PeopleSoft

Query Manager

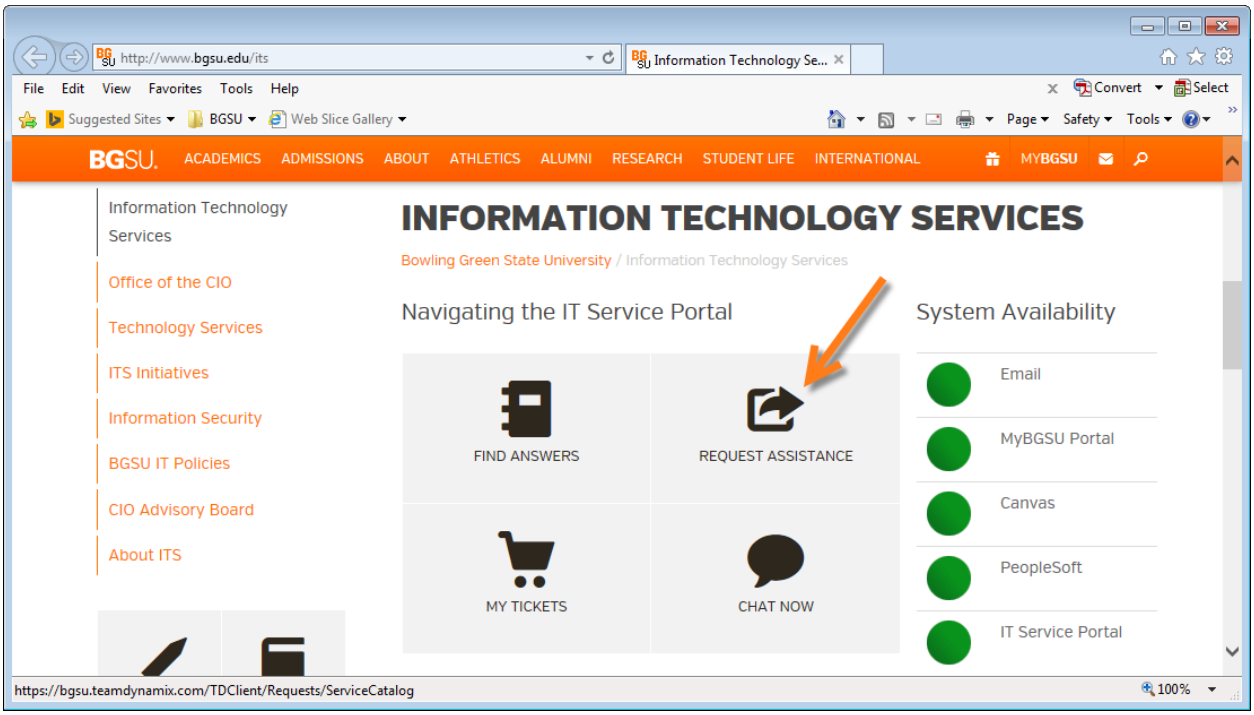
December 2015

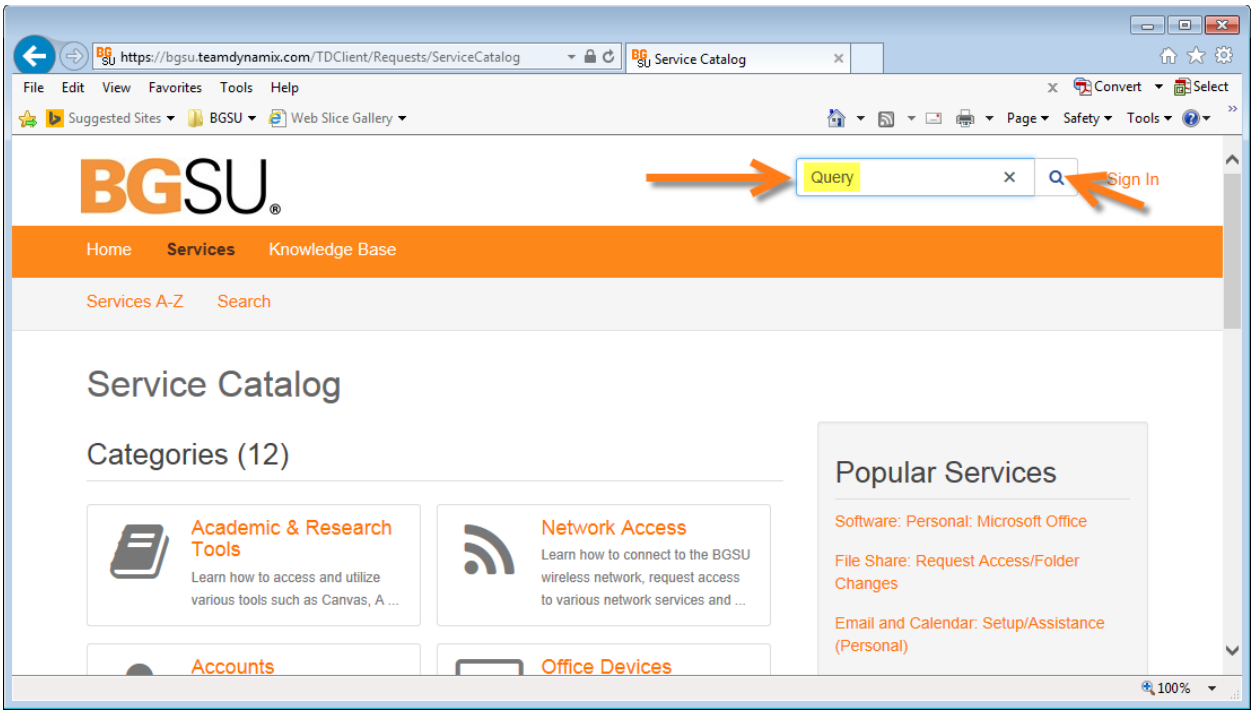
For assistance with building queries using Query Manager outside of this class, please contact the Technology Support Center (TSC).

Phone: 419-372-0999

Web:

1. Point your browser to http://www.bgsu.edu/its.
2. Click “Request Assistance”
3. Search for “Query”.
4. Click “PeopleSoft: Access Queries”.




https://bgsu.teamdynamix.com/TDClient/Requests/ServiceCatalog
Service Catalog
File Edit View Favorites Tools Help
Suggested Sites BGSU Web Slice Gallery
Page Safety Tools
BGSU
Query
Sign In
Home Services Knowledge Base
Services A-Z Search
Service Catalog
Categories (12)
Academic & Research Tools
Learn how to access and utilize various tools such as Canvas, A ...
Network Access
Learn how to connect to the BGSU wireless network, request access to various network services and ...
Accounts
Office Devices
Popular Services
Software: Personal: Microsoft Office
File Share: Request Access/Folder Changes
Email and Calendar: Setup/Assistance (Personal)
100%

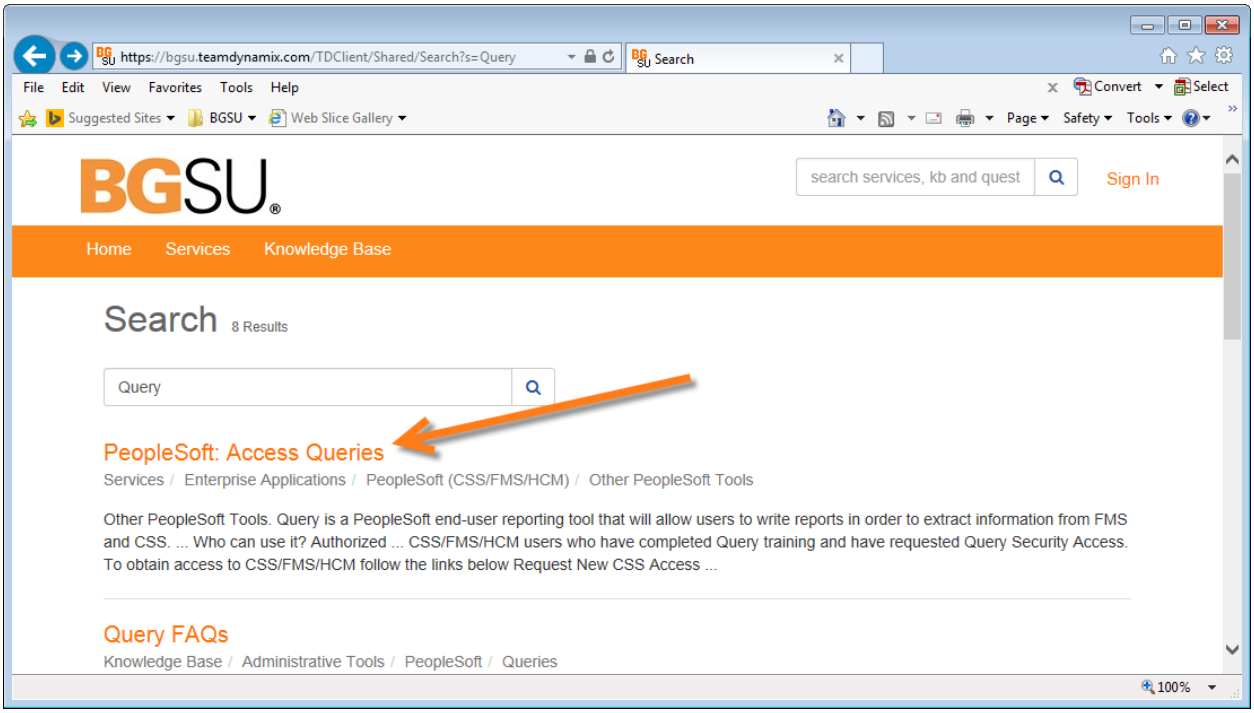

https://bgsu.teamdynamix.com/TDClient/Shared/Search?s=Query
Search
File Edit View Favorites Tools Help
Suggested Sites BGSU Web Slice Gallery
Page Safety Tools
BGSU
search services, kb and quest
Sign In
Home Services Knowledge Base
Search 8 Results
Query
PeopleSoft: Access Queries
Services / Enterprise Applications / PeopleSoft (CSS/FMS/HCM) / Other PeopleSoft Tools
Other PeopleSoft Tools. Query is a PeopleSoft end-user reporting tool that will allow users to write reports in order to extract information from FMS and CSS. ... Who can use it? Authorized ... CSS/FMS/HCM users who have completed Query training and have requested Query Security Access. To obtain access to CSS/FMS/HCM follow the links below Request New CSS Access ...
Query FAQs
Knowledge Base / Administrative Tools / PeopleSoft / Queries
100%

Table of Contents

INTRODUCTION

You have completed the first two classes of the BGSU PeopleSoft Query course: Fundamentals of Database Structure and Query Viewer. You are now ready for the third class, Query Manager. Query Manager is a hands on, two-full day class, providing useful query concepts. Query concepts include:

➢ Creating queries	➢ Edit Fields
➢ Edit Criteria	➢ Working with Prompts
➢ Writing Expressions	➢ Using Prompts & Expressions
➢ Using Aggregate Functions	➢ Inner & Outer Joins
➢ Subqueries	➢ Unions

Query Manager provides access to search for, view, run, create, modify, and save queries.

This material assumes that you are using Internet Explorer to access Query Manager. Your experience may differ when using other browsers such as Firefox, Chrome, and Safari.

About the Training Environment

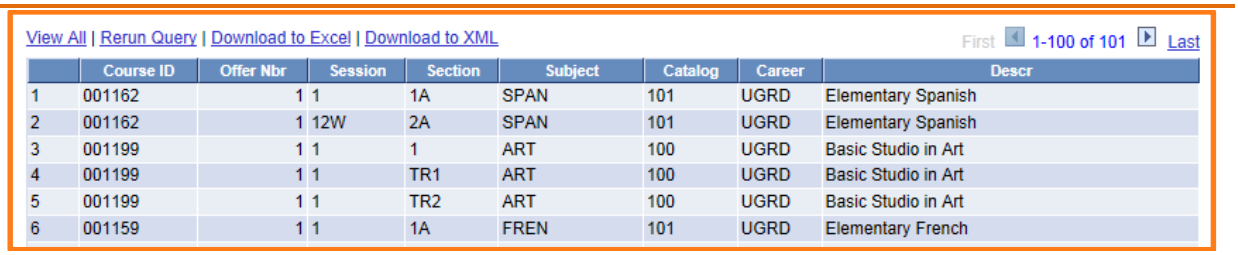
The data used in this class is the data supplied by Oracle Corporation with a vanilla (not customized) installation of the PeopleSoft Campus Solutions System (CSS). It will not necessarily resemble data that you will encounter in your daily work. For instance, the institution code for Bowling Green State University is “BGSUN”, but during training you will use “PSUNV”, the institution code for PeopleSoft University.

Also, while the exercises during training primarily involve data from CSS, the concepts and methods you will learn will also apply to Financial Management System (FMS) and Human Capital Management (HCM).

If you are uncertain about specific data values and records related to your area, contact the subject matter experts in your department.

What is a Query?

A query is a question or request for information. For example, your manager might ask for a list of laboratory classes that occurred during the fall 2007 semester. He would like to see the course ID, course offer number, session code, class section, subject, catalog number, academic career, and description fields. In order to obtain the results, several records can be joined together and saved as a query.



View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 101 Last

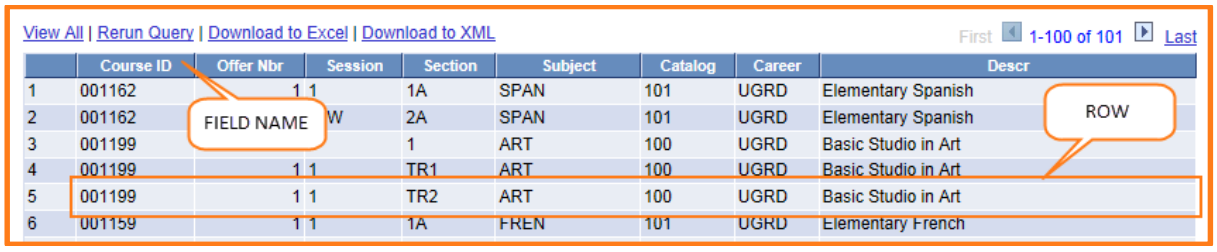
	Course ID	Offer Nbr	Session	Section	Subject	Catalog	Career	Descr
1	001162	1	1	1A	SPAN	101	UGRD	Elementary Spanish
2	001162	1	12W	2A	SPAN	101	UGRD	Elementary Spanish
3	001199	1	1	1	ART	100	UGRD	Basic Studio in Art
4	001199	1	1	TR1	ART	100	UGRD	Basic Studio in Art
5	001199	1	1	TR2	ART	100	UGRD	Basic Studio in Art
6	001159	1	1	1A	FREN	101	UGRD	Elementary French

When using Query Viewer the queries have already been created by a user that has access to Query Manager. In order to know what results are correct, you first have to understand the basic terminology of a query as well as how a query is structured.

If you recall from Query class I – Fundamentals of Database Structure the PeopleSoft query terms include:

Record:	Stores information about many objects of interest of the same type (laboratory classes).
Fields:	Individual characteristics for that object of interest (course ID, Offer Nbr, etc).
Rows:	All of the data that represents that object of interest. The entire row contains all of the information on the laboratory class.

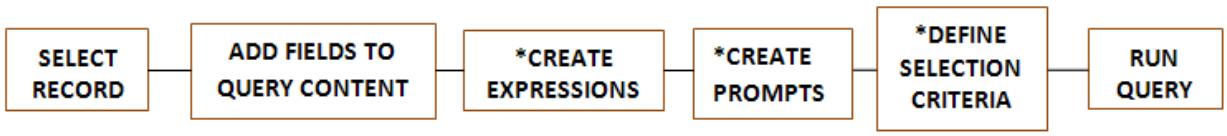
Let's take a deeper look at how a query is structured.



View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 101 Last

	Course ID	Offer Nbr	Session	Section	Subject	Catalog	Career	Descr
1	001162	1	1	1A	SPAN	101	UGRD	Elementary Spanish
2	001162		W	2A	SPAN	101	UGRD	Elementary Spanish
3	001199			1	ART	100	UGRD	Basic Studio in Art
4	001199	1	1	TR1	ART	100	UGRD	Basic Studio in Art
5	001199	1	1	TR2	ART	100	UGRD	Basic Studio in Art
6	001159	1	1	1A	FREN	101	UGRD	Elementary French

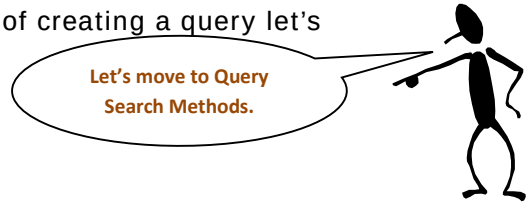
There is more to a query than records, fields, and rows. There is a process that assists in the creation of a query.



Records:	Search for records to add to your query (CLASS_TBL)
Fields:	Edit or change column headings, column order, and ordering results for the selected fields.
***Expressions:**	Formulas used to calculate a result from data in fields or other values.
***Prompts:**	Add, change, and delete the values the user is asked to enter when the query is run.

***Criteria:**	Set the conditions that determine which rows will appear in the results, based on the data in each row.
Run:	Display of the query results in a browser window.
* denotes optional task	

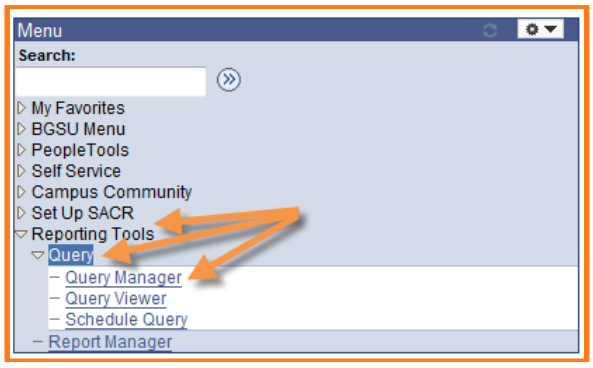
Now that you know what a query is and the process of creating a query let's move forward.



Basic Search

A Basic search can be used when some information about the Query name is known.

The navigation menu that appears may look very different from your navigation menu; a user's security access determines the CSS modules that are available to them.



Once signed into the system, click the arrow next to **Reporting Tools**. Next, click the arrow beside **Query**. Finally, **click** the **Query Manager** link.

The system takes you to the Basic search page. As you learned in Query class II – View a Query; located on the Basic search page is a Search By field with a drop down menu. The drop down menu has criteria that will assist in further defining a search. Only one criterion at a time can be used when performing a Basic search.

The system defaults to Search By: Query Name. There is a high volume of public queries that exist in the system. In order to identify custom queries (queries that are created by BGSU for BGSU) a naming convention has been created and is encouraged to be followed by users creating queries.

The following naming conventions apply to BGSU Custom *public* queries.

Name Component	R	Description	Valid Values
FACILITY	R	"BG" for BGSU	BG
UNDERSCORE	R	Separation	_
PRODUCT SUFFIX	R	Two character Product Abbreviation	See Table- Product Suffixes
UNDERSCORE	R	Separation	_
QUERY NAME	R	Descriptive name for the query	Developer Defined

Product Suffix (CSS)	
AD	Admissions
CC	Campus Community
FA	Financial Aid
HO	Housing
SF	Student Financials
SI	Student Insurance
SR	Student Records

Product Suffix (FMS)	
AP	Accounts Payable
GL	General Ledger
GM	Grants Management
PO	Purchasing Office

Product Suffix (HCM)	
BA	Benefits Administration
BN	Benefits
HR	Human Resources
PY	Payroll
TL	Time & Labor

EXAMPLES:

BG_SR_ALL_CLASS_RM_INFO

BG_AP_PMTS_ON_HOLD

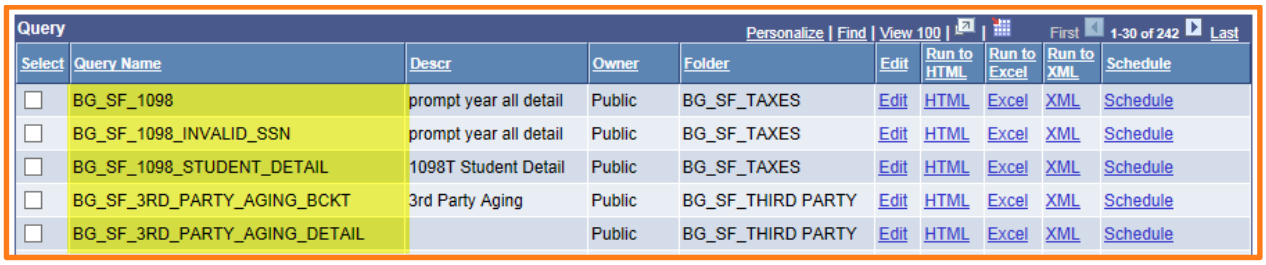
BG_FA_PKG_PLAN_SETUP

BG_GL_ACCOUNT_LIST

BG_HR_ALLDEPTMENTS

BG_HO_UGRD_HOUSING_REQUESTS

As Query Names appear in the system:



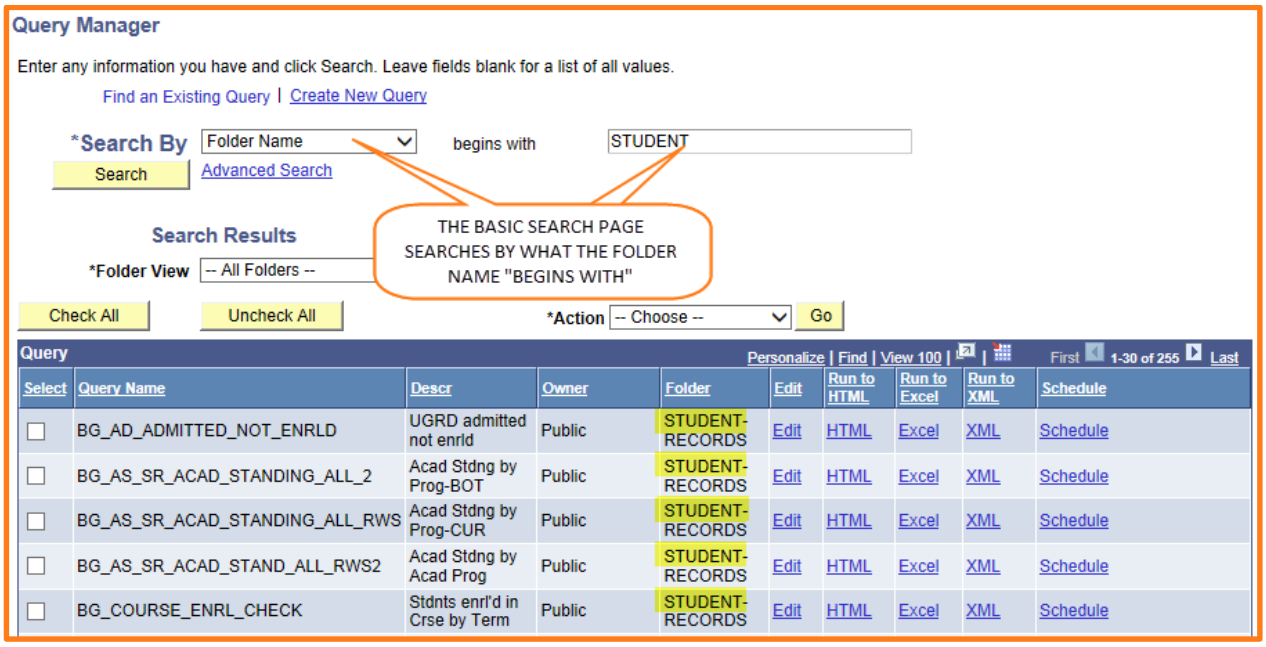
Query — Personalize | Find | View 100 | First 1-30 of 242 Last

Select	Query Name	Descr	Owner	Folder	Edit	Run to HTML	Run to Excel	Run to XML	Schedule
☐	BG_SF_1098	prompt year all detail	Public	BG_SF_TAXES	Edit	HTML	Excel	XML	Schedule
☐	BG_SF_1098_INVALID_SSN	prompt year all detail	Public	BG_SF_TAXES	Edit	HTML	Excel	XML	Schedule
☐	BG_SF_1098_STUDENT_DETAIL	1098T Student Detail	Public	BG_SF_TAXES	Edit	HTML	Excel	XML	Schedule
☐	BG_SF_3RD_PARTY_AGING_BCKT	3rd Party Aging	Public	BG_SF_THIRD PARTY	Edit	HTML	Excel	XML	Schedule
☐	BG_SF_3RD_PARTY_AGING_DETAIL		Public	BG_SF_THIRD PARTY	Edit	HTML	Excel	XML	Schedule

What if you don't know the Query Name? Then, a different criterion can be used to assist in your search. For example, you do not know the query name yet you know the Folder Name.

A Basic search will only search by the "begins with" condition type. When a query is placed in a public folder, often times the query owner will share the Folder Name.

Below is an example of using the criterion Folder Name with a "begins with" condition of "Student". As the example shows, all folders that begin with Student appear in the Search Results.



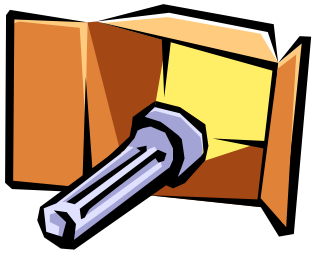
Query — Personalize | Find | View 100 | First 1-30 of 255 Last

Select	Query Name	Descr	Owner	Folder	Edit	Run to HTML	Run to Excel	Run to XML	Schedule
☐	BG_AD_ADMITTED_NOT_ENRLD	UGRD admitted not enrld	Public	STUDENT-RECORDS	Edit	HTML	Excel	XML	Schedule
☐	BG_AS_SR_ACAD_STANDING_ALL_2	Acad Stdng by Prog-BOT	Public	STUDENT-RECORDS	Edit	HTML	Excel	XML	Schedule
☐	BG_AS_SR_ACAD_STANDING_ALL_RWS	Acad Stdng by Prog-CUR	Public	STUDENT-RECORDS	Edit	HTML	Excel	XML	Schedule
☐	BG_AS_SR_ACAD_STAND_ALL_RWS2	Acad Stdng by Acad Prog	Public	STUDENT-RECORDS	Edit	HTML	Excel	XML	Schedule
☐	BG_COURSE_ENRL_CHECK	Stdnts enrl'd in Crse by Term	Public	STUDENT-RECORDS	Edit	HTML	Excel	XML	Schedule

Exercise 1

Practice using the Basic search method. Use the following criteria to record the number of queries that exist.

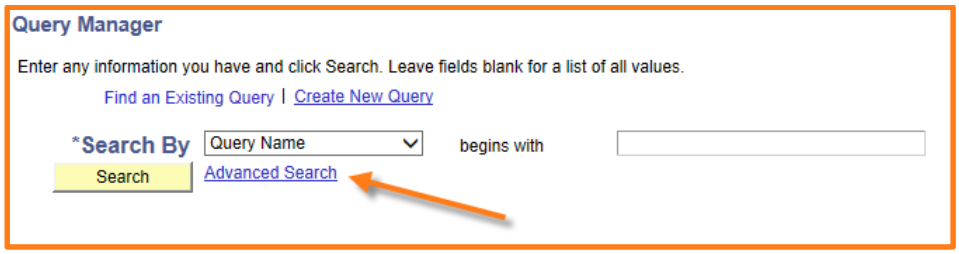
Search By	Begins With	Results
Query Name	FA	_____
	SR	_____
	SF	_____
	AD	_____
Folder Name	JPM	_____
Description	ISIR	_____
	Class	_____
	Course	_____
	Grade	_____
	Person	_____



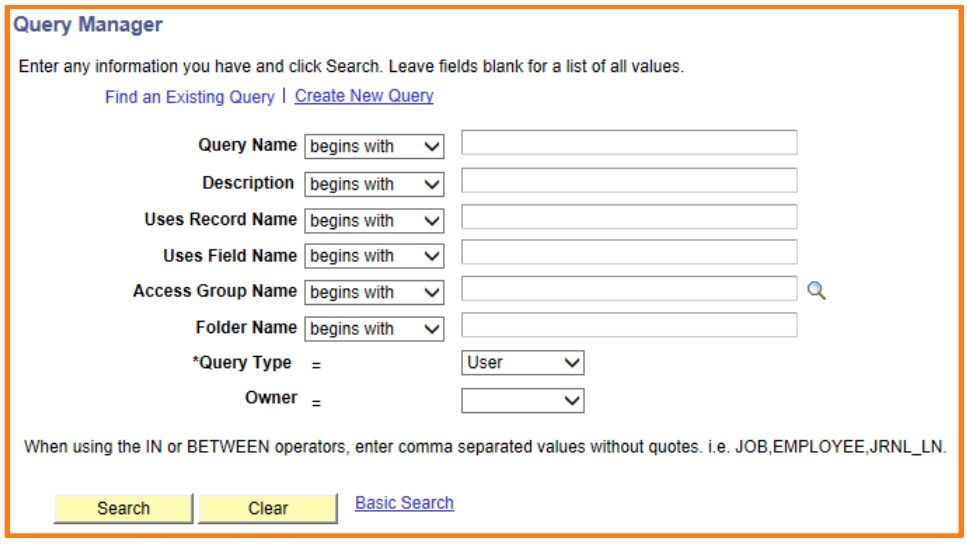
Advanced Search

Using the Basic search function is easy when you know what the query, folder, or description *begins with*. The Advanced search method allows several criteria to be used as well as a selection of conditions to place on the criteria.

Click the Advanced Search **link**



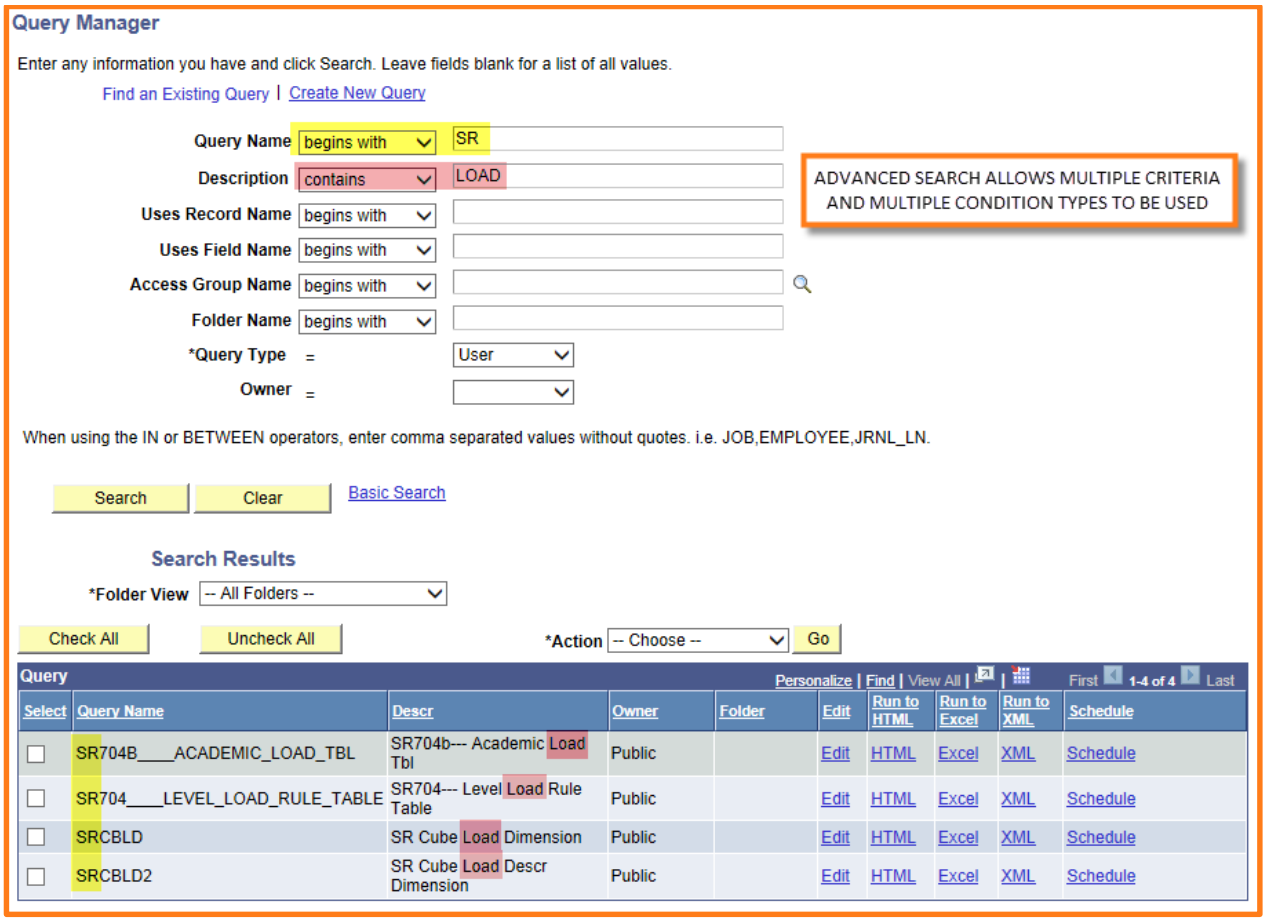
The system takes you to the Advanced search page.



Notice the fields on the Advanced search page are the same options that you find on the Basic search page in the Search By dropdown. The Advanced search page offers the use of several different criteria and condition types. Condition types are used with search criteria to limit the results you want to see when searching for a query.

begins with	applies to the start of <The folder name *begins with* Student>
between	find cases in which the criterion is between two values <amount owed is *between* $250.99 and $999.99>
contains	have within <the description *contains* the word "term">
in	the criterion is in a list of values <is *in* the list ACAD_PROG, ACAD_PLAN, DESCR>
not=	the value is not equal to what you enter <Term *not=* 0590>

Below is an example of using different criteria and different condition types to search for results.



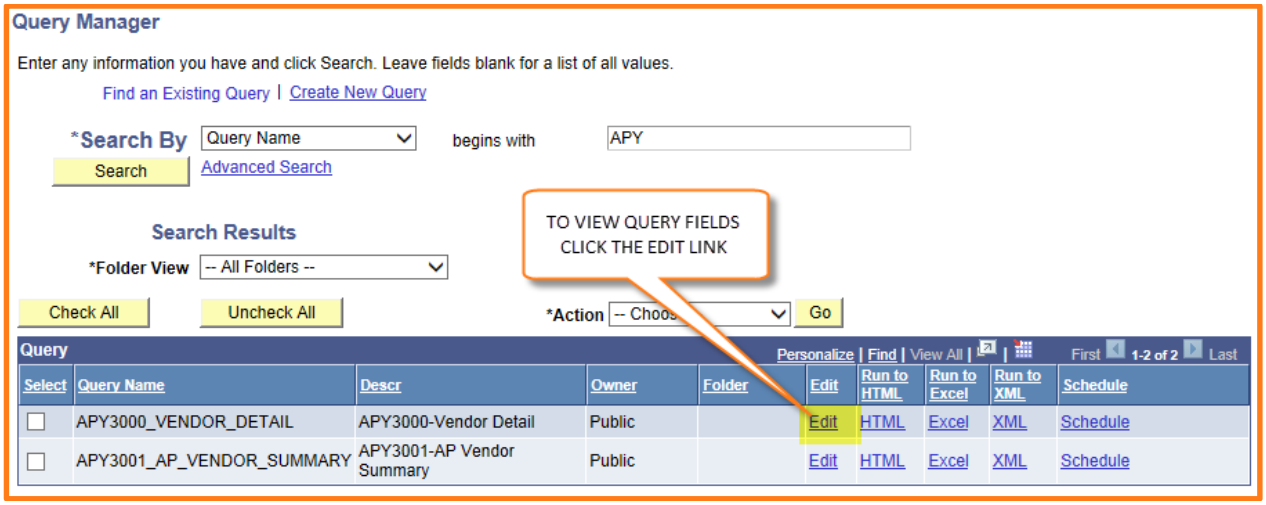
Exercise 2

This exercise will allow you to use multiple criteria and specific condition types selected for each criterion. When there are two criteria be sure to enter both of them before clicking the Search button.

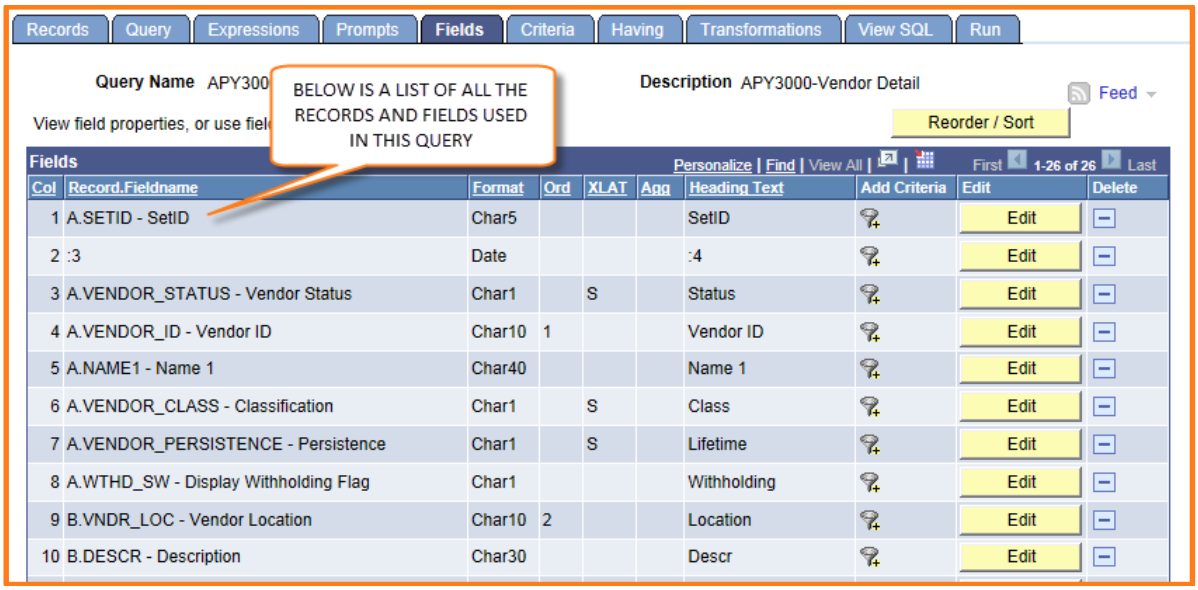
Criteria	Condition Type	Information
Query Name	contains	Campus
Description	contains	Location _____
Query Name	contains	Schedule
Description	contains	Train _____
Uses Record Name	begins with	ACAD_PLAN _____
Description	contains	Term _____
Uses Field Name	begins with	Acad _____
Uses Field Name	begins with	Session _____
Uses Field Name	begins with	INSTR _____

Retrieving Queries

Now that you are familiar with the different search methods and naming conventions, let's retrieve the query that we are searching for. For this example, the Basic Search method is used. The Search By field criteria is Folder Name and the search is on Career.



There are several results that meet the search criterion. Click the Edit link. The system places you on the Fields page. Additional query building pages are accessed by clicking the desired tab.



Page Description:

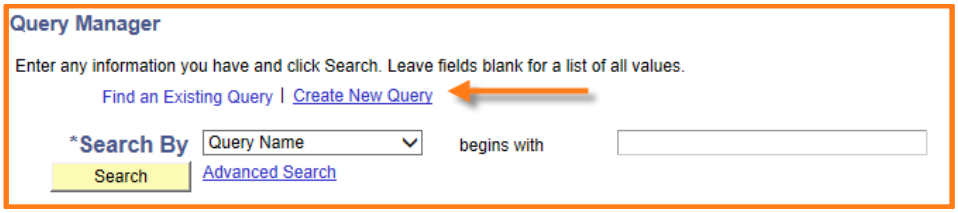
Records -	Search for records to add to your query, view a list of fields in a record, and add a record to your query.
Query -	Select fields to be displayed in the results. You can also join additional records to your query here and delete records.
Expressions -	Formulas used to calculate a result from data in fields and other values.
Prompts -	Add, change, and delete the values the user is asked to enter when the query is run.
Fields -	Edit or change column headings, column positions, and ordering results for the fields selected
Criteria -	Set the conditions that determine which rows will appear in the results, based on the data in each row.
Having -	Set the conditions that determine which rows will appear in the results, based on the data in a group of rows.
View SQL -	View SQL statement generated whenever the query is modified.
Run -	Display query results in the browser window.

Create a Simple Query

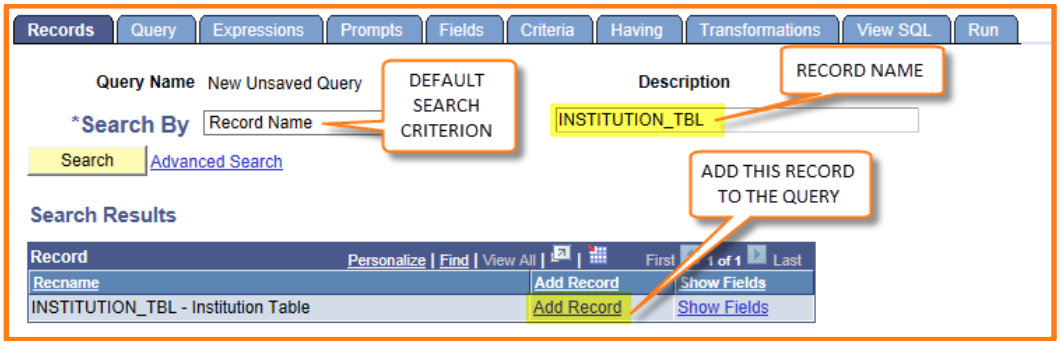
In this exercise you are asked to get the basic information about all institutions recorded in the system. Let's create a query using the **INSTITUTION_TBL** record to demonstrate the default criteria, Check All button, and the Effective Date criteria.

Creating a query isn't as hard as we think it is. Creating queries becomes challenging when you have to locate the records and know what data resides in them. For purposes of training, you will be provided with all of the record and field names.

Starting in Query Manager, **click Create New Query**.

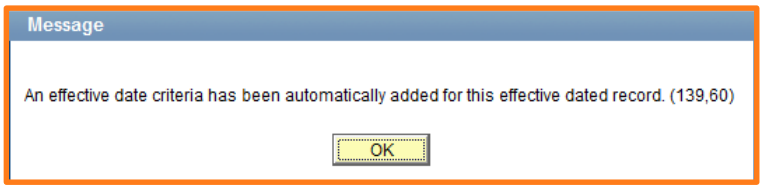


The Records page is displayed with default search criterion as Record Name. **Enter CAMPUS_TBL** then click the **Search** button.



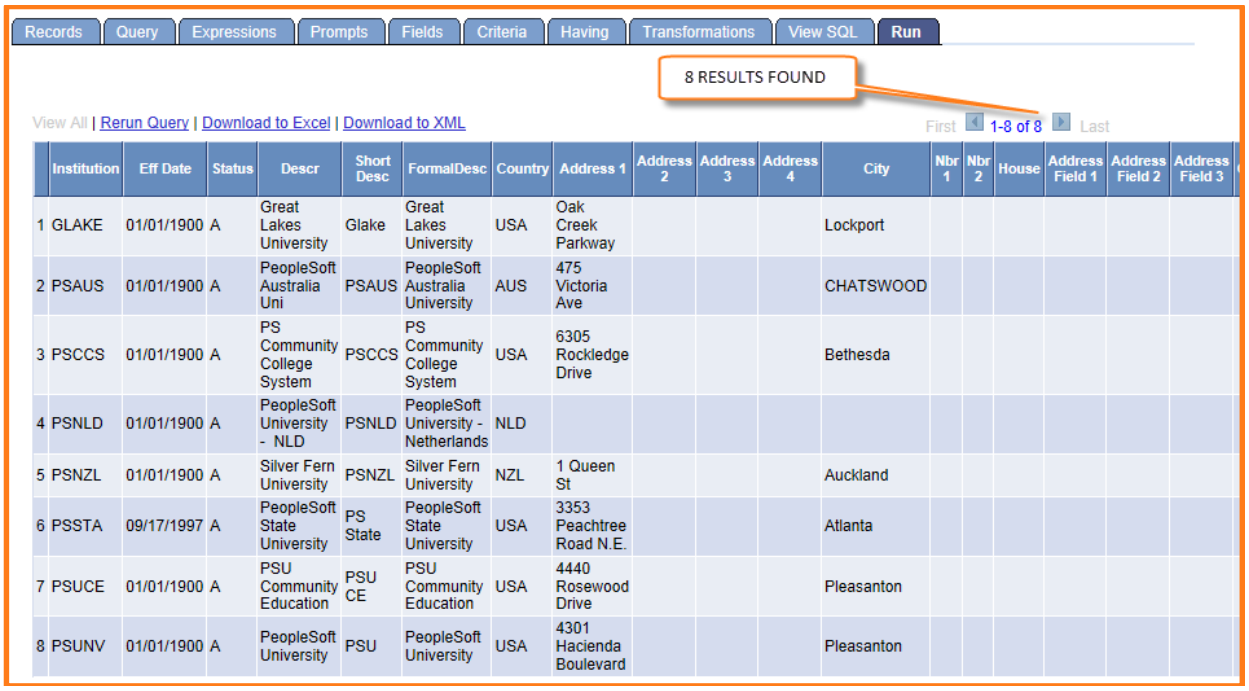
Click the **Add Record** link.

Click OK to acknowledge that an effective date criteria has been automatically added.



Click the **Check All** button to add all fields.

Click the **Run** tab.

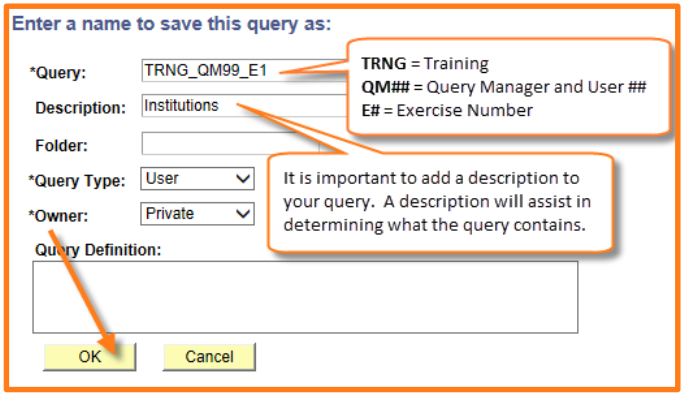


	Institution	Eff Date	Status	Descr	Short Desc	FormalDesc	Country	Address 1	Address 2	Address 3	Address 4	City	Nbr 1	Nbr 2	House	Address Field 1	Address Field 2	Address Field 3
1	GLAKE	01/01/1900	A	Great Lakes University	Glake	Great Lakes University	USA	Oak Creek Parkway				Lockport						
2	PSAUS	01/01/1900	A	PeopleSoft Australia Uni	PSAUS	PeopleSoft Australia University	AUS	475 Victoria Ave				CHATSWOOD						
3	PSCCS	01/01/1900	A	PS Community College System	PSCCS	PS Community College System	USA	6305 Rockledge Drive				Bethesda						
4	PSNLD	01/01/1900	A	PeopleSoft University - NLD	PSNLD	PeopleSoft University - Netherlands	NLD											
5	PSNZL	01/01/1900	A	Silver Fern University	PSNZL	Silver Fern University	NZL	1 Queen St				Auckland						
6	PSSTA	09/17/1997	A	PeopleSoft State University	PS State	PeopleSoft State University	USA	3353 Peachtree Road N.E.				Atlanta						
7	PSUCE	01/01/1900	A	PSU Community Education	PSU CE	PSU Community Education	USA	4440 Rosewood Drive				Pleasanton						
8	PSUNV	01/01/1900	A	PeopleSoft University	PSU	PeopleSoft University	USA	4301 Hacienda Boulevard				Pleasanton						

That's it! You have results!

As noted earlier, at Bowling Green State University we have only one institution, which has the designation BGSUN. The default data supplied by Oracle with PeopleSoft has several institutions. During this training in most cases we will use only data from the PeopleSoft University institution, which has the identifier PSUNV.

Now, let's save the query. The *only* page that *does not* have the save option is Run. **Click** the **Records** tab then **click Save As**.



During training, the same naming convention for saving will be used. Enter **TRNG_QM##_E1** for the query name. To help you remember what this query is about, you may also wish to enter a description.

Finally, **click** the **OK** button.

The query name and description each have a maximum length of 30 characters. You may need to abbreviate some terms.

A meaningful description can help you identify a query. The description appears in the list of queries when you search. You can also search for queries by using words in the description.

Question 1A

The first query you will create is a query to list all campuses. Show all fields from the campus record in the results.

- Campuses are in the **CAMPUS_TBL** record
- Save and Save often. Save this query with the name **TRNG_QM##_Q1A**

What is the code corresponding to the Columbia Campus?

_____

What does the code WALCR represent? _____

View All | Rerun Query | Download to Excel | Download to XML First 1-14 of 14 Last

	Institution	Campus	Eff Date	Status	Descr	Short Desc	Location	Conflict Check
1	GLAKE	MAIN	01/01/1900	A	Main Campus	MainCampus	PSCSCLWD	Y
2	PSAUS	MAIN	01/01/1900	A	Main Campus	Main	PSAUSMAIN	N
3	PSCCS	COLUM	01/01/1900	A	Columbia Campus	Columbia	COLUMBIA	Y
4	PSCCS	MAIN	01/01/1900	A	Main Campus	Main	BETHESDA	
5	PSCCS	NAVY	01/01/1900	A	Navy Campus	Navy	ANNAPOLIS	
6	PSNLD	AMS	01/01/1900	A	Amsterdam Campus	Amsterdam	PSCSNLD2	Y
7	PSNLD	MAIN	01/01/1900	A	Main Campus	Main	PSCSNLD2	Y
8	PSNZL	EAST	01/01/1900	A	East Campus	East	PSNZL01	
9	PSNZL	MAIN	01/01/1900	A	Main Campus	Main	PSNZL01	
10	PSNZL	NORTH	01/01/1900	A	North Campus	North	PSNZL01	
11	PSNZL	SOUTH	01/01/1900	A	South Campus	South	PSNZL01	
12	PSNZL	WEST	01/01/1900	A	West Campus	West	PSNZL01	
13	PSUNV	MAIN	01/01/1900	A	Main Hacienda Campus	Main	PSCSHCDA	Y
14	PSUNV	WALCR	01/01/1900	A	Walnut Creek Campus	Walnut	PSCSWAL2	Y

Observe that each campus belongs to an institution. An organization may have several institutions, each of which may have several campuses. In this training PeopleSoft University (PSUNV) has two campuses: Main Hacienda Campus (MAIN) and Walnut Creek Campus (WALCR).

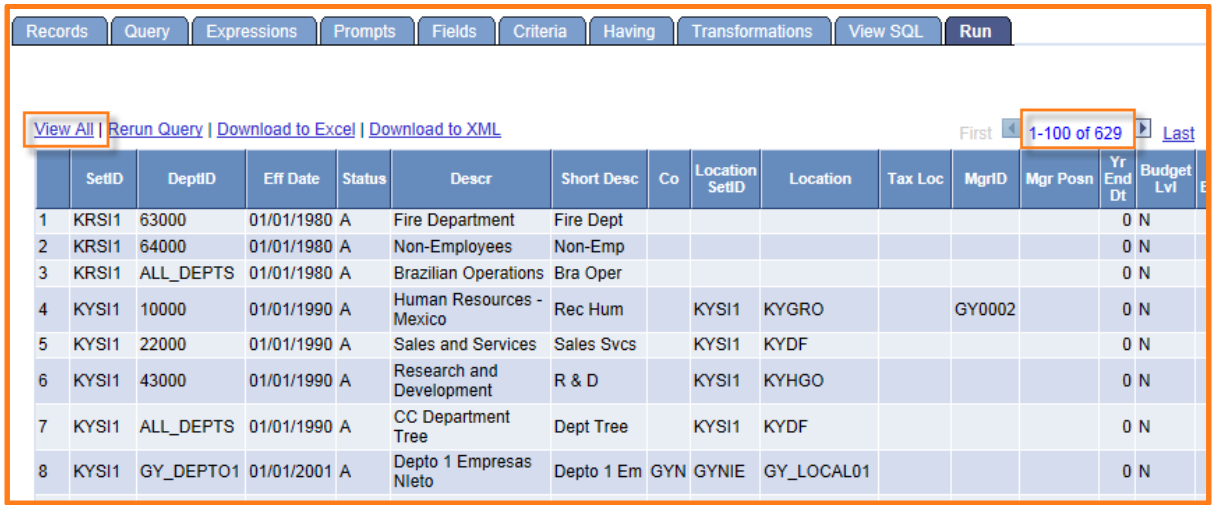
At BGSU there are four campuses belonging to the BGSUN institution: Distance (DIST), eCampus (ECAM), Firelands (FIRE), and Main Campus (MAIN).

Later in this class you will learn how to add restrictions to your queries to limit the results to those for the PeopleSoft University institution.

Question 1B

This question being asked is to create a query that lists all of the known departments. Show every field of each department.

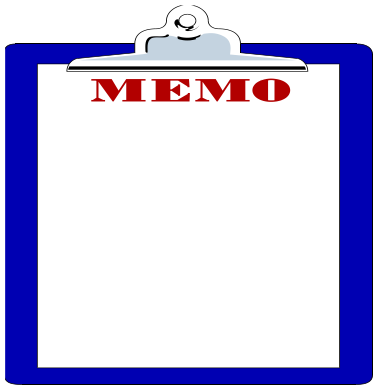
- Departments are stored in the **DEPT_TBL** record
- There will be several thousand rows in the results. Use the **View All** link and your browser's **Find** feature to locate the information you need
- Save and Save often. **Save As TRNG_QM##_Q1B**



	SetID	DeptID	Eff Date	Status	Descr	Short Desc	Co	Location SetID	Location	Tax Loc	MgrID	Mgr Posn	Yr End Dt	Budget Lvl
1	KRSI1	63000	01/01/1980	A	Fire Department	Fire Dept							0	N
2	KRSI1	64000	01/01/1980	A	Non-Employees	Non-Emp							0	N
3	KRSI1	ALL_DEPTS	01/01/1980	A	Brazilian Operations	Bra Oper							0	N
4	KYSI1	10000	01/01/1990	A	Human Resources - Mexico	Rec Hum		KYSI1	KYGRO		GY0002		0	N
5	KYSI1	22000	01/01/1990	A	Sales and Services	Sales Svcs		KYSI1	KYDF				0	N
6	KYSI1	43000	01/01/1990	A	Research and Development	R & D		KYSI1	KYHGO				0	N
7	KYSI1	ALL_DEPTS	01/01/1990	A	CC Department Tree	Dept Tree		KYSI1	KYDF				0	N
8	KYSI1	GY_DEPTO1	01/01/2001	A	Depto 1 Empresas Nleto	Depto 1 Em	GYN	GYNIE	GY_LOCAL01				0	N

What is the department ID for University Advancement? _____

Which department has the ID 97200? _____



Edit Fields – Select Fields to Display

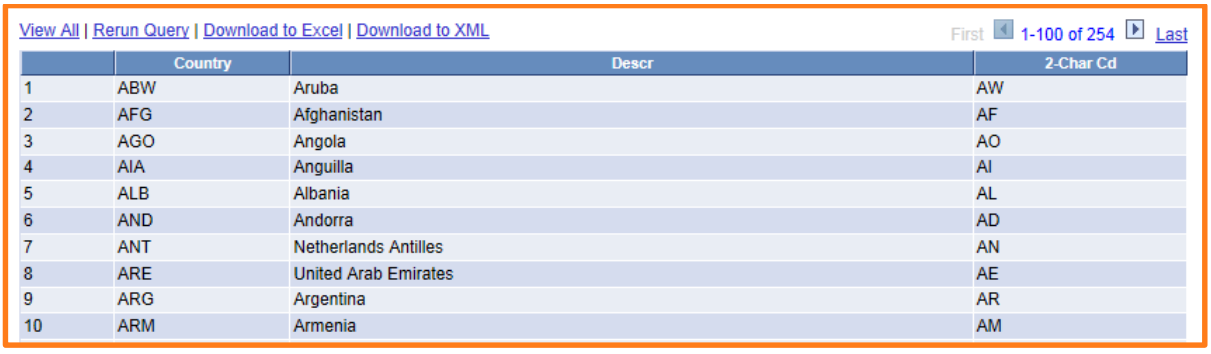
As you have seen, selecting all of the fields can result in dozens of columns. Often times we will want to select the fields we want to display in the results. If you have noticed, short descriptions are provided next to the fields, on the Query page. Let's see.

We will get a list of all countries known to PeopleSoft. Rather than showing all of the fields of each country, let's show only the country, description, and two-character code.

- Create a query on **COUNTRY_TBL**
- Show the following fields:
  - **COUNTRY**
  - **DESCR**
  - **COUNTRY_2CHAR**

- **Click Run** tab

Your results show three fields: Country, Description, and 2-Character Code.



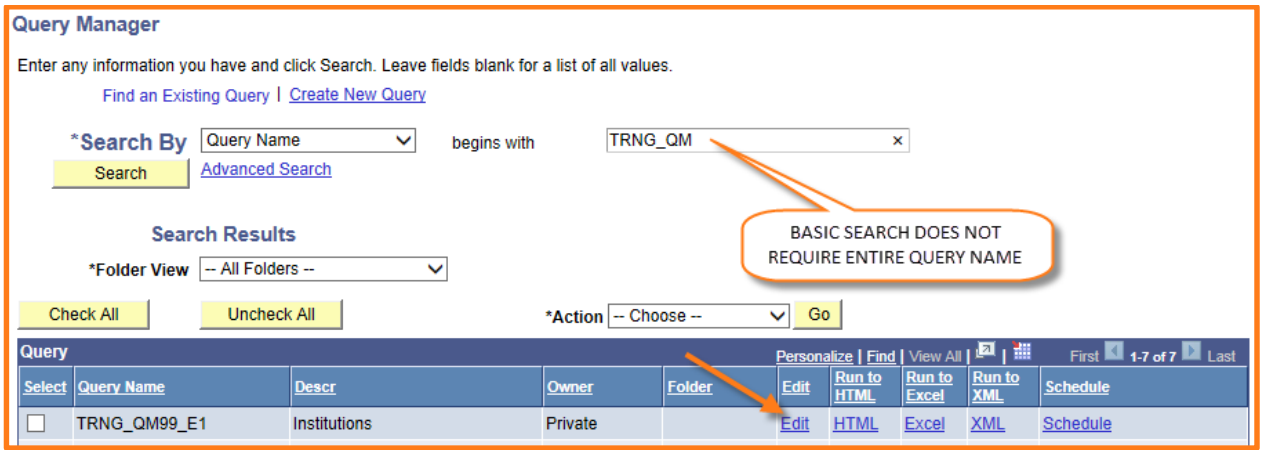
View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 254 Last

	Country	Descr	2-Char Cd
1	ABW	Aruba	AW
2	AFG	Afghanistan	AF
3	AGO	Angola	AO
4	AIA	Anguilla	AI
5	ALB	Albania	AL
6	AND	Andorra	AD
7	ANT	Netherlands Antilles	AN
8	ARE	United Arab Emirates	AE
9	ARG	Argentina	AR
10	ARM	Armenia	AM

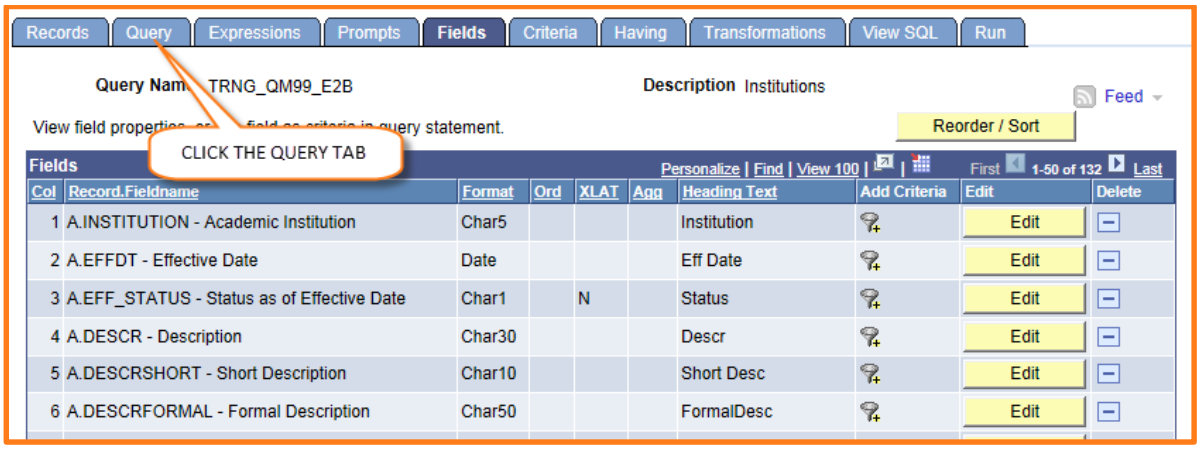
- Save and Save often. Save As **TRNG_QM##_E2A**

Next, assume that you do not need all of the information about each campus in the query you created earlier. Let's remove three of the fields from the results.

Go back to the Query Manager Basic search page. Search for the **TRNG_QM##_E1** query you built earlier.

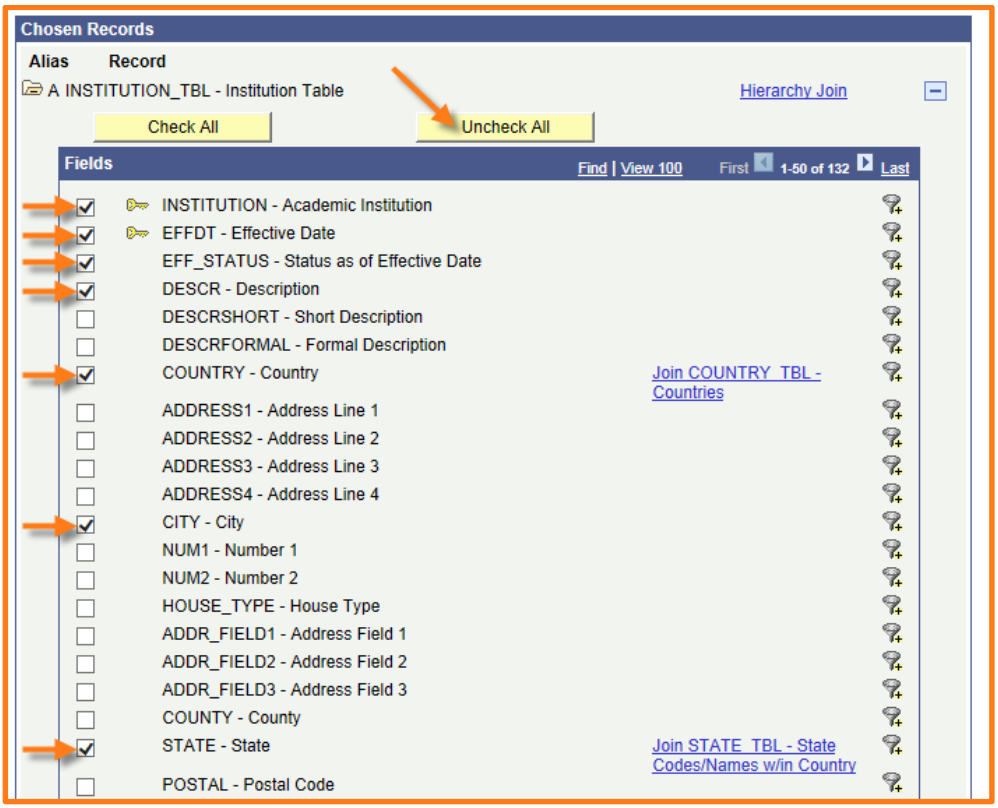


- **Click** on the **Edit link.** This will place you on the **Fields** page.
- You will be creating a copy of the original query then editing the copy. **Click** the **Save As** link and save this query with the name **TRNG_QM##_E2B**. (It is important to use **Save As** so you don't overwrite the original query.)
- **Click** the **Query** tab



Earlier we used this example to "check all" fields. Now we are going to uncheck fields then select only the fields we want. Click the **Uncheck All** button then select the following fields:

- **INSTITUTION**
- **EFFDT**
- **EFF_STATUS**
- **DESCR**
- **COUNTRY**
- **CITY**
- **STATE**



Save the query again then run the query. Notice anything with the results? The amount of rows remains the same. Three fields have been removed from the results.

View All | Rerun Query | Download to Excel | Download to XML — First 1-8 of 8 Last

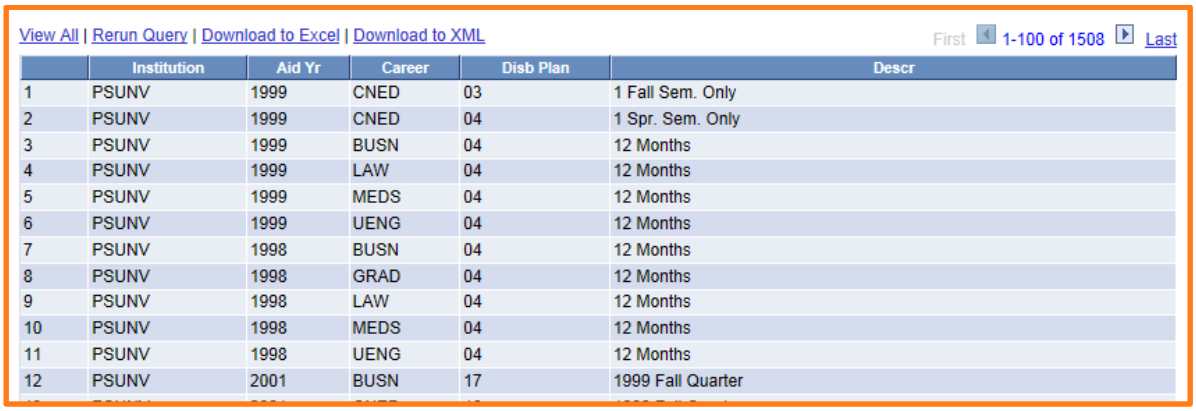
	Institution	Eff Date	Status	Descr	Country	City	State
1	GLAKE	01/01/1900	A	Great Lakes University	USA	Lockport	IL
2	PSAUS	01/01/1900	A	PeopleSoft Australia Uni	AUS	CHATSWOOD	NSW
3	PSCCS	01/01/1900	A	PS Community College System	USA	Bethesda	MD
4	PSNLD	01/01/1900	A	PeopleSoft University - NLD	NLD		
5	PSNZL	01/01/1900	A	Silver Fern University	NZL	Auckland	
6	PSSTA	09/17/1997	A	PeopleSoft State University	USA	Atlanta	GA
7	PSUCE	01/01/1900	A	PSU Community Education	USA	Pleasanton	CA
8	PSUNV	01/01/1900	A	PeopleSoft University	USA	Pleasanton	CA

The fields selected in the Query tab affect only which fields are shown in the results. Consider this example: your neighbor has 20 books on his bookshelf. If you ask him about all the books that are on his bookshelf, he can tell you only the title of the book, the title and author of each book, or the title, author, and publisher of each book, but he will always tell you about all 20 books.

Question 2

For this question, create a query that lists all aid disbursement plans at BGSU. Show only the aid year, career, disbursement plan code, and description of each plan.

- Use the **DISB_PLAN_TBL** record
- Show *only* the **INSTITUTION**, **AID_YEAR**, **ACAD_CAREER**, **DISBURSEMENT_PLAN**, and **DESCR** fields
- Save and Save often. Save this query as **TRNG_QM##_Q2**



View All | Rerun Query | Download to Excel | Download to XML

First 1-100 of 1508 Last

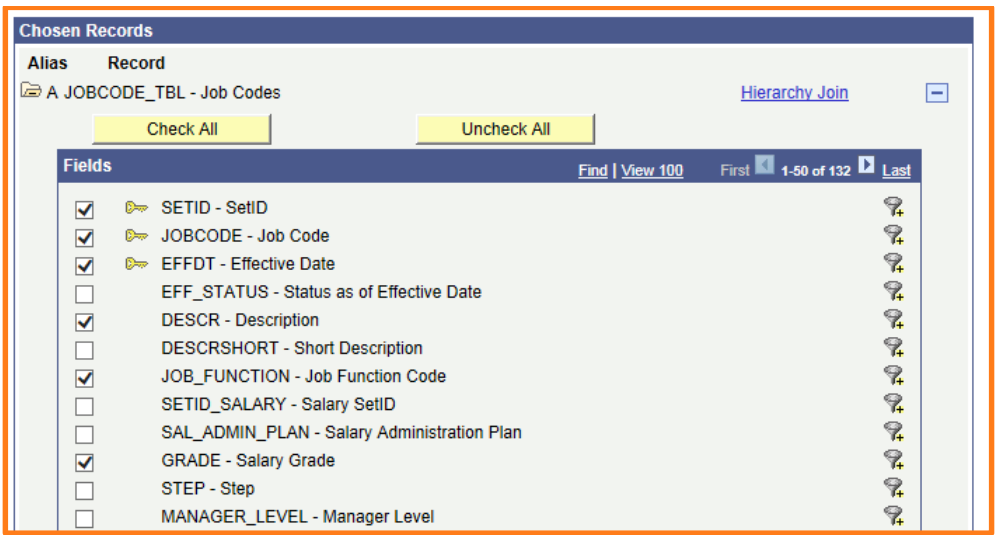
	Institution	Aid Yr	Career	Disb Plan	Descr
1	PSUNV	1999	CNED	03	1 Fall Sem. Only
2	PSUNV	1999	CNED	04	1 Spr. Sem. Only
3	PSUNV	1999	BUSN	04	12 Months
4	PSUNV	1999	LAW	04	12 Months
5	PSUNV	1999	MEDS	04	12 Months
6	PSUNV	1999	UENG	04	12 Months
7	PSUNV	1998	BUSN	04	12 Months
8	PSUNV	1998	GRAD	04	12 Months
9	PSUNV	1998	LAW	04	12 Months
10	PSUNV	1998	MEDS	04	12 Months
11	PSUNV	1998	UENG	04	12 Months
12	PSUNV	2001	BUSN	17	1999 Fall Quarter

Edit Fields – Display Order and Sort Order

Sometimes the results of the query do not look "organized". The order of columns from left to right may not be ideal for your purposes. The results may have no particular order, with values in primary fields going from low to high and back to low throughout. You can use the Reorder/Sort feature in the Fields tab to set the order in which columns are displayed from left to right and the ordering of rows in the results.

Let's get a list of job codes in PeopleSoft. Show only the SET ID, job code, effective date, description, job function, and grade for each job code. We will then make the description be the left-most column in the results and order the rows so that they are listed by ascending job code and descending grade.

- Use the following record **JOBCODE_TBL** (job codes)
- Show the following fields:
  - **SETID**
  - **JOBCODE**
  - **EFFDT**
  - **DESCR**
  - **JOB_FUNCTION**
  - **GRADE**



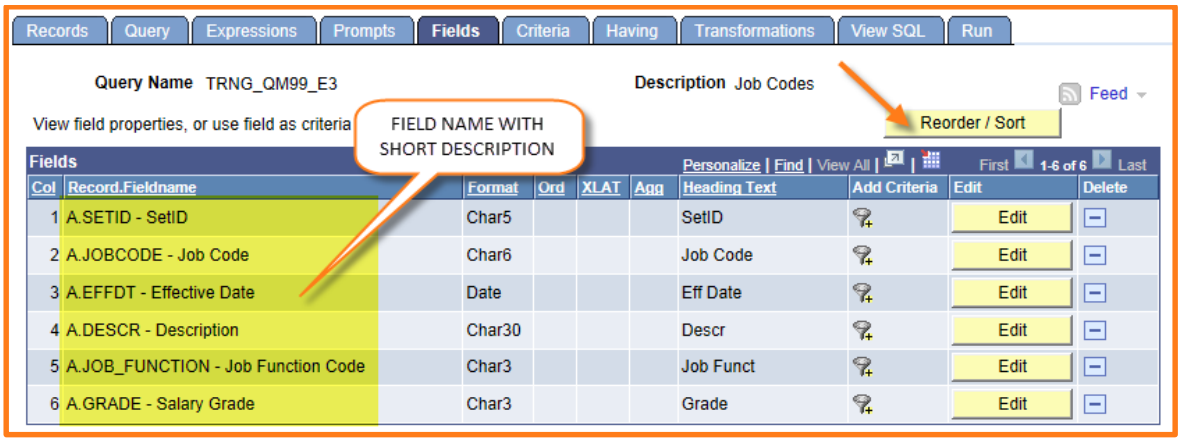
- **Save** the query as **TRNG_QM##_E3**.
- **Click Run** tab. Notice that the fields are in the same order as the fields listed in the Query tab

View All | Rerun Query | Download to Excel | Download to XML First 1-100 of 865 Last

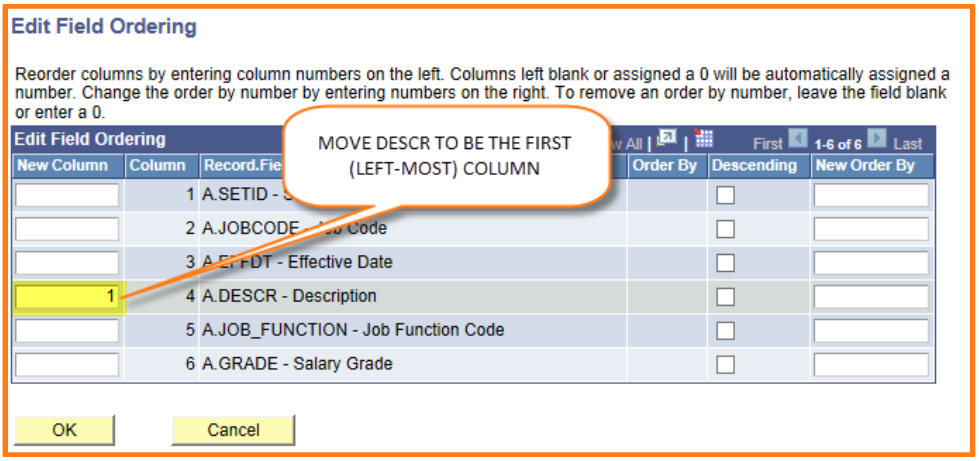
	SetID	Job Code	Eff Date	Descr	Job Funct	Grade
1	K1USA	910005	01/01/2004	Trainee		
2	K1USA	AADUMY	01/01/1900	Additional Appointment		
3	KMMYS	120010	01/01/1980	Administrator-Human Resources	HRS	
4	KMMYS	170005	01/01/1980	Assistant-Administrative	ADM	
5	KMMYS	170035	01/01/1980	Assistant-Marketing	MKT	
6	KMMYS	420050	01/01/1980	Director-Finance	FIN	
7	KMMYS	600085	01/01/1980	Manager-Finance	FIN	
8	KMMYS	610000	01/01/1980	Vice President	MGT	
9	KMMYS	620000	01/01/1980	President	MGT	
10	KMMYS	650000	01/01/1980	Chief Executive Officer	MGT	

To better organize the structure of the results, let's place the DESCR field as the left-most column. Notice on the Fields page there are short descriptions provided for the field names.

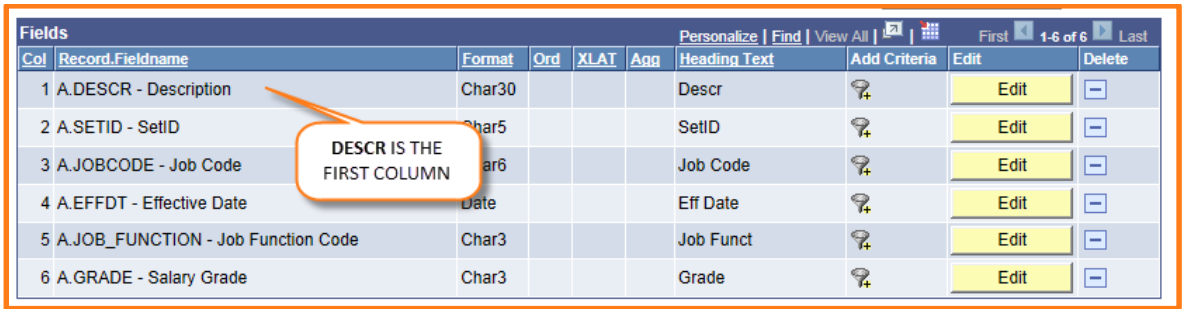
- **Click Fields** tab. This will allow us to reorder fields and sort rows
- **Click Reorder / Sort** button



- Enter a **1** in the **New Column** field of the **DESCR** row to move DESCR to the first column
- **Click OK**



The Fields page appears. DESCR is now the first column.

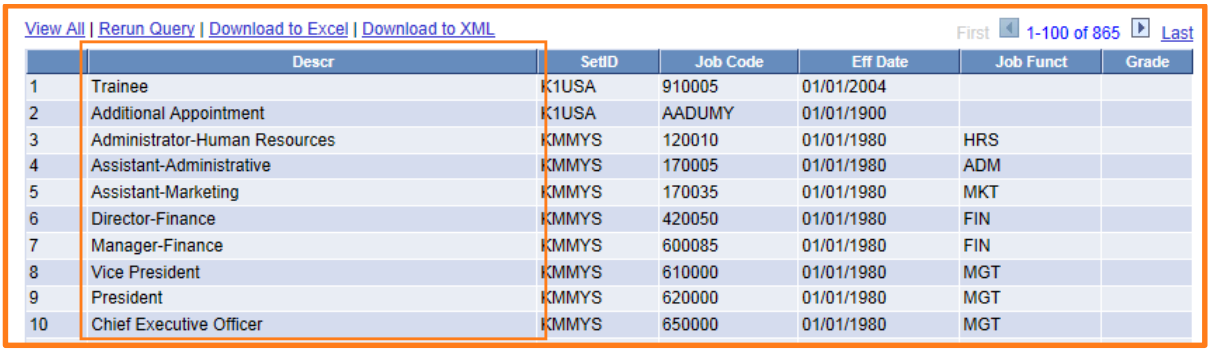


Fields — Personalize | Find | View All | First 1-6 of 6 Last

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.DESCR - Description	Char30				Descr		Edit	
2	A.SETID - SetID	Char5				SetID		Edit	
3	A.JOBCODE - Job Code	Char6				Job Code		Edit	
4	A.EFFDT - Effective Date	Date				Eff Date		Edit	
5	A.JOB_FUNCTION - Job Function Code	Char3				Job Funct		Edit	
6	A.GRADE - Salary Grade	Char3				Grade		Edit	

Save the query.

Click the **Run** tab. View the results. Notice the DESCR field is the left-most column

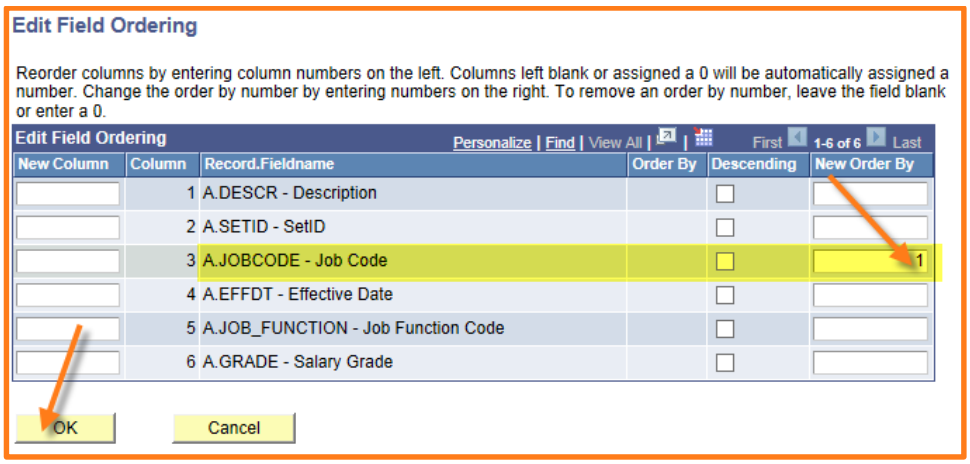


View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 865 Last

	Descr	SetID	Job Code	Eff Date	Job Funct	Grade
1	Trainee	K1USA	910005	01/01/2004		
2	Additional Appointment	K1USA	AADUMY	01/01/1900		
3	Administrator-Human Resources	KMMYS	120010	01/01/1980	HRS	
4	Assistant-Administrative	KMMYS	170005	01/01/1980	ADM	
5	Assistant-Marketing	KMMYS	170035	01/01/1980	MKT	
6	Director-Finance	KMMYS	420050	01/01/1980	FIN	
7	Manager-Finance	KMMYS	600085	01/01/1980	FIN	
8	Vice President	KMMYS	610000	01/01/1980	MGT	
9	President	KMMYS	620000	01/01/1980	MGT	
10	Chief Executive Officer	KMMYS	650000	01/01/1980	MGT	

Next, go back to the Edit Field Ordering screen and sort the results by job code.

- **Click Fields** tab
- **Click Reorder / Sort** button
- **Place** a "**1**" in the **New Order By** column in the **A.JOBCODE** row
- **Click OK**



Edit Field Ordering

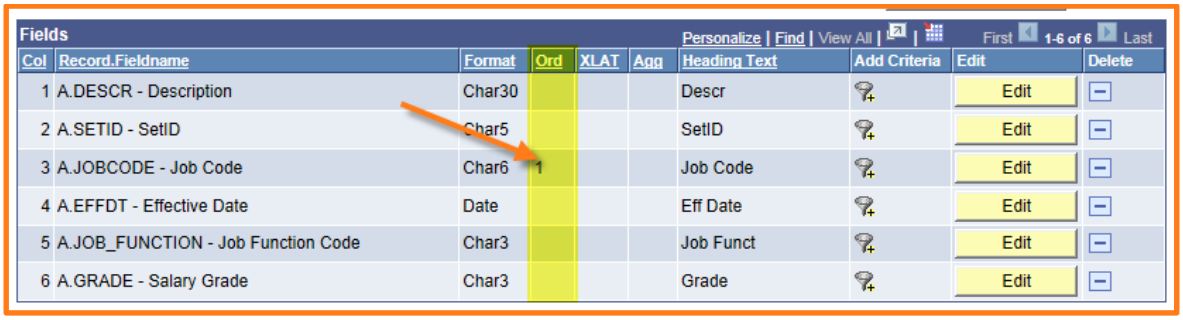
Reorder columns by entering column numbers on the left. Columns left blank or assigned a 0 will be automatically assigned a number. Change the order by number by entering numbers on the right. To remove an order by number, leave the field blank or enter a 0.

Edit Field Ordering — Personalize | Find | View All | First 1-6 of 6 Last

New Column	Column	Record.Fieldname	Order By	Descending	New Order By
	1	A.DESCR - Description			
	2	A.SETID - SetID			
	3	A.JOBCODE - Job Code			1
	4	A.EFFDT - Effective Date			
	5	A.JOB_FUNCTION - Job Function Code			
	6	A.GRADE - Salary Grade			

OK Cancel

In the Fields page, a "1" indicator is now shown in the Ord (Order By) column for the JOBCODE field.

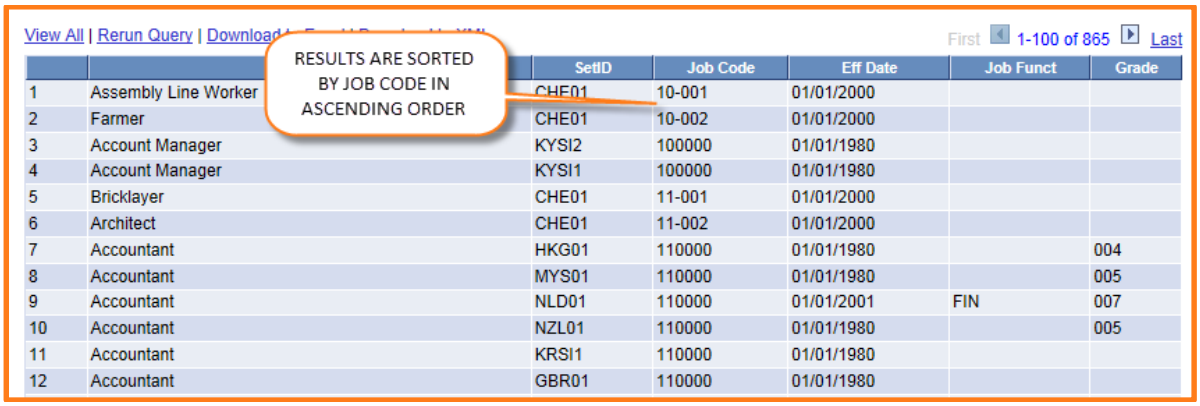


Fields — Personalize | Find | View All | First 1-6 of 6 Last

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.DESCR - Description	Char30				Descr		Edit	
2	A.SETID - SetID	Char5				SetID		Edit	
3	A.JOBCODE - Job Code	Char6	1			Job Code		Edit	
4	A.EFFDT - Effective Date	Date				Eff Date		Edit	
5	A.JOB_FUNCTION - Job Function Code	Char3				Job Funct		Edit	
6	A.GRADE - Salary Grade	Char3				Grade		Edit	

- **Click Run** tab to view results

Results show that the rows are ordered by job code in *ascending order*.



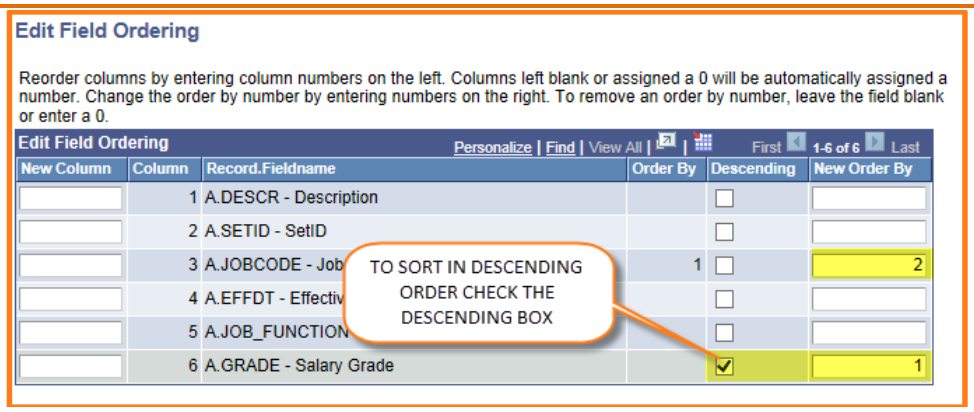
View All | Rerun Query | Download — First 1-100 of 865 Last

RESULTS ARE SORTED BY JOB CODE IN ASCENDING ORDER

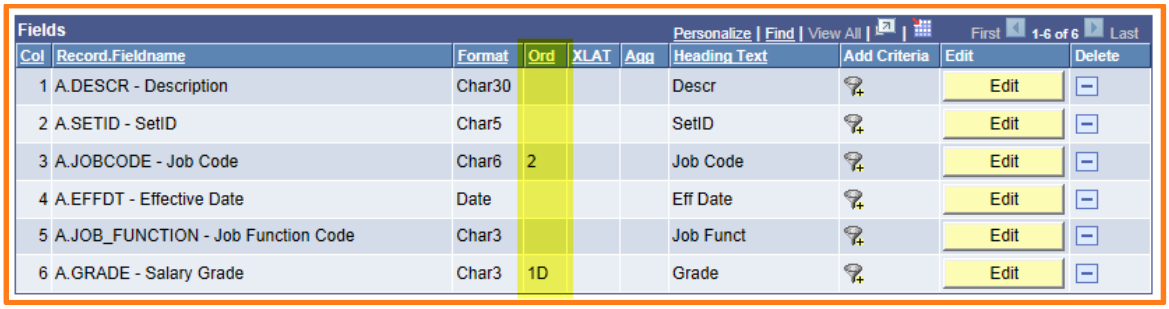
		SetID	Job Code	Eff Date	Job Funct	Grade
1	Assembly Line Worker	CHE01	10-001	01/01/2000		
2	Farmer	CHE01	10-002	01/01/2000		
3	Account Manager	KYSI2	100000	01/01/1980		
4	Account Manager	KYSI1	100000	01/01/1980		
5	Bricklayer	CHE01	11-001	01/01/2000		
6	Architect	CHE01	11-002	01/01/2000		
7	Accountant	HKG01	110000	01/01/1980		004
8	Accountant	MYS01	110000	01/01/1980		005
9	Accountant	NLD01	110000	01/01/2001	FIN	007
10	Accountant	NZL01	110000	01/01/1980		005
11	Accountant	KRSI1	110000	01/01/1980		
12	Accountant	GBR01	110000	01/01/1980		

Now, go back to the Edit Field Ordering screen and sort the results by descending grade first and job code second.

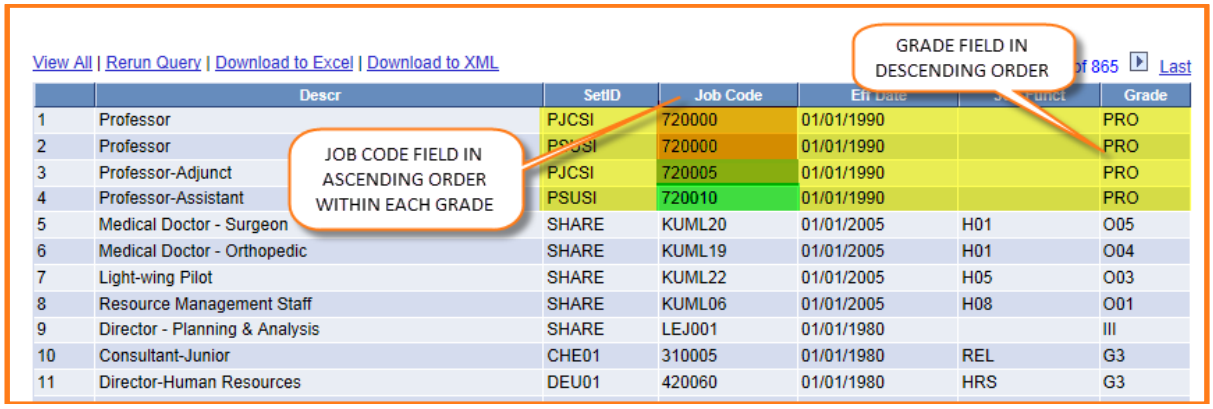
- **Click** the **Fields** tab
- **Click** the **Reorder / Sort** button
- **Sort** the **GRADE** field in *descending order*
  - Place a **"1"** in the **New Order By** column.
  - **Check** the **Descending** box
- **Place** a **2** in the **New Order By** column of the **A.JOBCODE** row
- **Click OK**



Look at the Fields page. Observe the "2" in the A.JOBCODE row and the "1D" in the A.GRADE row.



- **Save** your query
- **Click** the **Run** tab



Results show that the Grade rows are in descending order. The rows ordered by Job Code are in ascending order within each Grade.

It can be easy to confuse the ordering of columns with the ordering of rows. It may help to keep the following guidelines in mind:

Columns:

- How fields appear in the results from *left* to *right*
- Change their position by entering numbers in the New Column column
- Look for key phrases in the query request such as "order the *fields*," "order the *columns*," and "show <some field> *first*." Also look for the order in which fields are named in the request.

Rows:

- How data is ordered from *top* to *bottom*
- Change the ordering by entering numbers in the New Order By column
- Look for key phrases in the query request such as "sort the *data* by," "order the *data*," "sort the *results*, ""display the data *in order by*," "from highest to lowest,", etc. Also look for words as "sort," "order," "ascending," and "descending."

IMPORTANT: if you do not specify any row ordering, the results will be sorted in an arbitrary order! You should always have a specific sort order in your queries so the results will be consistently displayed in the same order. This also helps organize the rows in a meaningful way.

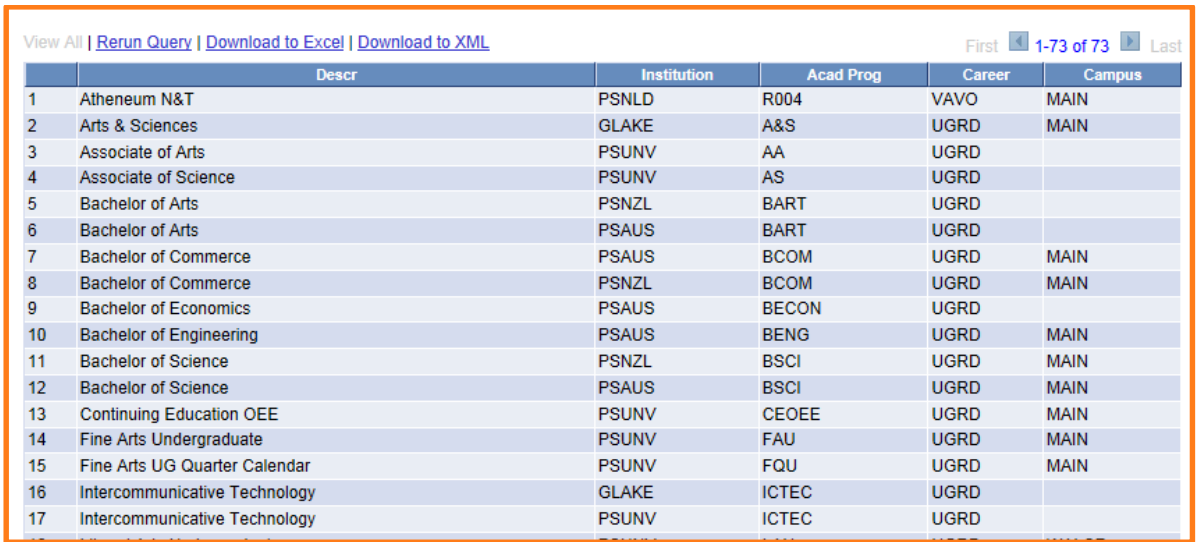
(In this course we won't always set a row ordering – but only so you can concentrate on the topic at hand!)

A Set ID (sometimes written as SetID) identifies configuration data that can be connected to a business unit, institution, or other organization. The "set" is a group of rows out of setup tables that apply to that unit. The set can be used to share common setup data across multiple business units; this may be indicated by having "SHARE" as the SetID. At BGSU notable SetID values include BGSUN (Common SetID for BGSU), BGHCM (BGHCM HR SetID Use Only), BGSUG (BGSU – Grants), and BGSUP (BGSU – Capital). In this training we will focus upon the set PSUNV (Peopleshort University).

Question 3

Create a query that lists the academic programs defined for the University. Show only the academic career, academic program, description, and campus. Display the description column first. Sort the results by career first in descending order (so UGRD comes before GRAD) and academic program second.

- Use the **ACAD_PROG_TBL** record.
- Select the **INSTITUTION**, **ACAD_PROG**, **DESCR**, **ACAD_CAREER**, and **CAMPUS** fields.
- Move the **DESCR** field to be the left-most displayed.
- Sort by **ACAD_CAREER** descending and **ACAD_PROG** ascending.
- Save the query with the name **TRNG_QM##_Q3**.



View All | Rerun Query | Download to Excel | Download to XML First 1-73 of 73 Last

	Descr	Institution	Acad Prog	Career	Campus
1	Atheneum N&T	PSNLD	R004	VAVO	MAIN
2	Arts & Sciences	GLAKE	A&S	UGRD	MAIN
3	Associate of Arts	PSUNV	AA	UGRD	
4	Associate of Science	PSUNV	AS	UGRD	
5	Bachelor of Arts	PSNZL	BART	UGRD	
6	Bachelor of Arts	PSAUS	BART	UGRD	
7	Bachelor of Commerce	PSAUS	BCOM	UGRD	MAIN
8	Bachelor of Commerce	PSNZL	BCOM	UGRD	MAIN
9	Bachelor of Economics	PSAUS	BECON	UGRD	
10	Bachelor of Engineering	PSAUS	BENG	UGRD	MAIN
11	Bachelor of Science	PSNZL	BSCI	UGRD	MAIN
12	Bachelor of Science	PSAUS	BSCI	UGRD	MAIN
13	Continuing Education OEE	PSUNV	CEOEE	UGRD	MAIN
14	Fine Arts Undergraduate	PSUNV	FAU	UGRD	MAIN
15	Fine Arts UG Quarter Calendar	PSUNV	FQU	UGRD	MAIN
16	Intercommunicative Technology	GLAKE	ICTEC	UGRD	
17	Intercommunicative Technology	PSUNV	ICTEC	UGRD	

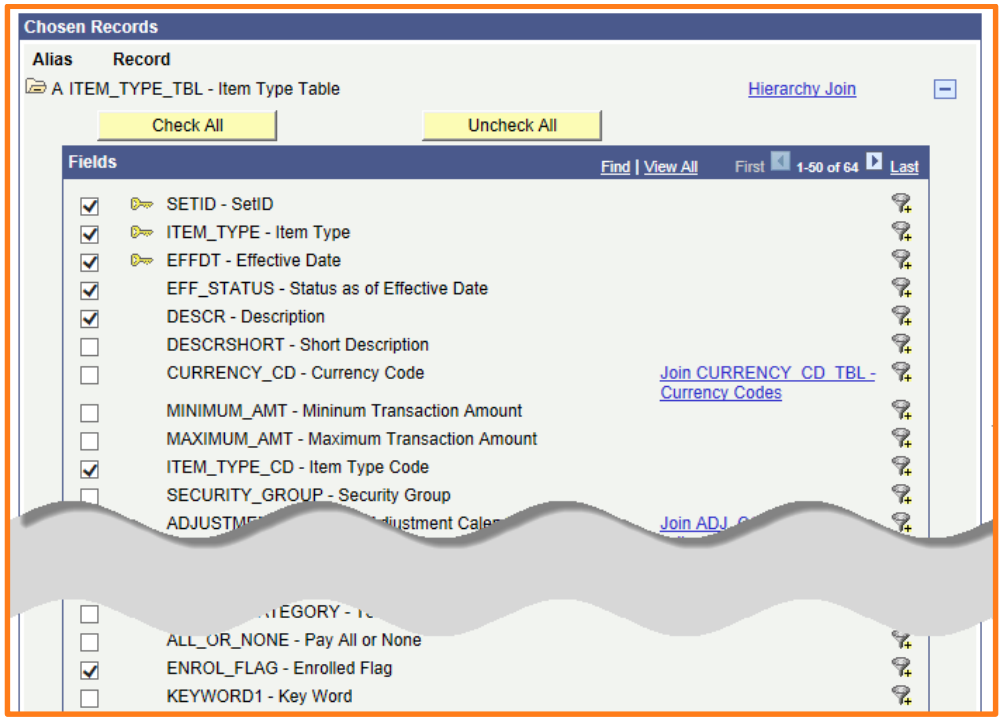
Edit Fields – Heading and Translate Values

The headings of columns can appear in short or long description format. PeopleSoft automatically places the short description on field headings. PeopleSoft does give us the option to place a long description on the field headings. Also, many values are stored and displayed as codes and abbreviations rather than words, such as "A" for "active" and "NWD" for "not withdrawn." Sometimes it is preferable to show words instead of these codes, called *translate values* in PeopleSoft. You can show a short or long description instead of a translate value in the results of a query. The next exercise will demonstrate how to change a heading to a long description and how to translate the value to something that is easily understood.

In the next exercise, you will create a query to list item types. The query will show the set ID, item type, effective date, effective status, description, item type code, and enrolled flag of each item type. You will then change the description field heading and show the meaning of each item type code.

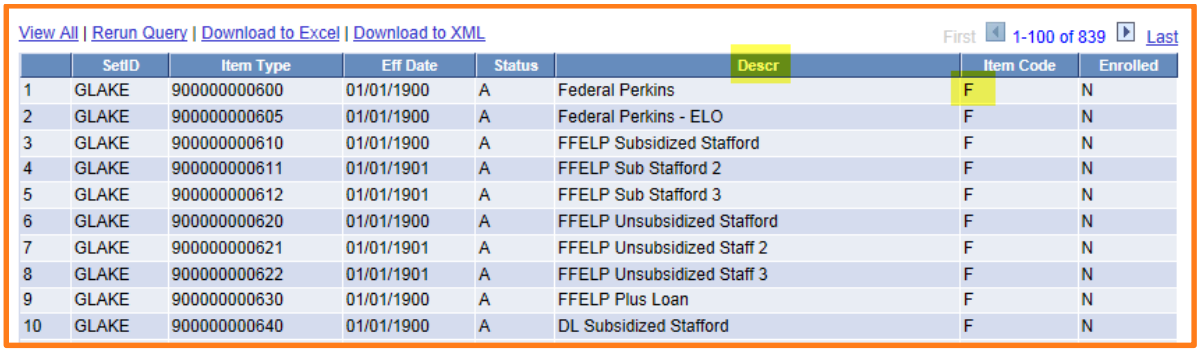
Create a new query on the **ITEM_TYPE_TBL** record. Show the following fields:

- **SETID**
- **ITEM_TYPE**
- **EFFDT**
- **EFF_STATUS**
- **DESCR**
- **ITEM_TYPE_CD**
- **ENROL_FLG**



- **Click** the **Run** tab

Observe the heading "Descr". Observe the Item Code "F".

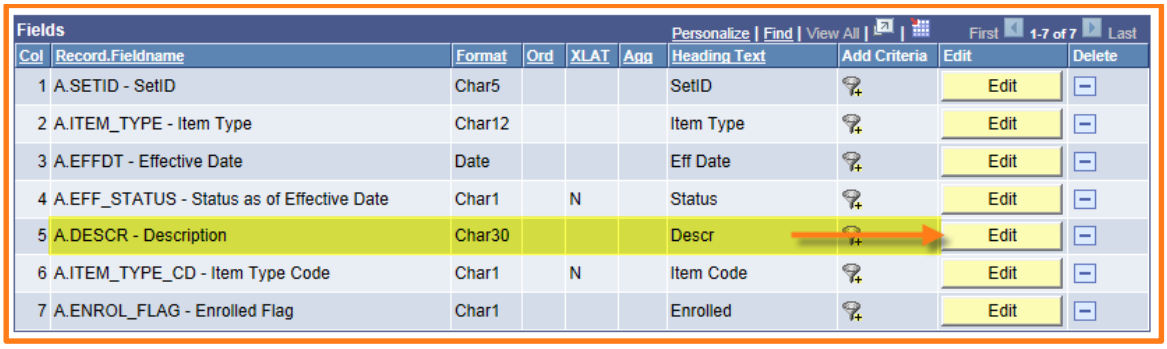


View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 839 Last

	SetID	Item Type	Eff Date	Status	Descr	Item Code	Enrolled
1	GLAKE	900000000600	01/01/1900	A	Federal Perkins	F	N
2	GLAKE	900000000605	01/01/1900	A	Federal Perkins - ELO	F	N
3	GLAKE	900000000610	01/01/1900	A	FFELP Subsidized Stafford	F	N
4	GLAKE	900000000611	01/01/1901	A	FFELP Sub Stafford 2	F	N
5	GLAKE	900000000612	01/01/1901	A	FFELP Sub Stafford 3	F	N
6	GLAKE	900000000620	01/01/1900	A	FFELP Unsubsidized Stafford	F	N
7	GLAKE	900000000621	01/01/1901	A	FFELP Unsubsidized Staff 2	F	N
8	GLAKE	900000000622	01/01/1901	A	FFELP Unsubsidized Staff 3	F	N
9	GLAKE	900000000630	01/01/1900	A	FFELP Plus Loan	F	N
10	GLAKE	900000000640	01/01/1900	A	DL Subsidized Stafford	F	N

- **Click** the **Fields** tab
- Save and save often. Save as **TRNG_QM##_E4**

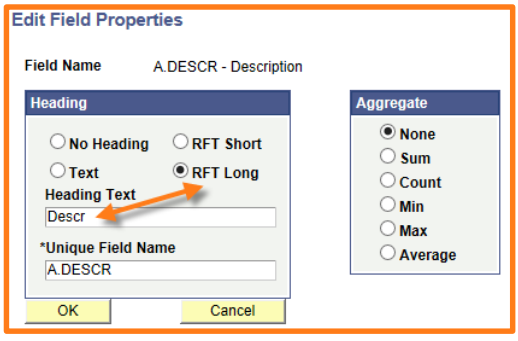
Click the **Edit** button in the DESCR row.



Fields — Personalize | Find | View All | First 1-7 of 7 Last

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.SETID - SetID	Char5				SetID		Edit	
2	A.ITEM_TYPE - Item Type	Char12				Item Type		Edit	
3	A.EFFDT - Effective Date	Date				Eff Date		Edit	
4	A.EFF_STATUS - Status as of Effective Date	Char1		N		Status		Edit	
5	A.DESCR - Description	Char30				Descr		Edit	
6	A.ITEM_TYPE_CD - Item Type Code	Char1		N		Item Code		Edit	
7	A.ENROL_FLAG - Enrolled Flag	Char1				Enrolled		Edit	

The Edit Field Properties page appears. The Field Name category shows the name of the field and the short and long description for the field. Four options appear in the Heading section. None, Text (enter any text), RFT Short (defined short name), and RFT Long (defined long name). Click on RFT Short or RFT Long to use the predefined short or long heading text; click on Text to enter your own heading text.



Edit Field Properties

Field Name A.DESCR - Description

Heading
- No Heading
- RFT Short
- Text
- RFT Long

Heading Text: Descr

*Unique Field Name: A.DESCR

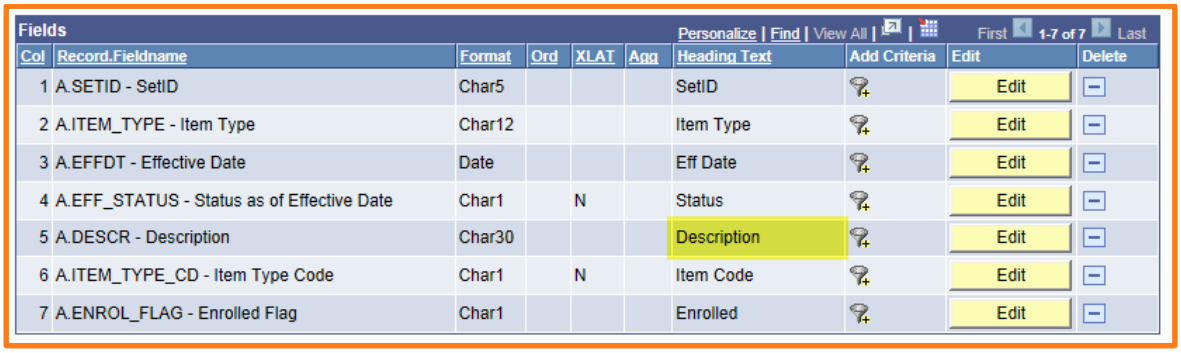
OK Cancel

Aggregate
- None
- Sum
- Count
- Min
- Max
- Average

If the Heading is switched from RFT Short to Text, the change in the Heading Text will not be shown until you click OK. This occurs if the Heading value is changed, no matter what its original value and new value are.

Click RFT Long to display a long description for the field heading.

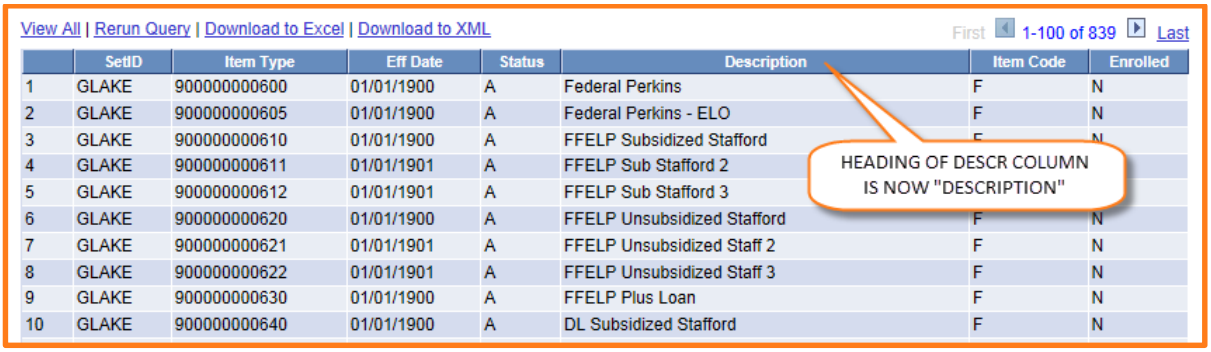
- **Click OK**. The Heading Text for DESCR is now "Description" instead of "Descr".



Fields — Personalize | Find | View All | First 1-7 of 7 Last

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.SETID - SetID	Char5				SetID		Edit	
2	A.ITEM_TYPE - Item Type	Char12				Item Type		Edit	
3	A.EFFDT - Effective Date	Date				Eff Date		Edit	
4	A.EFF_STATUS - Status as of Effective Date	Char1		N		Status		Edit	
5	A.DESCR - Description	Char30				Description		Edit	
6	A.ITEM_TYPE_CD - Item Type Code	Char1		N		Item Code		Edit	
7	A.ENROL_FLAG - Enrolled Flag	Char1				Enrolled		Edit	

- **Save** your query
- **Click** the **Run** tab



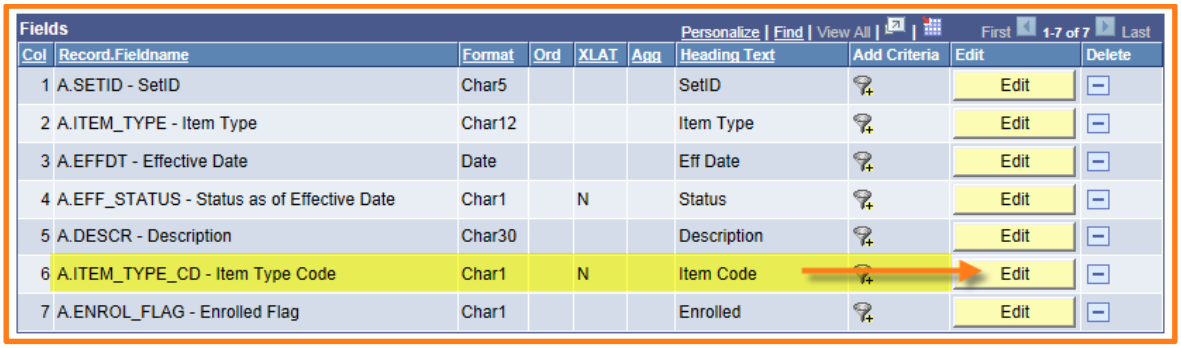
View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 839 Last

	SetID	Item Type	Eff Date	Status	Description	Item Code	Enrolled
1	GLAKE	900000000600	01/01/1900	A	Federal Perkins	F	N
2	GLAKE	900000000605	01/01/1900	A	Federal Perkins - ELO	F	N
3	GLAKE	900000000610	01/01/1900	A	FFELP Subsidized Stafford	F	N
4	GLAKE	900000000611	01/01/1901	A	FFELP Sub Stafford 2	F	N
5	GLAKE	900000000612	01/01/1901	A	FFELP Sub Stafford 3	F	N
6	GLAKE	900000000620	01/01/1900	A	FFELP Unsubsidized Stafford	F	N
7	GLAKE	900000000621	01/01/1901	A	FFELP Unsubsidized Staff 2	F	N
8	GLAKE	900000000622	01/01/1901	A	FFELP Unsubsidized Staff 3	F	N
9	GLAKE	900000000630	01/01/1900	A	FFELP Plus Loan	F	N
10	GLAKE	900000000640	01/01/1900	A	DL Subsidized Stafford	F	N

The Descr field heading now appears as Description.

Earlier, Item Code values were pointed out. Take the opportunity to scroll through the results to see if other values appear. (Use the **View All** link to display all of the rows to see more item codes.) Item Code values A, B, C, D, F, H, L, P, R, T, V, W, X, and Z appear in the results. What do they mean?

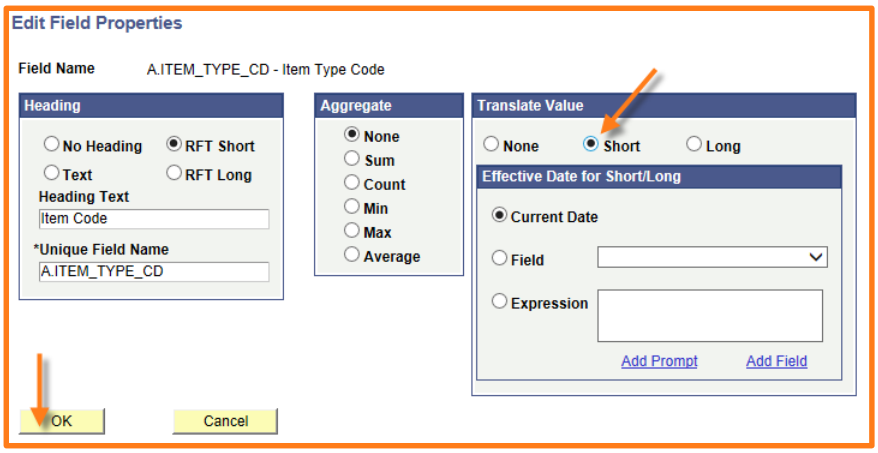
- **Click** the **Fields** tab
- Click **Edit** in the **ITEM_TYPE_CD** row



Fields — Personalize | Find | View All | First 1-7 of 7 Last

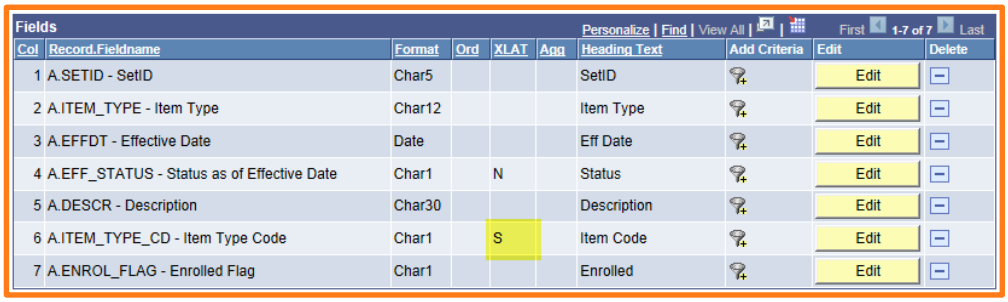
Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.SETID - SetID	Char5				SetID		Edit	
2	A.ITEM_TYPE - Item Type	Char12				Item Type		Edit	
3	A.EFFDT - Effective Date	Date				Eff Date		Edit	
4	A.EFF_STATUS - Status as of Effective Date	Char1		N		Status		Edit	
5	A.DESCR - Description	Char30				Description		Edit	
6	A.ITEM_TYPE_CD - Item Type Code	Char1		N		Item Code		Edit	
7	A.ENROL_FLAG - Enrolled Flag	Char1				Enrolled		Edit	

There is a set of meanings defined for the possible values of ITEM_TYPE_CD. Query Manager is aware of this and provides the option to show the meaning in place of the code in the results. Notice the Edit Fields Properties page has changed. A Translate Value section appears. The Translate Value section has three options: None (currently selected), Short, and Long.



- **Click Short**
- **Click OK**

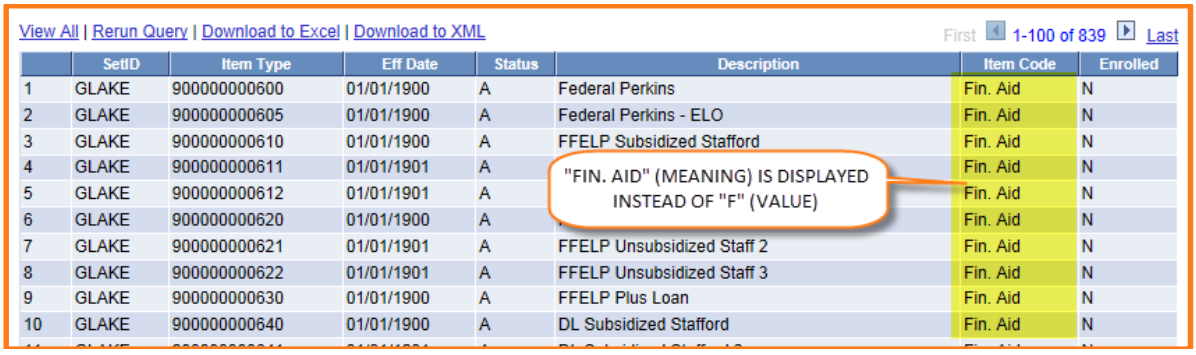
You are returned to the **Fields** tab. The XLAT column in the ITEM_TYPE_CD row now shows "S" to indicate the short translate value will be displayed in the results.



Fields — Personalize | Find | View All | First 1-7 of 7 Last

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.SETID - SetID	Char5				SetID		Edit	
2	A.ITEM_TYPE - Item Type	Char12				Item Type		Edit	
3	A.EFFDT - Effective Date	Date				Eff Date		Edit	
4	A.EFF_STATUS - Status as of Effective Date	Char1		N		Status		Edit	
5	A.DESCR - Description	Char30				Description		Edit	
6	A.ITEM_TYPE_CD - Item Type Code	Char1		S		Item Code		Edit	
7	A.ENROL_FLAG - Enrolled Flag	Char1				Enrolled		Edit	

- **Save** your query
- **Click** the **Run** tab



View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 839 Last

	SetID	Item Type	Eff Date	Status	Description	Item Code	Enrolled
1	GLAKE	900000000600	01/01/1900	A	Federal Perkins	Fin. Aid	N
2	GLAKE	900000000605	01/01/1900	A	Federal Perkins - ELO	Fin. Aid	N
3	GLAKE	900000000610	01/01/1900	A	FFELP Subsidized Stafford	Fin. Aid	N
4	GLAKE	900000000611	01/01/1901	A		Fin. Aid	N
5	GLAKE	900000000612	01/01/1901	A		Fin. Aid	N
6	GLAKE	900000000620	01/01/1900	A		Fin. Aid	N
7	GLAKE	900000000621	01/01/1901	A	FFELP Unsubsidized Staff 2	Fin. Aid	N
8	GLAKE	900000000622	01/01/1901	A	FFELP Unsubsidized Staff 3	Fin. Aid	N
9	GLAKE	900000000630	01/01/1900	A	FFELP Plus Loan	Fin. Aid	N
10	GLAKE	900000000640	01/01/1900	A	DL Subsidized Stafford	Fin. Aid	N

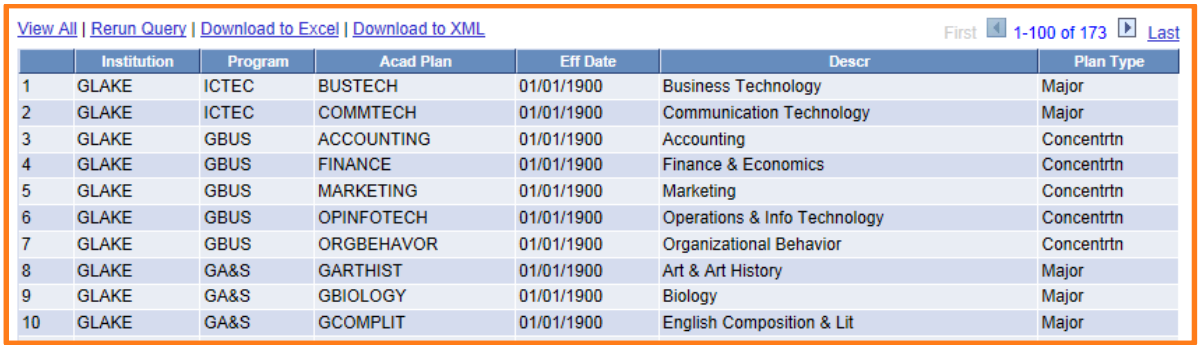
"FIN. AID" (MEANING) IS DISPLAYED INSTEAD OF "F" (VALUE)

Question 4

Create a new query on academic plans. Show the following fields: institution, academic plan, effective date, description, academic plan type, and academic program. Make the academic program the left-most column and order the results by academic programs in descending order followed by the academic plan in ascending order. Change the heading of academic programs to "Program" and show the short translation of academic plan type.

- Use the record **ACAD_PLAN_TBL**
- Show the following fields:
  - **INSTITUTION**
  - **ACAD_PLAN**
  - **EFFDT**
  - **DESCR**
  - **ACAD_PLAN_TYPE**
  - **ACAD_PROG**
- Move **ACAD_PROG** to be the second column
- Sort by **INSTITUTION** in ascending order, **ACAD_PROG** in descending order, and **ACAD_PLAN** in ascending order
- Change the heading of **ACAD_PROG** to "Program"
- Change the translate value type of **ACAD_PLAN_TYPE** to Short

Save and Save often. Save as **TRNG_QM###_Q4**



View All | Rerun Query | Download to Excel | Download to XML First 1-100 of 173 Last

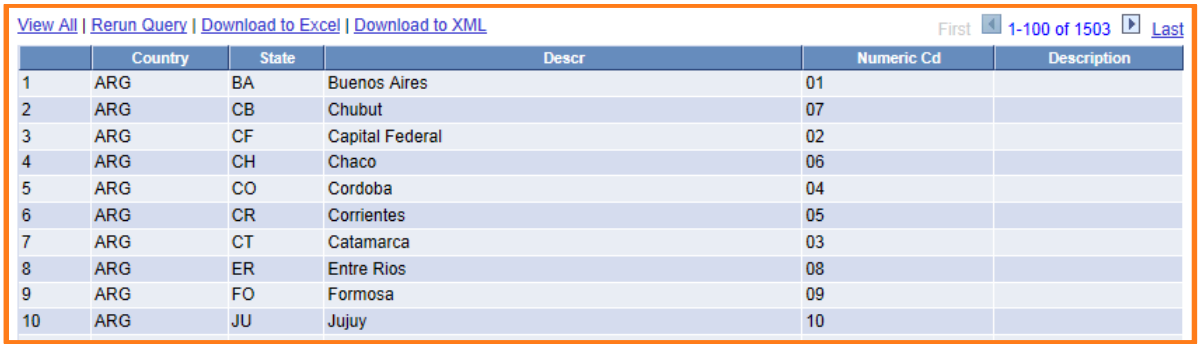
	Institution	Program	Acad Plan	Eff Date	Descr	Plan Type
1	GLAKE	ICTEC	BUSTECH	01/01/1900	Business Technology	Major
2	GLAKE	ICTEC	COMMTECH	01/01/1900	Communication Technology	Major
3	GLAKE	GBUS	ACCOUNTING	01/01/1900	Accounting	Concentrtn
4	GLAKE	GBUS	FINANCE	01/01/1900	Finance & Economics	Concentrtn
5	GLAKE	GBUS	MARKETING	01/01/1900	Marketing	Concentrtn
6	GLAKE	GBUS	OPINFOTECH	01/01/1900	Operations & Info Technology	Concentrtn
7	GLAKE	GBUS	ORGBEHAVOR	01/01/1900	Organizational Behavior	Concentrtn
8	GLAKE	GA&S	GARTHIST	01/01/1900	Art & Art History	Major
9	GLAKE	GA&S	GBIOLOGY	01/01/1900	Biology	Major
10	GLAKE	GA&S	GCOMPLIT	01/01/1900	English Composition & Lit	Major

Edit Criteria – Part I

Maybe you have to write a lot of letters to residents of Canada and you need a list of the Canadian provinces and their abbreviations. When you have much more data than you need or want you must add specific criteria to the query to retrieve only the desired data. Selection criteria apply to all data examined by the query and the system will only retrieve the rows with the specified criteria.

Create a query to show all state codes and names. Limit the results to Canadian provinces.

- Start a new query on **STATE_TBL** (state codes and names)
- **Show** all fields
- **Click Run** tab

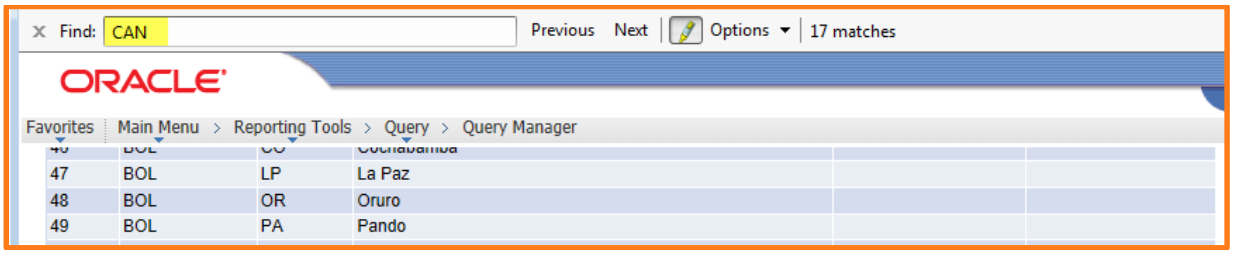

View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 1503 Last

	Country	State	Descr	Numeric Cd	Description
1	ARG	BA	Buenos Aires	01	
2	ARG	CB	Chubut	07	
3	ARG	CF	Capital Federal	02	
4	ARG	CH	Chaco	06	
5	ARG	CO	Cordoba	04	
6	ARG	CR	Corrientes	05	
7	ARG	CT	Catamarca	03	
8	ARG	ER	Entre Rios	08	
9	ARG	FO	Formosa	09	
10	ARG	JU	Jujuy	10	

States and provinces in PeopleSoft belong to countries; the country the state or province belongs to is in the COUNTRY field, which is shown in the Country column of the results.

We want to find the Canadian provinces. Use your browser's Find feature to locate all provinces that have a country of "CAN", the value representing Canada.

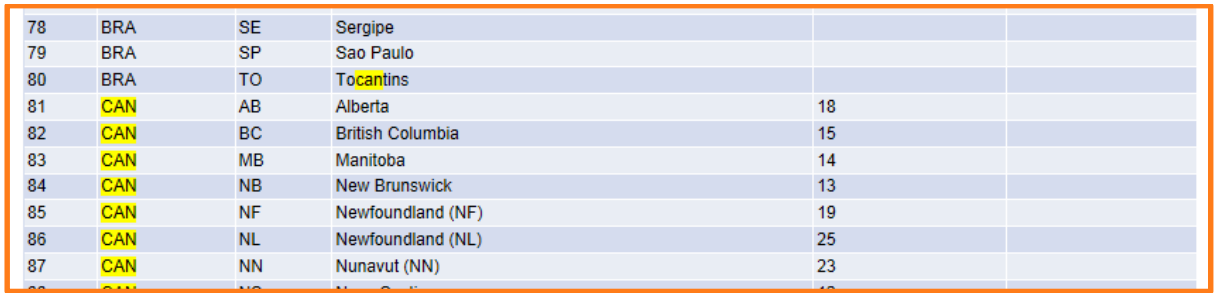
- **Click View All** link to show all results
- Use the browser **Find** feature and search for "CAN"


Find: CAN — Previous Next — Options — 17 matches

ORACLE

Favorites Main Menu > Reporting Tools > Query > Query Manager

47	BOL	LP	La Paz
48	BOL	OR	Oruro
49	BOL	PA	Pando



78	BRA	SE	Sergipe		
79	BRA	SP	Sao Paulo		
80	BRA	TO	Tocantins		
81	CAN	AB	Alberta	18	
82	CAN	BC	British Columbia	15	
83	CAN	MB	Manitoba	14	
84	CAN	NB	New Brunswick	13	
85	CAN	NF	Newfoundland (NF)	19	
86	CAN	NL	Newfoundland (NL)	25	
87	CAN	NN	Nunavut (NN)	23	

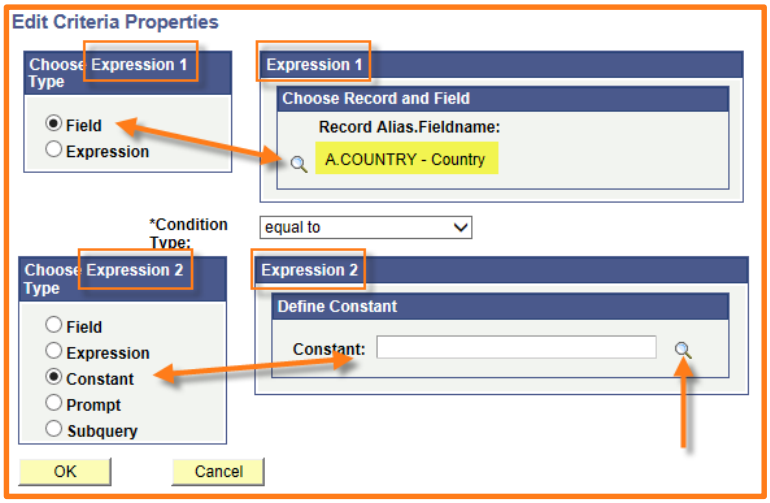
- Save and Save often. Save as **TRNG_QM##_E5**

It would be easier to restrict the query results to just the Canadian provinces. To do this we must add criteria to the query. *Criteria* are conditions added to a query that restrict the rows that are included in the results. Criteria are almost always a comparison between two *expressions*, which are sets of symbols, field names, and values that produce results. An *expression* can be a single field or values or can be a complex calculation. Most criteria will involve comparing the value in a field to a value in another field or to a constant, a number, work, or phrase that you enter. Comparisons between expressions usually check whether two expressions are equal or if one is greater than another.

- **Click Fields** tab

We have used the Reorder / Sort button and the Edit button; the only one left is the Add Criteria funnel . The Edit Criteria Properties page appears.

- **Click** in the COUNTRY row



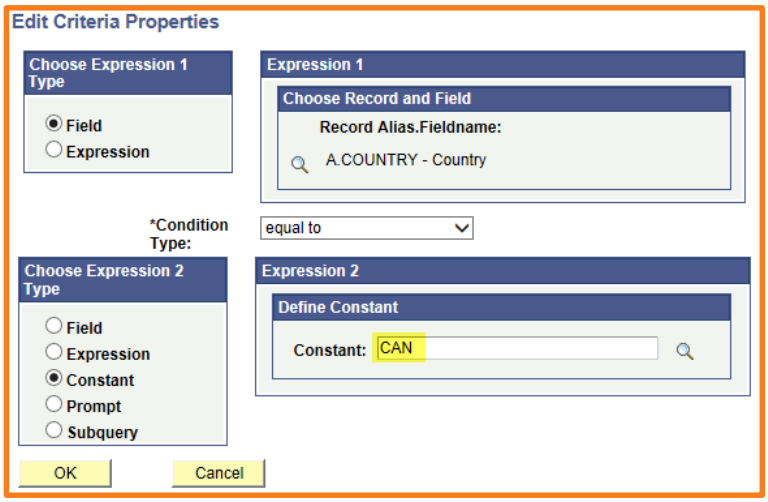
In the Edit Criteria Properties page, you must identify two expressions, Expression 1 and Expression 2. For each expression you must choose its type and define the expression. For Expression 1, you can set Expression 1 type to either Field to use a field from a record or Expression to use a new or already defined expression. Most of the time, you will select Field for the type. For Expression 2, you have three more Type options: Constant (enter a value), Prompt, and Subquery. The prompt and subquery options are discussed later in this course.

To obtain a list of Canadian provinces and their abbreviations, we started by building a query that listed all states and provinces known to the system. Now we are adding a criterion that COUNTRY field must have a value of "CAN," the country code representing Canada.

The first part of the criterion will refer to the COUNTRY field. **Set** the **Expression 1 Type** option to **Field**. **Select A.COUNTRY** for the Record Alias.Fieldname option; use the magnifying glass to look up records and fields available to the query. (If you click the Add Criteria funnel button while on the Query page, the Record Alias.Fieldname option is automatically filled in.)

We want only rows in which COUNTRY is equal to "CAN". **Select "equal to"** for the **Condition Type** option.

Now complete the second part of the criterion. We are looking for a specific value, so **set** the **Expression 2 Type** to **Constant**. In the Constant field, either **enter** CAN or use the magnifying glass to summon a list from which you can select it.



- **Click** the 🔍 in the Expression 2 Define Constant box. The system will prompt for a Country abbreviation.

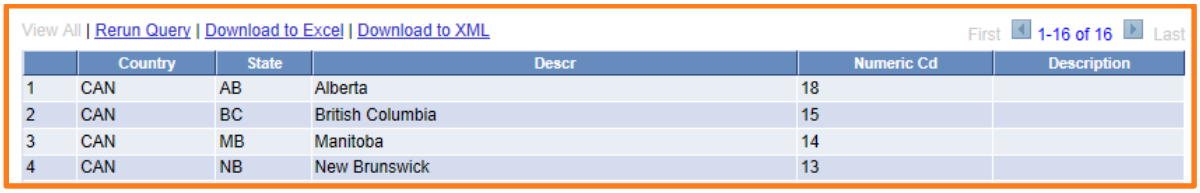
The system takes you to a "prompt." The system is prompting for a Country to identify as a single fixed value. We want our results to show the Canadian provinces. Use the 🔍 to look up Country abbreviations.

- **Place** Country abbreviation "CAN" in Constant box
- **Click OK**

The system takes you back to the Edit Criteria Properties page.

- **Click OK** to return to the Fields tab

- **Save** your query
- **Click Run** tab



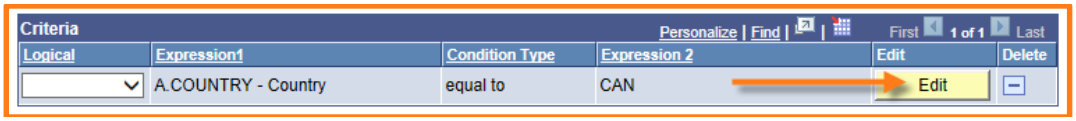
	Country	State	Descr	Numeric Cd	Description
1	CAN	AB	Alberta	18	
2	CAN	BC	British Columbia	15	
3	CAN	MB	Manitoba	14	
4	CAN	NB	New Brunswick	13	

We did it! The results show all Canadian provinces.

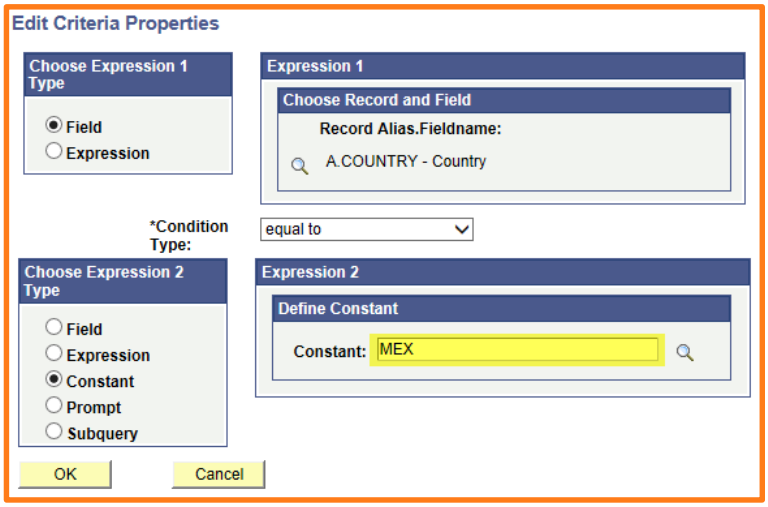
Now let's change this to retrieve states of Mexico instead of provinces of Canada.

- **Go** to the **Criteria** tab
- **Click Edit**

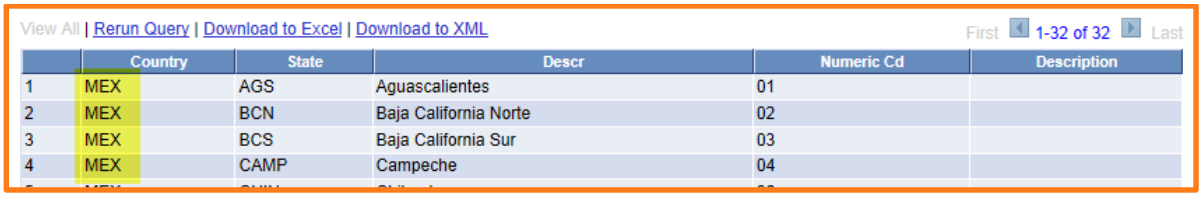
The same criteria information will be used for Mexico as was used for Canada.



- **Modify** criteria – change **CAN** to **MEX**
- **Click OK**



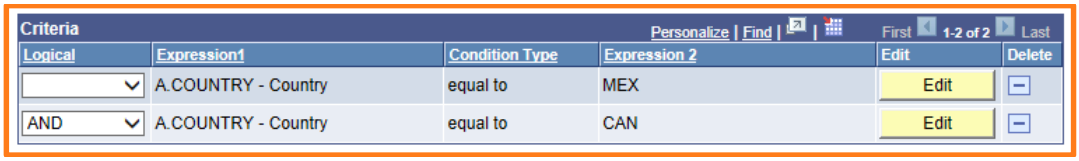
- **Save** the query
- Click the **Run** tab



	Country	State	Descr	Numeric Cd	Description
1	MEX	AGS	Aguascalientes	01	
2	MEX	BCN	Baja California Norte	02	
3	MEX	BCS	Baja California Sur	03	
4	MEX	CAMP	Campeche	04	

Next, let's list the states and provinces in both Mexico and Canada. Since we changed the criterion that COUNTRY is equal to "CAN" to be that COUNTRY is equal to "MEX", we'll add a new criterion that COUNTRY is equal to "CAN" to replace the criterion that was changed.

- **Click Fields** tab
- **Click**  in the **COUNTRY** row
- **Add CAN** as the COUNTRY, again
- **Click OK**
- **Click** the **Criteria** tab



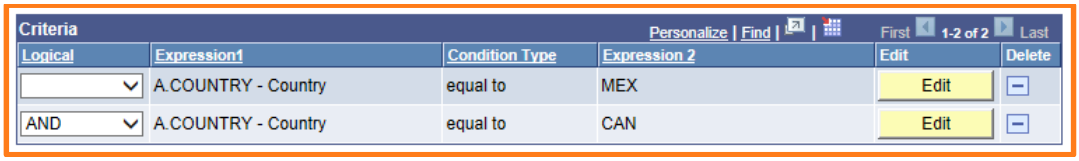
Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.COUNTRY - Country	equal to	MEX	Edit	
AND	A.COUNTRY - Country	equal to	CAN	Edit	

Click the **Run** tab.

	Country	State	Descr	Numeric Cd	Description
1					

There are NO matching rows!

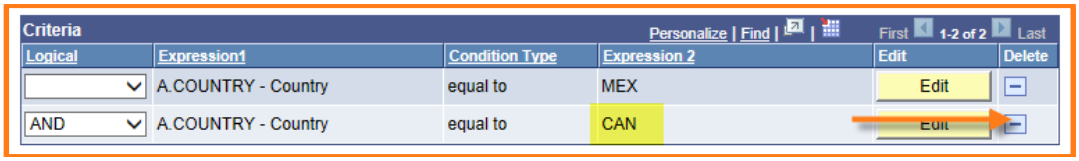
- **Go back** to the **Criteria** tab



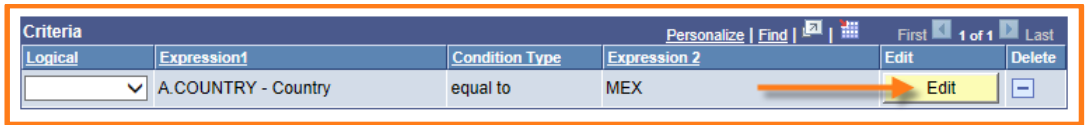
The Criteria as defined are that A.COUNTRY is *equal to* MEX *and* A.COUNTRY is *equal to* CAN. A state can't be in Canada and Mexico at the same time, so it is impossible to meet both criteria!

What we need instead is a criterion that checks whether A.COUNTRY is one out of several possible values.

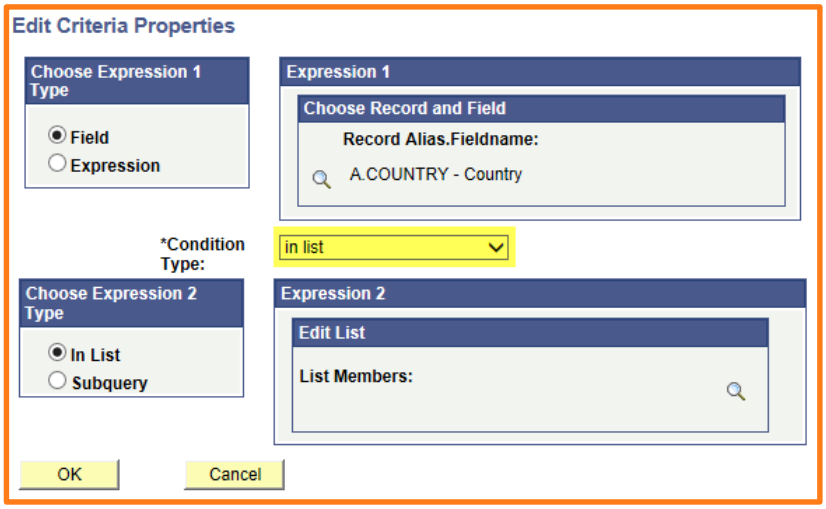
Delete the criterion just added by clicking the minus button in the row for CAN.



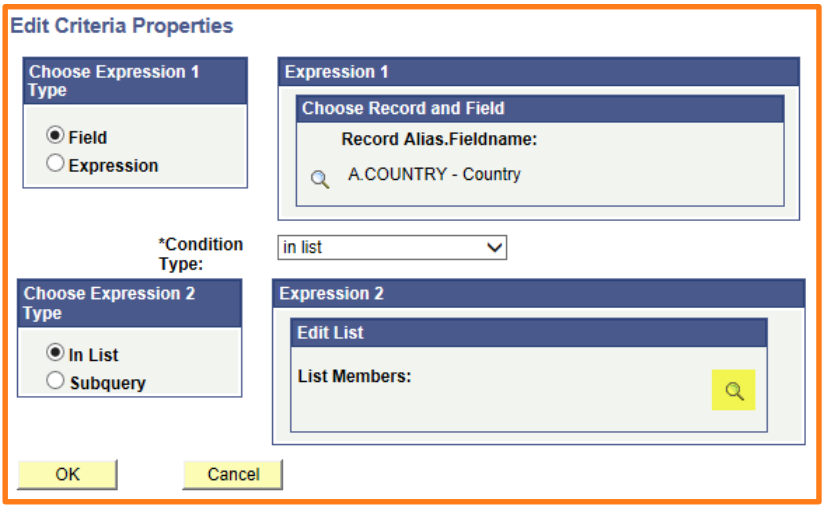
Next, change the remaining criterion by clicking the Edit button in its row.



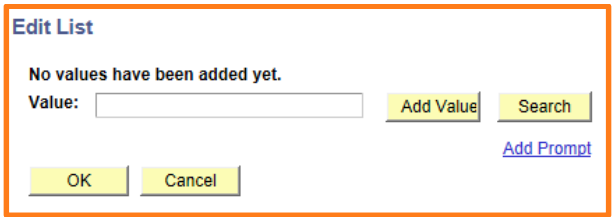
Open the Condition Type drop-down list. Several options are presented, which will be covered in detail in the next section. For now, select "in list".



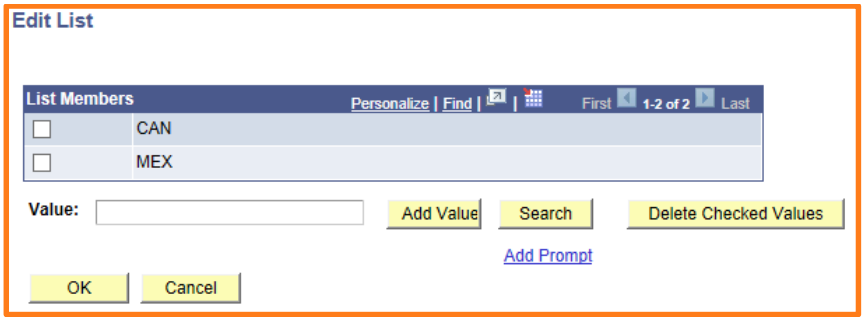
We now need to tell Query Manager which possible values to check whether the value in COUNTRY matches. Click the magnifying glass in the Edit List section.



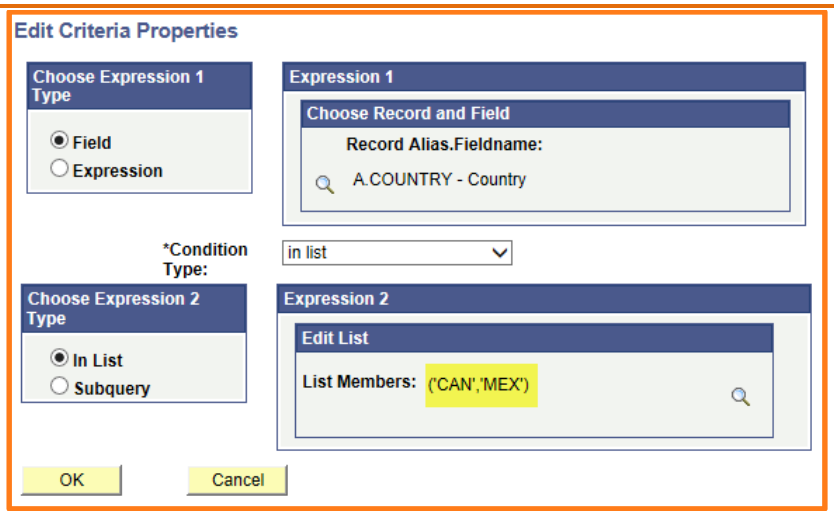
The Edit List dialog appears.



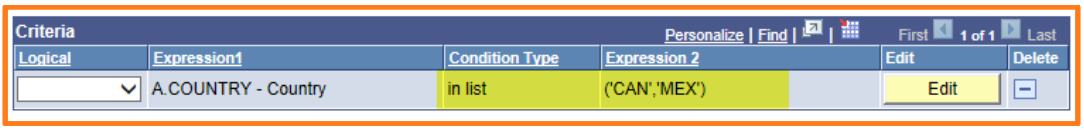
Enter **CAN** in the Value field then click the **Add Value** button. Next enter **MEX** in the Value field then click the **Add Value** button.



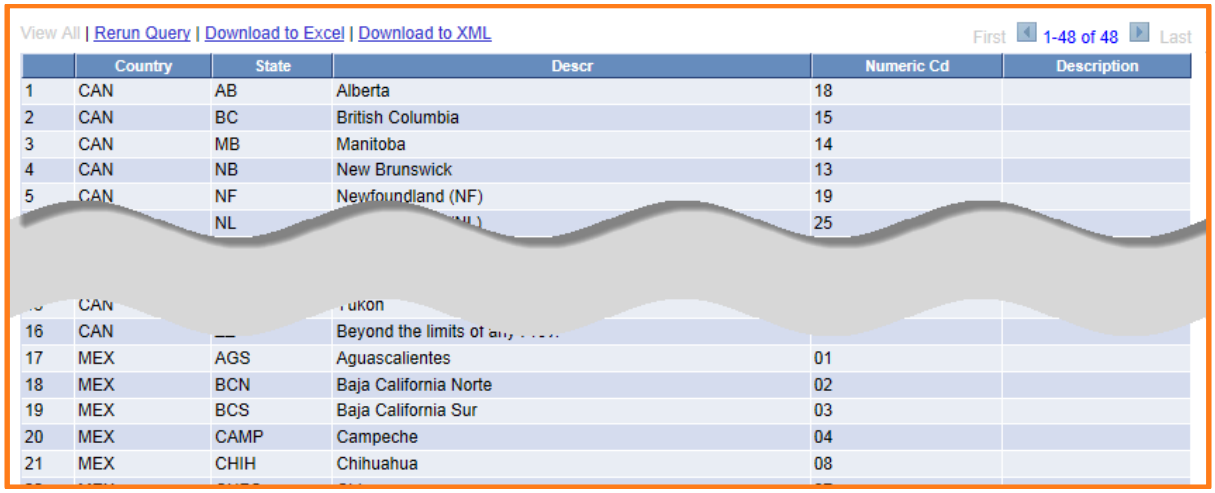
Click the **OK** button. The list members are displayed as a comma-separated list in parentheses.



Click the **OK** button. The criterion is now shown with a condition type of "in list" and Expression 2 having the comma-separated list of the values just entered.



Save the query again then click the **Run** tab.

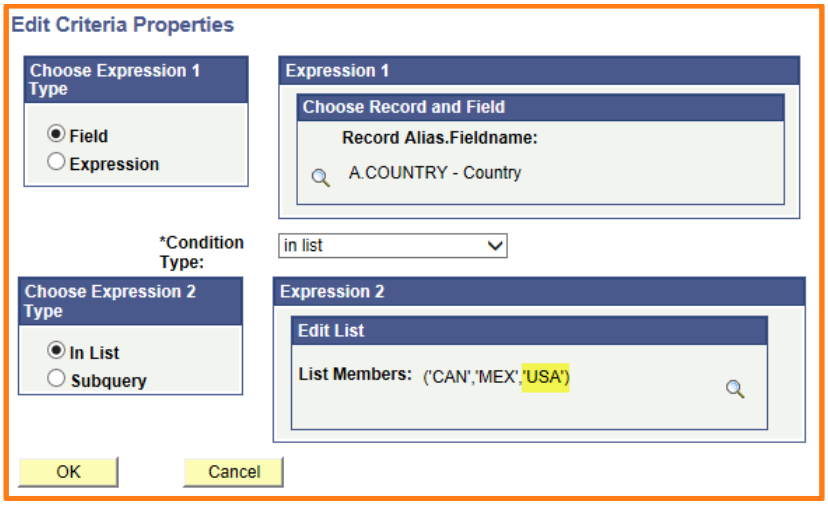


Now the results include both Canadian provinces and Mexican states!

Let's modify the query one more time to retrieve all provinces and states in North America by adding the United States to the list of allowable countries.

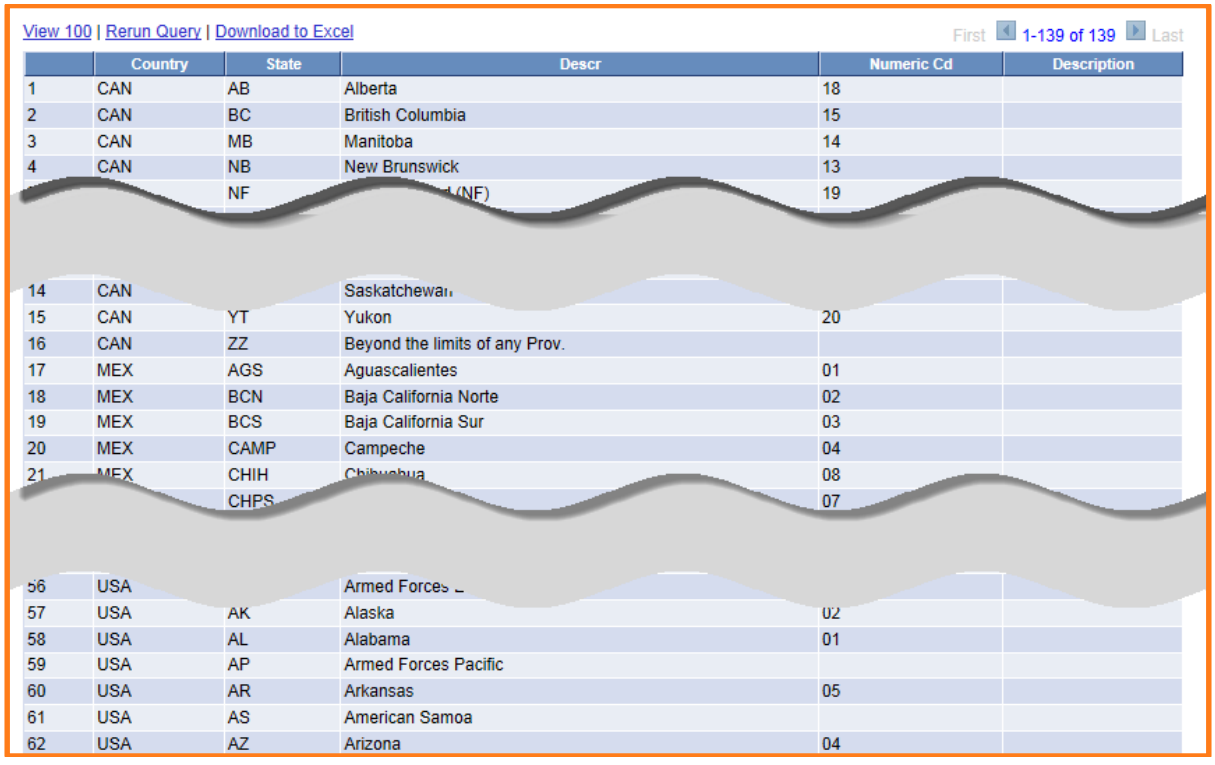
- **Click** the **Criteria** tab
- **Click** the **Edit** button in the row for A.COUNTRY
- **Click** the magnifying glass for List Members
- **Add** the value **USA**

- **Click** the **OK** button to return to Edit Criteria Properties



- **Click** the **OK** button to return to the **Criteria** page

Save the query again then click the **Run** tab.

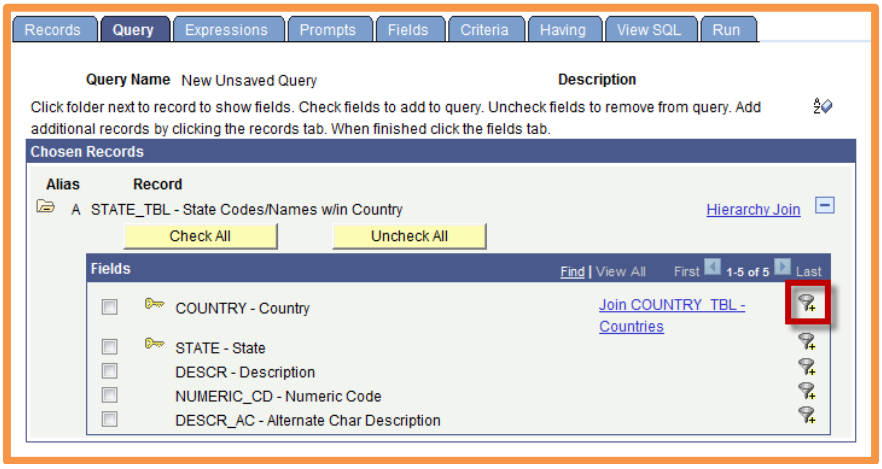


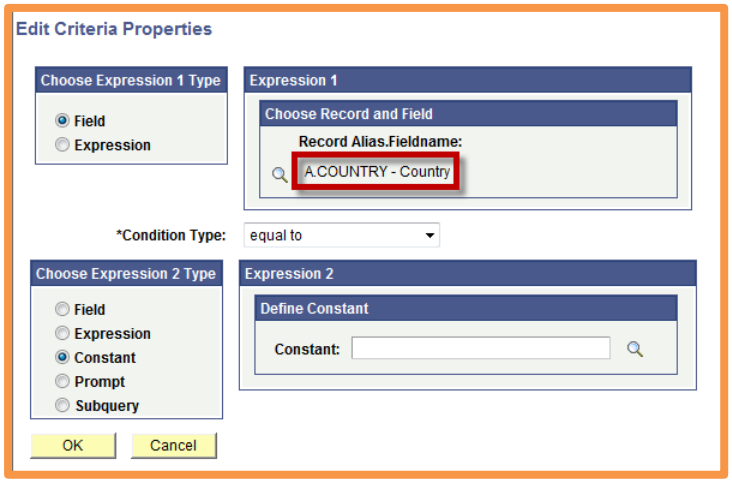
Now all states and provinces in North America are retrieved by the query.

Note that Query Manager uses the term “criteria” as both a singular and a plural, though the term “criterion” is more appropriate as a singular.

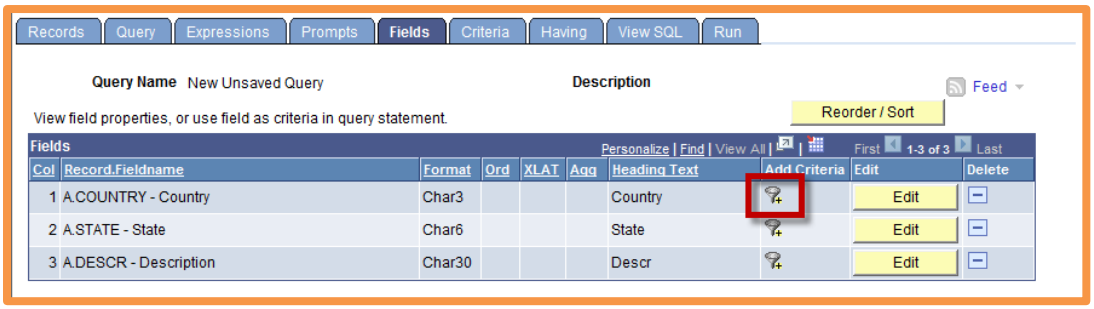
There are several different ways you can add criteria when building a query.

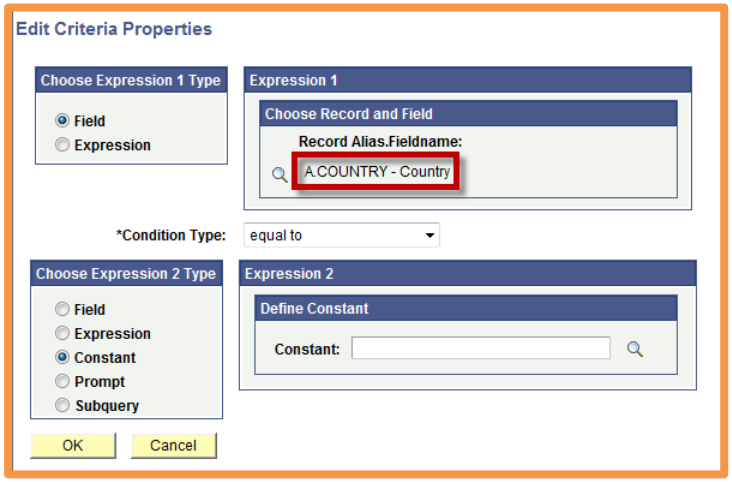
1. On the Query tab, click the Use as Criteria button. This takes you to the Edit Criteria Properties page; the field in the same row as the button is automatically selected as Expression 1.



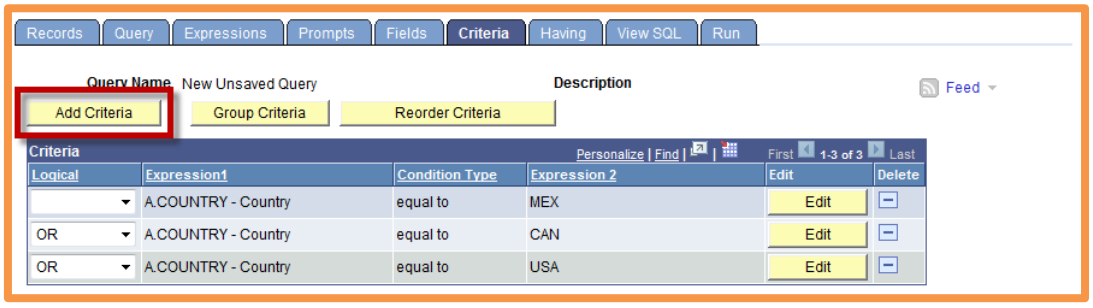


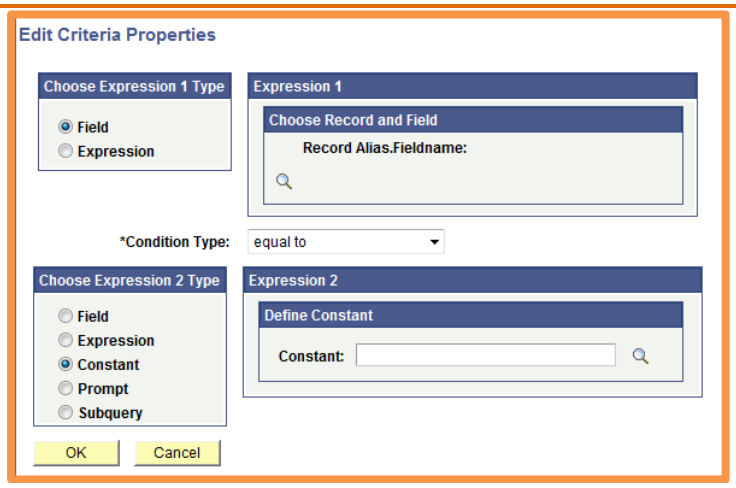
2. On the Fields tab, click the Add Criteria button. This takes you to the Edit Criteria Properties page; the field in the same row as the button is automatically selected as Expression 1.



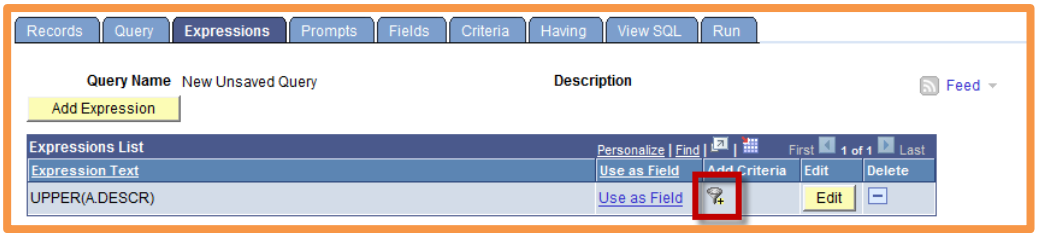


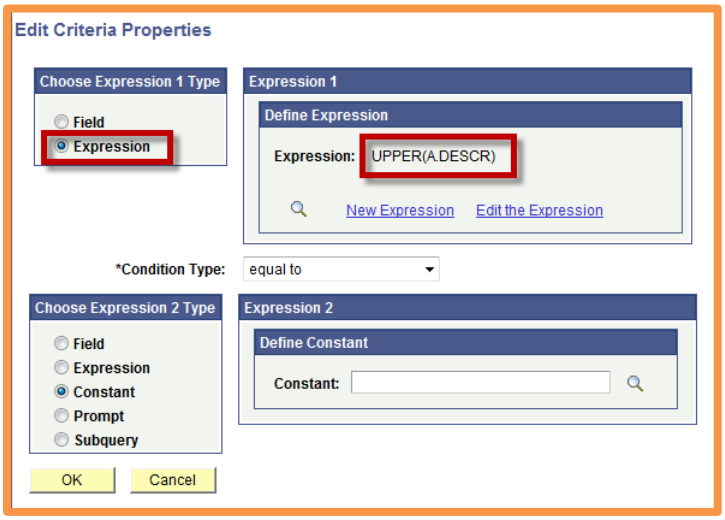
3. On the Criteria tab, click the Add Criteria button.





4. On the Expressions tab, click the Add Criteria button. This takes you to the Edit Criteria Properties page; Expression 1 Type is set to Expression instead of Field and the expression in the same row as the button is automatically selected as Expression 1. (Expressions are covered later in this class.)

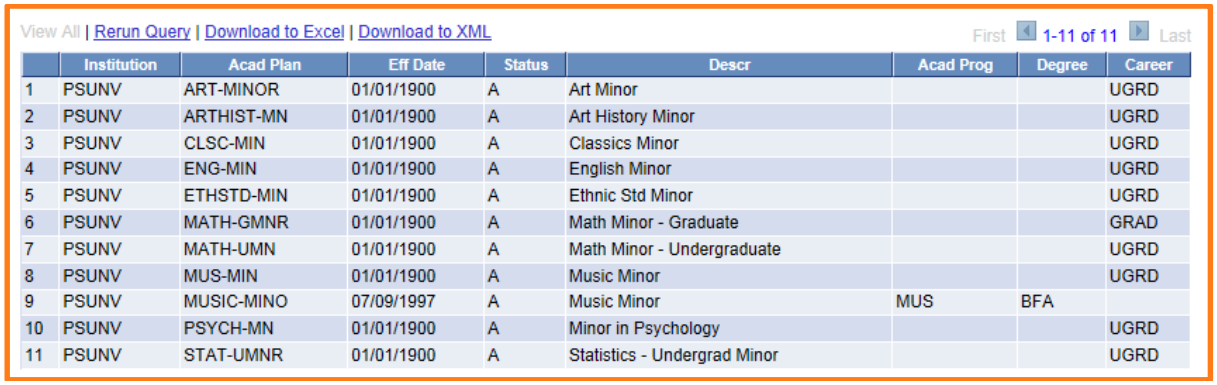




Question 5A

Create a new query to list minors offered at PeopleSoft University. For each plan, show the institution, academic plan, effective date, effective status, description, academic program, degree, and academic career.

- Use the **ACAD_PLAN_TBL** record
- Show the following fields:
  - **INSTITUTION**
  - **ACAD_PLAN**
  - **EFFDT**
  - **EFF_STATUS**
  - **DESCR**
  - **ACAD_PROG**
  - **DEGREE**
  - **ACAD_CAREER**
- Add two criteria:
  - The institution is **PSUNV**
  - The plan type is Minor (how will you find the code that corresponds to Minor?)
- Save and Save often. Save as **TRNG_QM##_Q5A**



View All | Rerun Query | Download to Excel | Download to XML — First 1-11 of 11 Last

	Institution	Acad Plan	Eff Date	Status	Descr	Acad Prog	Degree	Career
1	PSUNV	ART-MINOR	01/01/1900	A	Art Minor			UGRD
2	PSUNV	ARTHIST-MN	01/01/1900	A	Art History Minor			UGRD
3	PSUNV	CLSC-MIN	01/01/1900	A	Classics Minor			UGRD
4	PSUNV	ENG-MIN	01/01/1900	A	English Minor			UGRD
5	PSUNV	ETHSTD-MIN	01/01/1900	A	Ethnic Std Minor			UGRD
6	PSUNV	MATH-GMNR	01/01/1900	A	Math Minor - Graduate			GRAD
7	PSUNV	MATH-UMN	01/01/1900	A	Math Minor - Undergraduate			UGRD
8	PSUNV	MUS-MIN	01/01/1900	A	Music Minor			UGRD
9	PSUNV	MUSIC-MINO	07/09/1997	A	Music Minor	MUS	BFA	
10	PSUNV	PSYCH-MN	01/01/1900	A	Minor in Psychology			UGRD
11	PSUNV	STAT-UMNR	01/01/1900	A	Statistics - Undergrad Minor			UGRD

Question 5B

Obtain a list of math classes that were scheduled for the Fall 2007 semester at PeopleSoft University. Display the course ID, course offer number, session code, subject, class section, catalog number, academic career, and description fields.

- Use the **CLASS_TBL** record
- Show the following fields:
  - **CRSE_ID**
  - **CRSE_OFFER_NBR**
  - **SESSION_CODE**
  - **CLASS_SECTION**
  - **SUBJECT**
  - **CATALOG_NBR**
  - **ACAD_CAREER**
  - **DESCR**
- An **INSTITUTION** value of "**PSUNV**" is used for PeopleSoft University
- A **STRM** value of "**0590**" is used for the Fall 2007 semester
- A **SUBJECT** value of "**MATH**" indicates a math class
- Save and Save often. Save as **TRNG_QM##_Q5B**

View All | Rerun Query | Download to Excel | Download to XML — First 1-31 of 31 Last

	Course ID	Offer Nbr	Session	Section	Subject	Catalog	Career	Descr
1	001010	1	1	1	MATH	301	GRAD	Topology
2	003564	1	1	1	MATH	623	GRAD	Partl Diff Equation I
3	003564	1	1	2	MATH	623	GRAD	Partl Diff Equation I
4	003556	1	1	1	MATH	567	GRAD	Combinatorics I
5	003558	1	1	1	MATH	570	GRAD	Combinatorics II
6	003560	1	1	1	MATH	600	GRAD	Topology I
7	003562	1	1	1	MATH	605	GRAD	Topology II
8	001015	1	1	1	MATH	121	UENG	Calculus I for Engineering
9	001015	1	1	1A	MATH	121	UENG	Calculus I for Engineering
10	001003	1	1	1	MATH	10	UGRD	Remedial Algebra

If you are unsure of what code represents a particular term, subject, or similar item, you can sometimes use a query to retrieve rows from a *setup table* that contains definitions of those items. For instance, as you earlier queried INSTITUTION_TBL to see the institutions known to the system, you can query TERM_TBL to see terms and SUBJECT_TBL to see subjects.

Term codes at PeopleSoft University, as of the 2004-2005 academic year, follow a standard naming convention. Start with a base value of 530. Add a value for the particular term:

- 0 for Fall Semester
- 2 for Fall Quarter
- 5 for Winter Quarter
- 7 for Intersession
- 10 for Spring Semester
- 12 for Spring Quarter
- 15 for Summer Semester
- 17 for Summer Quarter

Add 20 for each academic year after 2005 and include leading zeroes to make a four-digit number. The term code for Summer Semester 2016 is 0765; 2016 is 11 years after 2005, so add 530, 11 times 20, and 15 to get 530 + 11 * 20 + 15 = 530 + 220 + 15 = 765 and include a leading zero to obtain “0765”.

Term codes at Bowling Green State University follow this convention: the first three digits are the calendar year without the second digit and the last digit represents the semester (2 = Spring, 5 = Summer, 8 = Fall). For example, Summer 2016 is term code 2165 since 2016 without the second digit is 216 and 5 represents the Summer semester.

Edit Criteria - Part II

Not every query will involve matching exact values. Part two will show us when to use other operators besides equal to.

In the last set of questions, all of the criteria used the Condition Type "equal to" since each of them checked if a field was equal to a particular value. There are many other condition types that you can use, allowing you to create criteria that check if a field is greater than or less than a value or between two values. Here are some useful condition types and their equivalents in mathematical operators:

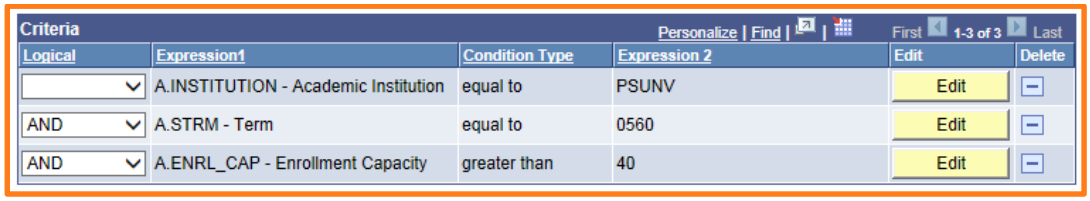
between -	Is Expression 1 between two numbers, two words, etc. (inclusive)? low-value <= Expr1 <= high value
equal to -	Is Expression 1 equal to Expression 2? Expr1 = Expr2
greater than -	Is Expression 1 greater than Expression 2? Expr1 > Expr2
less than -	Is Expression 1 less than Expression 2? Expr1 < Expr2
not between -	Is Expression 1 outside a range given by two values? Expr1 < low value or Expr1 > high value
not equal to -	Is Expression 1 not equal to Expression 2? Expr1 <> Expr2
not greater than -	Is Expression 1 less than or equal to Expression 2? Expr1 <= Expr2
not less than -	Is Expression 1 greater than or equal to Expression 2? Expr1 >= Expr2

Note that the common concept of "greater than or equal to" is defined in PeopleSoft Query as "not less than" and that "less than or equal to" is defined as "not greater than."

Create a new query to list all classes held at PeopleSoft University in Spring 2006 with an enrollment capacity of 40 or more.

- Start a query on **CLASS_TBL**
- **Select** the **SUBJECT**, **CATALOG_NBR**, **CLASS_SECTION**, **DESCR**, and **ENRL_CAP** fields
- **Click** the **Use as Criteria** button in the **INSTITUTION** row
- **Set** Condition Type to **equal to**
- **Set** constant to **PSUNV**
- **Click OK**
- **Click** the **Use as Criteria** button in the **STRM** row
- **Set** Condition Type to **equal to**
- **Set** constant to **0560**
- **Click OK**
- **Click** the **Use as Criteria** button in the **ENRL_CAP** row
- **Set** Condition Type to **greater than**
- **Set** constant to **40**

- **Click OK**
- **Click** the **Criteria** tab to review the criteria

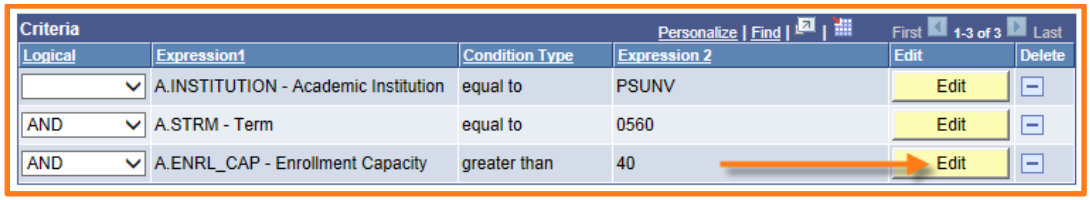


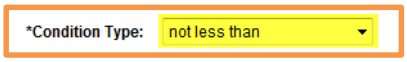
- **Save** the query as **TRNG_QM##_E6A**
- **Click** the **Run** tab.

	Section	Subject	Catalog	Descr	Cap Enrl
1	1	HISTORY	130	World History to WWII	50
2	1OE	HISTORY	130	World History to WWII	50
3	1	BIOLOGY	103	General Biochemistry	75
4	1A	ART	112	History of World Art	60
5	L001	FREN	101	Elementary French	60
6	L002	FREN	101	Elementary French	60
7	2	PHYSICS	150	Elements of Physics	100
8	1	ENGLLIT	120	Anglo-Saxon Lit	999
9	1	ENGLLIT	532	Grad Readings	999
10	1	BIOLOGY	100	General Biology I	100

Are these the results we are looking for? We want to result to yield classes with enrollment capacities of 40 or more. We chose "greater than" for the condition; this means classes with an enrollment capacity of exactly 40 will be missed. We need to change the condition so that it means "greater than or equal to," which in Query Manager is referred to as "not less than."

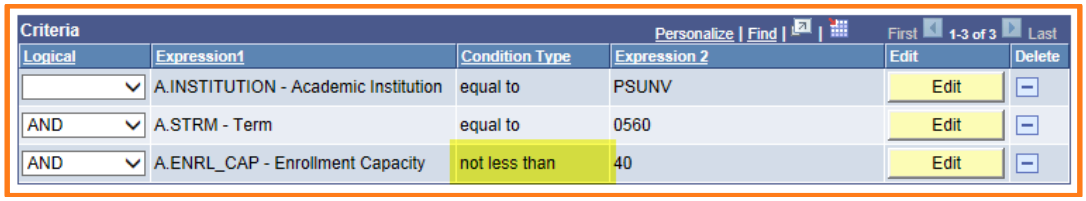
- **Click** the **Criteria** tab
- **Click Edit** in the **ENRL_CAP** row





- **Change** Condition Type to "**not less than**"

- **Click OK**.



- **Save** the query
- **Click Run** tab
- Now classes with an enrollment capacity of exactly 40 are included in the results

View All | Rerun Query | Download to Excel | Download to XML First 1-18 of 18 Last

	Section	Subject	Catalog	Descr	Cap Enrl
1	1	HISTORY	130	World History to WWII	50
2	1OE	HISTORY	130	World History to WWII	50
3	1	BIOLOGY	103	General Biochemistry	75
4	1	ART	112	History of World Art	40
5	1A	ART	112	History of World Art	60
6	1B	ART	112	History of World Art	40
7	L001	FREN	101	Elementary French	60
8	L002	FREN	101	Elementary French	60
9	2	PHYSICS	150	Elements of Physics	100
10	1	ENGLLIT	120	Anglo-Saxon Lit	999

Using Wildcards

Wildcards are symbols that substitute for other characters in search strings. They act much like "wild" cards in a card game; wild cards can be used as if they were any other card in a card game.

The primary use of wildcards in queries is to find rows in which a text field *contains* a string rather than *equals* a string. You may want to find instances in which that text has a particular word or a set of consecutive characters.

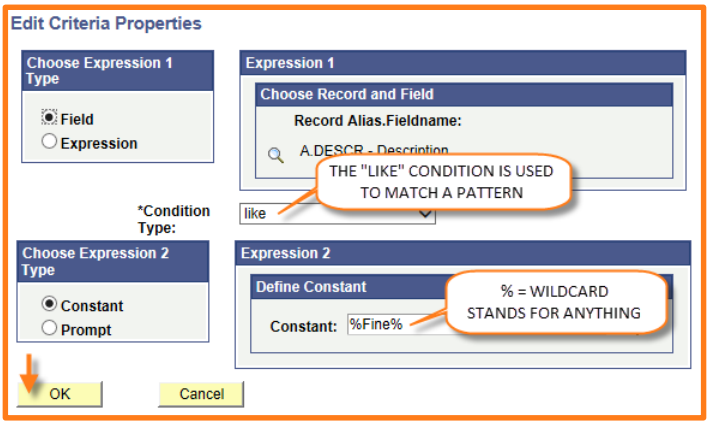
You can use the wildcard character % to represent any number of characters, including zero characters. For instance, "%ark" would be a match against "dark", "mark", "shark", as well as "ark". A wildcard character can appear several times in a search string. See the Supplemental Material for more examples of search strings containing wildcard characters.

To create criteria that involve matching partial strings, use the "like" condition type instead of "equal to". ("Like" in this case is used to mean "similar to".)

Let's obtain a list of item types that contain "Fine" in the description. We will use the wildcard symbol (%) in front of and behind the word Fine.

- Start a query on **ITEM_TYPE_TBL**
- **Select** fields:
  - **SETID**
  - **ITEM_TYPE**
  - **EFFDT**
  - **EFF_STATUS**
  - **DESCR**
- Save and save often. Save as **TRNG_QM##_E6B**

- **Click Fields** tab
- **Click** in the DESCR row
- **Set** Condition Type to "**like**"
- **Place %Fine%** in the Constant box
- **Click OK**
- **Save** the query
- **Click** the **Run** tab



View All | Rerun Query | Download to Excel | Download to XML

First 1-7 of 7 Last

	SetID	Item Type	Eff Date	Status	Descr
1	MODEL	000001000037	01/01/1900	A	Fineburg Fine Arts Scholarship
2	MODEL	021000000005	01/01/1900	A	Library Fines
3	MODEL	200000000001	01/01/1900	A	Parking Fines
4	PSUNV	000001000037	01/01/1900	A	Fineburg Fine Arts Scholarship
5	PSUNV	200000000001	01/01/1998	A	Parking Fines
6	PSUNV	210000000005	01/01/1998	A	Library Fines
7	PSUNV	210000010005	01/01/1998	A	Library Fines

Important note: "like" is case sensitive! If you change "Fine" to "fine" you'll get NO results. Refer to the "Criteria and Case Sensitive Data" Segment of the Supplemental Material.

For best practice it is better to build criteria based on a code or specific parts of a code rather than the description. Codes are unlikely to change and usually follow conventions, with specific characters or digits having particular meanings, whereas descriptions can be misspelled or have alternate spellings. For instance, if all item types that represent fines have 8 as the fifth digit, it would be better to add a criterion that checks the fifth digit of the item type than to look for "Fine" in the description – especially since not all instances of "Fine" represent a charge (see the "Fine Arts Scholarship" in the example). However, depending on how the codes are determined, the description may be the only field that carries the meaning; in the example, there is no digit in the item type that designates that it is for a fine.

Question 6

For this question, find the term codes, description, and term beginning date for a specified amount of time. Create a new query to get the term code, description, and term beginning date for all fall terms from 2010 through 2014 for undergraduate students at PeopleSoft University. Use the term description to determine whether or not a term is in the fall. Include only the fall semesters; do not include the fall quarters. (Fall semesters have descriptions ending with "Fall" whereas fall quarters have descriptions ending with "Fall Qtr".)

- Use the **TERM_TBL** record
- Show the following fields:
  - **STRM**
  - **DESCR**
  - **TERM_BEGIN_DT**
- An **INSTITUTION** value of "PSUNV" indicates PeopleSoft University
- An **ACAD_CAREER** value of "UGRD" indicates undergraduate
- Add Criteria to **TERM_BEGIN_DT** for terms starting *between* 01/01/2010 and 12/31/2014
- How will you set up a criterion to look for "Fall" at the *end* of the term description? Examine the DESCR field to determine if a term is in the Fall semester. What condition type will you use? Where will you place wildcards?
- Save and save often. Save as **TRNG_QM##_Q6**

View All | Rerun Query | Download to Excel | Download to XML — First 1-5 of 5 Last

	Term	Descr	Begin Date
1	0730	2014 Fall	08/30/2014
2	0710	2013 Fall	08/30/2013
3	0690	2012 Fall	08/30/2012
4	0670	2011 Fall	08/30/2011
5	0650	2010 Fall	08/30/2010

At PeopleSoft University there are eight careers: BUSN (Graduate Business), CNED (Continuing Education), GRAD (Graduate), LAW (Maw), MEDS (Medical School), TECH (Technical), UENG (Undergraduate Engineering), and UGRD (Undergraduate).

There are only two careers at Bowling Green State University: UGRD (Undergraduate) and GRAD (Graduate).

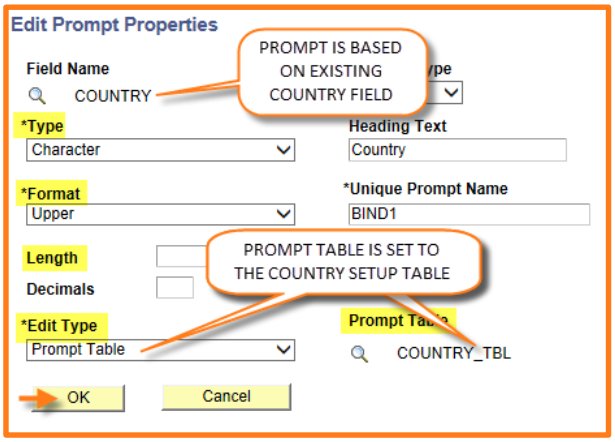
Working with Prompts

Prompts allow a user to enter different values into a query each time it runs. For instance, you might look for transactions over $100 in one instance and transactions over $2000 in another instance. Rather than writing two different queries, you can write one query that prompts for a value. This makes the query more flexible and reduces duplicate queries that do almost the same thing.

Let's create a query that lists all the states belonging to a country that the user selects at run time.

- Use the following record **STATE_TBL**
- **Click** the **Check All** button
- Save and save often. Save as **TRNG_QM##_E7**
- **Click** in the COUNTRY row
- **Set** Choose Expression 2 Type to "**Prompt**"
- **Click New Prompt** link for Expression 2

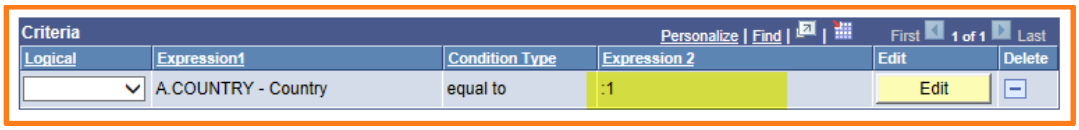
If the prompt is based on an existing field (COUNTRY), the prompt's characteristics (such as Type, Format, and Length) default to those of the field. In the case of COUNTRY, Type is already set to Character, Format to Upper, and Length to 3).



A prompt can restrict legal values to those from a table, a list of translate values, or Boolean values (yes and no). The Edit Type field is used for this. Observe the *Edit Type* is set to *Prompt Table*. Prompt Table is set to COUNTRY_TBL. This means the *user can only select a country known to the system*.

- **Click OK** to return to the Edit Criteria Properties dialog
- **Click OK** to return to the Query tab

The Expression 2 prompt is set to :1. Upon running the query :1 will be replaced by the value entered by the user (a country code). To see this in the criterion **click** the **Criteria** tab.

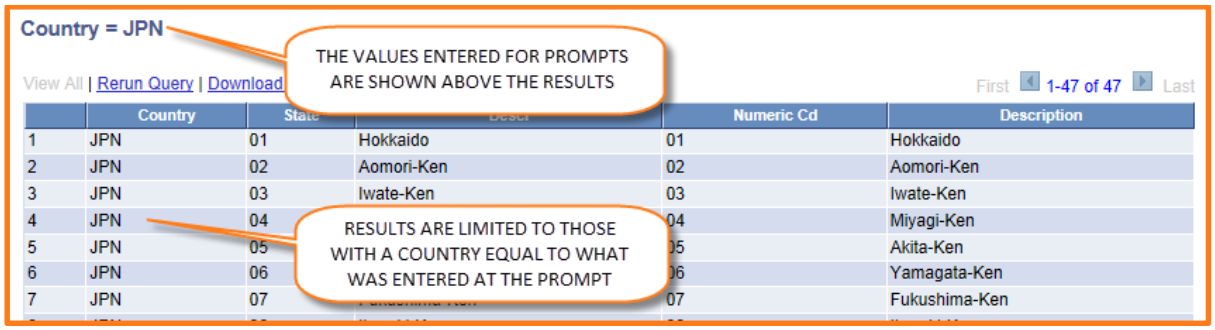


- **Click** the Prompts tab to confirm the addition of the new prompt

- **Save** the query
- **Click Run** tab

A prompt window appears in which you are asked to enter a country.

- **Enter JPN**
- **Click OK**



It is not possible to set a value to appear by default in the prompt window. Numeric prompts always default to zero and text and date prompts always default to blank.

If you create a prompt but do not reference it in an expression or in criteria, the prompt will not appear when you run the query.

Question 7

Let's use an existing query to find classes with a particular subject in Fall 2007. Modify the query you wrote earlier to display the math classes offered in Fall 2007 so that it prompts the user for a subject. This question offers a great learning opportunity on editing criteria and adding criteria.

Run the query twice. First, enter a subject of MATH to demonstrate that the query still returns the same results as before adding the prompt. Second, enter a subject of ENGLCOMP to test it out with a different subject.

- Use existing query **TRNG_QM##_Q5B**
- Save the query with a new name **TRNG_QM##_Q7**
- **Change** the criterion that **SUBJECT** is equal to MATH to be equal to a prompt instead
- Run query using subject *MATH*
- Rerun query using subject *ENGLCOMP* (English Composition)
- Save again when finished

Subject = MATH

View All | Rerun Query | Download to Excel | Download to XML First 1-31 of 31 Last

	Course ID	Offer Nbr	Session	Section	Subject	Catalog	Career	Descr
1	001010	1	1	1	MATH	301	GRAD	Topology
2	003564	1	1	1	MATH	623	GRAD	Partl Diff Equation I
3	003564	1	1	2	MATH	623	GRAD	Partl Diff Equation I
4	003556	1	1	1	MATH	567	GRAD	Combinatorics I
5	003558	1	1	1	MATH	570	GRAD	Combinatorics II

Subject = ENGLCOMP

View All | Rerun Query | Download to Excel | Download to XML First 1-13 of 13 Last

	Course ID	Offer Nbr	Session	Section	Subject	Catalog	Career	Descr
1	003334	1	1	1	ENGLCOMP	555	GRAD	Adv Fict Writ I
2	003336	1	1	1	ENGLCOMP	556	GRAD	Adv Fict Writ II
3	003336	1	1	2	ENGLCOMP	556	GRAD	Adv Fict Writ II
4	001212	1	1	1	ENGLCOMP	100	UGRD	English Composition I
5	003278	1	1	1	ENGLCOMP	101	UGRD	Intro Verse

When you create the prompt on the SUBJECT field you may notice that Edit Type defaults to "No Table Edit" and Prompt Table defaults to a blank. This means that what the user enters at the prompt will not be validated against known subjects. There is a setup table named SUBJECT_TBL that contains all available subjects; however, to use it you must also include a prompt on INSTITUTION_TBL since INSTITUTION is a field that precedes SUBJECT in the key of SUBJECT_TBL. This is a requirement of Query Manager; to use a prompt table for validation, you must have prompts on all the fields prior to that field in the key.

Writing Expressions

Expressions are ways to perform calculations and conversions and to edit or combine text strings. They can be displayed as a column in the result set and can be used in criteria.

Expressions are comprised of fields, values, operators, and functions. Operators generally perform a calculation on the fields and values preceding and following them. For instance, in "5 + 3", the "+" is the operator which performs a calculation on 5 and 3 to produce 8. Functions take a set of input values and produce an output value. The square root function is SQRT; SQRT (4) in an expression has a result of 2.

For the next exercise, we'll perform a "what-if" analysis. We'll examine what would happen if Student Financial Aid offers 2.5% more aid for each award type and if this exceeds the current budget for each award type. This would help us determine if more money needs to be allocated for that award.

Our analysis will be for the 2007 Aid Year (the most recent available in the training data set) and will only consider awards for which aid was offered in that year.

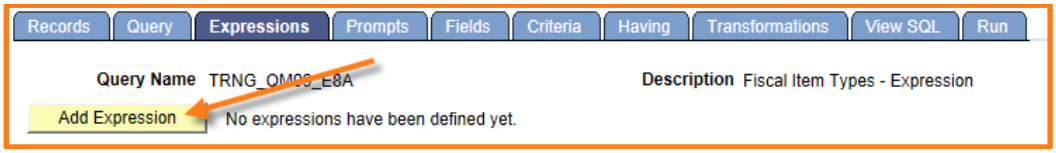
- Start a query on **ITEM_TYPE_FISCL**
- **Select** fields:
  - **ITEM_TYPE**
  - **MAX_OFR_BUDGT**
  - **OFR_GROSS**
- **Click** the **Use as Criteria** button for **SETID**.
- **Set** the Constant to '**PSUNV**'.
- **Click** the **OK** button.
- **Click** the **Use as Criteria** button for **AID_YEAR**.
- **Set** the Constant to **2007**.
- **Click** the **OK** button.
- **Click** the **Use as Criteria** button for **OFR_GROSS**.
- **Set** the Condition Type to "**greater than**".
- **Set** the Constant to **0**.
- **Click** the **OK** button.

- Save and save often. Save as **TRNG_QM##_E8A**
- **Click** the **Run** tab to make sure the query runs

View All | Rerun Query | Download to Excel | Download to XML — First 1-44 of 44 Last

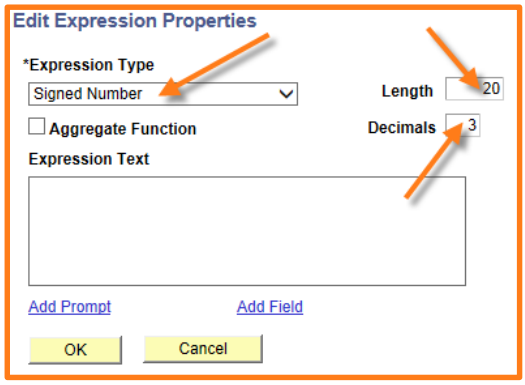
	Item Type	Budgeted	Gross
1	900000000001	1000000.000	143320.000
2	900000000006	1000000.000	396292.000
3	900000000009	1000000.000	12200.000
4	900000000010	1000000.000	16850.000
5	900000000040	500000.000	84000.000
6	900000000041	1000000.000	418734.000
7	900000000042	1000000.000	4500.000
8	900000000043	1000000.000	379776.000
9	900000000044	1000000.000	686203.000
10	900000000100	1000000.000	288275.000

- **Click Expressions** tab
- **Click Add Expression** button

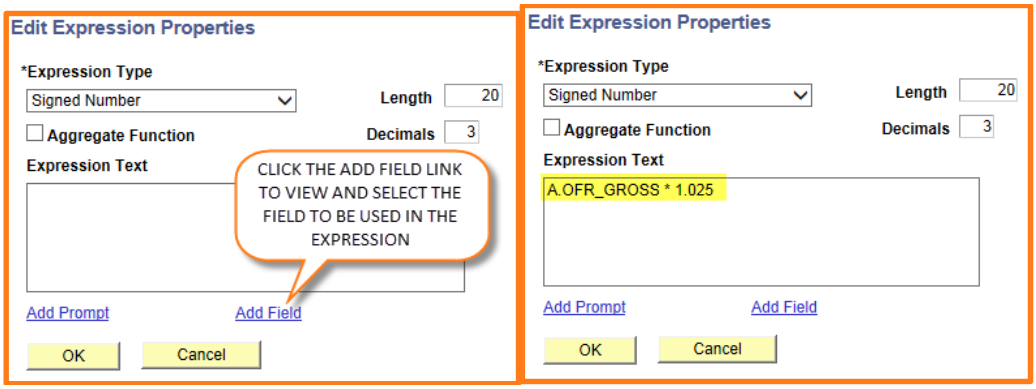


We must choose the type of expression, indicating what sort of result it gives. Will the result be a string (character), a number, a date, or something else? In this case, **select Signed Number** from the Expression Type drop-down.

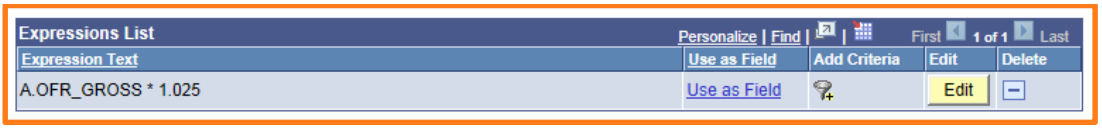
We must also set the size of the result. If the result was a string, the length would be the maximum number of characters in the string. For a number, we set the number of length in digits, including both before and after the decimal point. We also separately specify the number of digits after the decimal.



- **Set** Expression Type to **Signed Number**
- **Set Length** to **20**
- **Set Decimals** to **3** (this means there will be 17 digits before the decimal)
- **Click** the **Add Field** link and **select A.OFR_GROSS**
- In The Expression Text box, enter "*** 1.025**" (without the quotes) *after* A.OFR_GROSS (the finished expression should be "A.OFR_GROSS * 1.025")
- **Click OK**

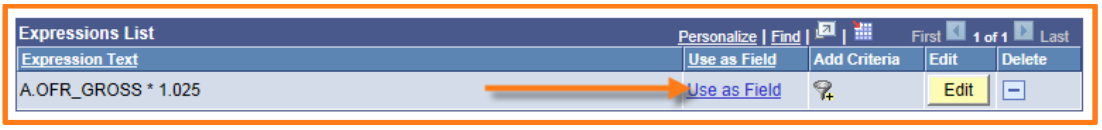


You are returned to the Expressions tab. The new expression is displayed in the Expressions List.

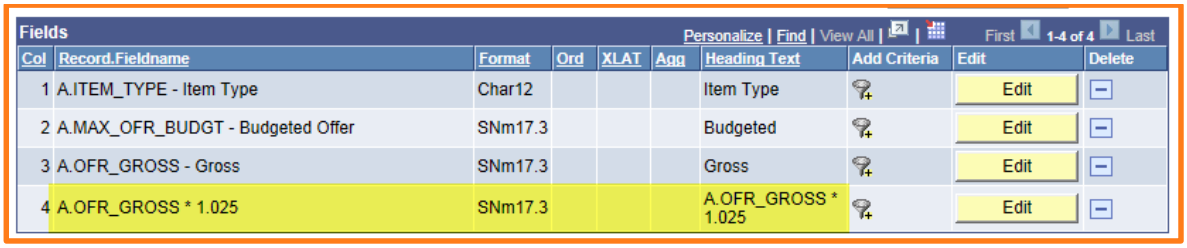


The *Use as Field* link is used when the expression is going to be used as a field. We want to create a column that provides us the gross offer amount including the 2.5% increase.

- **Click** the **Use as Field** link in the row for A.OFR_GROSS * 1.025

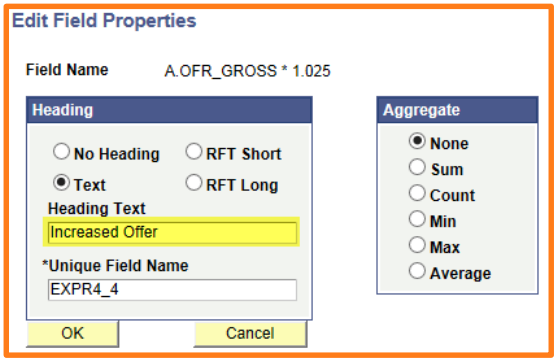


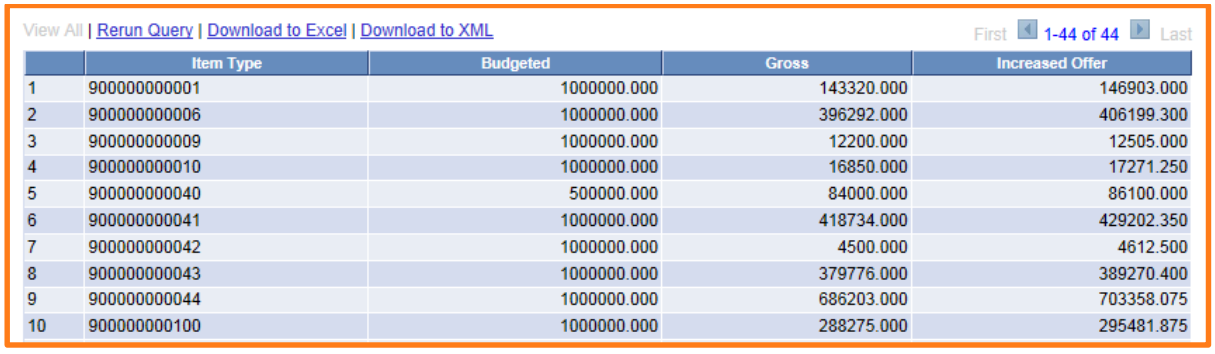
The expression A.OFR_GROSS * 1.025 appears on the Fields page.



Note that the heading of the column containing the expression defaults to the expression itself. It's best to change this to a meaningful column heading.

- **Click** the **Edit** button in the A.OFR_GROSS * 1.025 row
- **Observe** that the heading type is **Text**; this is needed to allow a custom column heading
- **Change** the Heading Text to "**Increased Offer**"
- **Click OK**
- **Save** the query
- **Click Run** tab



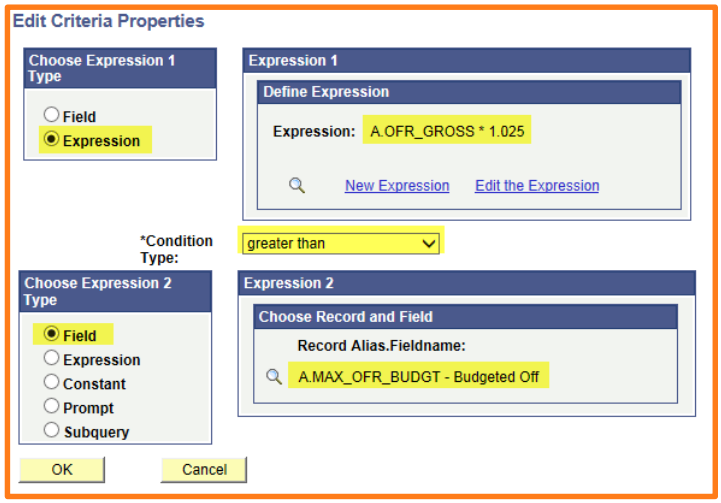


Now we want to see all the awards that will have a higher gross offer amount than budgeted amount *after* the increase to the offers.

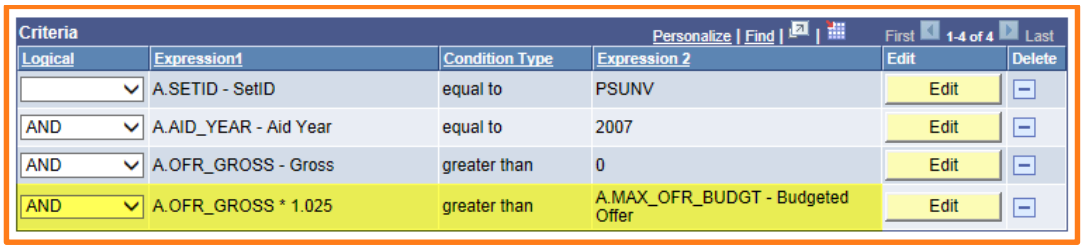
We'll use the current query as a starting point. Click the **Criteria** tab. Click the **Save As** link. Set the new name of the query to **TRNG_QM##_E8B** then click the **OK** button.

Next, perform the following steps:

- **Click** the **Add Criteria** button
- **Set** Expression 1 Type to **Expression**
- **Click** in Expression 1, Define Expression box
- Click **A.OFR_GROSS * 1.025**
- **Set** Condition Type to "**greater than**"
- **Set** Expression 2 Type to **Field**
- **Click** in Expression 2, Choose Record and Field box
- **Select A.MAX_OFR_BUDGET**
- **Click OK**



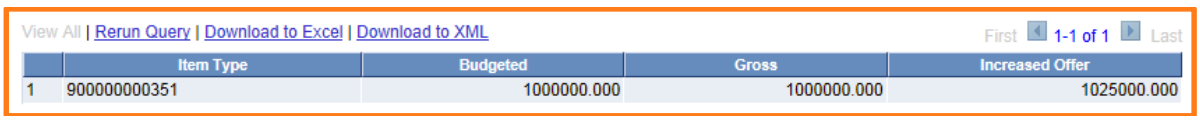
You should see a new criterion for A.OFR_GROSS * 1.025 being greater than A.MAX_OFR_BUDGT listed on the Criteria tab.



Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.SETID - SetID	equal to	PSUNV	Edit	
AND	A.AID_YEAR - Aid Year	equal to	2007	Edit	
AND	A.OFR_GROSS - Gross	greater than	0	Edit	
AND	A.OFR_GROSS * 1.025	greater than	A.MAX_OFR_BUDGT - Budgeted Offer	Edit	

This exercise has shown that you can not only show the results of calculations in the rows returned by the query, you can also use the results of calculations to control which rows are returned.

- **Save** your query
- Click **Run** tab



	Item Type	Budgeted	Gross	Increased Offer
1	900000000351	1000000.000	1000000.000	1025000.000

Now we will write a new query that will find all senior-level classes offered at PeopleSoft University in Fall 2007. As you might expect by now, this will involve a query on CLASS_TBL with criteria that INSTITUTION must be equal to "PSUNV" and STRM must be equal to "0590". The query will also need a criterion that examines each class' catalog number to determine if that class is senior-level.

Start a new query. Add the record **CLASS_TBL** to it. Select the following fields:

- **CRSE_ID**
- **CRSE_OFFER_NBR**
- **SESSION_CODE**
- **CLASS_SECTION**
- **SUBJECT**
- **CATALOG_NBR**
- **DESCR**

Add the following criteria:

- **INSTITUTION** is equal to "PSUNV"
- **STRM** is equal to "0590"

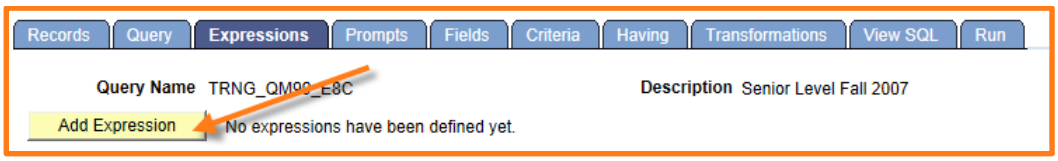
Save this query as **TRNG_QM##_E8C**. **Run** the query to see the results for what you have built so far. Notice that classes at all levels – freshman, sophomore, etc. – are in the results.

View All | Rerun Query | Download to Excel | Download to XML First 1-100 of 731 Last

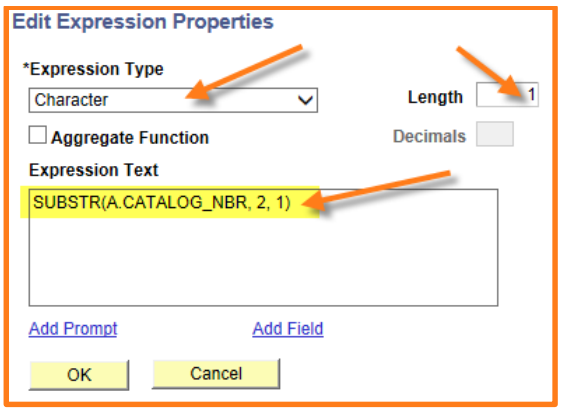
	Course ID	Offer Nbr	Session	Section	Subject	Catalog	Descr
1	001003	1	1	1	MATH	10	Remedial Algebra
2	001003	1	1	2	MATH	10	Remedial Algebra
3	001003	1	1	3	MATH	10	Remedial Algebra
4	001004	1	1	1	MATH	107	Precalculus
5	001005	1	1	1	MATH	111	Calculus I
6	001005	1	1	2	MATH	111	Calculus I
7	001006	1	1	1	MATH	212	Calculus II
8	001008	1	1	1	MATH	136	Introduction of Linear Algebra
9	001002	1	1	1	MATH	104	Finite Mathematics
10	001026	1	1	1	HISTORY	130	World History to WWII

Next we need to build the criterion that limits the results to just senior-level classes. We will be comparing a field (CATALOG_NBR) against an expression. We will use a function to extract the second character from CATALOG_NBR then check whether it is a 4.

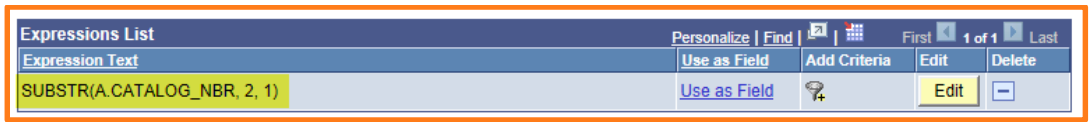
Click the **Expressions** tab then click the **Add Expression** button to create a new expression.



The Edit Expression Properties dialog appears. CATALOG_NBR is a textual value rather than an actual number, so we will leave **Expression Type** set to **Character**. We will only examine one character of the string, so the **Length** of the expression can be left at **1**. In the **Expression Text** field, enter the following (without the quotes): "**SUBSTR(A.CATALOG_NBR, 2, 1)**". This expression means get the *substring* (part of a string) starting at position 2 and having a length of 1 – just the second character of the catalog number, which will be the level of the class.

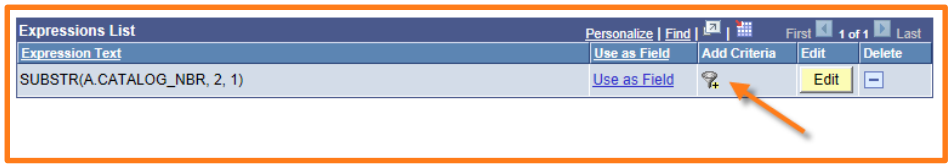


Click the **OK** button to return to the Expressions tab. The new expression will be in the Expressions list.



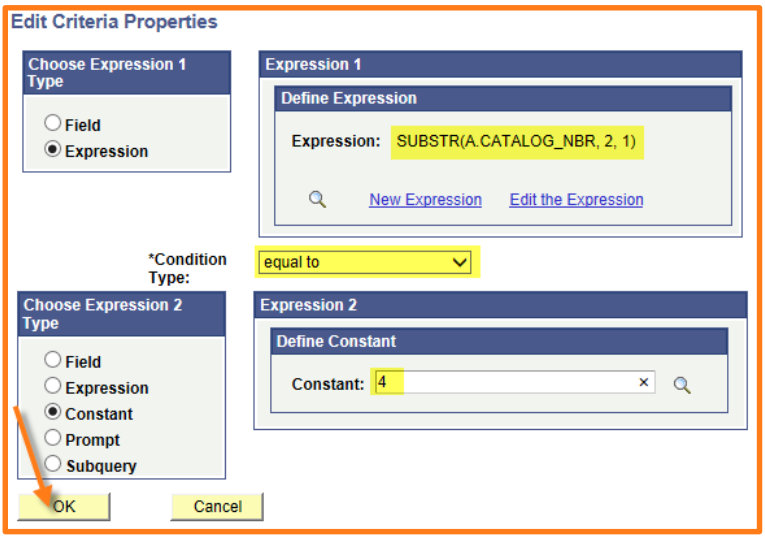
We need to restrict the results to only senior-level classes; we are doing this through a criterion that the second character of the catalog number is 4. To do this, we need to add a criterion that uses the new expression.

- In the row of the Expressions List for the **SUBSTR** expression,

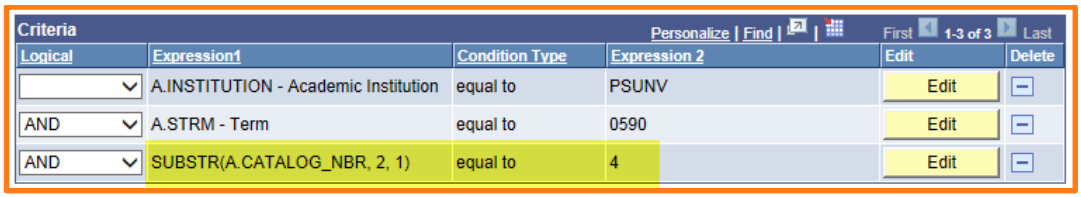


click the **Add Criteria** button

- The Edit Criteria Properties screen appears. Observe that the **SUBSTR** expression has already been selected
- The Condition Type is already "equal to." In the **Constant** field for Expression 2, **enter 4**
- **Click** the **OK** button



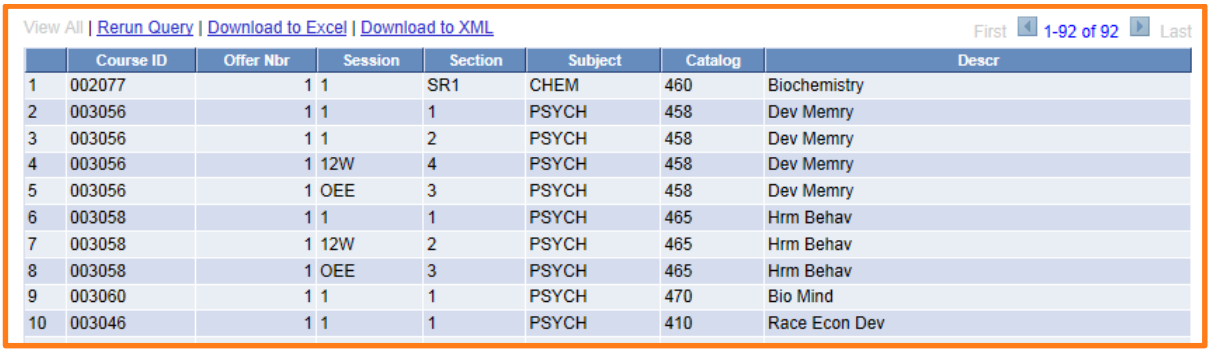
Now let's see how the new criterion looks. **Click** the **Criteria** tab. Observe that the bottom row shows "**SUBSTR(A.CATALOG_NBR, 2, 1)** equal to 4".



Criteria — Personalize | Find | First 1-3 of 3 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.INSTITUTION - Academic Institution	equal to	PSUNV	Edit	
AND	A.STRM - Term	equal to	0590	Edit	
AND	SUBSTR(A.CATALOG_NBR, 2, 1)	equal to	4	Edit	

- **Save** your changes
- **Click** the **Run** tab to view the results. Each class shown will be senior-level (4xx).



View All | Rerun Query | Download to Excel | Download to XML — First 1-92 of 92 Last

	Course ID	Offer Nbr	Session	Section	Subject	Catalog	Descr
1	002077	1	1	SR1	CHEM	460	Biochemistry
2	003056	1	1	1	PSYCH	458	Dev Memry
3	003056	1	1	2	PSYCH	458	Dev Memry
4	003056	1	12W	4	PSYCH	458	Dev Memry
5	003056	1	OEE	3	PSYCH	458	Dev Memry
6	003058	1	1	1	PSYCH	465	Hrm Behav
7	003058	1	12W	2	PSYCH	465	Hrm Behav
8	003058	1	OEE	3	PSYCH	465	Hrm Behav
9	003060	1	1	1	PSYCH	470	Bio Mind
10	003046	1	1	1	PSYCH	410	Race Econ Dev

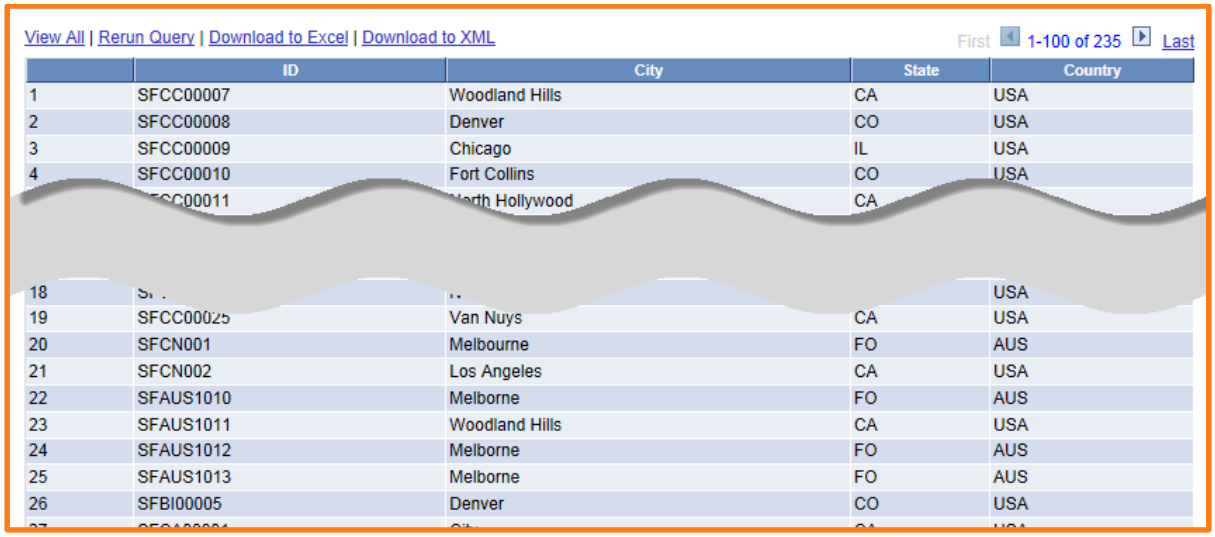
Question 8

Obtain a list of billing addresses in the system for transmission to an outside vendor for processing. The vendor's software requires that the State field have a value of "FO" (Foreign) for all addresses outside the United States. Write your query using an expression that will substitute "FO" in the State field for non-domestic addresses. Include the person's ID and the city, state, and country of each address.

- Use the **ADDRESSES** record
- Display the **EMPLID**, **COUNTRY**, and **CITY** fields
- Billing addresses are those in which **ADDRESS_TYPE** is "BILL"
- Create an *expression* that uses the value of **COUNTRY** to display the value of **STATE** if **COUNTRY** is "USA" or "FO" if **COUNTRY** is not "USA". Enter the following as the Expression Text:

  CASE WHEN A.COUNTRY = 'USA' THEN A.STATE ELSE 'FO' END

  (See the Supplemental Material for an explanation of how to use CASE expressions.)
- Use the expression you created as a field. Give the field a heading of "State"
- Change the column order so that **EMPLID** is the left-most column, followed by **CITY**, your **CASE** expression, and **COUNTRY**
- Save the query as **TRNG_QM##_Q8**



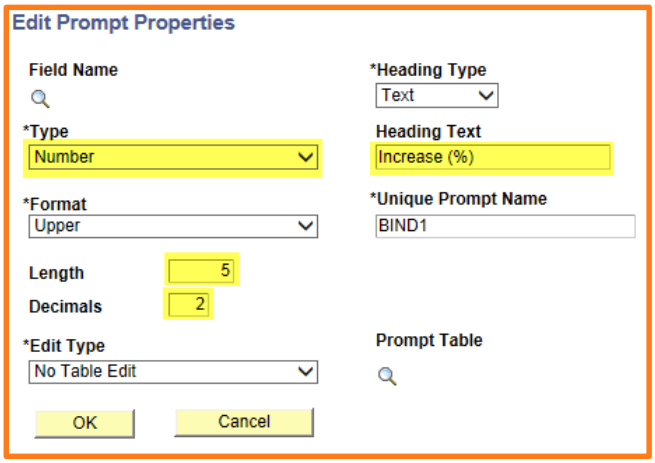
View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 235 Last

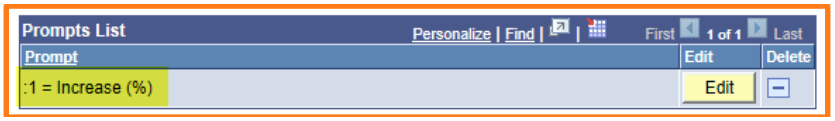
	ID	City	State	Country
1	SFCC00007	Woodland Hills	CA	USA
2	SFCC00008	Denver	CO	USA
3	SFCC00009	Chicago	IL	USA
4	SFCC00010	Fort Collins	CO	USA
	[illegible]	[illegible]	CA	
18	[illegible]	[illegible]		USA
19	SFCC00025	Van Nuys	CA	USA
20	SFCN001	Melbourne	FO	AUS
21	SFCN002	Los Angeles	CA	USA
22	SFAUS1010	Melborne	FO	AUS
23	SFAUS1011	Woodland Hills	CA	USA
24	SFAUS1012	Melborne	FO	AUS
25	SFAUS1013	Melborne	FO	AUS
26	SFBI00005	Denver	CO	USA

Using a Prompt with Expressions

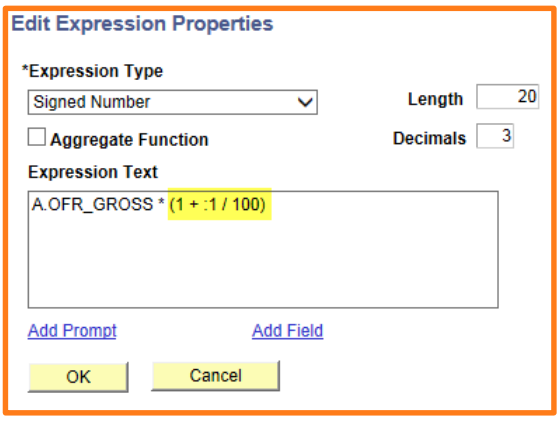
It is possible to use a prompt inside an expression, allowing the user great flexibility in determining what calculations are performed. We will use the previous query on salary grades and change this query to accept a percentage.

- Find the existing query **TRNG_QM##_E8B**
- Save as **TRNG_QM##_E9**
- **Click** the **Prompts** tab
- **Click** the **Add Prompt** button
- **Set** Type to **Number**
- **Set** Heading Text to **Increase (%)**
- **Set** Length to "**5**"
- **Set** Decimals to "**2**"
- **Click OK**



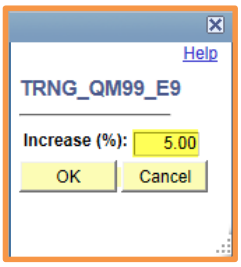


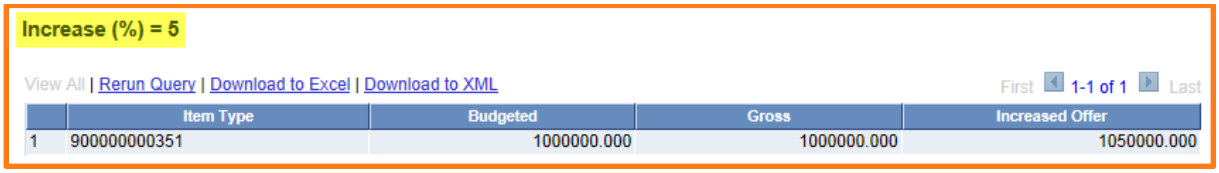
- **Click** the **Expressions** tab
- **Click** the **Edit** button in the row for A.OFR_GROSS * 1.025
- **Change** the **Expression Text** to **A.OFR_GROSS * (1 + :1 / 100)**
- **Click OK**



Recall from the Working with Prompts lesson that the prompt indicator :1 will be replaced with the value that you enter when the query is run. When you run the query and enter "5.00" for the increase amount, the expression becomes A.OFR_GROSS *(1 + 5.00 / 100), which evaluates to A.OFR_GROSS * 1.05. By including a prompt in the expression, you can enter a different increase amount each time the query is run.

- **Save** the query
- **Click** the **Run** tab
- **Enter** 5.00 for the increase percentage

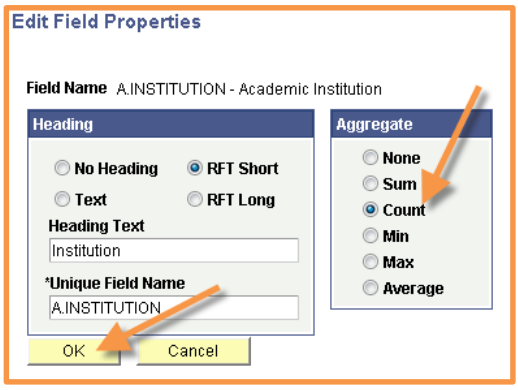




Use Aggregate Functions

All queries built so far operate on one row at a time. It is common to want to collect information on groups of rows, such as a count, a sum, a minimum, a maximum, and an average. There are aggregate functions that are used to compile this information on either all rows in a query or broken down by certain fields. Let's get a count of academic programs across all institutions in the system.

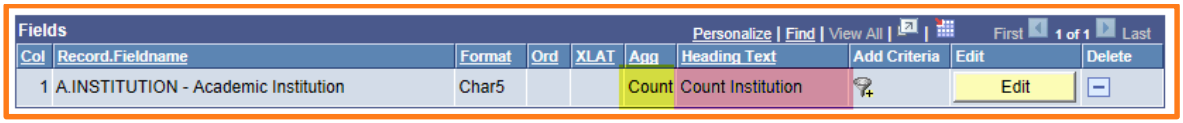
- Start a query on **ACAD_PROG_TBL**
- **Select INSTITUTION**
- **Click Fields** tab
- **Edit INSTITUTION** row
- **Set** Aggregate to **Count**



The Aggregate column now shows Count.

- **Click OK**
- Save and Save often. Save as **TRNG_QM##_E10A**

Once the aggregate function has been placed at Count, in the row for INSTITUTION, the Agg (Aggregate) column shows Count and the Heading Text on the Fields page shows Count Institution.



- **Click Run** tab

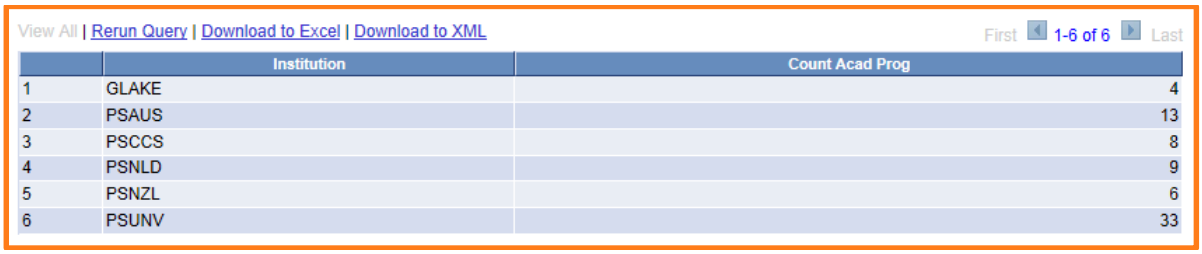
Note that this is not a count of institutions; it is a count of all academic programs. The heading is Count Institution because we set the aggregate function on the Institution field.

Let's get a count of programs at each institution.

- **Click Query** tab
- **Save** the query as **TRNG_QM##_E10B**
- **Select** the **ACAD_PROG** field for display
- **Click Fields** tab

At this time the query is counting the number of institutions that have each program. Instead, we want the number of programs at each institution. To do this we will remove the aggregate from INSTITUTION and add the count aggregate to ACAD_PROG.

- **Click** the **Edit** button in the **INSTITUTION** row
- Set **Aggregate** to **None**
- **Click** the **OK** button
- **Click** the **Edit** button in the **ACAD_PROG** row
- Set **Aggregate** to **Count**
- **Click** the **OK** button
- **Save** the query
- **Click** the **Run** tab


View All | Rerun Query | Download to Excel | Download to XML First 1-6 of 6 Last

	Institution	Count Acad Prog
1	GLAKE	4
2	PSAUS	13
3	PSCCS	8
4	PSNLD	9
5	PSNZL	6
6	PSUNV	33

Observe that the sum of all the counts of programs at each institution is the same as the count of programs overall (4 + 13 + 8 + 9 + 6 + 33 = 73).

An important consideration when using aggregate functions is how the rows are grouped together. As seen in this example, you can obtain a count of all rows returned by a query and you can obtain a count of rows within groups by values in a field such as **ACAD_PROG**.

When you have a field that has an aggregate function, the rows are grouped by *all* fields being displayed that are *not* being aggregated.

In the first part of the exercise, the only field being selected was **INSTITUTION**, and this had the Count aggregate applied to it. Since there were no fields that did not have an aggregate, the only group was all rows returned by the query. Therefore, the result was the count of plans returned by the query.

In the second part of the exercise, both **ACAD_PROG** and **INSTITUTION** were selected. Only **ACAD_PROG** had an aggregate function. **INSTITUTION** did not have an aggregate, so the rows were grouped by **INSTITUTION**. Therefore, the result was the count of programs *at each institution*.

If you instead apply the aggregate function to **INSTITUTION**, your query will produce a count of institutions that have each academic program; the results will look like this:

View All | Rerun Query | Download to Excel | Download to XML First 1-65 of 65 Last

	Acad Prog	Count Institution
1	A&S	1
2	AA	2
3	AS	1
4	BART	2
5	BCOM	2
6	BECON	1
7	BENG	1
8	BSCI	2
9	CEOEE	1
10	CERT	2

SEARCHING FOR MORE . . .

- The Count, Min, and Max aggregate functions can be applied to any field. For numeric fields, Min and Max return the lowest and highest numeric values in the group, respectively. For text fields, they return the earliest and latest values in alphabetical order in the group, respectively. (When dealing with mixed-case data, uppercase is considered to be “less than” lowercase and shorter strings are “less than” longer strings.) The Sum and Average functions can only be used on numeric fields.

- If you want to have criteria that use the results of an aggregate, such as restricting the results to those rows with a count greater then 50, you must add these in the Having tab instead of the Criteria tab. (If you click the  button for a field that has an aggregate function, you will automatically add a having criterion instead of a regular criterion.)

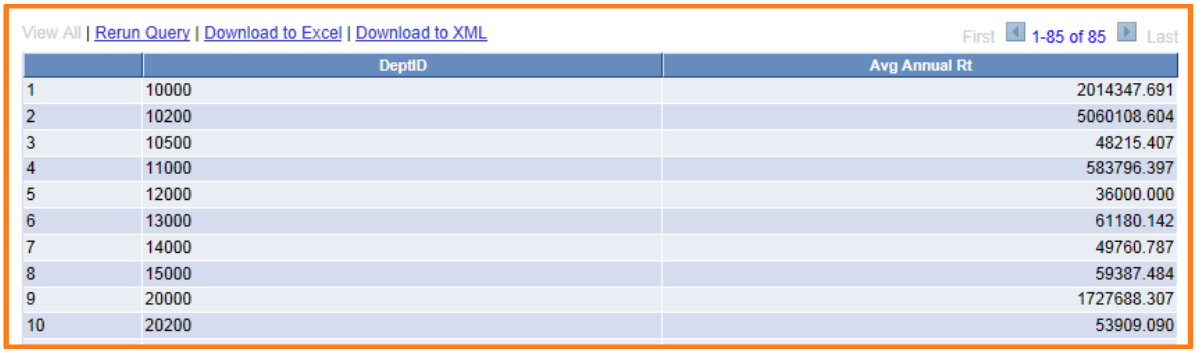
- You can use aggregate functions in expressions. The names of the functions are SUM, COUNT, MIN, MAX, and AVG; each of these accepts a field name as an argument. You must check the Aggregate Function checkbox in the Edit Expressions Properties page to use one of these functions in an expression.

Question 10

Find the average pay rate of active, full-time employees by department. Start a new query on JOB. Select department ID and annual rate. Change the aggregate function on annual rate to average. Add criteria that employee status is equal to A (active), full part time is equal to F (full-time), standard hours is equal to 40, and the job indicator is P (primary job).

- Use the **JOB** record
- Show fields for **Department ID** and **Annual Rate**
- Obtain an **average** on **Annual Rate**
- Add Criteria:
  - **EMPL_STATUS** is equal to A
  - **FULL_PART_TIME** is equal to F
  - **STD_HOURS** is equal to 40
  - **JOB_INDICATOR** is equal to P

Save and save often. Save as **TRNG_QM##_Q10A**



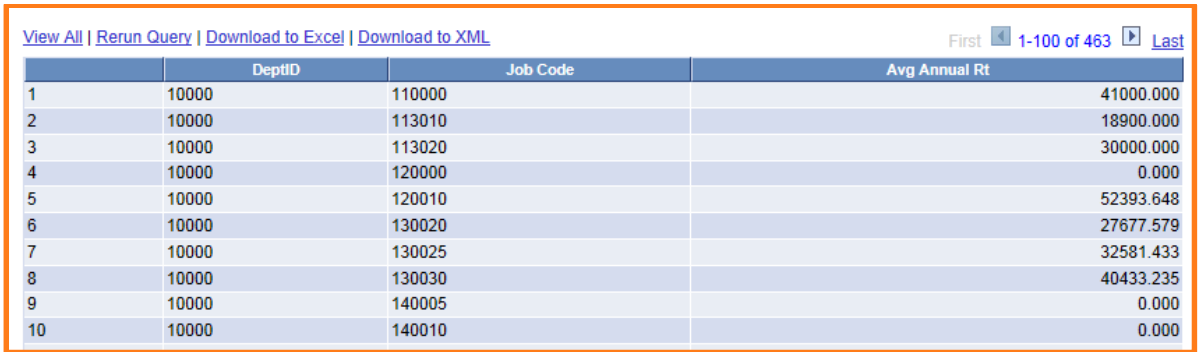
View All | Rerun Query | Download to Excel | Download to XML — First 1-85 of 85 Last

	DeptID	Avg Annual Rt
1	10000	2014347.691
2	10200	5060108.604
3	10500	48215.407
4	11000	583796.397
5	12000	36000.000
6	13000	61180.142
7	14000	49760.787
8	15000	59387.484
9	20000	1727688.307
10	20200	53909.090

Next modify the query to find the average salary for individual job codes within each department.

- Display the **Job Code** field
- Move **Job Code** to be after the department ID but before the average annual rate

Save this query as **TRNG_QM##_Q10B**.



View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 463 Last

	DeptID	Job Code	Avg Annual Rt
1	10000	110000	41000.000
2	10000	113010	18900.000
3	10000	113020	30000.000
4	10000	120000	0.000
5	10000	120010	52393.648
6	10000	130020	27677.579
7	10000	130025	32581.433
8	10000	130030	40433.235
9	10000	140005	0.000
10	10000	140010	0.000

Joins – Introduction

Most queries that you will write will involve retrieving information from several records. The PeopleSoft data is stored in a relational database; this means information about something is recorded in many different tables, with each table holding data only about one aspect of that thing. For instance, a student has names, addresses, and classes; the names are stored in one table, the addresses in another table, and the class schedule in a third table. This is done to better organize the data and to reduce duplication, such as by ensuring names are all in one table and making it unnecessary to replicate those names in other tables.

Each table in a relational database has a *primary key*, which is one or more fields that uniquely identify each row in that table. For example, a person in PeopleSoft is uniquely identified by his or her EMPLID, since no two people can have the same EMPLID. Tables that contain data that belong to a person or relate to a person will therefore have an EMPLID field. Since a person can have several addresses, such as a home address and a work address, the addresses table might have a primary key of both the EMPLID and the address type (home or work) to uniquely identify an address. (Fields that are in the primary key are shown with a key icon next to them on the Query tab.)

To use information from several tables in your query, you must *join* tables together so that data related to the same thing is presented in the same row in the results. If you want to show a student's name alongside that student's address, you would join the names and addresses tables together. You must specify in your query which fields to use to relate the tables to each other. In the case, you would join on the EMPLID field, since the student ID is what uniquely identifies a student; rows in the names table and rows in the addresses table that have the same EMPLID belong to the same student.

There are two main types of joins: an *inner join*, in which only rows in which there is matching data in both tables are returned, and an *outer join*, in which rows from one table can still be returned even if there is not a match in the other table. Outer joins are discussed later in this course.

An example of an inner join will best demonstrate how a join operates.

Consider these excerpts from the countries table (COUNTRY_TBL) and states table (STATE_TBL).

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
MEX	Mexico
USA	United States

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
MEX	DGO	Durango
MEX	JAL	Jalisco
MEX	YUC	Yucatan
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

They will be joined together so the countries and the states and provinces within them can be listed in query results. They will be joined on the COUNTRY field since this uniquely identifies a country. For instance, a row in STATE_TBL with a COUNTRY equal to "USA" represents a state in the United States.

When an inner join takes place, every row in the left table is compared against every row in the right table. When the values in the join field or fields match, the rows are combined and added as one row to the results. When the values in the join field or fields do not match, no row is added to the results.

In this example, the first comparison is of the Canada row in COUNTRY_TBL and the Alberta row in STATE_TBL. Since the join field COUNTRY has the same value CAN in both tables, a row combining the Canada row with the Alberta row is added to the results.

COUNTRY	DESCR
CAN	Canada
MEX	Mexico
USA	United States

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
MEX	DGO	Durango
MEX	JAL	Jalisco
MEX	YUC	Yucatan
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta

Next, the Canada row in COUNTRY_TBL is compared against the British Columbia row in STATE_TBL. Since the join field is COUNTRY and the COUNTRY field in both COUNTRY_TBL and STATE_TBL has the value CAN, there is a match. A new row is added to the results with a combination of the Canada row and the British Columbia row.

COUNTRY	DESCR
CAN	Canada
MEX	Mexico
USA	United States

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
MEX	DGO	Durango
MEX	JAL	Jalisco
MEX	YUC	Yucatan
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia

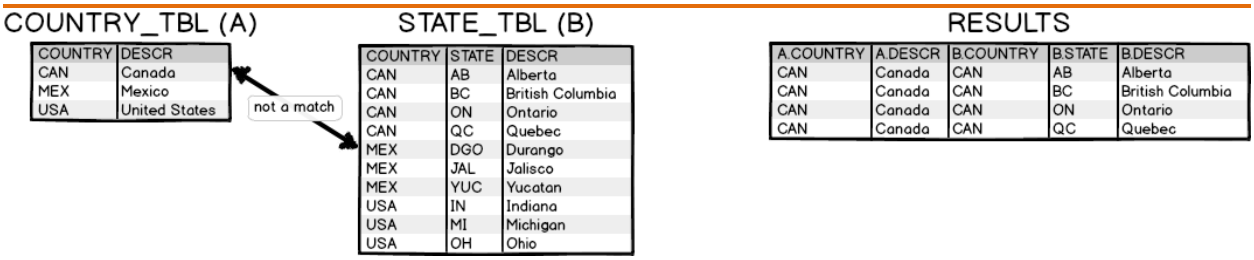
The comparisons continue with the Ontario and Quebec rows in STATE_TBL. These are both combined with Canada and added to the results.

COUNTRY	DESCR
CAN	Canada
MEX	Mexico
USA	United States

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
MEX	DGO	Durango
MEX	JAL	Jalisco
MEX	YUC	Yucatan
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec

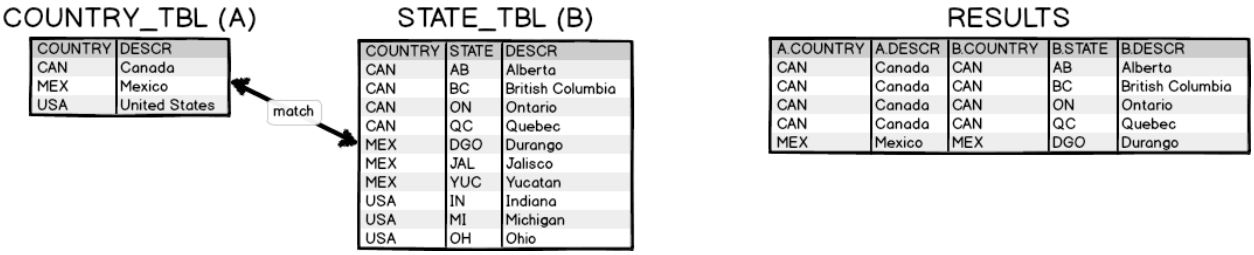
The next comparison is between the Canada row in COUNTRY_TBL and the Durango row in STATE_TBL. Since COUNTRY is CAN in the row of COUNTRY_TBL and MEX in the row of STATE_TBL, there is no match; nothing is added to the results.



The same occurs as the Canada row in COUNTRY_TBL is compared with the remaining rows in STATE_TBL for Mexico and the United States. Since the COUNTRY fields do not have matching values, no more rows are added to the results.

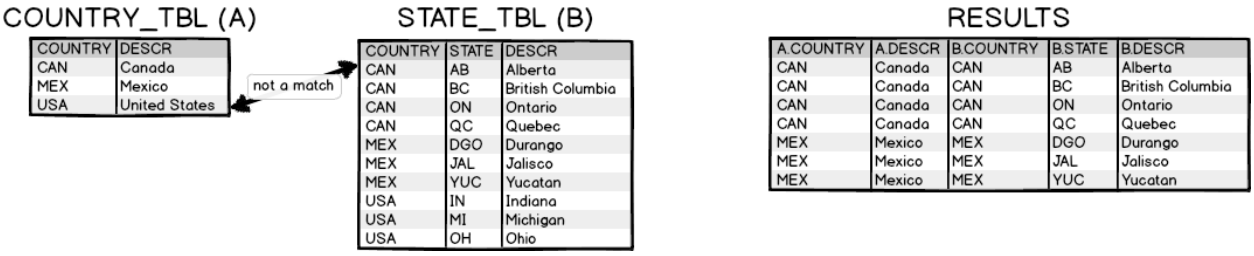
After the Canada row has been compared against all the rows of STATE_TBL, processing moves on to the Mexico row, which will also be compared against every row in STATE_TBL. In the first comparison, the Mexico row is compared to the Alberta row. Since the COUNTRY values differ, nothing is added to the results.

The Mexico row is then compared against the British Columbia, Ontario, and Quebec rows, with no matches on the COUNTRY field. Next, the Mexico row is compared to the Durango row; since COUNTRY is MEX in both the row from COUNTRY_TBL and the row from STATE_TBL, a new row is added to the results, combining data from COUNTRY_TBL and STATE_TBL.

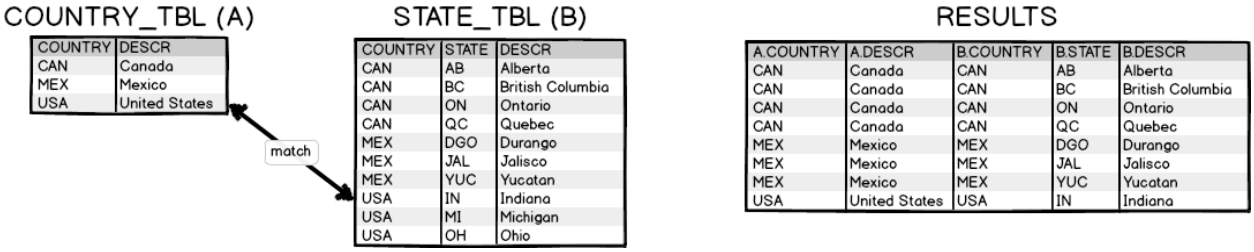


Jalisco and Yucatan are checked next and both are added to the results since there is a match on the COUNTRY value of MEX. When the Mexico row is then compared to the Indiana row, since MEX does not equal USA, Indiana is not added to the results.

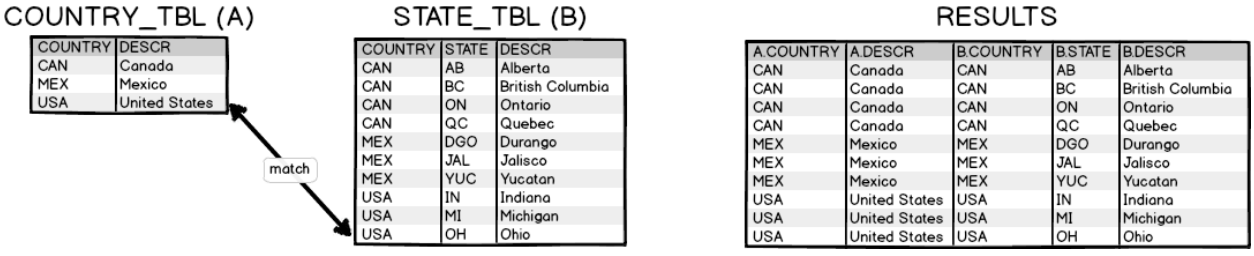
After the Mexico row of COUNTRY_TBL has been compared to every row in STATE_TBL, processing continues with the United States row of COUNTRY_TBL. The United States row is first compared to the Alberta row of STATES_TBL; since USA and CAN do not match, nothing is added to the results.



Scanning continues through the rows of STATE_TBL. No matches are found among all the rows in STATE_TBL in which COUNTRY is CAN or MEX. Once the processing reaches the Indiana row in STATE_TBL, the COUNTRY value is USA in the rows being compared in both COUNTRY_TBL and STATE_TBL, so a row is added to the results.



Michigan and Ohio are added to the results as well since COUNTRY is USA in the rows of STATE_TBL for those states.



At the end, the results contain every row from COUNTRY_TBL combined with each row in STATE_TBL in which the COUNTRY fields of COUNTRY_TBL and STATE_TBL match each other.

Criteria can be added to further restrict the results if desired. For instance, one could add a condition that the name of the state contains "an", expressed as B.STATE is like "%an%". The results would then be the following:

RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
MEX	Mexico	MEX	DGO	Durango
MEX	Mexico	MEX	YUC	Yucatan
USA	United States	USA	IN	Indiana
USA	United States	USA	MI	Michigan

In PeopleSoft Query, inner joins are further classified as one of three types:

- Record Hierarchy – all data in one table represents details about or children of data in another table
- Related Record – data in a table gives more information, such as descriptions and meanings about something referenced in another table
- Any Record – manually join two tables on any fields

The distinctions between these types and how to construct inner joins are discussed in the following sections.

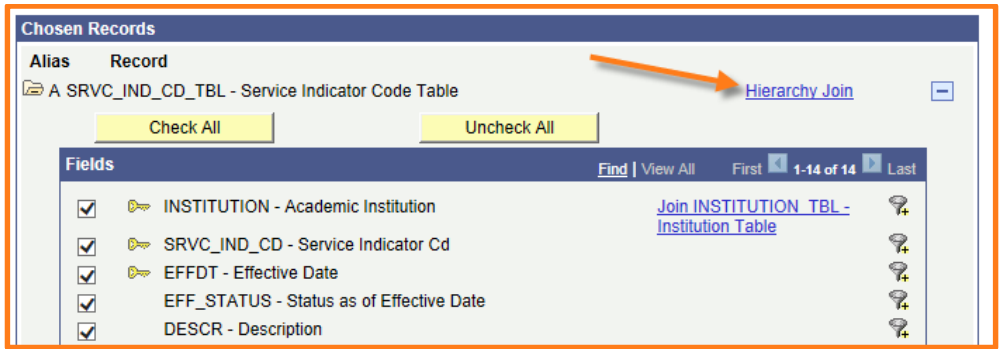
This space intentionally left blank.

Joins – Record Hierarchy

The hierarchy join is used to get information about something at two or more levels; this is sometimes referred to as a *master/detail* or *parent/child* relationship. One example of this is an invoice; the invoice itself is one level and the line items are another level.

For the next exercise, we will get a list of currently defined service indicators (holds) at PeopleSoft University. These have codes (in SRVC_IND_CD_TBL) and reasons (in SRVC_IN_RSN_TBL). If we want to show the descriptions of both the code and the reason, we need to *join* the tables. For a parent/child relationship, the join is on *all the key fields (* = key field)*.

- Start a query on **SRVC_IND_CD_TBL** (this table provides codes)
- **Select** fields:
  - **INSTITUTION***
  - **SRVC_IND_CD***
  - **EFFDT***
  - **EFF_STATUS**
  - **DESCR**
- Add a criterion that **INSTITUTION** is equal to "PSUNV"
- Save and save often. Save as **TRNG_QM##_E11**
- **Click Hierarchy Join** link

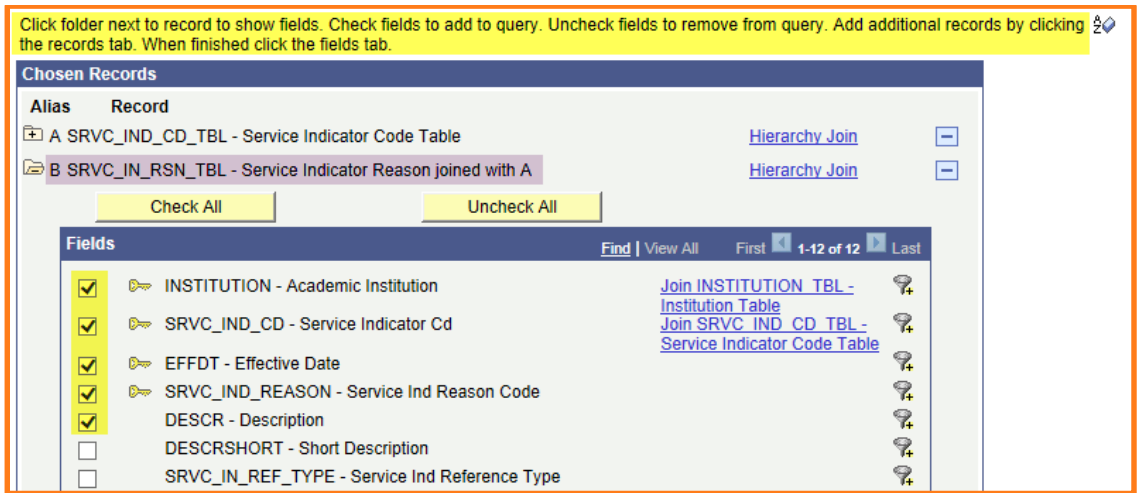


The hierarchy join window shows the parent/child relationship. The relationship is joined on all key fields of the parent table. The primary keys in the parent table are INSTITUTION, SRVC_IND_CD, and EFFDT. The primary key of the child table, SRVC_IN_RSN_TBL, contains all fields of the primary key of its parent, as well as the reason code, SRVC_IND_REASON. The "Select record for hierarchy join" window allows you to select a child record and automatically add the criteria for the join.

- **Click SRVC_IN_RSN_TBL** (this record provides reasons).
- **Show** fields:
  - **INSTITUTION**
  - **SRVC_IND_CD**

- **EFFDT**
- **SRVC_IND_REASON**
- **DESCR**

Notice that there is an additional record that has been added to the Chosen Records section. The second alias record is **B.SRVC_IN_RSN_TBL – Service Indicator Reason joined with A**.



- Save the query.
- **Click Run** tab

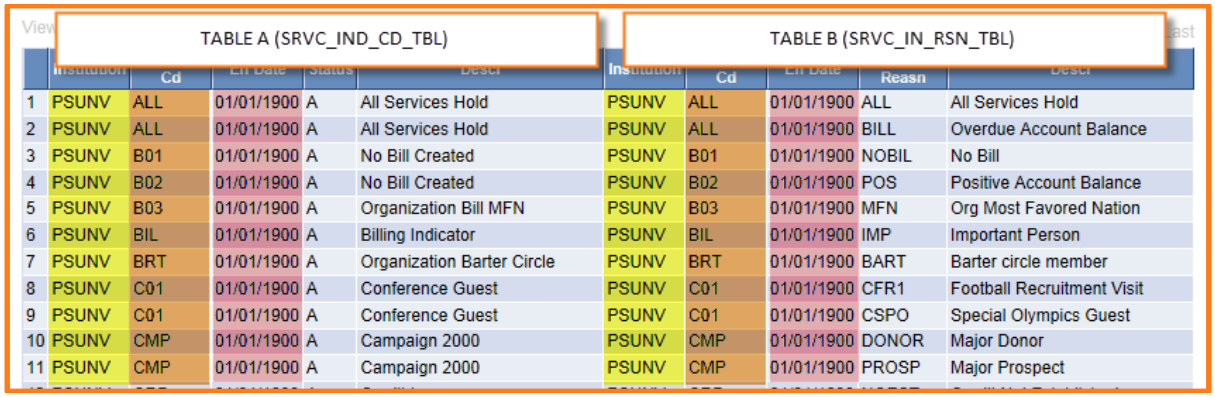


TABLE A (SRVC_IND_CD_TBL)

	Institution	Cd	Eff Date	Status	Descr
1	PSUNV	ALL	01/01/1900	A	All Services Hold
2	PSUNV	ALL	01/01/1900	A	All Services Hold
3	PSUNV	B01	01/01/1900	A	No Bill Created
4	PSUNV	B02	01/01/1900	A	No Bill Created
5	PSUNV	B03	01/01/1900	A	Organization Bill MFN
6	PSUNV	BIL	01/01/1900	A	Billing Indicator
7	PSUNV	BRT	01/01/1900	A	Organization Barter Circle
8	PSUNV	C01	01/01/1900	A	Conference Guest
9	PSUNV	C01	01/01/1900	A	Conference Guest
10	PSUNV	CMP	01/01/1900	A	Campaign 2000
11	PSUNV	CMP	01/01/1900	A	Campaign 2000

TABLE B (SRVC_IN_RSN_TBL)

	Institution	Cd	Eff Date	Reasn	Descr
1	PSUNV	ALL	01/01/1900	ALL	All Services Hold
2	PSUNV	ALL	01/01/1900	BILL	Overdue Account Balance
3	PSUNV	B01	01/01/1900	NOBIL	No Bill
4	PSUNV	B02	01/01/1900	POS	Positive Account Balance
5	PSUNV	B03	01/01/1900	MFN	Org Most Favored Nation
6	PSUNV	BIL	01/01/1900	IMP	Important Person
7	PSUNV	BRT	01/01/1900	BART	Barter circle member
8	PSUNV	C01	01/01/1900	CFR1	Football Recruitment Visit
9	PSUNV	C01	01/01/1900	CSPO	Special Olympics Guest
10	PSUNV	CMP	01/01/1900	DONOR	Major Donor
11	PSUNV	CMP	01/01/1900	PROSP	Major Prospect

Notice the Institution, Srv Ind Cd and Eff Date fields are the same; the Descr fields are different because the first Descr field is for codes (A. table) and the second Descr field is for reason (B. table).

- **Click Query** tab
- **Remove** fields from **SRVC_IN_RSN_TBL**:
  - **INSTITUTION**
  - **SRVC_IND_CD**
  - **EFFDT**
- **Save** the query.
- **Click Run** tab

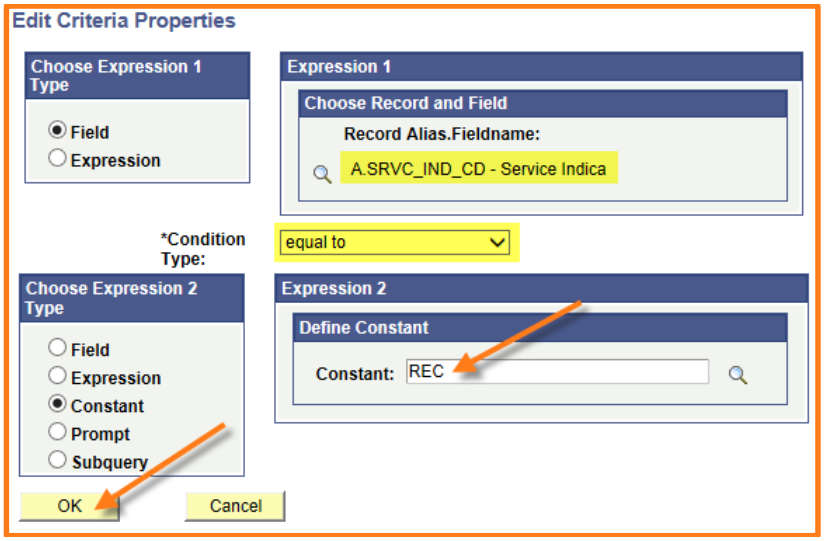
View All | Rerun Query | Download to Excel | Download to XML — First 1-73 of 73 Last

	Institution	Srv Ind Cd	Eff Date	Status	Descr	Srvc Reasn	Descr
1	PSUNV	ALL	01/01/1900	A	All Services Hold	ALL	All Services Hold
2	PSUNV	ALL	01/01/1900	A	All Services Hold	BILL	Overdue Account Balance
3	PSUNV	B01	01/01/1900	A	No Bill Created	NOBIL	No Bill
4	PSUNV	B02	01/01/1900	A	No Bill Created	POS	Positive Account Balance
5	PSUNV	B03	01/01/1900	A	Organization Bill MFN	MFN	Org Most Favored Nation
6	PSUNV	BIL	01/01/1900	A	Billing Indicator	IMP	Important Person
7	PSUNV	BRT	01/01/1900	A	Organization Barter Circle	BART	Barter circle member
8	PSUNV	C01	01/01/1900	A	Conference Guest	CFR1	Football Recruitment Visit
9	PSUNV	C01	01/01/1900	A	Conference Guest	CSPO	Special Olympics Guest
10	PSUNV	CMP	01/01/1900	A	Campaign 2000	DONOR	Major Donor
11	PSUNV	CMP	01/01/1900	A	Campaign 2000	PROSP	Major Prospect

The results are the same but the redundant fields are gone.

Let's limit the results to only the REC (Recreation Services) codes and reasons.

- **Click** the **Fields** tab
- **Click**  in the **SRVC_IND_CD** row
- **Enter "REC"** in for the Constant
- **Click OK** to finish the new criterion

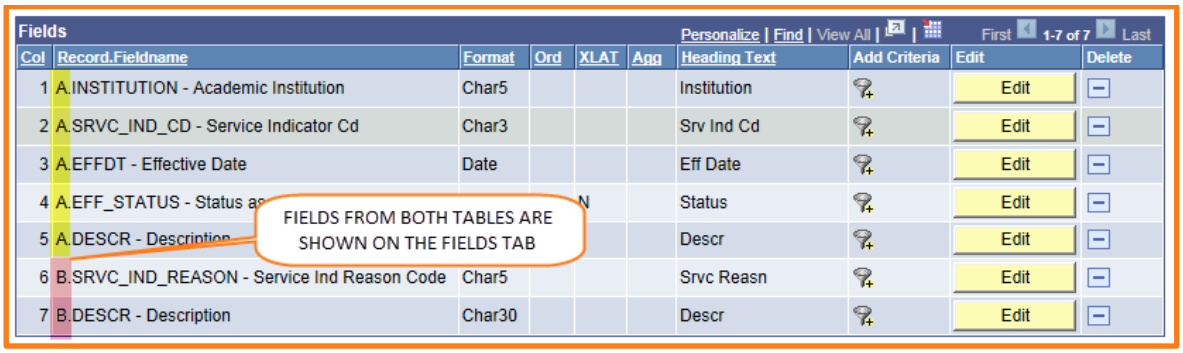


- **Save** the query
- **Click** the **Run** tab

View All | Rerun Query | Download to Excel | Download to XML — First 1-3 of 3 Last

	Institution	Srv Ind Cd	Eff Date	Status	Descr	Srvc Reasn	Descr
1	PSUNV	REC	01/01/1900	A	Recreation Services	BEHV	Behavior Complaints
2	PSUNV	REC	01/01/1900	A	Recreation Services	CHRG	Outstanding Charges
3	PSUNV	REC	01/01/1900	A	Recreation Services	NEW	New student

Criteria can still be added even when there are multiple records in a query. What about sorting fields and rows? Yes, fields can still be reordered/sorted using the Fields tab. The fields from both A. record and B. record appear on the Fields page.



Let's sort the list by the description of the reason.

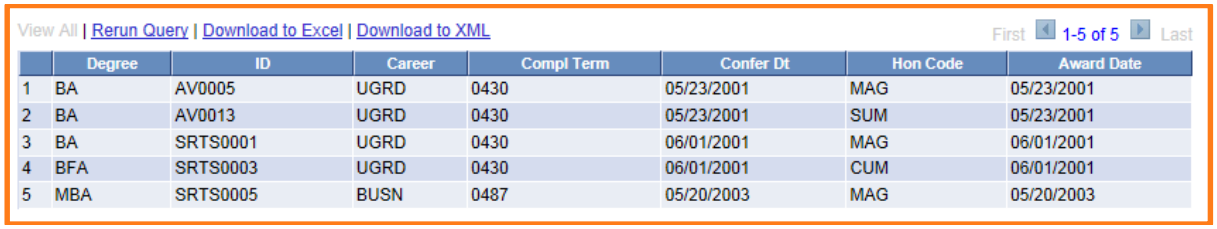
- **Click Fields** tab
- **Click** the **Reorder / Sort** button
- **Set New Order By** to **1** for **B.DESCR**. (A is the *alias* for SRVC_IND_CD_TBL and B is the alias for SRVC_IN_RSN_TBL. Since we want the description of the reason, we choose B.DESCR instead of A.DESCR.)
- **Click OK**
- **Save** the query
- **Click Run** tab

	Institution	Srv Ind Cd	Eff Date	Status	Descr	Srvc Reasn	Descr
1	PSUNV	REC	01/01/1900	A	Recreation Services	BEHV	Behavior Complaints
2	PSUNV	REC	01/01/1900	A	Recreation Services	NEW	New student
3	PSUNV	REC	01/01/1900	A	Recreation Services	CHRG	Outstanding Charges

Question 11

Obtain a list of students who earned degrees with honors at PeopleSoft University using the ACAD_DEGR record. Show the employee ID, degree, academic career, completion term, and degree conferred date. Join the record with ACAD_DEGR_HONS and show the honors code and honors award date. Order the results by degree first and ascending and EMPLID second. Also display the degree as the left-most column.

- Use the **ACAD_DEGR** record
- Show Fields for:
  - **EMPLID**
  - **DEGREE**
  - **ACAD_CAREER**
  - **COMPLETION_TERM**
  - **DEGR_CONFER_DT**
- Join with **ACAD_DEGR_HONS**
- Show Fields:
  - **HONORS_CODE**
  - **HONORS_AWARD_DT**
- Set criteria that **INSTITUTION** is "PSUNV"
- Show the results *sorted* first by Degree and next by EMPLID, both in ascending order
- Display the results with Degree as the first column and EMPLID as the second column
- Save and save often. Save as **TRNG_QM##_Q11**



View All | Rerun Query | Download to Excel | Download to XML First 1-5 of 5 Last

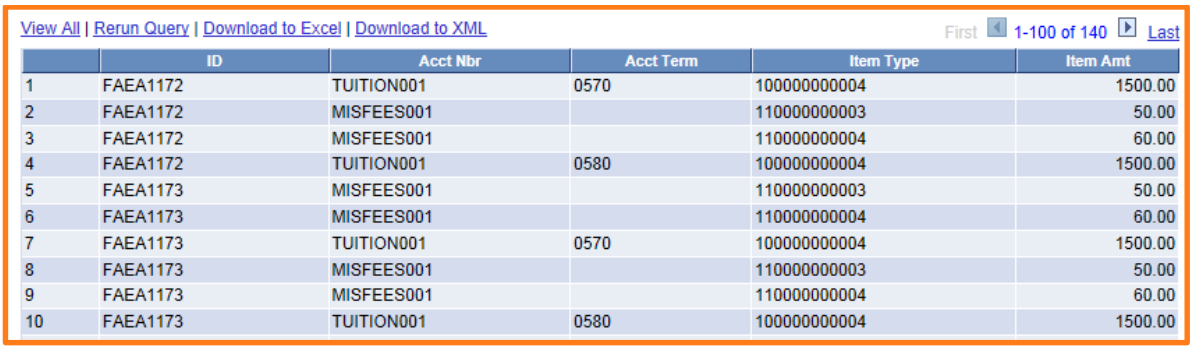
	Degree	ID	Career	Compl Term	Confer Dt	Hon Code	Award Date
1	BA	AV0005	UGRD	0430	05/23/2001	MAG	05/23/2001
2	BA	AV0013	UGRD	0430	05/23/2001	SUM	05/23/2001
3	BA	SRTS0001	UGRD	0430	06/01/2001	MAG	06/01/2001
4	BFA	SRTS0003	UGRD	0430	06/01/2001	CUM	06/01/2001
5	MBA	SRTS0005	BUSN	0487	05/20/2003	MAG	05/20/2003

Joins – Related Record

This type of join is used to get information from a prompt table (sometimes called setup table). In a relational database, duplication of information is generally avoided. For instance, data on academic program is stored in one row of ACAD_PROG_TBL, with a unique identifier (the combination of INSTITUTION and ACAD_PROG) used to refer to it. Other records that make use of academic programs will have INSTITUTION and ACAD_PROG fields but not the description, academic group, and grading scheme fields. INSTITUTION and ACAD_PROG are the "key" fields used to get data from the other table.

Let's get a list of Student Financials items – credits and charges – from the 2007 academic year for PeopleSoft University. When displaying these items, to help the user of the query, show the description of each item type alongside the item type code.

- Start a new query
- Use the record **ITEM_SF**
- **Show** the following fields:
  - **COMMON_ID**
  - **ACCOUNT_NBR**
  - **ACCOUNT_TERM**
  - **ITEM_TYPE**
  - **ITEM_AMT**
- **Add** the following criteria:
  - **BUSINESS_UNIT** is "PSUNV"
  - **ACAD_YEAR** is 2007
- Save and save often. Save as **TRNG_QM##_E12**.
- **Run** the query.

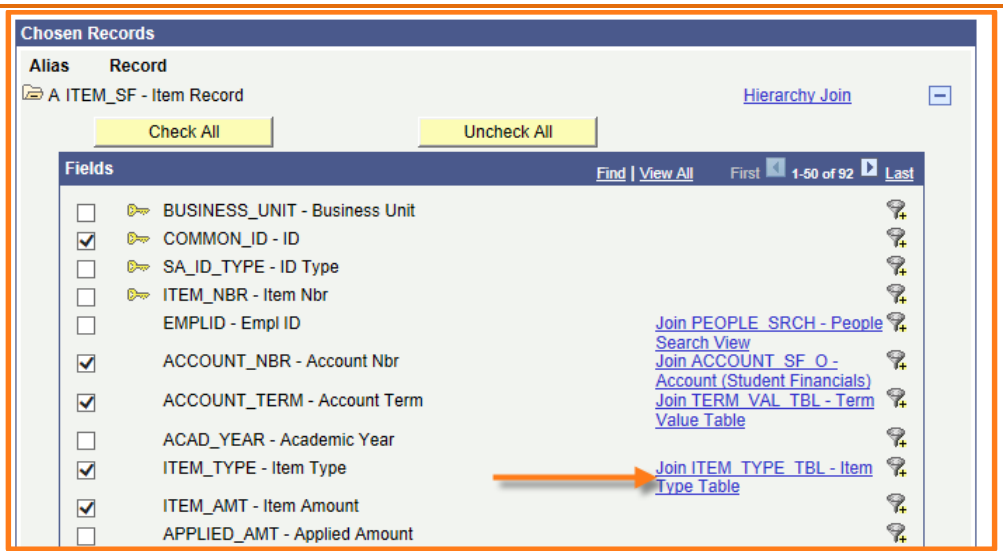


View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 140 Last

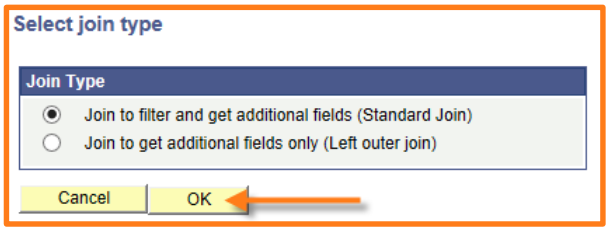
	ID	Acct Nbr	Acct Term	Item Type	Item Amt
1	FAEA1172	TUITION001	0570	100000000004	1500.00
2	FAEA1172	MISFEES001		110000000003	50.00
3	FAEA1172	MISFEES001		110000000004	60.00
4	FAEA1172	TUITION001	0580	100000000004	1500.00
5	FAEA1173	MISFEES001		110000000003	50.00
6	FAEA1173	MISFEES001		110000000004	60.00
7	FAEA1173	TUITION001	0570	100000000004	1500.00
8	FAEA1173	MISFEES001		110000000003	50.00
9	FAEA1173	MISFEES001		110000000004	60.00
10	FAEA1173	TUITION001	0580	100000000004	1500.00

The query returns the items from the 2007 academic year. Now we need to add the item type descriptions to the results. We will join a table containing descriptions of the item types so that the results can show the descriptions.

- **Click** the **Query** tab
- **Click** the "**Join ITEM_TYPE_TBL – Item Type Table**" link

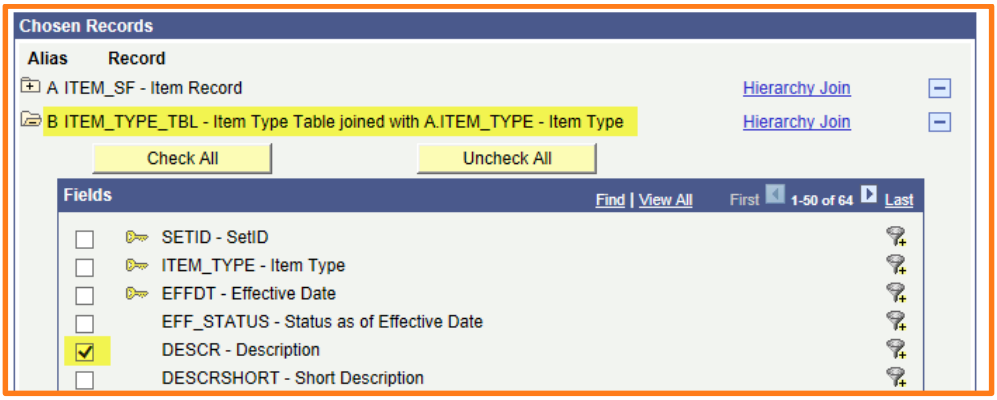


The "Select join type" dialog box appears. There are two options: "Join to filter and get additional fields (Standard Join)" and "Join to get additional fields only (Left outer join)". We want to filter and get additional fields (the system defaults to a Standard Join).

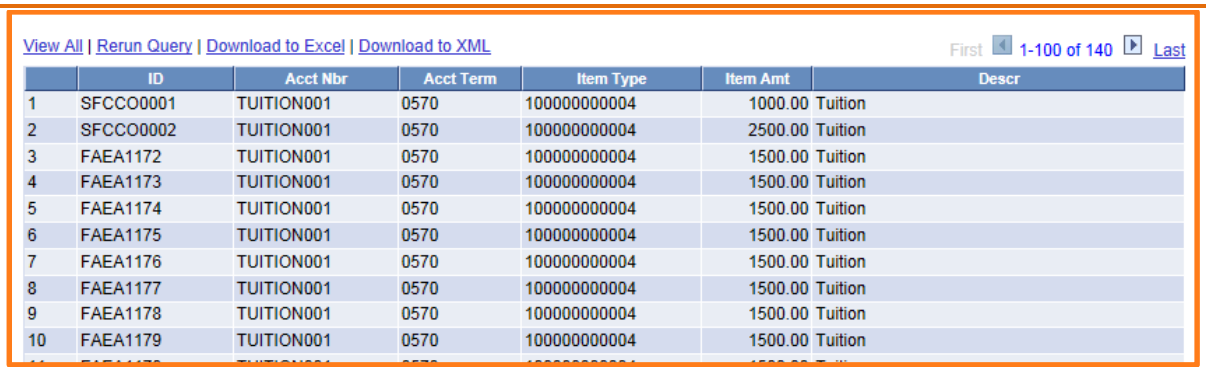


- **Click OK** (note that the OK button is the rightmost button instead of the leftmost)

Next, choose the fields from the joined table to display in the results. Select the **DESCR** field from **B.ITEM_TYPE_TBL- Item Type Table joined with A.ITEM_TYPE – Item Type**.



- **Add** a criterion that **SETID** is equal to "PSUNV"
- **Save** the query
- **Click** the **Run** tab

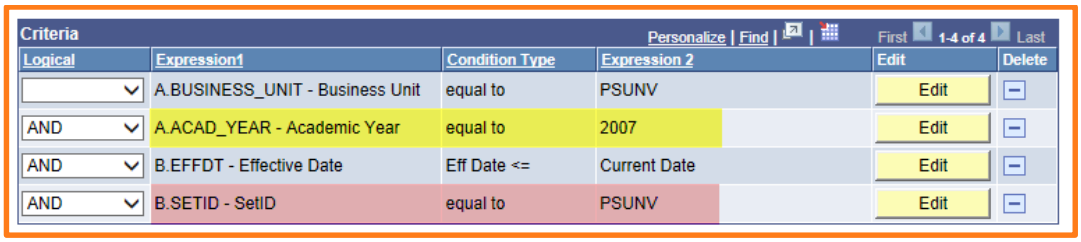


View All | Rerun Query | Download to Excel | Download to XML

First 1-100 of 140 Last

	ID	Acct Nbr	Acct Term	Item Type	Item Amt	Descr
1	SFCCO0001	TUITION001	0570	100000000004	1000.00	Tuition
2	SFCCO0002	TUITION001	0570	100000000004	2500.00	Tuition
3	FAEA1172	TUITION001	0570	100000000004	1500.00	Tuition
4	FAEA1173	TUITION001	0570	100000000004	1500.00	Tuition
5	FAEA1174	TUITION001	0570	100000000004	1500.00	Tuition
6	FAEA1175	TUITION001	0570	100000000004	1500.00	Tuition
7	FAEA1176	TUITION001	0570	100000000004	1500.00	Tuition
8	FAEA1177	TUITION001	0570	100000000004	1500.00	Tuition
9	FAEA1178	TUITION001	0570	100000000004	1500.00	Tuition
10	FAEA1179	TUITION001	0570	100000000004	1500.00	Tuition

Observe that you were able to apply criteria to each table that was joined together. Click the **Criteria** tab and see that there are criteria applied to both table A (the alias for ITEM_SF) and B (the alias for ITEM_TYPE_TBL).



Criteria

Personalize | Find | First 1-4 of 4 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.BUSINESS_UNIT - Business Unit	equal to	PSUNV	Edit	
AND	A.ACAD_YEAR - Academic Year	equal to	2007	Edit	
AND	B.EFFDT - Effective Date	Eff Date <=	Current Date	Edit	
AND	B.SETID - SetID	equal to	PSUNV	Edit	

Question 12

Create a query on academic programs, showing the institution, program (both abbreviated and description), effective date, career, and group for each program at PeopleSoft University. In addition, obtain the description of each group from the academic groups record. Finally, change the headings of the two description columns to “Program Description” and “Group Description.”

- Use the **ACAD_PROG_TBL** record
- Show Fields for
  - **INSTITUTION**
  - **ACAD_PROG**
  - **EFFDT**
  - **DESCR**
  - **ACAD_CAREER**
  - **ACAD_GROUP**
- Limit results to rows where **INSTITUTION** is equal to “PSUNV”
- Join with **ACAD_GROUP_TBL**
  - Show field for **Description**
- Change the headings of the academic program description to “Program Description” and academic group description to “Group Description”
- Save and save often. Save as **TRNG_QM##_Q12**

View All | Rerun Query | Download to Excel | Download to XML First 1-33 of 33 Last

	Institution	Acad Prog	Eff Date	Program Description	Career	Acad Group	Group Description
1	PSUNV	AA	01/01/1900	Associate of Arts	UGRD	PSUNV	PeopleSoft University
2	PSUNV	AS	01/01/1900	Associate of Science	UGRD	PSUNV	PeopleSoft University
3	PSUNV	CEOEE	01/01/1900	Continuing Education OEE	UGRD	LBART	College of Liberal Arts
4	PSUNV	CERT	01/01/1900	Cont Ed - Certificate Program	CNED	CNTED	Continuing Education
5	PSUNV	CNTGQ	01/01/1900	Cont Ed - GR Quarter Calendar	CNED	CNTED	Continuing Education
6	PSUNV	CNTGR	01/01/1900	Continuing Education Graduate	CNED	CNTED	Continuing Education
7	PSUNV	CNTUG	01/01/1900	Continuing Education Undergrad	CNED	CNTED	Continuing Education
8	PSUNV	CNTUQ	01/01/1900	Cont Ed UG - Quarter Calendar	CNED	CNTED	Continuing Education
9	PSUNV	FAU	01/01/1900	Fine Arts Undergraduate	UGRD	FA	College of Fine Arts
10	PSUNV	FQU	01/01/1900	Fine Arts UG Quarter Calendar	UGRD	FA	College of Fine Arts

Joins – Any Record

On some occasions you will want to get information from records that do not have a parent/child relationship and where one is not defined as the prompt table for another. Query Manager allows you to build a query such that any fields are joined together, but this must be done manually and with care.

Fields to be joined together will likely have the same name or similar names and must have data that represents the same thing. It would make no sense to join EMPLID in one table to STRM in another, as these are completely different; either no row will be returned because the fields can never match or the ones that are returned are mistakes because the rows aren't really related.

Let's get a list of financial aid awards for the 2007 aid year in which the accepted amount is greater than $3500. Include the description of the award by joining to the item type table.

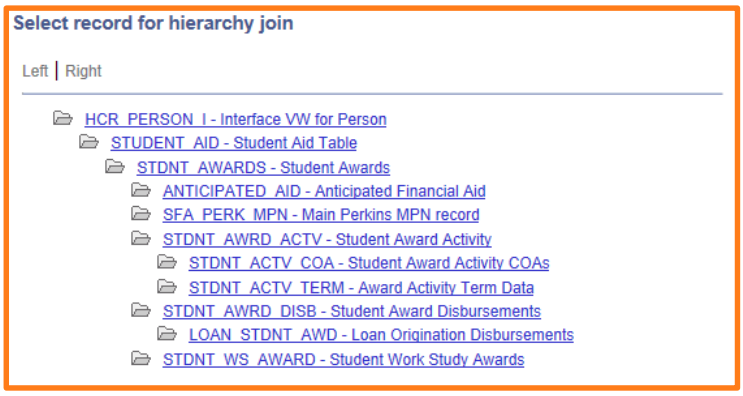
- Start a query on **STDNT_AWARDS**
- **Select** fields:
  - **EMPLID**
  - **INSTITUTION**
  - **AID_YEAR**
  - **ITEM_TYPE**
  - **ACAD_CAREER**
- Add criteria:
  - **INSTITUTION** is equal to **PSUNV**
  - **AID_YEAR** is equal to **2007**
  - **ACCEPT_AMOUNT** is greater than $3500.
- Save and save often. Save the query as **TRNG_QM##_E13**.
- **Click** the **Run** tab to test what you have built so far.

View All | Rerun Query | Download to Excel | Download to XML First 1-100 of 704 Last

	ID	Institution	Aid Yr	Item Type	Career
1	FAD0047	PSUNV	2007	900000000312	UGRD
2	FAD0047	PSUNV	2007	900000000351	UGRD
3	FAD0048	PSUNV	2007	900000000312	UGRD
4	FAD0048	PSUNV	2007	900000000351	UGRD
5	FAD0018	PSUNV	2007	900000000351	UGRD
6	FAD0019	PSUNV	2007	900000000384	UGRD
7	FAD0029	PSUNV	2007	900000000351	UGRD

The results show the awards in which more than $3500 was accepted, but it is not clear what those awards are. To get the description of the item type, we will join STDNT_AWARDS to ITEM_TYPE_TBL.

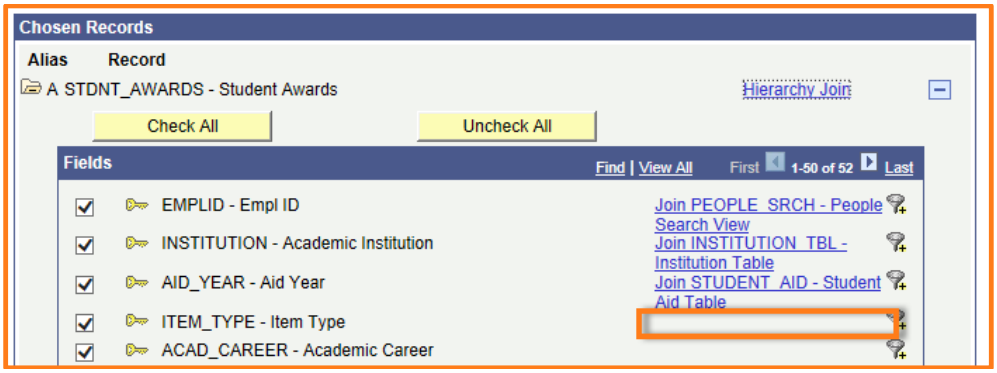
- **Click** the **Query** tab.
- **Click** the **Hierarchy Join** link.



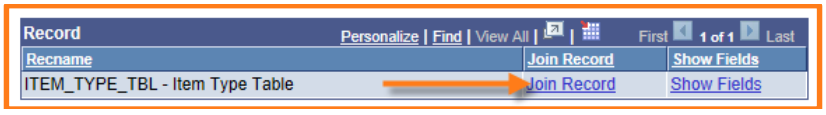
There is NO parent/child relationship between awards and item types. This is because awards don't own item types (the types are independent and used for things other than awards) and item types don't extend information for an award.

- **Click** the **Cancel** button

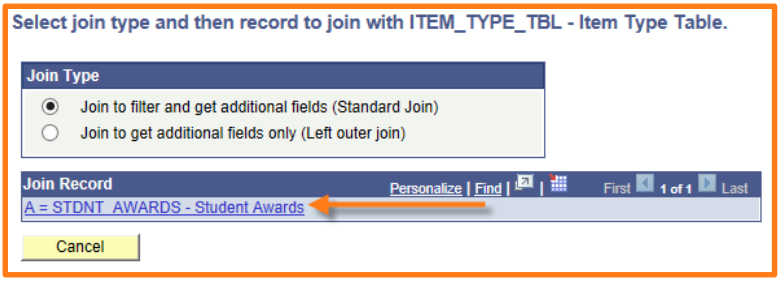
Also observe that there is no Related Record join for ITEM_TYPE.



- **Click** the **Records** tab
- **Search** for **ITEM_TYPE_TBL**
- **Click** the **Join Record** link in the **ITEM_TYPE_TBL** row



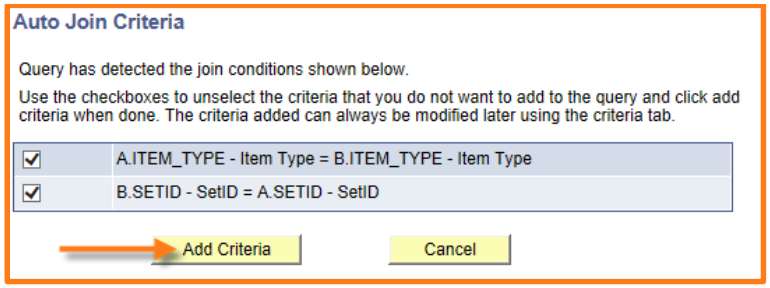
A "select join type and then record to join" page appears. You are asked to first choose which type of join to use. In this case you will select the default option, "Join to filter and get additional fields". This is known as a *standard join* or *inner join*. (Note that the Hierarchy Join and Related Record joins from previous sections are also inner joins.) You then click the link corresponding to the record to which you want to join the new record. In this case you are joining ITEM_TYPE_TBL to STDNT_AWARDS.



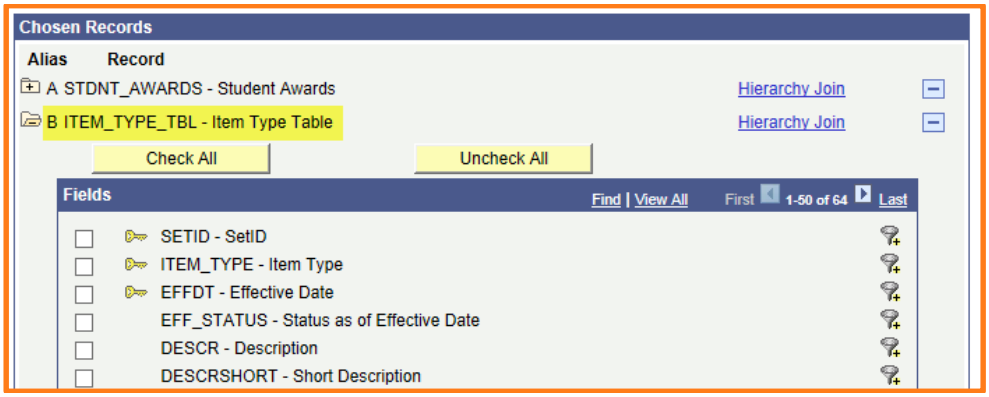
- **Click A = STDNT_AWARDS – Student Awards**

Next, Query Manager tries to detect fields in common on which the join will be built, usually on key fields and fields with the same name. These automatic joins are listed on the Auto Join Criteria page. For each automatic join that you accept, a criterion will be built. You can remove these later if you find they are unnecessary and add some later in the Criteria tab. Query Manager won't always be correct – for instance, it will miss cases in which the join needs to be on two fields with different names – but it usually provides a good start.

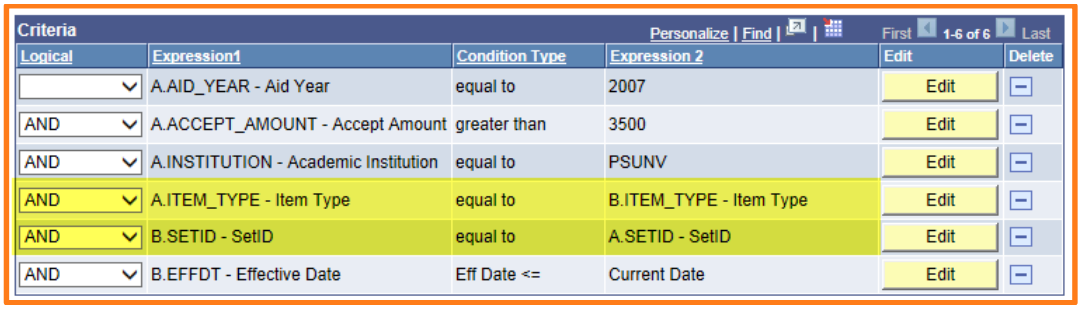
For this exercise, leave both criteria selected and click the Add Criteria button.



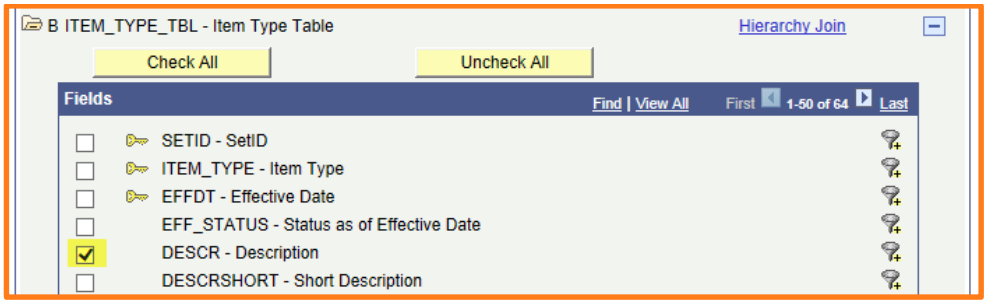
The Query page appears and ITEM_TYPE_TBL has been added to the chosen records.



To see that the join criteria were automatically created, click the **Criteria** tab. The join criteria on ITEM_TYPE and SETID, selected on the Auto Join Criteria page, were added to the query.



Go back to the **Query** tab and select the **DESCR** field for display.



- **Save** your query
- **Click** the **Run** tab. The results now include the description of each item type.

View All | Rerun Query | Download to Excel | Download to XML
First 1-100 of 704 Last

	ID	Institution	Aid Yr	Item Type	Career	Descr
1	FAD0047	PSUNV	2007	900000000312	UGRD	DL UNSUB LOAN
2	FAD0047	PSUNV	2007	900000000351	UGRD	TERI Loan
3	FAD0048	PSUNV	2007	900000000312	UGRD	DL UNSUB LOAN
4	FAD0048	PSUNV	2007	900000000351	UGRD	TERI Loan
5	FAD0018	PSUNV	2007	900000000351	UGRD	TERI Loan
6	FAD0019	PSUNV	2007	900000000384	UGRD	TERI Loan - Qtr
7	FAD0029	PSUNV	2007	900000000351	UGRD	TERI Loan
8	FAD0030	PSUNV	2007	900000000383	UGRD	DL QTR Unsub Stafford Loan
9	FAD0030	PSUNV	2007	900000000384	UGRD	TERI Loan - Qtr
10	FAD0007	PSUNV	2007	900000000312	UGRD	DL UNSUB LOAN

Question 13

Let's find active employees in a department that the user will select. Create a query using the Job table. Display the employee ID, employee record, job code, and position number. Create prompts for enabling the user to select a Set ID for the department and to select the department itself. Join to record to POSITION_DATA to include a position description.

- Start a query on the **JOB** table
- Select fields:
  - **EMPLID**
  - **EMPL_RCD**
  - **JOBCODE**
  - **POSITION_NBR**
- Add a criterion that **SETID_DEPT** is equal to a value entered through a prompt. Set the **Edit Type** of the prompt to "Prompt Table" and the **Prompt Table** to **SETID_TBL**.
- Add a criterion that **DEPTID** is equal to a value entered through a prompt. Set the **Edit Type** of the prompt to "Prompt Table" and the **Prompt Table** to **SET_DEPT_VW**.
- Add these criteria:
  - **HR_STATUS** is equal to "A"
  - **JOB_INDICATOR** is equal to "P"
- Join **POSITION_DATA**
  - Select field **DESCR** (agree to the automatic join B.POSITION_NBR = A.POSITION_NBR)
- Save and save often. Save as **TRNG_QM##_Q13**
- Run the query, entering these values for the prompts:
  - Dept. SetID: "PSUNV", DeptID: "ADMISSIONS"
  - Dept. SetID: "SHARE", DeptID: "10500"

This space intentionally left blank.

Dept. SetID = PSUNV,DeptID=ADMISSIONS

View All | Rerun Query | Download to Excel | Download to XML First 1-13 of 13 Last

	ID	Empl Rcd#	Job Code	Position	Descr
1	SEV4019	0	PS0002	PS000001	Teacher's Assistant
2	SEV4020	0	PS0003	PS000002	Assistant Professor
3	SEV4021	0	PS0003	PS000002	Assistant Professor
4	SEV4022	0	PS0004	PS000003	Professor
5	SEVE0506	0	PS0004	PS000003	Professor
6	SEVE0507	0	PS0003	PS000002	Assistant Professor
7	SEVE0508	0	PS0002	PS000001	Teacher's Assistant

Dept. SetID = SHARE,DeptID=10500

View All | Rerun Query | Download to Excel | Download to XML First 1-6 of 6 Last

	ID	Empl Rcd#	Job Code	Position	Descr
1	KC0018	0	820005	19000103	Benefits Specialist
2	KU0031	0	820090	19000090	Pension Specialist
3	KU0058	0	820090	19000090	Pension Specialist
4	KU0104	0	820090	19000090	Pension Specialist
5	KUZ009	0	600035	19360010	Manager-Compensation/Benefits
6	KUZ012	0	140065	19360011	Human Resource Analyst

Recall from the note on Question 7 that when you want to offer a prompt on a field that features validation – checking that the value the user enters is one known to the system – you have to include prompts on every field the key. In question 13 we could provide a prompt on DEPTID without also providing a prompt on SETID_DEPT, but we would then not be able to validate what the user enters nor provide a list of allowed values. It is a helpful convenience to the users of your queries to provide prompts that also validate entries and give a list of valid selections.

Notice that we had to change the default prompt table for the DEPTID field from DEPT_TBL to SET_DEPT_VW. This is because Query Manager requires that the field names in the table match those in the prompt table. The JOB table has a field named SETID_DEPT for the Set ID of the department but DEPT_TBL uses the field name SETID. (This is because JOB contains several different Set ID fields – SETID_DEPT, SETID_JOBCODE, SETID_LOCATION, etc.) SET_DEPT_VW is a view – a pre-built query that acts like a table – that retrieves data from DEPT_TBL but replaces the field name SETID with SETID_DEPT. This allows SET_DEPT_VW to be used as a prompt table for DEPTID in JOB.

Outer Joins

When you're joining two tables, the first table (the one on the left) may have rows that don't have matching counterparts in the second table (the one on the right). The table on the right may have rows that don't have matching counter parts in the table on the left. If you perform an inner (standard) join on those tables, all the unmatched rows are excluded from the output. Outer joins do not exclude the unmatched rows. Outer joins come in three types: the left outer join, the right outer join, and the full outer join. Query Manager supports only the left outer join.

Consider these excerpts from the countries table (COUNTRY_TBL) and states table (STATE_TBL).

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

They will be joined together so the countries and the states and provinces within them can be listed in query results. They will be joined on the COUNTRY field since this uniquely identifies a country. For instance, a row in STATE_TBL with a COUNTRY equal to "USA" represents a state in the United States.

When a left outer join takes place, every row in the left table is compared against every row in the right table. When the values in the join field or fields match, the rows are combined and added as one row to the results. When the row of the left table has no matches with any row in the right table, a row is added to the results containing all fields from the left table and empty values where values from the right table would normally appear.

In this example, the first comparison is of the Canada row in COUNTRY_TBL and the Alberta row in STATE_TBL. Since the join field COUNTRY has the same value CAN in both tables, a row combining the Canada row with the Alberta row is added to the results.

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

match

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta

As processing continues, the Canada row from COUNTRY_TBL is compared against the British Columbia, Ontario, and Quebec rows from STATE_TBL. Since the values of the join field COUNTRY match in all cases, new rows are added to the results for each province.

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

match

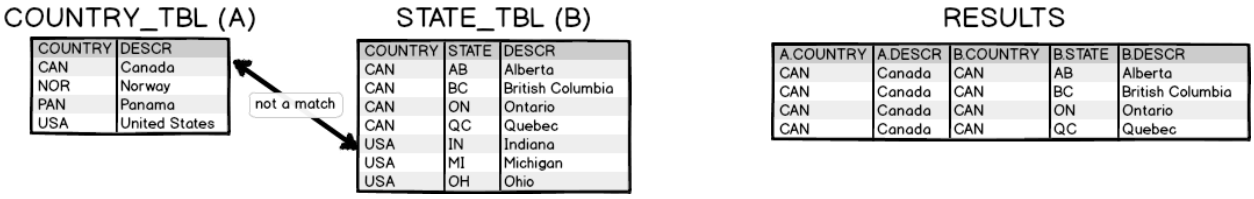
STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec

The next comparison is between the Canada row of COUNTRY_TBL and the Indiana row of STATE_TBL. Since the COUNTRY values of CAN and USA do not match, a new row is not added to the results.



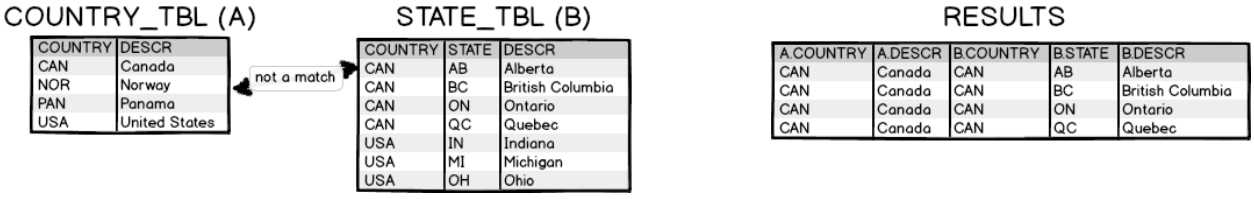
COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec

Similarly, no rows are added when the Canada row is compared against the Michigan and Ohio rows.

Next, the Norway row in COUNTRY_TBL is compared to the Alberta row of STATE_TBL. Since the COUNTRY values of NOR and CAN do not match, nothing is added to the results.

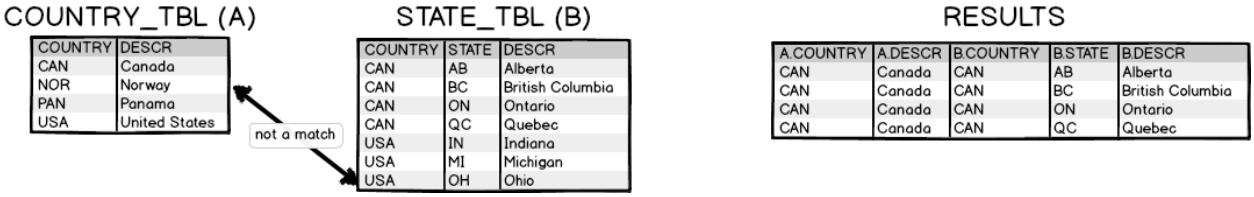


COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec

Processing continues with each of the rows in STATE_TBL compared to the Norway row of COUNTRY_TBL. When processing reaches a comparison of the Ohio row, nothing has been added to the results for Norway.

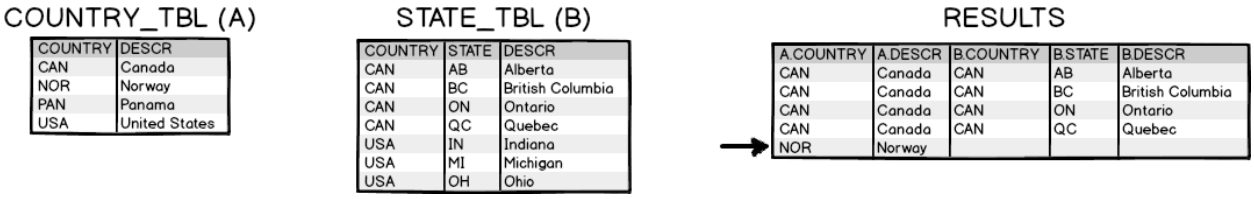


COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec

Since there have been no matches for Norway and this is a left outer join, a row is added to the results containing all fields from the left table (COUNTRY_TBL) and empty values in place of all fields from the right table (STATE_TBL). In this case, A.COUNTRY and A.DESCR are filled with NOR and Norway while B.COUNTRY, B.STATE, and B.DESCR are filled with empty values.



COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec
NOR	Norway			

Processing follows with the Panama row of COUNTRY_TBL being matched against each row in STATE_TBL. Since there are no rows in STATE_TBL with a COUNTRY value of PAN, nothing will be added to the results.

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

not a match

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec
NOR	Norway			

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

not a match

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec
NOR	Norway			

After the last row of STATE_TBL is checked and no matches have been found, a row is added to results containing the fields of the Panama row of COUNTRY_TBL and empty values filling in the fields of STATE_TBL.

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec
NOR	Norway			
PAN	Panama			

Finally, the United States row of COUNTRY_TBL is compared to each row of STATE_TBL. The rows for Indiana, Michigan, and Ohio have a matching COUNTRY value of USA, so rows will be added to the results for each of these states. When the outer join is complete, the results will be the following:

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec
NOR	Norway			
PAN	Panama			
USA	United States	USA	IN	Indiana
USA	United States	USA	MI	Michigan
USA	United States	USA	OH	Ohio

Below is a comparison of the results of using an inner join on COUNTRY_TBL and STATE_TBL and an outer join on these two tables.

INNER JOIN RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec
USA	United States	USA	IN	Indiana
USA	United States	USA	MI	Michigan
USA	United States	USA	OH	Ohio

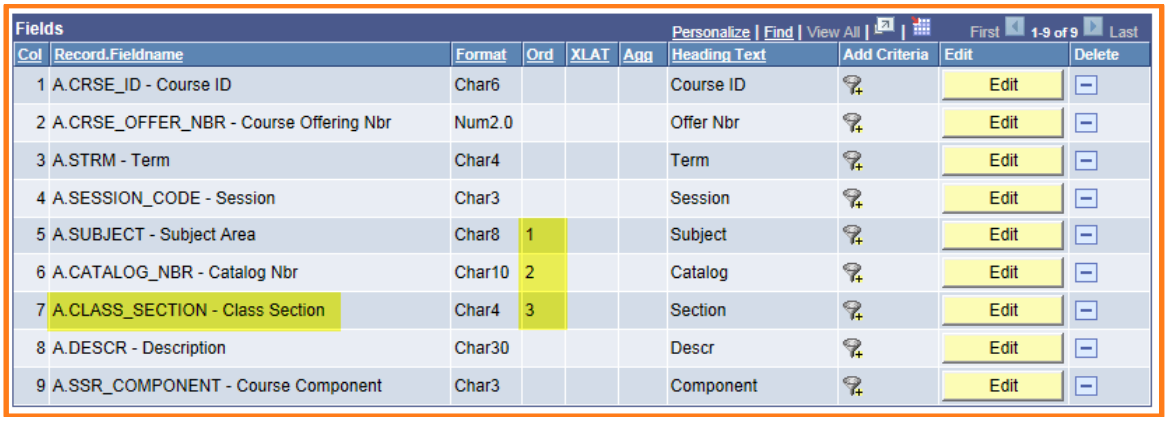
OUTER JOIN RESULTS

A.COUNTRY	A.DESCR	B.COUNTRY	B.STATE	B.DESCR
CAN	Canada	CAN	AB	Alberta
CAN	Canada	CAN	BC	British Columbia
CAN	Canada	CAN	ON	Ontario
CAN	Canada	CAN	QC	Quebec
NOR	Norway			
PAN	Panama			
USA	United States	USA	IN	Indiana
USA	United States	USA	MI	Michigan
USA	United States	USA	OH	Ohio

Now let us build a query that utilizes an outer join. We will get a list of History classes from the Fall 2006 semester at PeopleSoft University. In cases in which there is an exam date and time recorded for the class, display it. If there is no exam date and time make sure the class is still shown.

Class data is in CLASS_TBL and exam data is in CLASS_EXAM. We will need to join CLASS_TBL and CLASS_EXAM, but will use an outer join because if there is no row in CLASS_EXAM matching the row in CLASS_TBL (which will be common because if a class has multiple components, such as lecture, lab, and recitation, only one will have an exam), we still want the data from CLASS_TBL to be in the results.

- Start a query on **CLASS_TBL**
- **Select** fields:
  - **CRSE_ID**
  - **CRSE_OFFER_NBR**
  - **STRM**
  - **SESSION_CODE**
  - **CLASS_SECTION**
  - **SUBJECT**
  - **CATALOG_NBR**
  - **DESCR**
  - **SSR_COMPONENT**
- **Add criteria**:
  - **STRM** equal to **0570**
  - **SUBJECT** equal to **HISTORY**
- Go to the **Fields** tab
- **Click** the **Reorder/Sort** button
- **Set** the **New Order By** of:
  - **SUBJECT** to **1**
  - **CATALOG_NBR** to **2**
  - **CLASS_SECTION** to **3**
- **Set** the **New Column** value of **CLASS_SECTION** to be immediately after **CATALOG_NBR**
- **Click** the **OK** button



Fields — Personalize | Find | View All | First 1-9 of 9 Last

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.CRSE_ID - Course ID	Char6				Course ID		Edit	
2	A.CRSE_OFFER_NBR - Course Offering Nbr	Num2.0				Offer Nbr		Edit	
3	A.STRM - Term	Char4				Term		Edit	
4	A.SESSION_CODE - Session	Char3				Session		Edit	
5	A.SUBJECT - Subject Area	Char8	1			Subject		Edit	
6	A.CATALOG_NBR - Catalog Nbr	Char10	2			Catalog		Edit	
7	A.CLASS_SECTION - Class Section	Char4	3			Section		Edit	
8	A.DESCR - Description	Char30				Descr		Edit	
9	A.SSR_COMPONENT - Course Component	Char3				Component		Edit	

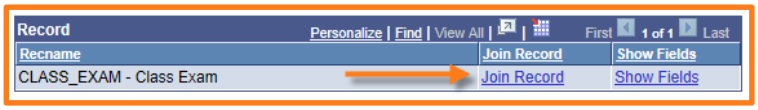
- Save and save often. Save as **TRNG_QM##_E14**.
- **Run** the query to test what you have built so far.

View All | Rerun Query | Download to Excel | Download to XML First 1-52 of 52 Last

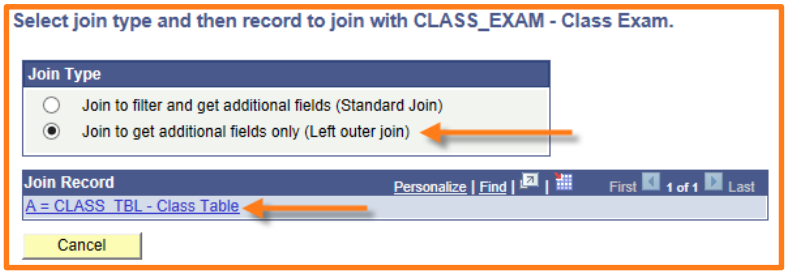
	Course ID	Offer Nbr	Term	Session	Subject	Catalog	Section	Descr	Component
1	007129	1	0570	1	HISTORY	100	1	Perspectives on the Present	LEC
2	007130	1	0570	1	HISTORY	101	1	History of the US	LEC
3	007228	1	0570	1	HISTORY	109	001	World History	LEC
4	007228	1	0570	1	HISTORY	109	002	World History	LEC
5	007228	1	0570	1	HISTORY	109	003	World History	LEC
6	001253	1	0570	1	HISTORY	120	1	American History	LEC
7	007229	1	0570	1	HISTORY	121	001	United States History I	LEC
8	007229	1	0570	1	HISTORY	121	002	United States History I	LEC
9	007229	1	0570	1	HISTORY	121	003	United States History I	LEC
10	007229	1	0570	1	HISTORY	121	004	United States History I	LEC

We now have a list of History classes from the Fall 2006 semester. Next we will join these results to CLASS_EXAM to get the exam times for each class.

- **Click** the **Records** tab
- **Search for** and **select** the **CLASS_EXAM** record
- **Click** the **Join Record** link

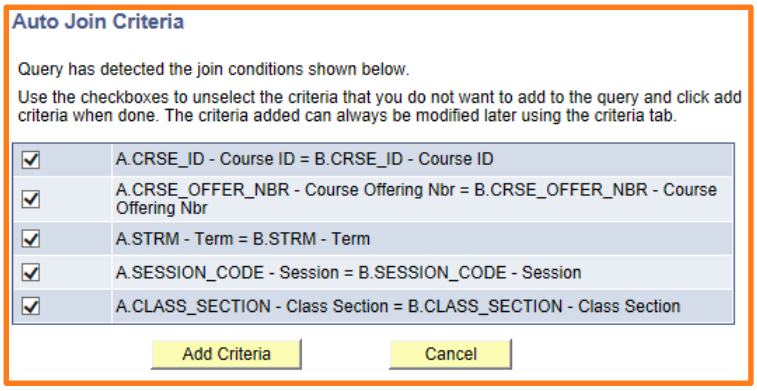


- In the "Select join type" page, **select** the **"Join to get additional fields only (Left outer join)"** option
- **Click** the **A = CLASS_TBL – Class Table** link

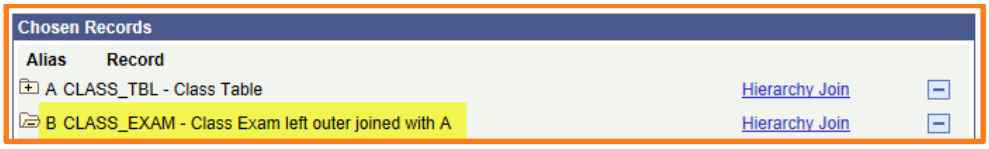


As when you did the Any Record join in the previous section, Query Manager attempts to find the proper join conditions and offers to create criteria for them.

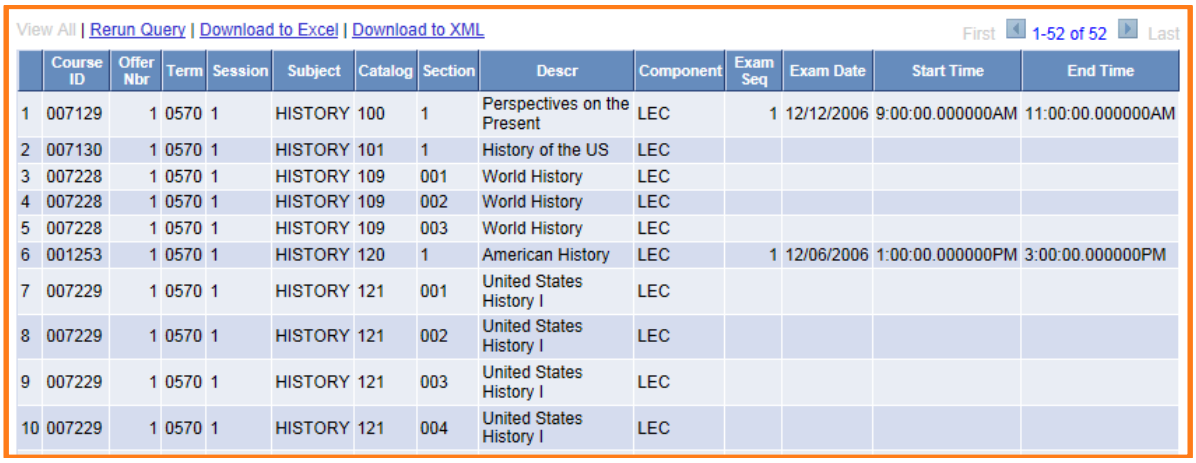
- Accept the default join conditions by leaving them all checked
- **Click** the **Add Criteria** button



The Query tab is shown. The CLASS_EXAM record is added to the query and the description indicates it has been outer joined to CLASS_TBL (referring to it by its alias of "A").



- **Select** the following fields of **CLASS_EXAM**:
  - **CLASS_EXAM_SEQ**
  - **EXAM_DT**
  - **EXAM_START_TIME**
  - **EXAM_END_TIME**
- **Save** the query
- **Click** the **Run** tab



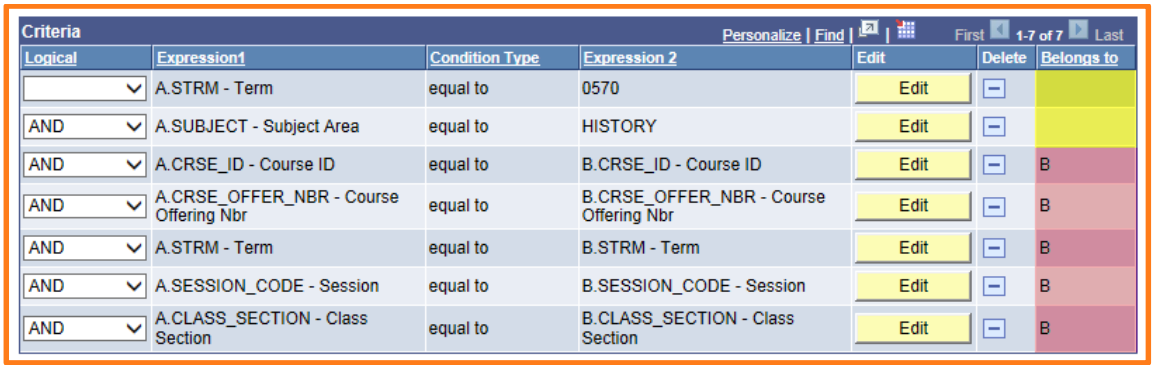
View All | Rerun Query | Download to Excel | Download to XML — First 1-52 of 52 Last

	Course ID	Offer Nbr	Term	Session	Subject	Catalog	Section	Descr	Component	Exam Seq	Exam Date	Start Time	End Time
1	007129	1	0570	1	HISTORY	100	1	Perspectives on the Present	LEC	1	12/12/2006	9:00:00.000000AM	11:00:00.000000AM
2	007130	1	0570	1	HISTORY	101	1	History of the US	LEC				
3	007228	1	0570	1	HISTORY	109	001	World History	LEC				
4	007228	1	0570	1	HISTORY	109	002	World History	LEC				
5	007228	1	0570	1	HISTORY	109	003	World History	LEC				
6	001253	1	0570	1	HISTORY	120	1	American History	LEC	1	12/06/2006	1:00:00.000000PM	3:00:00.000000PM
7	007229	1	0570	1	HISTORY	121	001	United States History I	LEC				
8	007229	1	0570	1	HISTORY	121	002	United States History I	LEC				
9	007229	1	0570	1	HISTORY	121	003	United States History I	LEC				
10	007229	1	0570	1	HISTORY	121	004	United States History I	LEC				

Now there are exams shown for classes that have exam dates and times defined in the system and the classes that do not have exams defined are still displayed.

There is an important factor to consider when adding criteria to a query that has an outer join. You must make sure that the criteria "belong" to the correct *clause* of the query. This determines whether the criteria is applied *before* the join is made (limiting the rows that could be joined) or *after* the join is made (limiting the results of the join).

Click the **Criteria** tab to show all the criteria in the query.



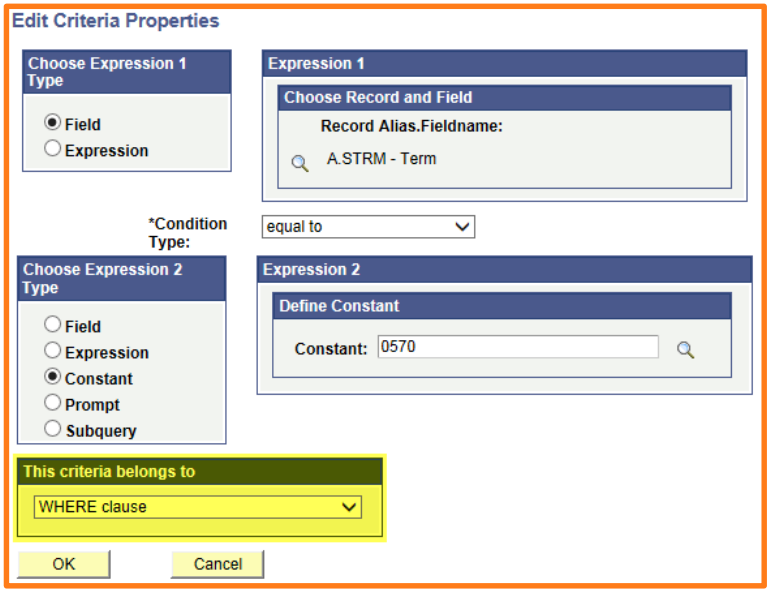
Criteria — Personalize | Find | First 1-7 of 7 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete	Belongs to
	A.STRM - Term	equal to	0570	Edit		
AND	A.SUBJECT - Subject Area	equal to	HISTORY	Edit		
AND	A.CRSE_ID - Course ID	equal to	B.CRSE_ID - Course ID	Edit		B
AND	A.CRSE_OFFER_NBR - Course Offering Nbr	equal to	B.CRSE_OFFER_NBR - Course Offering Nbr	Edit		B
AND	A.STRM - Term	equal to	B.STRM - Term	Edit		B
AND	A.SESSION_CODE - Session	equal to	B.SESSION_CODE - Session	Edit		B
AND	A.CLASS_SECTION - Class Section	equal to	B.CLASS_SECTION - Class Section	Edit		B

Since there is an outer join in the query, the **Belongs to** column is displayed. This indicates whether a criterion belongs to the main query (the value is blank), so that it is applied *after* the join, or to a specific record (the value is an alias letter – B in this case), so that it is applied *before* the join.

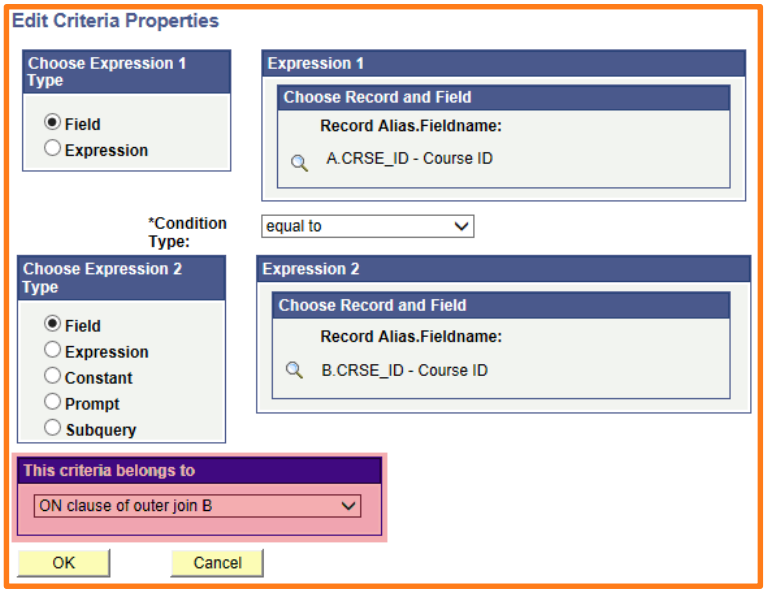
Click the **Edit** button in the row for **STRM**. In the Edit Criteria Properties page, there is a section named "This criteria belongs to". The value shown is "WHERE clause". This indicates that the criterion is applied after the join. ("WHERE clause" refers to the SQL generated by Query Manager that is sent to the database to execute the query.)

Click the **Cancel** button.



Now click the **Edit** button in the row for **CRSE_ID**. In the Edit Criteria Properties page, the section "This criteria belongs to" shows "ON clause of outer join B". This indicates that the criterion is applied to record B (CLASS_EXAM) before the join. ("ON clause" also refers to the generated SQL.)

Click the **Cancel** button.



When Query Manager added the join criteria from the Auto Join Criteria step, it set them to belong to the ON clause of outer join B because we need to limit the rows being joined to those having matching values in record A (CLASS_TBL).

The two criteria we added before the outer join – those on STRM and SUBJECT – belong to the WHERE clause because term and subject are not in CLASS_EXAM table and have no effect on whether there is a matching row in CLASS_EXAM.

What if we wanted to show only exams on particular days? You would add a criterion that belongs to the ON clause because you'd want to check only rows in CLASS_EXAM that have those days before joining. If you did it after the join, all the classes with no exam would be thrown out because the condition that EXAM_DT has the desired value would be applied to a rows with an empty EXAM_DT.

In general:

Select "ON clause of outer join" if:

- this criterion is part of the join condition (such as A.CRSE_ID is equal to B.CRSE_ID)

OR

- this criterion restricts rows that are in the record being outer joined (such as B.EXAM_DT is 5/2/2013)

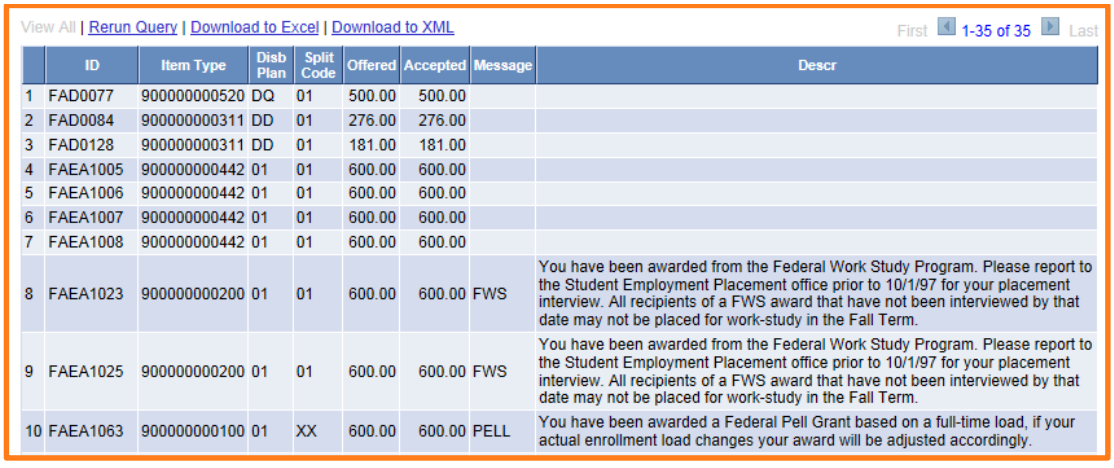
Select "WHERE clause" option if:

- this criterion is related to the main record or records in the query (such as A.STRM is equal to "2132")

Question 14

Create a query to list offers of awards of less than $750 of financial aid that were accepted by students at PeopleSoft University. Limit the results to the 2007 aid year and to undergraduates. In cases in which there is a message for the award, show the message; if there is no such message, the award must still be displayed in the results. *Order* the results by student ID and item type. Make the award message code and descriptions the rightmost two *columns* in the results.

- Use the **STDNT_AWARDS** and **AWD_MESSAGE_TBL** records
- Consider carefully whether to use an inner join or an outer join. What should happen if there is no row in **AWD_MESSAGE_TBL** matching the row in **STDNT_AWARDS**?
- Show the following fields from **STDNT_AWARDS**:
  - **EMPLID**
  - **ITEM_TYPE**
  - **DISBURSEMENT_PLAN**
  - **SPLIT_CODE**
  - **AWARD_MSG_CD**
  - **OFFER_AMOUNT**
  - **ACCEPT_AMOUNT**
- Save this query as **TRNG_QM##_Q14**
- Show the **DESCRLONG** field from **AWD_MESSAGE_TBL**
- Add the following criteria:
  - **INSTITUTION** is “PSUNV”
  - **AID_YEAR** is “2007”
  - **ACAD_CAREER** is “UGRD”
  - **AWARD_STATUS** is “A” (accepted)
  - **OFFER_AMOUNT** is less than $750
- Position **AWARD_MSG_CD** as the second-to-last column in the results
- Position **DESCRLONG** as the last (rightmost) column in the results
- Sort the results by **EMPLID** and **ITEM_TYPE**
- **Save** the query again



View All | Rerun Query | Download to Excel | Download to XML — First 1-35 of 35 Last

	ID	Item Type	Disb Plan	Split Code	Offered	Accepted	Message	Descr
1	FAD0077	900000000520	DQ	01	500.00	500.00		
2	FAD0084	900000000311	DD	01	276.00	276.00		
3	FAD0128	900000000311	DD	01	181.00	181.00		
4	FAEA1005	900000000442	01	01	600.00	600.00		
5	FAEA1006	900000000442	01	01	600.00	600.00		
6	FAEA1007	900000000442	01	01	600.00	600.00		
7	FAEA1008	900000000442	01	01	600.00	600.00		
8	FAEA1023	900000000200	01	01	600.00	600.00	FWS	You have been awarded from the Federal Work Study Program. Please report to the Student Employment Placement office prior to 10/1/97 for your placement interview. All recipients of a FWS award that have not been interviewed by that date may not be placed for work-study in the Fall Term.
9	FAEA1025	900000000200	01	01	600.00	600.00	FWS	You have been awarded from the Federal Work Study Program. Please report to the Student Employment Placement office prior to 10/1/97 for your placement interview. All recipients of a FWS award that have not been interviewed by that date may not be placed for work-study in the Fall Term.
10	FAEA1063	900000000100	01	XX	600.00	600.00	PELL	You have been awarded a Federal Pell Grant based on a full-time load, if your actual enrollment load changes your award will be adjusted accordingly.

Subqueries – Checking for Existence

Many times, to answer one question, you need to ask several other questions. There may be cases in which you have to run one query to collect information to be used in another query. Rather than write and run two separate queries, you can use a subquery within your main query to collect that information. The result of the subquery can then be used like a field or expression in your query criteria.

To answer some questions, you may need to collect specific information, but to answer others, you may only need to find out whether something – usually a child row or a related row – exists. In a query, this is done by including an *exists* subquery or a *does not exist* subquery in your main query.

For each row returned by the main query, the subquery will report whether or not there is a matching row matching the conditions in the subquery. This is similar to an outer join, which reports the data from a second record regardless of whether there is a match, but no data is actually returned.

Consider again the excerpts from the countries (COUNTRY_TBL) and states (STATE_TBL) tables.

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

STATE_TBL (B)

COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

Let us ask the question of which countries have states or provinces. We do not need to know what those states or provinces are, so it is not necessary to join COUNTRY_TBL to STATE_TBL. In this case we would use an *exists* subquery. For each country in COUNTRY_TBL, the subquery would check STATE_TBL for any rows with the same value in COUNTRY as the country being examined.

Processing starts with the Canada row of COUNTRY_TBL. The subquery examines the states table for the existence of a row with the same value in COUNTRY – CAN. Since one is found, the Canada row is added to the results.

COUNTRY_TBL (A)

COUNTRY	DESCR
CAN	Canada
NOR	Norway
PAN	Panama
USA	United States

match

STATE_TBL (B)

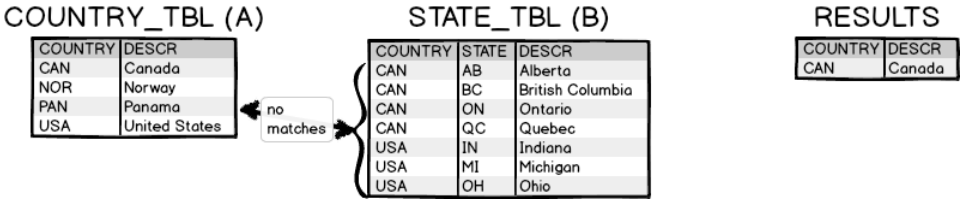
COUNTRY	STATE	DESCR
CAN	AB	Alberta
CAN	BC	British Columbia
CAN	ON	Ontario
CAN	QC	Quebec
USA	IN	Indiana
USA	MI	Michigan
USA	OH	Ohio

RESULTS

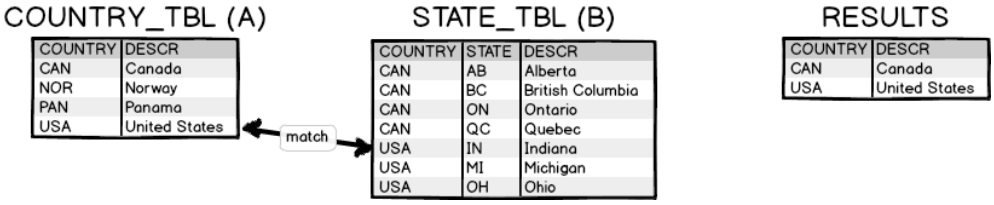
COUNTRY	DESCR
CAN	Canada

Processing continues with the Norway row. The subquery checks for the existence of a row in STATE_TBL with NOR in the COUNTRY field. Since there is no such row, nothing is added to the results.

The Panama row is examined next. Since there are no rows in STATE_TBL with a COUNTRY value of PAN, no rows are added to the results.

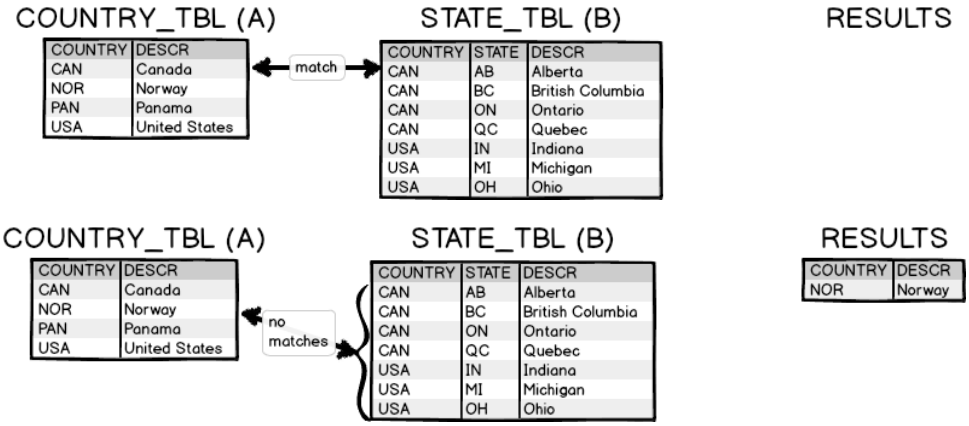


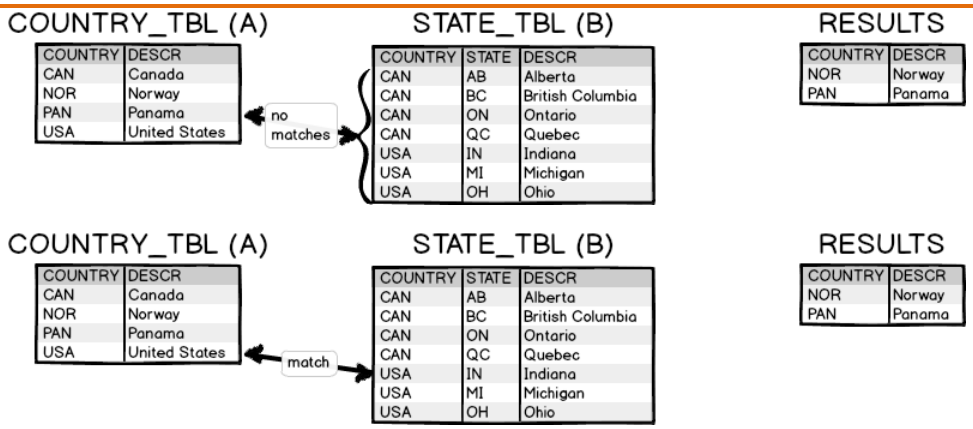
Lastly, the United States row is examined. The subquery finds a row in STATE_TBL that has a COUNTRY value of USA, matching the COUNTRY_VALUE in the United States row; the United States row is therefore added to the results.



If we wanted to instead know which countries did not have any states or provinces, we would use a *not exists* subquery. For every row country in COUNTRY_TBL, the subquery would check if a row with a matching COUNTRY exists in STATES_TBL. Only if there is *not* a matching row in STATE_TBL will the row in COUNTRY_TBL be added to the results. This is the opposite of an *exists* subquery.

Using the same excerpts of COUNTRY_TBL and STATE_TBL from the previous example, here is how the countries that do not have states or provinces would be found using a not exists subquery:





This space intentionally left blank.

Next we will build an exists subquery in Query Manager. Consider the following exercise. Write a query to find undergraduate classes offered in Spring 2006 that had graduate students enrolled in them. You can think of this as a two-step process: collect all undergraduate classes from Spring 2006, then for each class, check if there were any graduate students enrolled. The main query will list classes and the subquery will check for the existence of any graduate students in each class.

- Start a query on **CLASS_TBL**
- **Select** fields:
  - **CRSE_ID**
  - **SESSION_CODE**
  - **CLASS_SECTION**
  - **SUBJECT**
  - **CATALOG_NBR**
  - **DESCR**
  - **CLASS_NBR**
- **Add Criteria**:
  - **INSTITUTION** is equal to **PSUNV**
  - **STRM** is equal to **0560**
  - **ACAD_CAREER** is equal to **UGRD**

Save and save often. **Save** as **TRNG_QM##_E15**.

Run the query to see a list of all undergraduate classes offered in Spring 2006.

View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 277 Last

	Course ID	Session	Section	Subject	Catalog	Descr	Class Nbr
1	001003	1	1	MATH	10	Remedial Algebra	1169
2	001004	1	1	MATH	107	Precalculus	1170
3	001005	1	1	MATH	111	Calculus I	1171
4	001006	1	1	MATH	212	Calculus II	1172
5	001008	1	1	MATH	136	Introduction of Linear Algebra	1173
6	001002	1	1	MATH	104	Finite Mathematics	1168
7	001026	1	1	HISTORY	130	World History to WWII	1002
8	001026	OEE	1OE	HISTORY	130	World History to WWII	1219
9	001026	OEE	1OF	HISTORY	130	World History to WWII	1220
10	001028	1	1	MATH	255	Graph Theory	1177

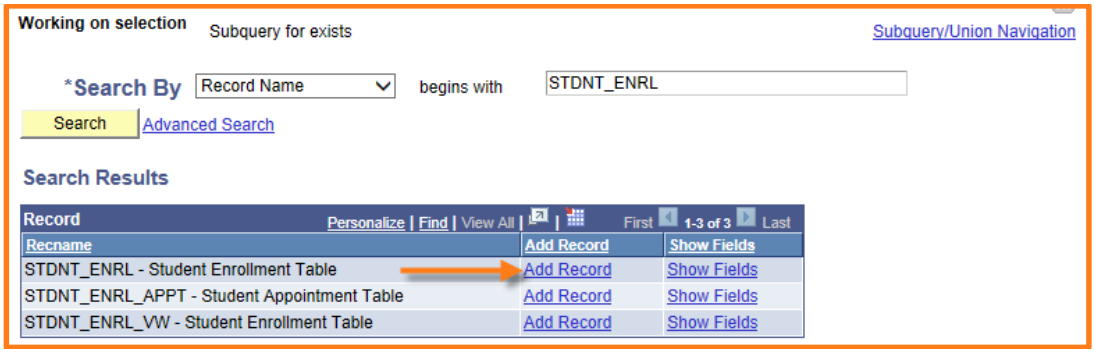
Now we will add the limitation that the results will show only classes that had a graduate student enrolled.

- **Go** to **Criteria tab**
- **Click Add Criteria**
- **Change** Condition Type to "**exists**". The Expression 1 section disappears and the Expression 2 section shows only the Define/Edit Subquery link.
- **Click** the **Define/Edit Subquery** link

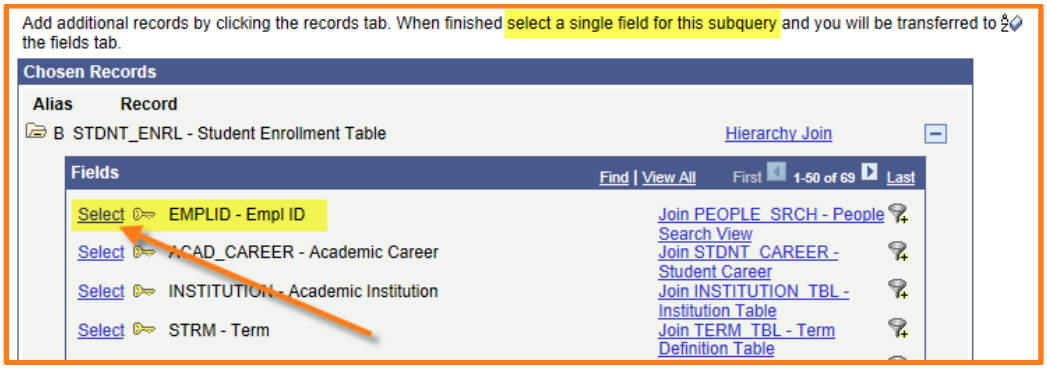
You are taken to the Records page. Below the Query Name field you will see "Working on selection: Subquery for exists". This informs you that you are now building the subquery that will be used to report information back to the main query.

Now we will build the subquery that checks for graduate student enrollment in the classes being processed in the main query.

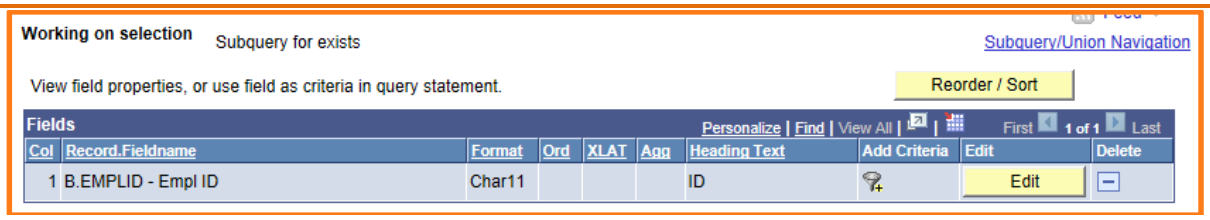
- **Search** for and add **STDNT_ENRL**. (Observe that the link is titled Add Record instead of Join Record. A subquery is not a join.)



- **Select** the **EMPLID** field. (For subqueries you can select only one field. For exists and not exists subqueries it does not matter which field you choose.)
- **Save** the query

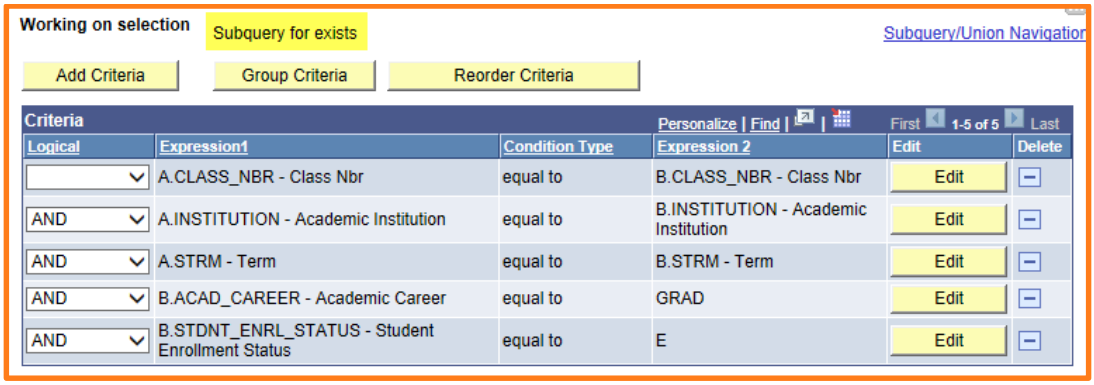


After selecting the field you are taken to the Fields tab. The field that you selected is displayed in the Fields list.

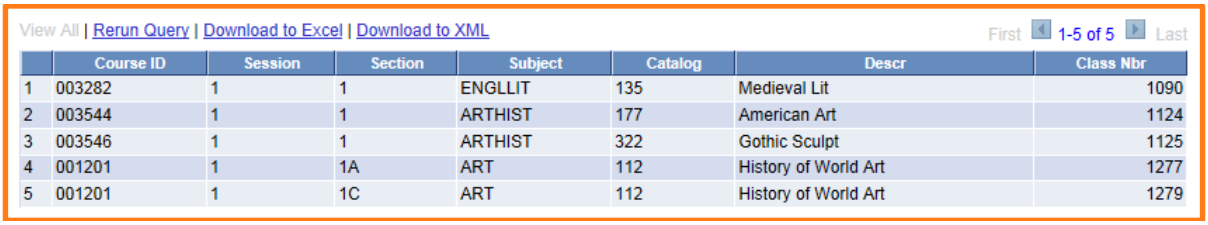


Next the subquery must be tied back to the main query so that only students from the classes in the main query are examined. It must also be restricted to enrolled graduate students.

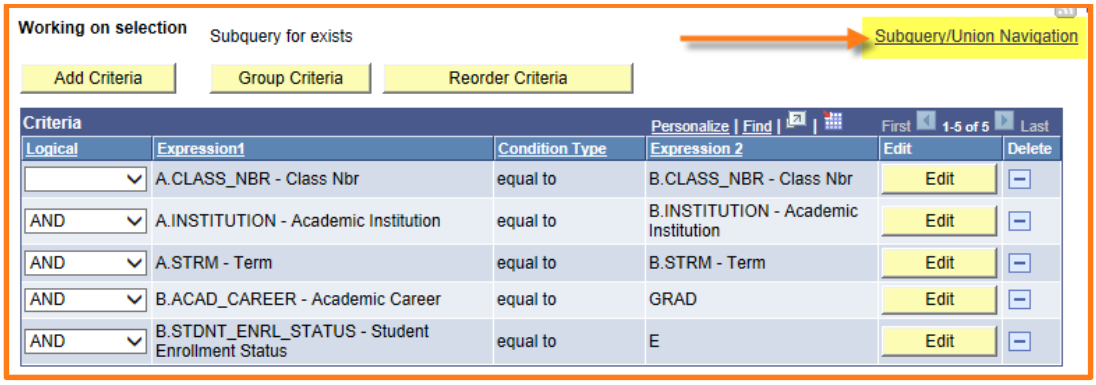
- **Click** the **Criteria** tab
- **Add Criteria**:
  - **A.CLASS_NBR** equal to **B.CLASS_NBR**
  - **A.INSTITUTION** equal to **B.INSTITUTION**
  - **A.STRM** equal to **B.STRM**
  - **B.ACAD_CAREER** equal to **GRAD**
  - **B.STDNT_ENRL_STATUS** equal to **E** (enrolled)



- **Save** the query
- **Click** the **Run** tab



Note: if you need to work on different parts of the query, click the Subquery/Union Navigation link near the top of every Query Manager page except for Run.



You will be taken to a page that lists the main part of the query, all subqueries, and all unions (which are described in the next section). Click the link corresponding to the part of the query you want to work on.

This space intentionally left blank.

Subqueries – Single Value

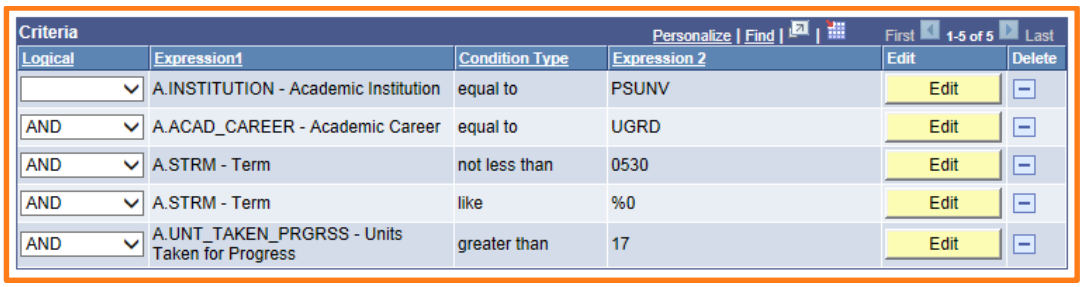
In the previous segment we used a subquery to find out whether or not there was a graduate student enrolled in an undergraduate class. The subquery did not need to return any information about the student – just that a student existed. In this segment we will use a subquery to get information about what is being examined in the main query, then use that information to restrict the rows being kept in the results.

Let's find undergraduate students who were enrolled at PeopleSoft University for more than 17 credit hours in consecutive terms. (We will only consider fall and spring terms and only terms starting from Fall 2004 to make finding the next term simpler.) List the student's ID, the term, the number of units taken in that term, and the number of units taken overall as of that term. Sort the results by student ID and term.

As we build this query, keep the idea of the subquery in mind. For each student, we will ask how many credit hours that student is taking in the following term. This will be done with a subquery. The result of that subquery – the number of credit hours – will then be used in the criteria of the main query.

Note that this question will be asked for *every* student and term; the credit hours may be different for different students. This is why a subquery is useful and appropriate – without one, we'd have to get the enrollment manually for every student in every term!

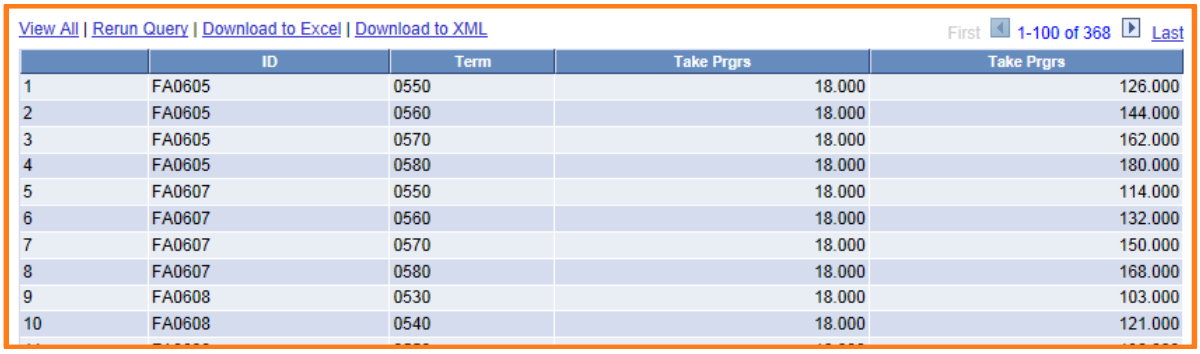
- Start a query on **STDNT_CAR_TERM**
- **Select** fields:
  - **EMPLID**
  - **STRM**
  - **UNT_TAKEN_PRGRSS**
  - **TOT_TAKEN_PRGRSS**
- **Add** criteria:
  - **INSTITUTION** is equal to **PSUNV**
  - **ACAD_CAREER** is equal to **UGRD**
  - **STRM** is not less than **0530** (greater than or equal to 0530, the code for Fall 2004)
  - **STRM** is like "%0" (ends with zero, making it a fall term or a spring term)
  - **UNT_TAKEN_PRGRSS** is greater than 17
- Click the **Criteria** tab and confirm that your criteria look like this:



Criteria — Personalize | Find | First 1-5 of 5 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.INSTITUTION - Academic Institution	equal to	PSUNV	Edit	
AND	A.ACAD_CAREER - Academic Career	equal to	UGRD	Edit	
AND	A.STRM - Term	not less than	0530	Edit	
AND	A.STRM - Term	like	%0	Edit	
AND	A.UNT_TAKEN_PRGRSS - Units Taken for Progress	greater than	17	Edit	

- **Sort** the results by **EMPLID** and **STRM**.
- Save and save often. Save as **TRNG_QM##_E16**.
- Run the query. The results list each student who, as an undergraduate, took more than 17 credit hours in a term, and the terms in which the student did so.

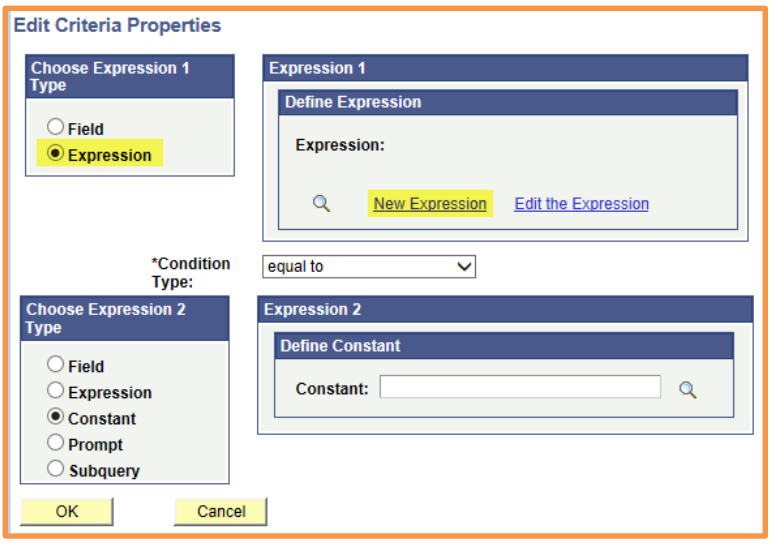


View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 368 Last

	ID	Term	Take Prgrs	Take Prgrs
1	FA0605	0550	18.000	126.000
2	FA0605	0560	18.000	144.000
3	FA0605	0570	18.000	162.000
4	FA0605	0580	18.000	180.000
5	FA0607	0550	18.000	114.000
6	FA0607	0560	18.000	132.000
7	FA0607	0570	18.000	150.000
8	FA0607	0580	18.000	168.000
9	FA0608	0530	18.000	103.000
10	FA0608	0540	18.000	121.000

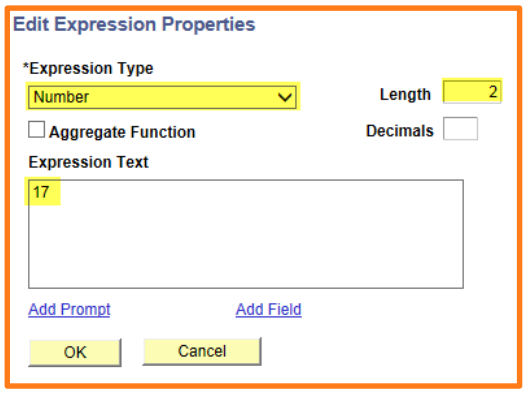
Next we will limit the results to just students who took over 17 credit hours in two consecutive terms. This will be done with a subquery.

- **Click** the **Criteria** tab
- **Click** the **Add Criteria** button
- **Select Expression** for **Expression 1 Type**
- **Click** the **New Expression** link

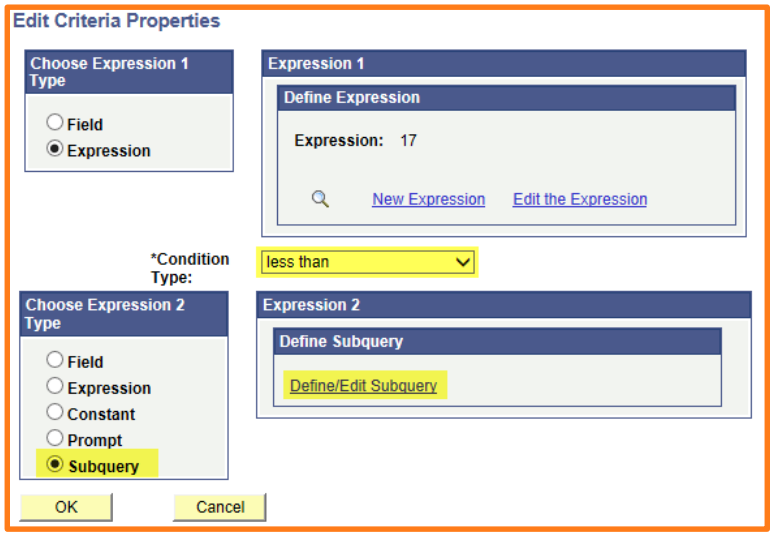


- Create the expression:
  - Select **Number** for **Expression Type**
  - Set **Length** to **2**
  - Set **Expression Text** to **17**
  - **Click OK**

(To compare a constant to the result of a subquery it is necessary to create an expression; even though there is a Constant option for Expression 2 Type, there is no Subquery option for Expression 1 Type.)

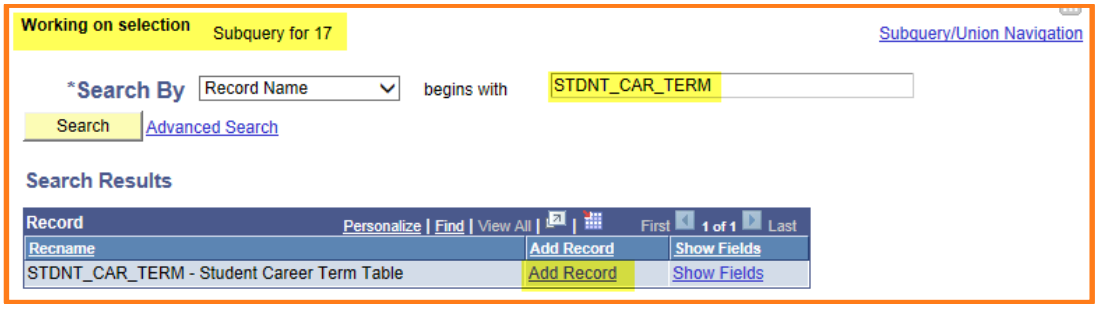


- **Set Condition Type** to **less than**
- **Set Expression 2 Type** to **"Subquery"**
- **Click Define/Edit Subquery**



You are taken to the Records page. Near the top of the page the message "Working on selection: Subquery for 17" is displayed. This indicates you are building a subquery rather than working on the main query.

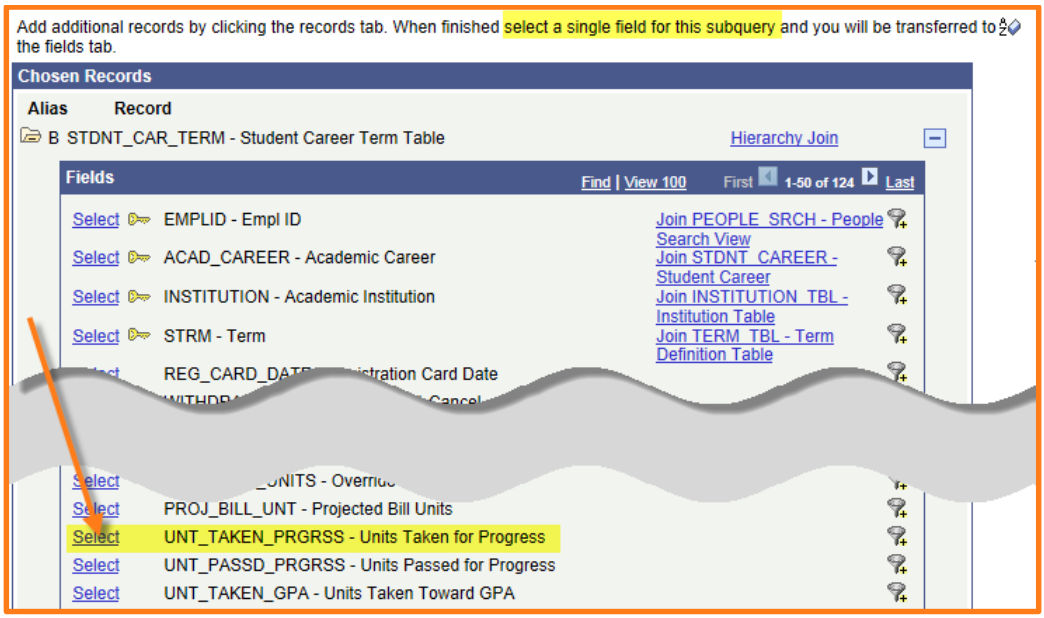
- **Select** and add **STDNT_CAR_TERM** to the subquery, then click on the **Add Record** link.



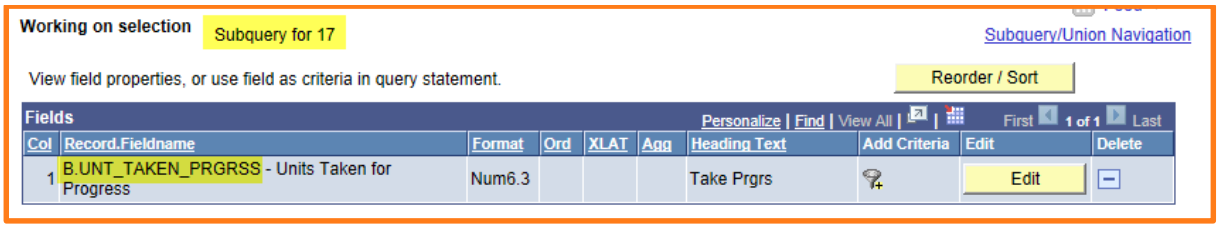
After clicking on the Add Record link, the Query page appears. Fields of STDNT_CAR_TERM are displayed, but instead of each field having a checkbox, they each have a Select link.

This is because a subquery can return only one value. In this case we want the UNT_TAKEN_PRGRSS.

- **Click** the **Select** link in the **UNT_TAKEN_PRGRSS** row.



You are taken to the **Fields** tab. **B.UNT_TAKEN_PRGRSS** is listed as the single field for this subquery.



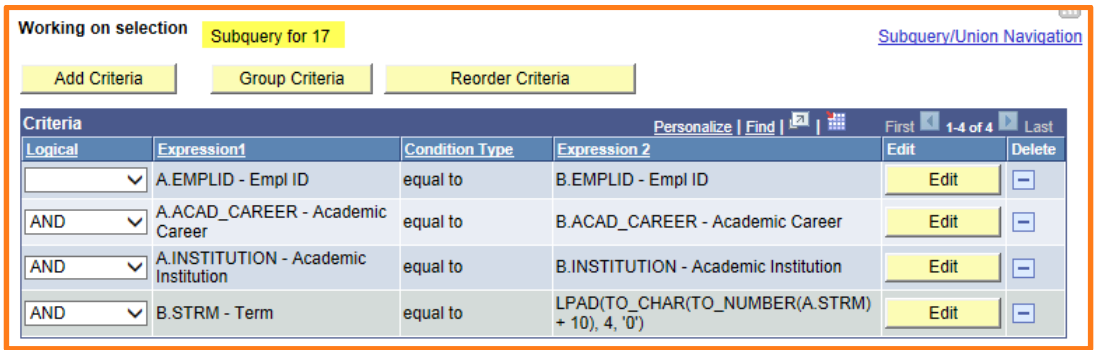
- **Save** the query.

Now we need to limit the subquery to the correct student and to the desired term. We will do this by adding criteria to the subquery.

- **Go to** either the **Query** or **Criteria** tab – making sure that "Working on selection: Subquery for 17" is still shown – then add the following criteria:
  - **A.EMPLID** equal to **B.EMPLID**
  - **A.ACAD_CAREER** equal to **B.ACAD_CAREER**
  - **A.INSTITUTION** equal to **B.INSTITUTION**
  - (these three criteria make sure the subquery provides results on the same student that the main query is processing)
  - **B.STRM** is equal to this expression:
    LPAD(TO_CHAR(TO_NUMBER(A.STRM) + 10), 4, '0')

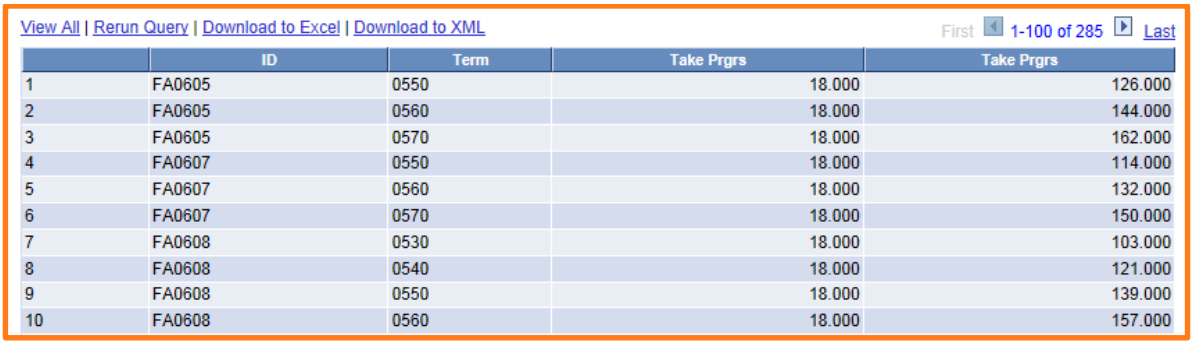
The convention for term codes at PeopleSoft University starting with Fall 2004 is that each consecutive fall and spring term has a code 10 higher than that of the previous term. For

instance, the code for Spring 2005 is “0540” so the code for Fall 2005 is 10 higher, which is “0550”. All term codes are padded on the left with zeroes until they are four digits long. The expression converts the term in the main query from a string to a number, adds 10 to that number, converts it back to a string, then adds zeroes as needed.



Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.EMPLID - Empl ID	equal to	B.EMPLID - Empl ID	Edit	
AND	A.ACAD_CAREER - Academic Career	equal to	B.ACAD_CAREER - Academic Career	Edit	
AND	A.INSTITUTION - Academic Institution	equal to	B.INSTITUTION - Academic Institution	Edit	
AND	B.STRM - Term	equal to	LPAD(TO_CHAR(TO_NUMBER(A.STRM) + 10), 4, '0')	Edit	

- **Save** the query
- **Click** the **Run** tab

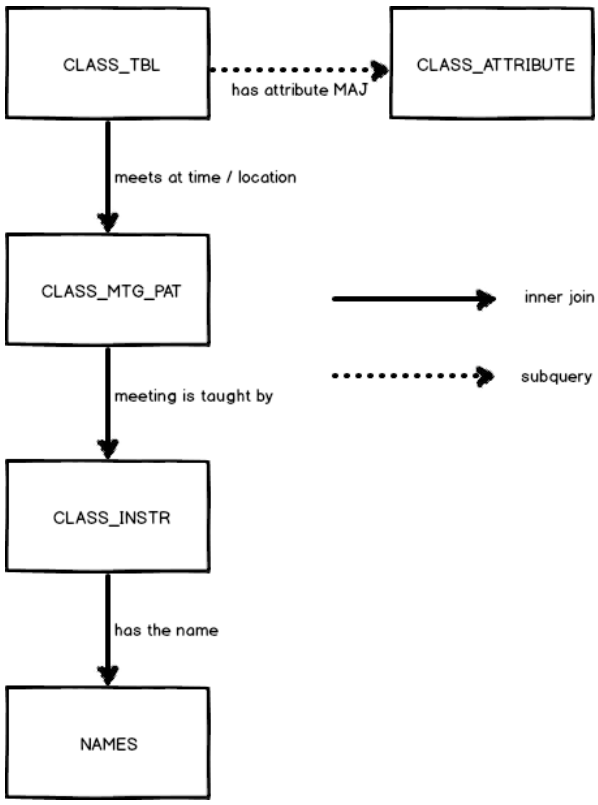


	ID	Term	Take Prgrs	Take Prgrs
1	FA0605	0550	18.000	126.000
2	FA0605	0560	18.000	144.000
3	FA0605	0570	18.000	162.000
4	FA0607	0550	18.000	114.000
5	FA0607	0560	18.000	132.000
6	FA0607	0570	18.000	150.000
7	FA0608	0530	18.000	103.000
8	FA0608	0540	18.000	121.000
9	FA0608	0550	18.000	139.000
10	FA0608	0560	18.000	157.000

These results list all undergraduates who took over 17 credit hours in a term as well as the following term.

Question 16

Find classes offered at PeopleSoft University in Fall 2006 that were reserved for students majoring in the subject matter. For each class, show the section, subject, catalog number, and description, as well as the primary name of the primary instructor of each class. This will require joining CLASS_TBL to CLASS_MTG_PAT, CLASS_MTG_PAT to CLASS_INSTR, and CLASS_INSTR to NAMES. It will also require a subquery criterion on CLASS_TBL to check for the existence of rows in CLASS_ATTRIBUTE with a CRSE_ATTR of "MAJ".



- Show the following fields from **CLASS_TBL**:
  - **CLASS_SECTION**
  - **SUBJECT**
  - **CATALOG_NBR**
  - **DESCR**
- Add a criterion that **STRM** is equal to **0570**
- Add an exists subquery using the **CLASS_ATTRIBUTE** record
- Add criteria to the subquery:
  - **B.CRSE_ID** is equal to **A.CRSE_ID**
  - **B.CRSE_OFFER_NBR** is equal to **A.CRSE_OFFER_NBR**
  - **B.STRM** is equal to **A.STRM**
  - **B.SESSION_CODE** is equal to **A.SESSION_CODE**
  - **B.CLASS_SECTION** is equal to **A.CLASS_SECTION**
  - **B.CRSE_ATTR** is equal to **MAJ** (Majors only)
- **Join** the **CLASS_MTG_PAT** record to **CLASS_TBL**
- **Join** the **CLASS_INSTR** record to **CLASS_MTG_PAT**
- Add a criterion that **INSTR_ROLE** is equal to **PI** (primary instructor)
- **Join** the **NAMES** record to **CLASS_INSTR**
- Show the **NAME** field from **NAMES**
- Add a criterion that **NAME_TYPE** is equal to **PRI** (primary)
- Save as **TRNG_QM##_Q16**

View All | Rerun Query | Download to Excel | Download to XML — First 1-4 of 4 Last

	Section	Subject	Catalog	Descr	Name
1	01	PSYCH	495	Psychology Special Topics	Litman,Edward
2	01	PSYCH	495	Psychology Special Topics	Edmundson,Aurelia
3	01	PSYCH	495	Psychology Special Topics	Baylor,Mara
4	01A	PSYCH	495	Psychology Special Topics	Edmundson,Aurelia

Unions

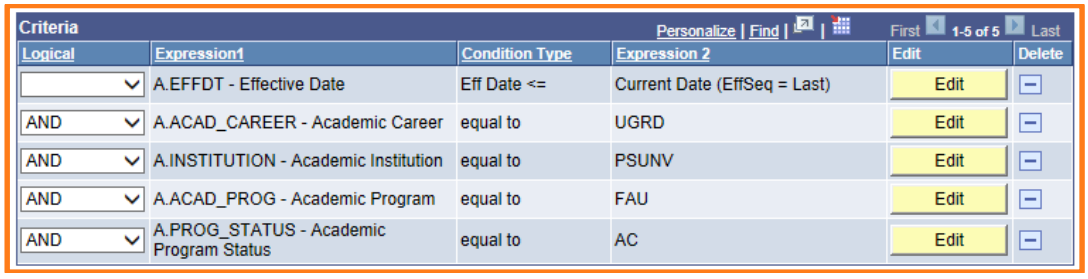
A union is the combination of results from two or more queries. If two queries are combined in a union, the result is all of the rows from the first query and all of the rows from the second query. Duplicate rows are discarded.

In order for two queries to be combined in a union, the queries must have the same number of columns and each column from the first query must have the same data type (date, name, or string) as the corresponding column from the second query. For instance, if the third column in the first query is a date, the third column in the second query must also be a date.

For the next exercise, you have an important communication for undergraduate students in the Fine Arts Undergraduate program, both current and recently admitted (for Fall 2000 or later, which will be considered "recent" for this exercise). Current students are listed in the **ACAD_PROG** table but admitted students are listed in **ADM_APPL_PROG**. Instead of building two queries you will use a **union** to collect the current and admitted students into the same set of results. Since the communication will be sent by e-mail you will join to **EMAIL_ADDRESSES** to obtain e-mail addresses for each student.

- Start a new query. Add the **ACAD_PROG** record
- Select the **EMPLID** field
- Add the following criteria:
  - **ACAD_CAREER** is equal to **UGRD**
  - **INSTITUTION** is equal to **PSUNV**
  - **ACAD_PROG** is equal to **FAU**
  - **PROG_STATUS** is equal to **AC**
- Save and save often. Save as **TRNG_QM##_E17**

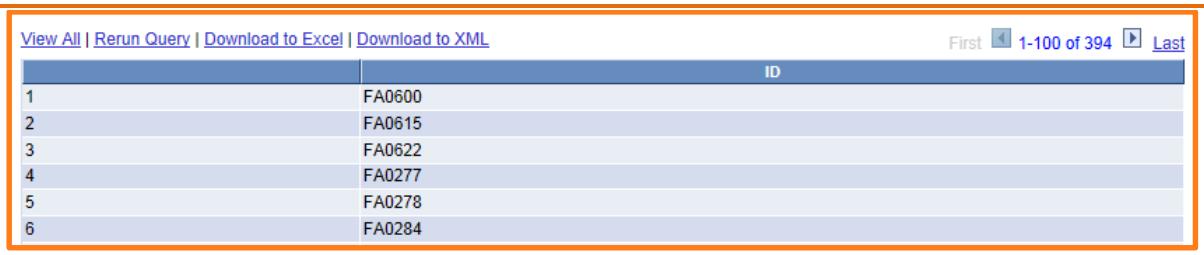
The Criteria page should look like this.


Criteria — Personalize | Find | First 1-5 of 5 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	Edit	
AND	A.ACAD_CAREER - Academic Career	equal to	UGRD	Edit	
AND	A.INSTITUTION - Academic Institution	equal to	PSUNV	Edit	
AND	A.ACAD_PROG - Academic Program	equal to	FAU	Edit	
AND	A.PROG_STATUS - Academic Program Status	equal to	AC	Edit	

- **Run** the query

The results should look like this.

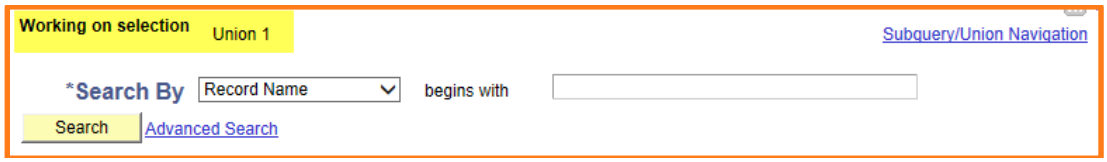


	ID
1	FA0600
2	FA0615
3	FA0622
4	FA0277
5	FA0278
6	FA0284

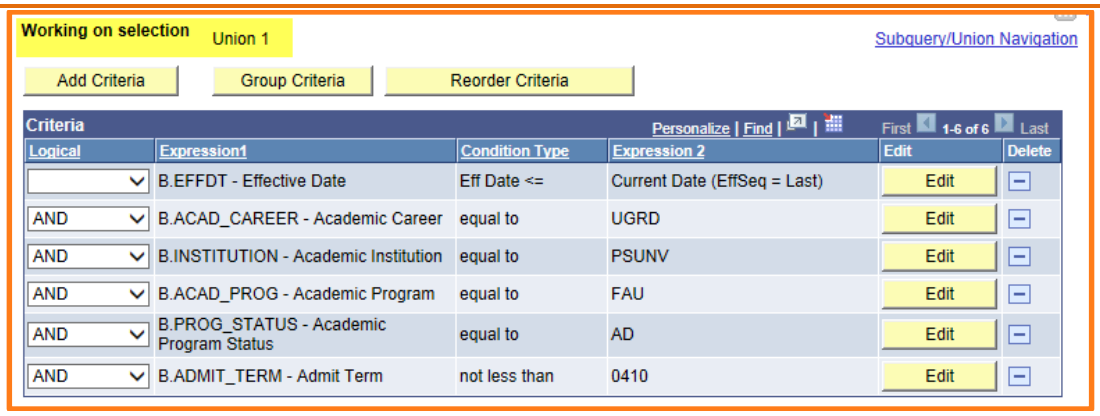
Now that we have the students currently in the Fine Arts Undergraduate program, we need to obtain the admitted students. We will do this by building a second query that retrieves the students and combining them in a *union* with the first query.

- **Click** on any tab except for Run. **Click** the New Union link located at the bottom of the page

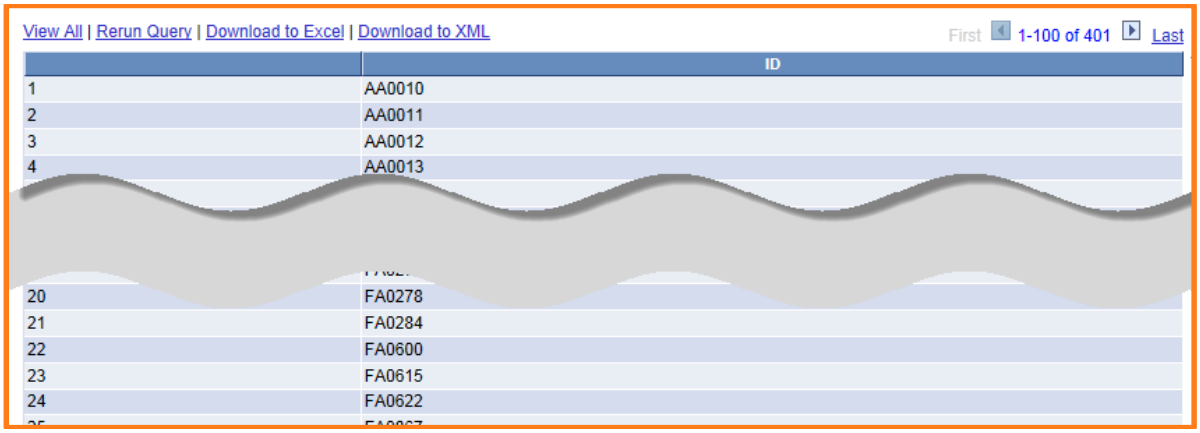
You are taken to the Records page. The indicator "Working on selection: Union 1" shows that you are working on the second part of this query.



- Search for and add the **ADM_APPL_PROG** record
- Select the **EMPLID** field. (Since this is a union, rather than a subquery, you can select multiple fields. However, remember that the number of fields in each union must be the same.)
- **Add** the following criteria:
  - **ACAD_CAREER** is equal to **UGRD**
  - **INSTITUTION** is equal to **PSUNV**
  - **ACAD_PROG** is equal to **FAU**
  - **PROG_STATUS** is equal to **AD**
  - **ADMIT_TERM** is not less than **0410**



- **Save** the query
- **Click** the **Run** tab

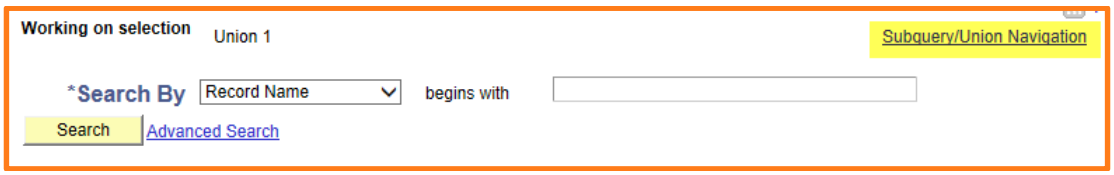


Now the results list students who are currently in the Fine Arts Undergraduate program and those admitted to it for Fall 2000 and later.

In order to send an e-mail to these students, we must obtain the e-mail addresses of the students. E-mail addresses are stored in the **EMAIL_ADDRESSES** table. You will join **EMAIL_ADDRESSES** to both **ACAD_PROG** and **ADM_APPL_PROG**.

First we will work on the part of the query dealing with current students. When you are editing a query that contains a union, the **Subquery / Union Navigation** link is shown on all tabs except for the **Run** tab. Follow this link to get a list of all parts of the query then follow the link corresponding to the part of the query you want to work on.

- **Click** the **Records** tab
- **Click** the **Subquery / Union Navigation** link



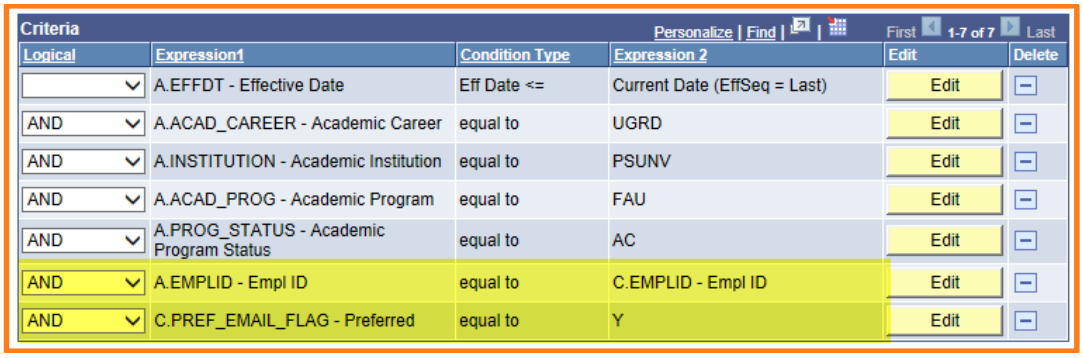
All parts of the query that are connected by a union are displayed.

In this exercise, **Top Level of Query** corresponds to the part getting current students (from **ACAD_PROG**) and **Union 1** corresponds to the part getting admitted students (from **ADM_APPL_PROG**).

- **Click** the **Top Level of Query** link

The Query tab for the first part of the query is shown. Now we need to join **EMAIL_ADDRESSES**.

- **Click** the **Records** tab.
- Search for and join the **EMAIL_ADDRESSES** record to **ACAD_PROG**.
- Accept the default join criteria by clicking the **Add Criteria** button.
- **Select** the **EMAIL_ADDR** field.
- **Add** the following criteria:
  - **PREF_EMAIL_FLAG** is equal to **Y**.



Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	Edit	
AND	A.ACAD_CAREER - Academic Career	equal to	UGRD	Edit	
AND	A.INSTITUTION - Academic Institution	equal to	PSUNV	Edit	
AND	A.ACAD_PROG - Academic Program	equal to	FAU	Edit	
AND	A.PROG_STATUS - Academic Program Status	equal to	AC	Edit	
AND	A.EMPLID - Empl ID	equal to	C.EMPLID - Empl ID	Edit	
AND	C.PREF_EMAIL_FLAG - Preferred	equal to	Y	Edit	

- **Save** the query

When two or more parts of a query are connected by a union, they must select the same number of fields and the field types must match. (For instance, if the first column in one part is a date, the first column in all other parts must also be a date.)

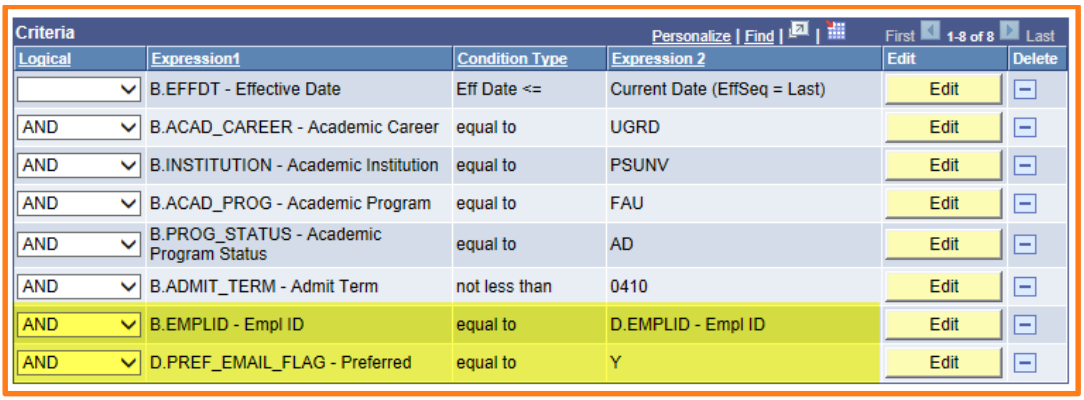
We still need to add the e-mail address to the second part of the query.

- **Click** the **Records** tab
- **Click** the **Subquery/Union Navigation** link
- **Click** the **Union 1** link

The Query tab for the second part of the query is shown. Now we will join **EMAIL_ADDRESSES**. (Keep in mind that you joined **EMAIL_ADDRESSES** to **ACAD_PROG** in the first part of the query, but the second part is completely separate from the first part.)

- **Click** the **Records** tab.
- Search for and join the **EMAIL_ADDRESSES** record to **ADM_APPL_PROG**.

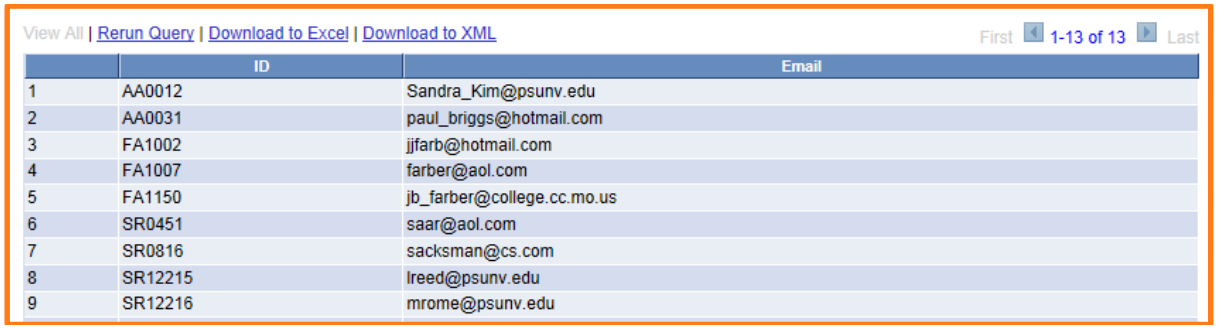
- Accept the default join criteria by clicking the **Add Criteria** button.
- **Select** the **EMAIL_ADDR** field.
- **Add** the following criteria:
  - **PREF_EMAIL_FLAG** is equal to **Y**.



Criteria — Personalize | Find | First 1-8 of 8 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	B.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	Edit	
AND	B.ACAD_CAREER - Academic Career	equal to	UGRD	Edit	
AND	B.INSTITUTION - Academic Institution	equal to	PSUNV	Edit	
AND	B.ACAD_PROG - Academic Program	equal to	FAU	Edit	
AND	B.PROG_STATUS - Academic Program Status	equal to	AD	Edit	
AND	B.ADMIT_TERM - Admit Term	not less than	0410	Edit	
AND	B.EMPLID - Empl ID	equal to	D.EMPLID - Empl ID	Edit	
AND	D.PREF_EMAIL_FLAG - Preferred	equal to	Y	Edit	

- **Save** the query
- **Run** the query



View All | Rerun Query | Download to Excel | Download to XML — First 1-13 of 13 Last

	ID	Email
1	AA0012	Sandra_Kim@psunv.edu
2	AA0031	paul_briggs@hotmail.com
3	FA1002	jjfarb@hotmail.com
4	FA1007	farber@aol.com
5	FA1150	jb_farber@college.cc.mo.us
6	SR0451	saar@aol.com
7	SR0816	sacksman@cs.com
8	SR12215	lreed@psunv.edu
9	SR12216	mrome@psunv.edu

Now the query results include both current and admitted students and the e-mail address of each student.

Duplicate rows are automatically discarded when parts are connected in a union. A duplicate row in this context is based on all fields in the row. In addition, the results are automatically sorted by first field, second field, and so on, but this order can be overridden if desired.

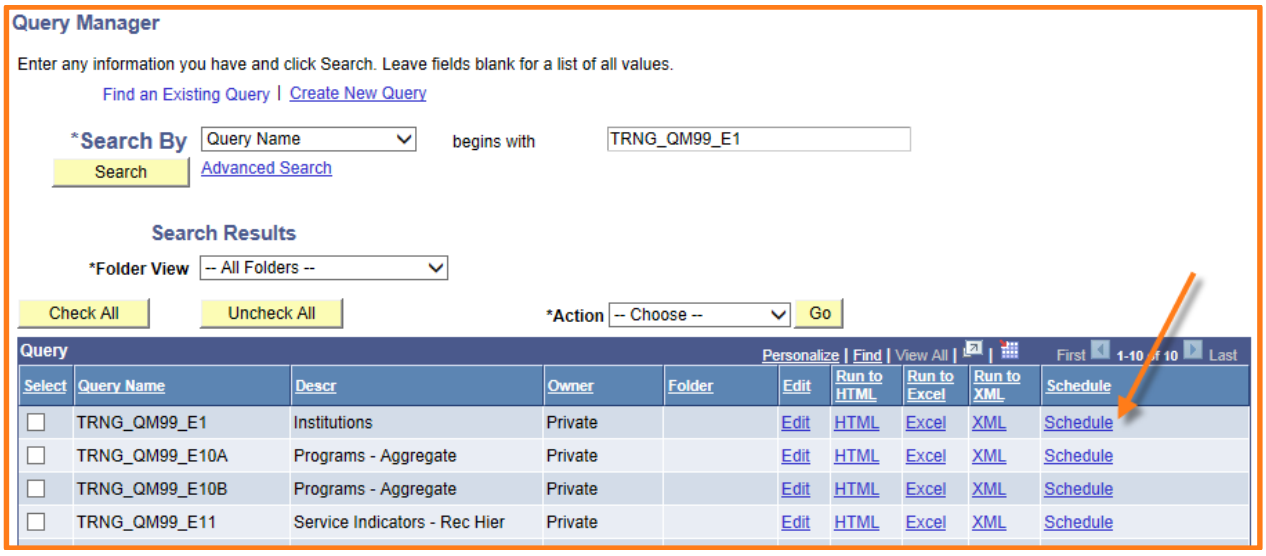
In this example you selected each person's preferred e-mail address (PREF_EMAIL_FLAG is equal to "Y"), but at Bowling Green State University you would instead select each person's campus e-mail address (E_ADDR_TYPE is equal to "CAMP") for communicating University business. Also, all current students and most, if not all, admitted students should have BGSU e-mail addresses.

Scheduling Queries

Some queries can be so complex that it can take several minutes to get results; during the time the query is running it is usually not possible to perform other tasks in PeopleSoft unless another PeopleSoft window is open. There are also cases in which a query needs to be run on a regular basis, such as once a day or once a week, for reporting and monitoring purposes.

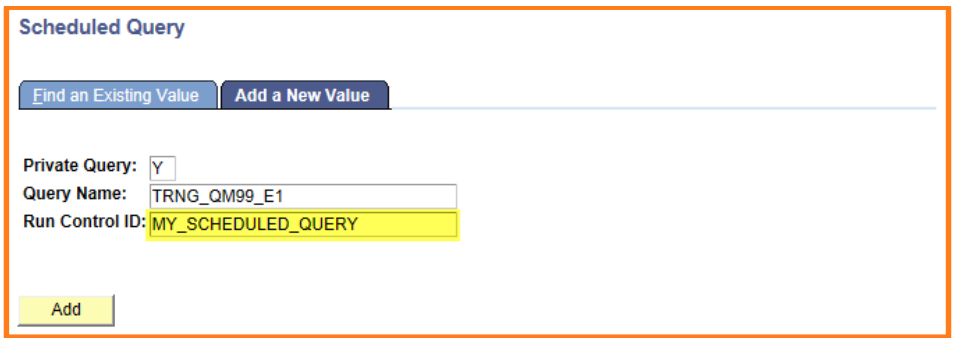
Scheduling a query to run at a particular time can address both of these items. When a query is scheduled, this allows it to be run without user intervention. The user can work in other pages in PeopleSoft then pick up the results when the query has finished running. If the query is scheduled to run under a *recurrence*, the query will be run at the days and times defined for that recurrence, automatically being rescheduled to run again at the appropriate time.

To schedule a query, search for it in the Query Manager search page, then click the **Schedule** link.

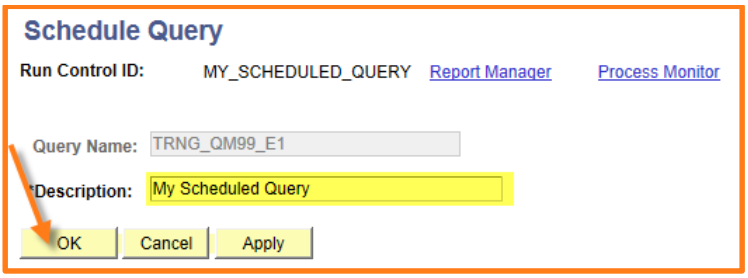


You will then be asked to select or create a *run control* to identify the process submitted to the PeopleSoft Process Monitor for running your query on a schedule. If you have no run controls that run the selected query you will automatically be put on the **Add a New Value** tab for creating a new run control; otherwise you will be taken to the **Find an Existing Value** tab from which you can select a run control you have already made for scheduling the query.

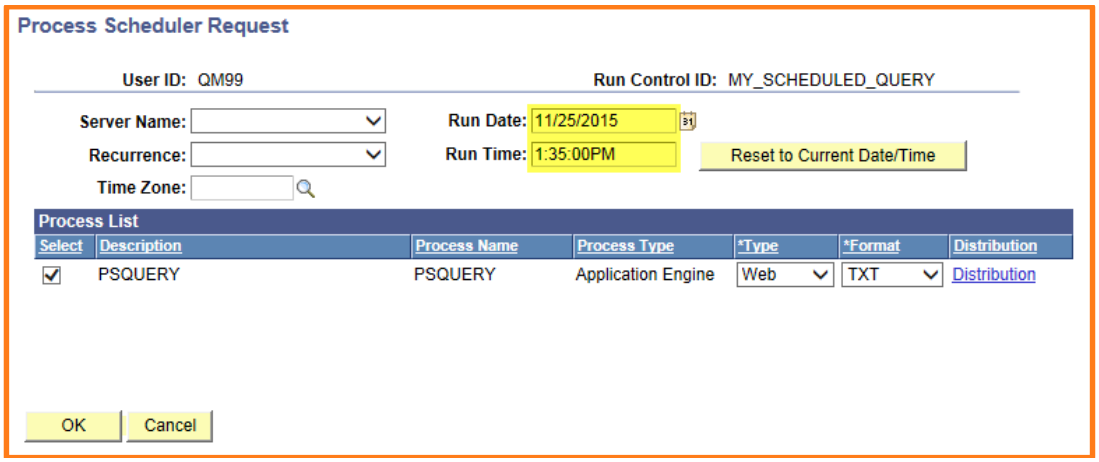
This example will involve creating a new run control by clicking the Add a New Value tab. The Private Query and Query Name fields are filled in automatically. Enter a Run Control ID to assign to this run control then click the **Add** button. If the query has any prompts you will be asked to enter values for each prompt.



You will then be asked to enter a description for this scheduled run. Enter this then click the **OK** button.



The Process Scheduler Request page appears. To pick a specific date and time at which the query should be run, enter the date and time into the **Run Date** and **Run Time** fields. To instead have the query run repeatedly at the same time every day or every week, select a predefined recurrence from the **Recurrence** dropdown.



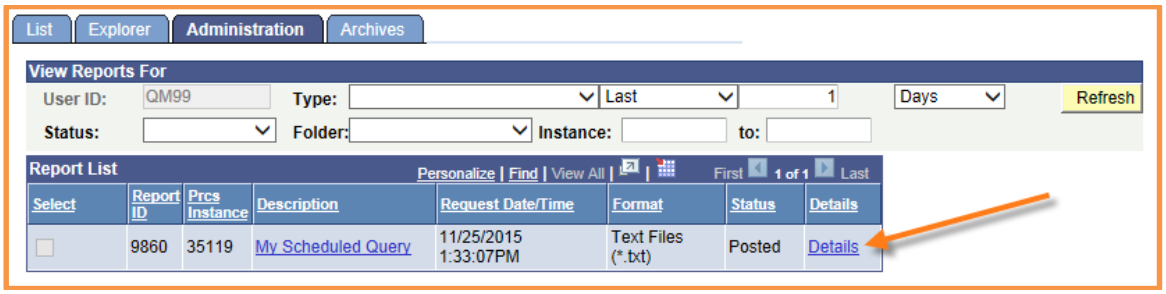
In the Process List section you will see one process named PSQUERY; this is the internal process that will run the query. In the **Format** dropdown there are several different formats for the output of the query, including:

- HTM: web page, similar to the output of a query when run from Query Manager
- PDF: a document in the Adobe PDF format
- TXT: a comma-separated variable (CSV) text file

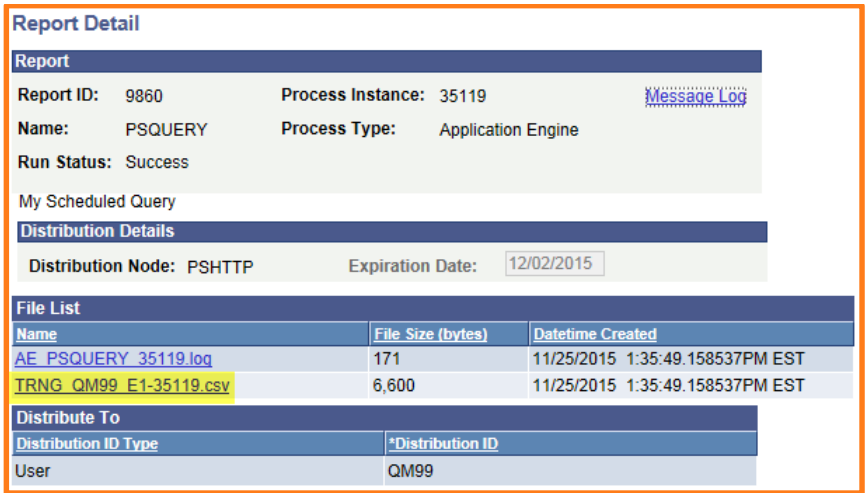
- XLS: an Excel workbook

Select the format that you want for the output. When you are ready to submit, click the OK button. You will be returned to the Query Manager search page.

To see the results of the query navigate to Reporting Tools > Report Manager then click the Administration tab. Your recently generated reports are displayed.



Click the link in the Details column corresponding to the scheduled query for which you want to view results. The details of that report are displayed. In the File List section, find the link with the name of the query (not the run control ID) that you scheduled.



Click the file link to open the file directly or right-click it and choose Save Target As to save it to your computer.

If you selected the TXT format to generate a CSV file and you are running a computer with Excel, make sure to save the file to your computer instead of opening directly and to save it with an extension of CSV instead of the default of XLS. This will prevent Excel from giving a warning that the file is in a different format than that specified by the extension. In addition, if there are values with leading zeroes such as EMPLIDs and ZIP codes in the results, create a new workbook then import the CSV file using Data > From Text and set the data type of those columns to Text; this will prevent Excel from dropping leading zeroes from those values. If you do not need to import the data into a database or other program you could

change the format to XLS when submitting the query to get the data as a true Excel workbook.

In Campus Solutions (CSS) and Human Capital Management (HCM) there are many defined recurrences to choose from. The name of the recurrence indicates the days, times, and frequency of that recurrence. Some examples of the naming convention are as follows:

- DLY5CN0700A – daily (DLY) on weekdays (5) at 7:00 AM (0700A); schedule when the current request starts (C) and schedule missed runs (N)
- DLY7PN0400P – daily (DLY) every day of the week (7) at 4:00 PM (0400P); schedule when the previous recurrence finishes (P) and schedule missed runs (N)
- WKYFPN0915P – weekly (WKY) on Fridays (F) at 9:15 PM (0915P); schedule when the previous request finishes (P) and schedule missed runs (N)
- WKYMWPY0630A – weekly (WKY) on Mondays and Wednesdays (MW) at 6:30 AM (0630A); schedule when the previous request finishes (P) and do not schedule missed runs (Y)

The letters U and R are used to indicate Sunday and Thursday, respectively.

In Financial Management Solutions (FMS) there are also many defined recurrences. The naming convention is less strict than in CSS. Most recurrences are just named after the days and times on which they run; examples include “M-F at 5pm” and “T – S 8:00 AM”.

NOTES:

Supplemental Material

Finding Records and Fields

Normally a request for a query will be in the form of a question or a request for certain information. It will usually not come with the fields, records, and criteria given to you directly. Instead, it will be like a story problem. You'll be responsible for translating the request into a query that PeopleSoft understands.

What if you don't already know where to find the information?

- Use the Advanced Search capabilities. You can search by record name, description, and fields used in a record, among others. For instance, if you know a query is about programs, you can search for records having "program" in the description or PROGRAM in the name. If this does not provide enough options or any correct ones, try abbreviations such as PGM and PROG. You could also search for records having a field with PROG in the name.
- Look at the descriptions. When you search for records, the description of each record is given along with the name of the record. When you show fields in a record, the description of each field is displayed.
- Ask the requestor to show you where the underlying data is in PeopleSoft. If he or she can show you some pages, you can get some field labels that may appear in the descriptions of fields. If the requestor shows you a field labeled "Contact Person," then you can search for records that have fields with descriptions containing "contact" or "person" or names containing "PERS".
- Use references that others in your department have created.
- Ask your coworkers. ☺

Effective Date, Effective Sequence, and Effective Status

Frequently there is a need to keep track of the history of changes to something in a database. The status of a student will change as he applies to a program, matriculates, possibly adds minors, and completes that program. There may also be a need to retain information about something, such as its name, flags, and amounts. In addition, to aid in planning ahead, there may be a need to store something that will come into effect in the future.

PeopleSoft uses special fields in many records to enable having data effective only at certain times – Effective Date (EFFDT), Effective Sequence (EFFSEQ), and Status as of Effective Date (EFF_STATUS).

Basics of "Effective" Data

Effective Date is the most commonly used of these three fields; the other two fields will not be in a record without an EFFDT field. It indicates that the record is effective as of a certain date.

This concept may be easiest to understand using an example. Consider the following rows of ACAD_PROG_TBL, which contains information about academic programs.

INSTITUTION	ACAD_PROG	EFFDT	EFF_ STATUS	DESCR	ACAD_CAREER	ACAD_ CALENDAR_ID	ACAD _ORG
BGSUN	MUSIC	5/20/2013	A	College of Musical Arts	UGRD	USEM	MUCLG
BGSUN	MUSIC	8/29/1982	A	College of Musical Arts	UGRD	USEM	
BGSUN	MUSIC	9/21/1976	A	College of Musical Arts	UGRD	UQTR	
BGSUN	MUSIC	9/24/1968	A	School of Music	UGRD	UQTR	
BGSUN	MUSIC	1/1/1910	A	School of Music	UGRD	USEM	

The row with the highest EFFDT that is not in the future contains the information current for today. In the example above, since August 29, 1982, the name of the music program has been "College of Musical Arts" and the academic calendar was based on semesters (USEM). From September 21, 1976 through August 28, 1982, the name was the same, but the calendar was based on quarters (UQTR).

If there is an Effective Date field in a record, the Effective Date is always part of the key, and it is the last part of the key unless there is also an Effective Sequence. The Effective Date applies to all of the key fields preceding it. In the example, EFFDT applies to the combination of INSTITUTION and ACAD_PROG. The Effective Date for program MUSIC does not indicate when data for effective for program ARTSC, program BUSN, etc.

By convention, the Effective Date used for the first instance of something that is effective dated is 1/1/1900 for base PeopleSoft data and 1/1/1910 for data added by BGSU.

If it is likely for there to be several changes to something on the same day, an Effective Sequence field is included in the record. This is a number that starts at 0 on a particular day and increases by 1 for each change to that something (identified by key) on that day. The most current information has the highest non-future EFFDT and the highest EFFSEQ *for that EFFDT*.

Again, an example can more clearly demonstrate this concept.

EMPLID	EMPL_RCD	EFFDT	EFFSEQ	DEPTID	JOBCODE	SUPERVISOR_ID	HR_STATUS	ACTION
9106	0	4/20/2008	0	310200	600001	2164	I	TER
9106	0	7/29/2007	1	310200	600001	2164	A	DTA
9106	0	7/29/2007	0	071100	600001	1316	A	HIR

Employee 9106 was hired on 7/29/2007, into department 071100 under supervisor 1316; this is shown in the row with EFFDT of 7/29/2007 and EFFSEQ of 0. Later that day, the employee was transferred to department 310200 with supervisor 2164; this is shown in the row with EFFDT of 7/29/2007 and EFFSEQ of 1. This assignment was effective until the employee's termination on 4/20/2008.

There may be cases in which something is deactivated or will be deactivated or is added to the database before it will become effective. These cases are handled by using a Status as of Effective Date field. Consider these two examples from ACAD_PLAN_TBL, which contains information on academic plans (majors).

INSTITUTION	ACAD_PLAN	EFFDT	EFF_STATUS	DESCR	ACAD_PLAN_TYPE	ACAD_PROG	DEGREE
BGSUN	AERO-BSTC	8/25/2025	I	Aerotechnology	MAJ	TECH	BSTC
BGSUN	AERO-BSTC	1/1/1910	A	Aerotechnology	MAJ	TECH	BSTC

The above rows indicate that the Aerotechnology major, with a Bachelor of Science in Technology degree (DEGREE = 'BSTC') upon graduation, is active (EFF_STATUS = 'A') until 8/24/2025. On 8/25/2025, it will become inactive (EFF_STATUS = 'I'). This means that AERO-BSTC will be inactive *in the future* but is active now. (This is an example and does not necessarily reflect the fate of the Aerotechnology major.)

INSTITUTION	ACAD_PLAN	EFFDT	EFF_STATUS	DESCR	ACAD_PLAN_TYPE	ACAD_PROG	DEGREE
BGSUN	ENVHLTH-BS	6/18/2011	I	Environmental Health	MAJ	ARTSC	BS
BGSUN	ENVHLTH-BS	1/1/1915	A	Environmental Health	MAJ	ARTSC	BS

The second example shows that plan ENVHLTH-BS – a Bachelor of Science for Environmental Health in the College of Arts and Sciences – was active until 6/17/2011 and was discontinued on 6/18/2011, placing it in an inactive status. This means that ENVHLTH-BS is *currently* inactive. (There is an equivalent ENVHBSENVH under program HLTH that is still active.)

Using Effective Data in Queries

The most common use of effective date logic in queries is to report only active data current at the time the query is run. This may be referred to as the "maximum non-future effective date."

Writing effective date logic manually involves adding a subquery on the same record as in the main query, joining on all key fields except for the date. This can be cumbersome for records with many fields in the key.

Query Manager makes this unnecessary! When you add a record that has an Effective Date field to a query, an effective date criterion is automatically added to the query. A message stating this appears as soon as you add that record. You can see this criterion in the Criteria tab as "Eff Date <= Current Date." You can change the criterion to look at the date in a field, expression, or a specific date of your choice. You can also show rows having the first effective date or last effective date regardless of whether the effective date is in the future.

If the record also has an Effective Sequence field, the criterion will be "Eff Date <= Current Date (EffSeq = Last)," indicating that only the last row created on the effective date will be included in the results. You can edit this criterion to use the first row instead of the last or to show all rows regardless of Effective Sequence.

If you accidentally delete a criterion on Effective Date, you can add one manually. If you choose EFFDT as a field in the criterion, you can choose from special Condition Types that apply only to effective dates:

- Eff Date < – effective date is less than the selected date
- Eff Date <= – effective date is less than or equal to the selected date
- Eff Date > – effective date is greater than the selected date
- Eff Date >= – effective date is greater than or equal to the selected date
- First Eff Date – effective date is the earliest for the key
- Last Eff Date – effective date is the latest for the key

The selected date can be the current date, a constant, a value in a field, or the result of an expression. Recall that effective dates are tied to keys, so "First Eff Date" refers to the row having the earliest effective date for all rows having the same values in their key fields (except Effective Date and Effective Sequence).

Query Manager does not automatically add criteria on Status of Effective Date. If you are interested in only active or only inactive rows, you must manually add the criterion on EFF_STATUS to your query. There are just two possible values of EFF_STATUS: 'A' for active and 'I' for inactive.

Query Organization

As the number of queries that you work with grows, you may want to group them by purpose, department, or other characteristics. You might also want to send them to other users, rename them, or delete old ones.

Query Manager enables you to organize your queries in a manner similar to how you may organize your files with Windows 7 and Mac OS X. However, there are some important differences.

Files in a file system are uniquely identified by an internal ID. Two or more files can have the same name, though such files usually must be stored in different folders. Queries are uniquely identified by name in a storage area; two queries in the same user's storage space cannot share the same name even if they are in different folders. Two queries can only have the same name if they are in two different users' private storage. No private query can have the same name as a public query.

Queries may be grouped into folders. This can aid in finding queries in a large institution, since you can search for only those queries that are in a particular folder. This is similar to storing files in different folders or directories in a file system. However, unlike with file systems, you cannot store a folder inside another folder. With Query Manager, there are only two levels of folders: inside a folder and not inside a folder.

With a file system, you must create new folders and delete unneeded folders manually. Query Manager implicitly creates a folder when you save or move a query to a folder that does not already exist. Query Manager also automatically deletes a folder when you remove the last query from that folder.

Copy a Query to a User

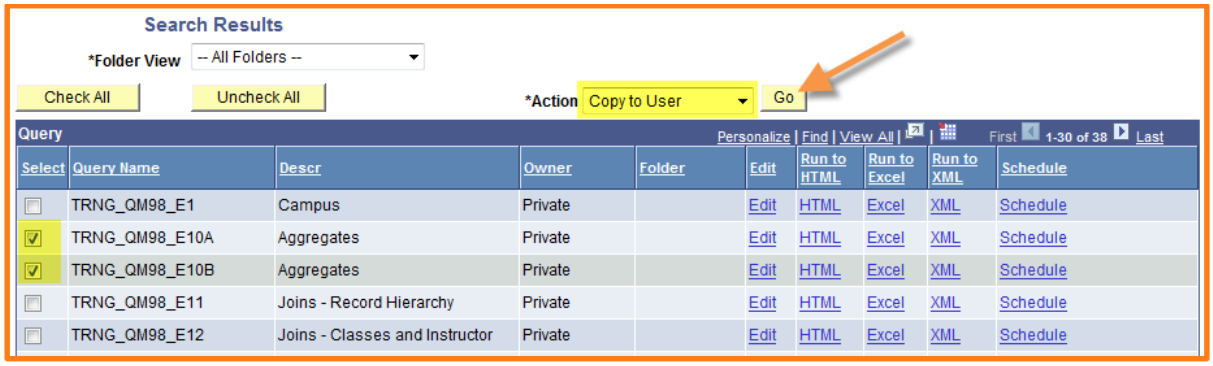
It may be useful to send a query to another user if that user needs to make minor modifications before running it, to see how it was built, or to run the query with different row-level security than that of the first user. Query Manager enables you to copy a query to another user's storage space.

To copy a query to another user, do the following:

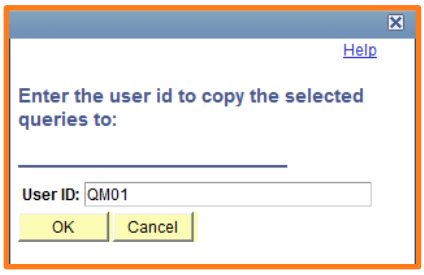
- From the Query Manager search page, search for the query to be copied.

In the row for the query in the Search Results area, **check** the **Select checkbox**. (You may copy multiple queries by checking the checkbox corresponding to each query to be copied.)

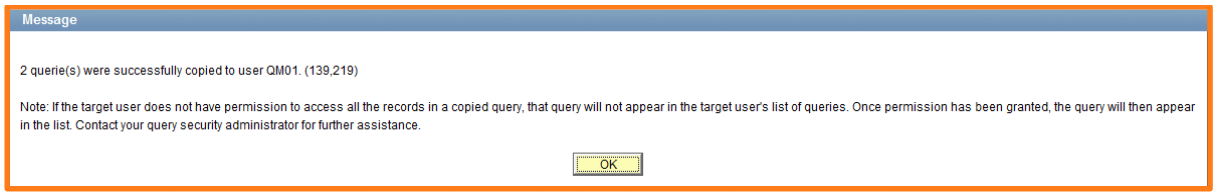
- From the Action dropdown, choose Copy to User.
- Click the Go button.



- Enter the User ID of the user who will receive the query.
- Click the OK button.



If the copy is successful, you will receive a message confirming this.



If the receiving user does not have access to the records in the copied query, that query will not appear when the receiving user searches for queries.

A public query cannot be copied to another user because this would result in a private query having the same name as a public query. If you want to copy a public query, the receiving user must edit the query and save it to his or her private storage. (See the *Save a Query with a New Name* segment.)

A query cannot be copied to another user if that user has a query with the same name; you cannot overwrite an existing query by copying one.

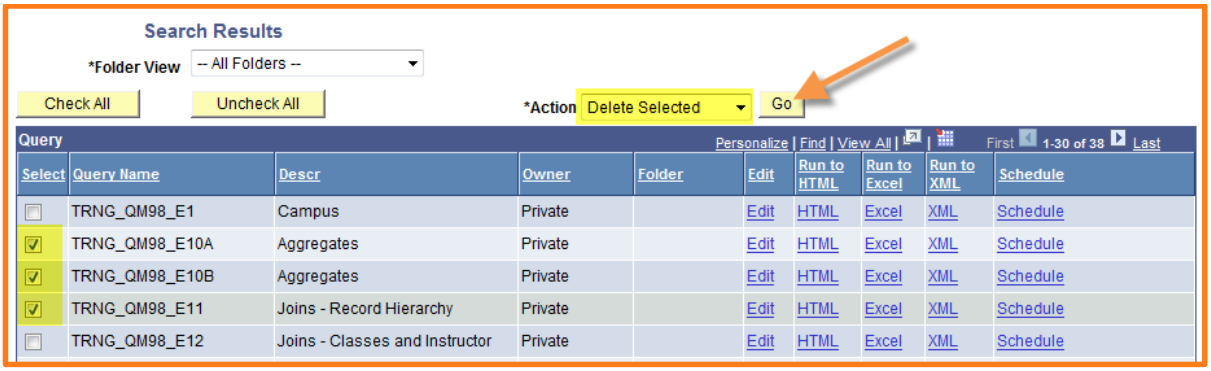
Delete a Query

Deleting unneeded queries, such as those that are out of date, or created as a test, can be done easily through Query Manager. To delete a query, do the following:

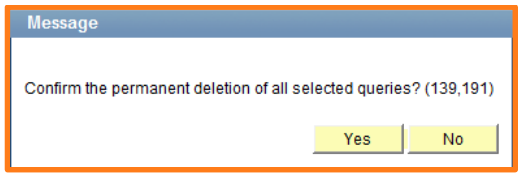
- From the Query Manager search page, search for the query to be deleted.

In the row for the query in the Search Results area, check the Select checkbox. (You may delete multiple queries by checking the checkbox corresponding to each query to be deleted.)

- From the Action dropdown, choose Delete Selected.
- Click the Go button.



When asked to confirm the deletion, click the Yes button.



If the last query is deleted from a folder, that folder is also deleted.

It is possible for you to delete a public query. Be **very** careful when selecting queries to delete since you may accidentally delete a query someone else needs!

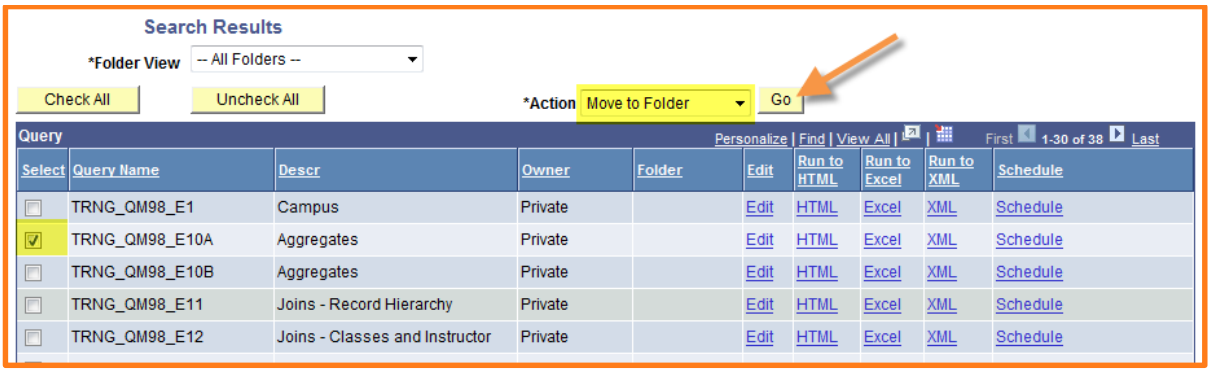
Move a Query to a Folder

It can be useful to group queries with a similar purpose or for the same department or college into a folder. Both Query Viewer and Query Manager allow you to search for queries by a folder name and filter the results of a search for queries by a folder name. To move a query to another folder, do the following:

- From the Query Manager search page, search for the query to be moved.

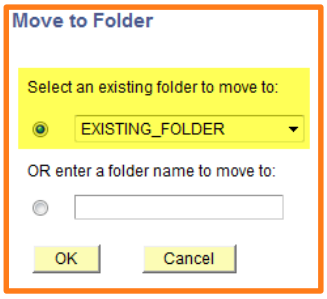
In the row for the query in the Search Results area, check the Select checkbox. (You may move multiple queries by checking the checkbox corresponding to each query to be moved.)

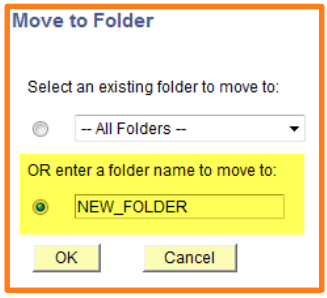
- From the Action dropdown, **choose Move to Folder**.
- **Click** the **Go** button.



A Move to Folder page appears.

- To move the selected query or queries to an existing folder, **click** the "**Select an existing folder to move to**" radio button and choose a folder from the dropdown.





- To move the selected query or queries to a new folder, **click** the "**OR enter a folder name to move to**" radio button and enter a folder name in the text box. A folder name may be at most 18 characters long.

*Make sure to ***both*** click a radio button *and* choose a folder

- **Click** the **OK** button.

*Query Manager will take the action indicated by the radio button. If you enter a new folder name but do not click the second radio button, the query will be moved to whatever existing folder is showing in the dropdown!

If the last query in a folder is moved out of that folder, the folder is deleted. If a new folder name is entered and a query is moved to it, a folder with that name is automatically created.

To move a query so that it is not in any folder, move it to a folder with a blank name.

Private queries remain private after being moved. Similarly, public queries remain public after being moved.

It is possible for you to move a public query to another folder. Be very careful when selecting queries to move since you may accidentally move a query someone else needs!

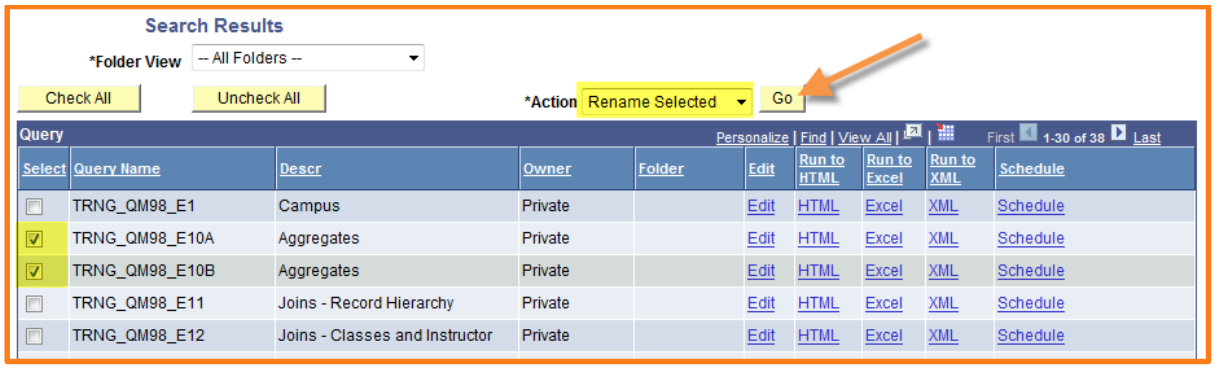
Rename a Query

On occasion, you may want to change the name of a query to make it easier to identify, correct a spelling mistake, or reduce confusion with another query. To rename a query, do the following:

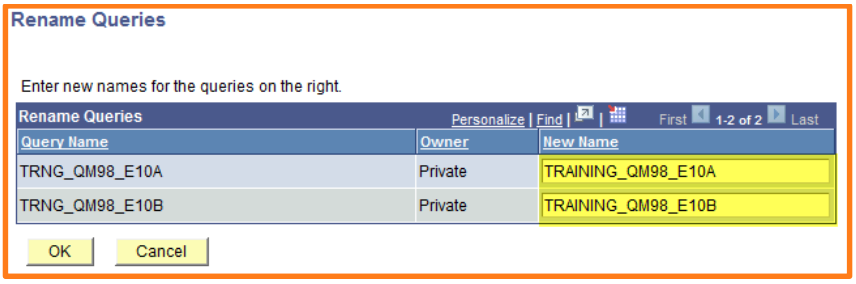
- From the Query Manager search page, search for the query to be renamed.

In the row for the query in the Search Results area, check the Select checkbox. (You may rename multiple queries by checking the checkbox corresponding to each query to be renamed.)

- From the Action dropdown, **choose Rename Selected**.
- **Click** the **Go** button.



- **Ente**r a **new name** for each selected query next to the old name of each query.
- **Click** the **OK** button.



Renaming a query does not change who owns the query or the folder in which it is stored.

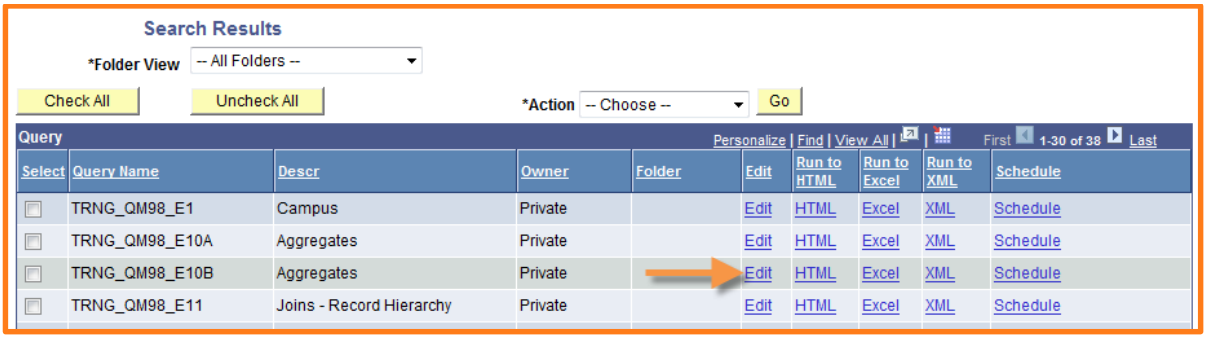
You cannot rename a query such that it has the same name as a private query in your storage area or a public query.

It is possible to rename a public query. Be very careful when selecting queries to rename since you may accidentally rename a query someone else needs.

Save a Query with a New Name

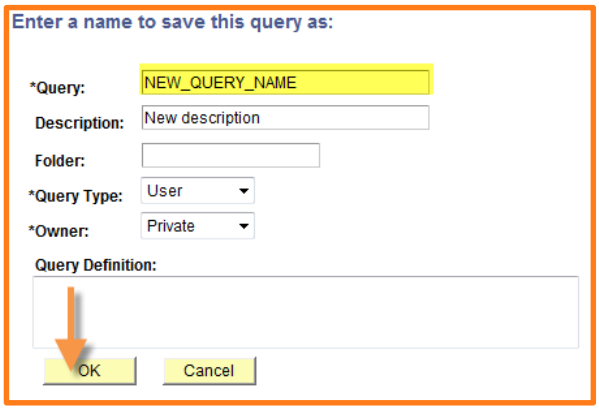
There are instances in which you may want to give a query a new name while keeping the existing query intact instead of renaming it. These include testing changes to a query and retaining historical versions of a query. To save a query with a new name, do the following:

- From the Query Manager search page, search for the query to be saved with a new name.
- In the row for the query in the Search Results area, **click** the **Edit** link.



At the bottom of any tab page except Run, **click** the **Save As** link.

- **Enter** a **new name** in the Query text box.
- **Enter** a new **description** or **folder** name or change the **Owner** between Private and Public if desired.
- **Click** the **OK** button.



You can overwrite an existing query by saving a query with the same name as an existing query. If you attempt this, you will be told a query with that name exists and asked if you want to continue. Click Yes to overwrite or No to abandon the save.

You cannot save a query to your private storage such that the query has the same name as a public query.

No change is made to the original query.

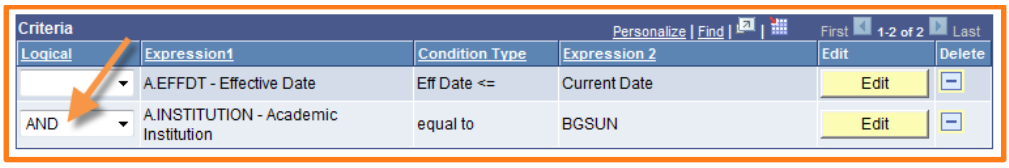
It is possible for you to overwrite a public query by saving a query with the same name as another public query. Be very careful when saving a public query since you may accidentally overwrite a query someone else needs.

Grouping Criteria and the OR operator

- On some occasions, you will need to write a query that returns rows that meet some criteria, but not all of them. For instance, you may be asked to limit the results to those for which the program status is either 'AC' or 'LA'. This could be implemented using the criterion "PROG_STATUS in list 'AC', 'LA'", but if the criteria involve multiple fields or cannot be enumerated in a list, you will need to create multiple criteria and link them with the OR operator.
- When you add criteria to a query, by default they are linked with the AND operator. This means that in order for a row to be included in the results, criterion 1 *and* criterion 2 *and* criterion 3 etc. must *all* be met. With an OR operator, if criterion 1 is true *or* criterion 2 is true, then the row will be included (assuming the other criteria are also met). Note that if both criterion 1 and criterion 2 are true, the row will still be in the results; it is only required that one of the criteria be met.

Choosing Logical Operators

- To change the operator linking two criteria, go to the Criteria tab and select the operator from the Logical column. For the first criterion, you can only select NOT to negate that criterion. For the other criteria, you can choose AND, AND NOT, OR, and OR NOT. Typically you will use this to change from AND to OR.
- In the screen shot below, the AND operator links the criterion above (A.EFFDT <= Current Date) and the criterion in the same row (A.INSTITUTION equal to BGSUN).



- The operators are called "Logical" because they operate on values that are either *true* or *false*. Each criterion has a comparison that produces a true result or a false result. In a row in which INSTITUTION is "BGSUN," the criterion A.INSTITUTION equal to BGSUN evaluates to *true*.

Grouping Criteria

- There is an important consideration when using both the AND and OR operators in a query. As in mathematics, there is a defined order of operations. The AND operators are given higher priority than OR operators. The AND operator is roughly equivalent to multiplication and the OR operator is roughly equivalent to addition in this sense. You can use parentheses to affect how the criteria are grouped, usually with the goal of combining the criteria linked with OR operators.

- Consider an example in which you are asked to list the academic program (ACAD_PROG record) information about students who are at BGSU (INSTITUTION = 'BGSUN') in the Arts and Sciences program (ACAD_PROG = 'ARTSC') and were either admitted in Fall 2012 (ADMIT_TERM = '2128') *or* completed the program in Spring 2012 (COMPLETION_TERM = '2122'). Note that academic program data is effective dated.

- Here is a first attempt at building these criteria in our query:

Criteria — Personalize | Find | First 1-5 of 5 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	Edit	
AND	A.INSTITUTION - Academic Institution	equal to	BGSUN	Edit	
AND	A.ACAD_PROG - Academic Program	equal to	ARTSC	Edit	
AND	A.ADMIT_TERM - Admit Term	equal to	2128	Edit	
OR	A.COMPLETION_TERM - Completion Term	equal to	2122	Edit	

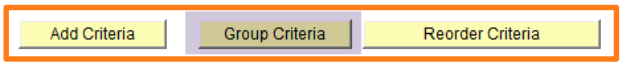
Below is an excerpt from the results returned by the query:

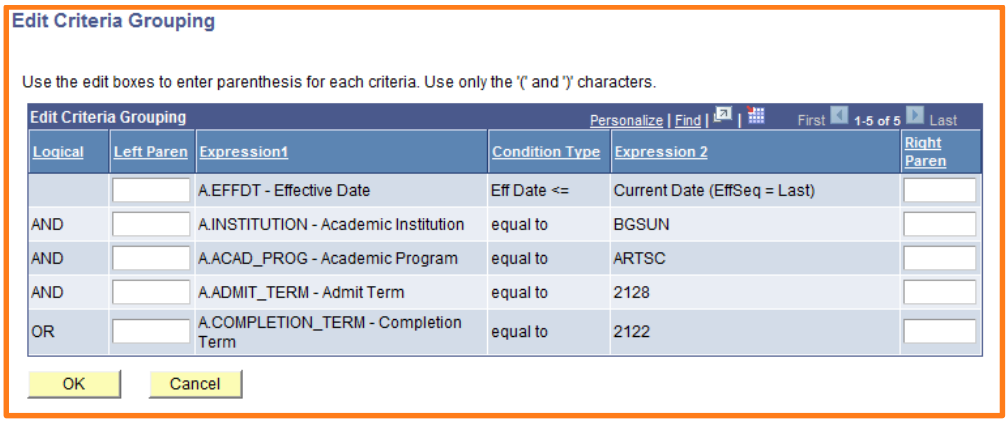
View All | Rerun Query | Download to Excel | Download to XML — First 201-300 of 3677 Last

	ID	Career	Career Nbr	Eff Date	Sequence	Acad Prog	Status	Admit Term	Compl Term
201	[redacted]843	UGRD	0	05/25/2012	1	ARTSC	DC	2128	
202	[redacted]691	UGRD	1	05/05/2012	1	HLTH	CM	2088	2122
203	[redacted]875	UGRD	0	05/05/2012	1	FIRE	CM	1972	2122
204	[redacted]729	UGRD	0	06/15/2012	1	ARTSC	AC	2128	
205	[redacted]753	UGRD	0	06/15/2012	1	ARTSC	AC	2128	

- Row 201 matches what was expected since the program is ARTSC and the admit term is Fall 2012. However, row 202 doesn't match what was intended; while the completion term is Spring 2012, the program is HLTH. This row was included because it meets the COMPLETION_TERM equal to 2122 criterion. Recall that in a case of criteria linked by an OR operator, either what is before the OR or after the OR must be true to be included in the results. There are four criteria linked by AND operators before the OR, so either these four criteria *in combination* must be true or the one criterion after the OR have to be met for the row to appear among the results.

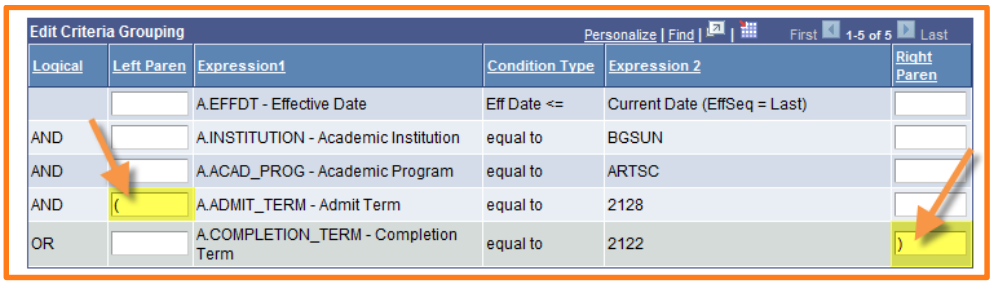
- In order to have the query worked as intended – that the results are all in the ARTSC program and that either the admit term is Fall 2012 or the completion term is Spring 2012 – parentheses are needed. To add parentheses to a query, go to the Criteria tab and click the Group Criteria button. The Edit Criteria Grouping page appears.



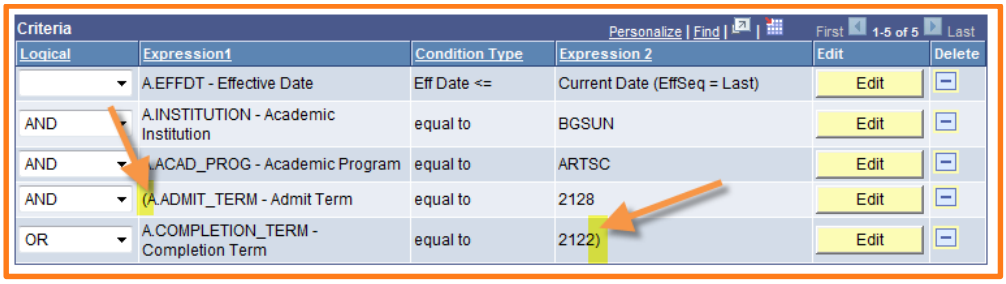


- In the column between Logical and Expression 1, you enter an opening parenthesis "(" before the first criterion in the group that you want to create. In the column to the right of Expression 2, you enter a closing parenthesis ")" after the last criterion in the group.

- In this case, we are creating a group containing the criterion on ADMIT_TERM and the criterion on COMPLETION_TERM. Parentheses are added are indicated in the screen shot below:



- Click OK to confirm the changes to the grouping. The parentheses are displayed on the Criteria page.

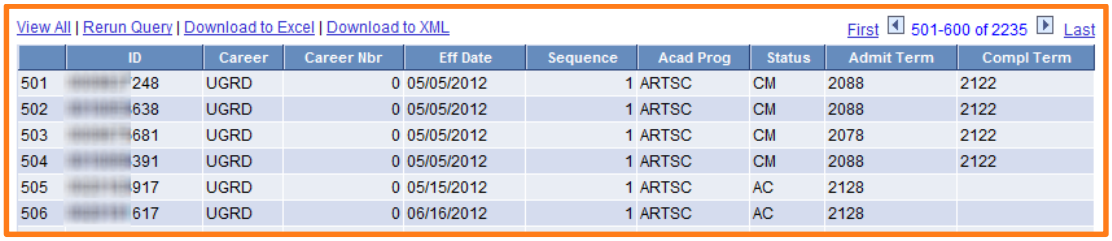


Criteria — Personalize | Find | First 1-5 of 5 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	Edit	
AND	A.INSTITUTION - Academic Institution	equal to	BGSUN	Edit	
AND	A.ACAD_PROG - Academic Program	equal to	ARTSC	Edit	
AND	(A.ADMIT_TERM - Admit Term	equal to	2128	Edit	
OR	A.COMPLETION_TERM - Completion Term	equal to	2122)	Edit	

- This shows that the two criteria on ADMIT_TERM and COMPLETION_TERM are now to be evaluated first. If either ADMIT_TERM is equal to 2128 or COMPLETION_TERM is equal to 2122, the result of the OR operation will be *true*. If neither of these are true, the result of the OR operation will be *false*, which will cause the row to be excluded.

- Running the modified query produces results like the following:



View All | Rerun Query | Download to Excel | Download to XML — First 501-600 of 2235 Last

	ID	Career	Career Nbr	Eff Date	Sequence	Acad Prog	Status	Admit Term	Compl Term
501	[redacted]248	UGRD	0	05/05/2012	1	ARTSC	CM	2088	2122
502	[redacted]638	UGRD	0	05/05/2012	1	ARTSC	CM	2088	2122
503	[redacted]681	UGRD	0	05/05/2012	1	ARTSC	CM	2078	2122
504	[redacted]391	UGRD	0	05/05/2012	1	ARTSC	CM	2088	2122
505	[redacted]917	UGRD	0	05/15/2012	1	ARTSC	AC	2128	
506	[redacted]617	UGRD	0	06/16/2012	1	ARTSC	AC	2128	

- Observe that all of the rows in the result set have either the admit term of Fall 2012 or the completion term of Spring 2012, and that no matter which of the two terms match the criteria, the academic program is ARTSC.

- In general, if you are going to use the OR operator in a query, you will likely need to group criteria together using parentheses.

Wildcards

In the *Edit Criteria – Part 2* segment, you learned about the "like" condition type, which allows you to find rows in which a text field is "like" a word or phrase, meaning the field contains that word or phrase. To use like in this manner, you must employ wildcards. (If you have no wildcards in your constant, then like is the same as equal!)

There are two wildcards that you can use in queries. The percent sign (%) is used to substitute for *zero or more* characters. The underscore (_) is used to substitute for *any single character*.

The use of wildcards is best demonstrated through examples. Consider the following search strings that use wildcards and some string that would match them.

Search String	Value in Field	Match?	Reason
abc%	abc	Yes	Starts with "abc" and is followed by zero characters
	abcd	Yes	Starts with "abc" and is followed by one character
	abcdefgh	Yes	Starts with "abc" and is followed by many characters
	ab	No	Does not start with "abc"
	abz	No	Does not start with "abc"
	aabc	No	Does not start with "abc" (even though "abc" is in the value)
%def	def	Yes	Begins with no characters and ends with "def"
	cdef	Yes	Begins with one character and ends with "def"
	abcdef	Yes	Begins with many characters and ends with "def"
	ef	No	Does not end with "def"
	ref	No	Does not end with "ref"
	deff	No	Does not end with "def" (even though "def" is in the value)
%ghi%	ghi	Yes	Begins with no characters, followed by "ghi," followed by no characters
	efghi	Yes	Begins with several characters, followed by "ghi," followed by no characters
	ghijk	Yes	Begins with no characters, followed by "ghi," followed by no characters
	efghijk	Yes	Begins with several characters, followed by "ghi," followed by no characters
	efgghiijk	Yes	Begins with "efg," followed by "ghi," ending with "ijk"
	efgjk	No	Does not contain "ghi"
	efgpijk	No	Does not contain "ghi"
	gh	No	Does not contain "ghi"
a_c	abc	Yes	"a" followed by one character followed by "c"
	a7c	Yes	"a" followed by one character followed by "c"
	ac	No	No characters between "a" and "c"
	abbc	No	Too many characters between "a" and "c"

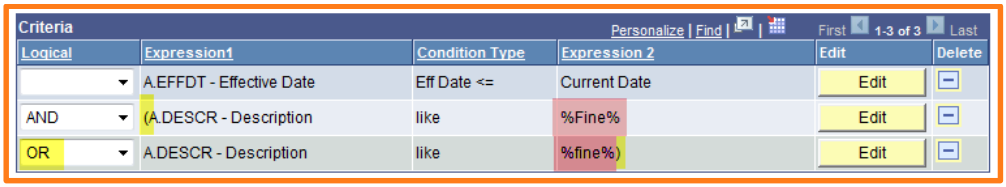
Criteria and Case-Sensitive Data

Criteria that involve textual data use case-sensitive comparisons. This means that "hello" and "Hello" are two distinct values that are *not* considered to be equal. If your query has a criterion that DESCR is equal to "Psychology department" but the row for this department has a DESCR of "Psychology Department" (note the capital D), the row will not be included in the results.

By convention, codes such as those used for statuses and types are fully uppercase, to prevent issues of case sensitivity from arising and from having codes that are the same except for case. (It would be confusing to have "AC," "Ac," and "ac" all as valid options having different meanings for the same field!) In queries that use criteria involving codes, make sure to enter the code using all uppercase letters. (If the field is defined in PeopleSoft as allowing only uppercase letters, then your value will be transformed to uppercase. However, not every field that has codes is defined in this manner.)

Consider exercise 6B, in which you are obtaining a list of item types that have "Fine" in the description. The current data is set up such that "Fine" is always capitalized. What if this was not the case? How might you work around this?

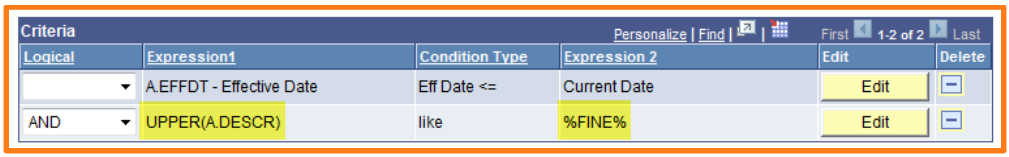
One alternative is to have two criteria, one which checks if DESCR contains "Fine" and one which checks if DESCR contains "fine."


Criteria — Personalize | Find | First 1-3 of 3 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.EFFDT - Effective Date	Eff Date <=	Current Date	Edit	
AND	(A.DESCR - Description	like	%Fine%	Edit	
OR	A.DESCR - Description	like	%fine%)	Edit	

Note that because of the other criteria in the query, it is necessary to group the two criteria on DESCR in parentheses, and use the OR operator instead of the AND operator on those criteria. (We want rows in which either the description contains "Fine" *or* the description contains "fine.")

Another alternative is to use an expression to convert DESCR to a known case – either uppercase or lowercase – and compare that against a constant in the same case.


Criteria — Personalize | Find | First 1-2 of 2 Last

Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.EFFDT - Effective Date	Eff Date <=	Current Date	Edit	
AND	UPPER(A.DESCR)	like	%FINE%	Edit	

In this example, we check if the uppercase version of DESCR, returned by the expression UPPER(A.DESCR), contains the text "FINE," which is also in uppercase. Cases of "Fine" and "fine" will be matched since the comparison will be against their uppercase equivalent, which is "FINE" for both.

Key Fields

An important aspect of relational database structure is that information about an object is separated into multiple tables, with each table having data about a certain set of characteristics. For instance, in PeopleSoft, the PERSON record is the basis of personal information, the NAMES record contains the names of each person, and the ADDRESSES record has the postal and physical addresses of each person. The data about one person is spread in PERSON, NAMES, ADDRESSES, and other records.

In order to collect the data about a person, the relevant records must be joined together in a query. The database management system needs to know how to find the data in one record based on data in another. This is done through the use of *key fields*.

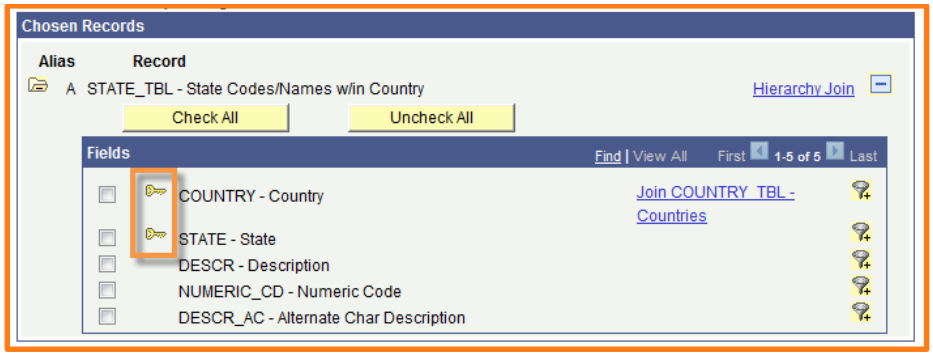
Primary Keys

A *primary key* is a field or set of fields that uniquely identifies an object. No other object can have the same primary key as another. No two cars have the same vehicle identification number (VIN). No two dollar bills have the same serial number. In PeopleSoft, no two people can have the same employee ID number, so the primary key of the PERSON record is EMPLID.

A primary key may contain multiple fields. For instance, the primary key of STATE_TBL is the combination of COUNTRY and STATE. The state abbreviation or code alone cannot uniquely identify a state or province; the code "MI" represents the province of Misiones in Argentina, the province of Milano in Italy, the state of Michigan in the United States, and the state of Miranda in Venezuela. However, all countries have a unique code, and no country has two states with the same code, so the country and state codes together can uniquely identify a state. The province of Misiones is identified by country code ARG and state code MI whereas Michigan is identified by country code USA and state code MI.

When tables have a parent-child relationship, the primary key of the parent is contained within the primary key of the child. This enables all children rows of a parent row to be found by joining the tables on the primary key fields. A query for gathering information from both COUNTRY_TBL and STATE_TBL would join these two tables on COUNTRY; the COUNTRY field is the primary key of COUNTRY_TBL and is part of the key of STATE_TBL, the child table of COUNTRY_TBL. When using a hierarchy join in Query Manager, the parent and child records are automatically joined on the fields their primary keys have in common.

The primary key is generally the first field or fields in the list of fields of a record. In Query Manager, the primary key is indicated on the Query tab by a key icon.

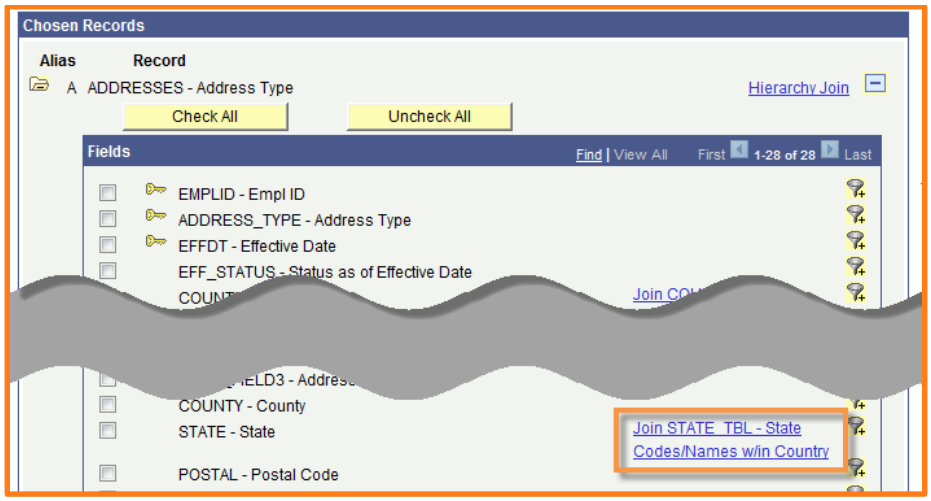


Foreign Keys

A *foreign key* is a field or set of fields that refer to the primary key of another table. These are typically used for restricting data in a field to only allowable values and to reduce data duplication. For example, in the ADDRESSES record, there is both a COUNTRY and a STATE

field; these in combination are a foreign key to STATE_TBL. It will not be possible to enter a country/state combination that does not exist in STATE_TBL into an address.

In order to get information about a state, such as its full name, one would have to join a record to STATE_TBL using both COUNTRY and STATE. A related record join in Query Manager automatically joins the foreign key of one record to the primary key of another record.



Keep in mind that when you are manually joining two records by adding criteria that you will likely need to add them for every field in the foreign key. In the exercise in which DEPT_TBL was joined to PERSON_NAME, you had to add a criterion that requires that MANAGER_ID of DEPT_TBL was the same as EMPLID of PERSON_NAME; EMPLID is the primary key of PERSON_NAME.

Keys with Effective Dates

The PeopleSoft database design includes the concept of effective data (see *Effective Date, Effective Sequence, and Effective Status* elsewhere in this manual), in which information about an object is recorded at different points in time, and the history about that object can be retrieved.

When effective data fields are in a record, they are considered to be part of the primary key. This is because the date (and sequence, if used) helps uniquely identify the *information* about an object at a point in time.

Consider the following rows of ACAD_PROG, which contains data about the programs in which a student is participating and has participated.

EMPLID	ACAD_ CAREER	STDNT_ CAR_NBR	EFFDT	EFFSEQ	INSTITUTION	ACAD_PROG	PROG_STATUS	PROG_ACTION
1379	UGRD	0	1/12/2012	0	BGSUN	EDUC	AC	PLNC
1379	UGRD	0	1/7/2011	0	BGSUN	EDUC	AC	PRGC
1379	UGRD	0	8/20/2010	0	BGSUN	ACEN	AC	ACTV

Student 1379 entered the Academic Enhancement program on 8/20/2010, then switched to Education & Human Development on 1/7/2011, then changed a plan within that program on 1/12/2012. All three of the rows are about the same student's participation, but represent different states of that participation by date. The EFFDT and EFFSEQ fields help uniquely identify the state of participation – by using these fields, one can find out which program a student was in at a given point in time.

The full primary key of ACAD_PROG is the combination of EMPLID, ACAD_CAREER, STDNT_CAR_NBR, EFFDT, and EFFSEQ.

When joining records that have effective data fields, one must be careful to consider whether or not it is appropriate to join on EFFDT (and EFFSEQ, if present). If a parent has effective data fields and it is being joined to a child record, then the join must include EFFDT. For example, the service indicator code record SRVC_IND_CD_TBL has a key of INSTITUTION, SRVC_IND_CD, and EFFDT. The reasons belonging to service indicators are in SRVC_IN_RSN_TBL and have a key of INSTITUTION, SRVC_IND_CD, EFFDT, and SRVC_IND_REASON. To join information on codes and reasons together, one must join on INSTITUTION, SRVC_IND_CD, and EFFDT, to ensure that the reason data is linked to the proper code.

However, when there is not a parent-child relationship, then EFFDT will not be part of the join. Data about academic plans is in ACAD_PLAN_TBL and the key is INSTITUTION, ACAD_PLAN, and EFFDT. Data about subplans is in ACAD_SUBPLN_TBL and the key is INSTITUTION, ACAD_PLAN, ACAD_SUB_PLAN, and EFFDT. The key of ACAD_PLAN_TBL is *not* contained in the key of ACAD_SUBPLN_TBL; the effective date in ACAD_PLAN_TBL is

about the plan whereas the effective date in ACAD_SUBPLN_TBL is about the subplan. To join these two records, use INSTITUTION and ACAD_PLAN alone.

Query Manager automatically takes care of these considerations. When performing a hierarchy join, the EFFDT (and EFFSEQ, if needed) field used as part of the join. When performing a related record join, EFFDT will be left out of the join. With a manual join (such as between ACAD_PLAN_TBL and ACAD_SUBPLN_TBL), Query Manager will not detect EFFDT as a possible field for the join criteria that can be automatically added.

What is a View?

A "view" is a special kind of query. It is a query that acts like a table in queries. Many views are delivered with PeopleSoft and developers can add their own. (However, this cannot be done through Query Manager.) Views are created for many purposes:

- Save a commonly-used subquery so that it does not need to be rewritten in several queries
- Reduce the number of fields returned from a table
- Return only current rows so that effective date logic need not be added to the main query
- Join data from many tables together into one convenient result set that can be used as if it were a table

In Query Manager, tables and views are both considered records, so you will see no difference in how they are used. By convention, views in PeopleSoft often have names ending with "VW," though some have names just containing "VW." You may find it beneficial to use views when they are available, as they can simplify your queries by hiding some details like fields, joins, and effective date logic.

Expressions

Recall from the Query Manager class that expressions are calculations, usually performed on fields from the records in your query, which produce a result. The result can be text, a date, or a number, and may be displayed and used in criteria as if it were a real field.

Reference fields in expressions by using their actual name, not their description and preceding the name with their alias and a period. For example, if you have SAL_GRADE_TBL aliased as "A" and want to use field MID_RT_ANNUAL in your expression, refer to the field as "A.MID_RT_ANNUAL" (without the quotes). Reference prompts by using a colon followed by the prompt number, such as ":1" (without the quotes).

One type of calculation that can be performed is mathematical. You can add, subtract, multiply and divide numeric values. The mathematical *operators* are:

- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)

Standard mathematical order of operations (multiplication and division are done before addition and subtraction; computation goes from left to right) is used. Parentheses can be used to group calculations together and force them to be done out of this order.

The only operator that applies to strings is concatenation, which is appending one string to the end of another string. The concatenation operator is the double pipe, ||. (The pipe character can usually be entered by holding down the Shift key while pressing the backslash \ key, which is normally above the Enter key.) For example, if A.STR1 contains 'ABC' and A.STR2 contains 'DEFG', then A.STR1 || A.STR2 results in 'ABCDEFG'.

There are not operators that perform calculations on two dates. You can add to or subtract from a date. The numeric value is translated into a number of days. If A.START_DT is 2/13/2012 then the expression A.START_DT + 7 results in 2/20/2012. If the numeric value is a decimal amount, then the time will also be affected. If A.START_DT is 2/13/2012 10:00 AM then the expression A.START_DT + 0.5 results in 2/13/2012 10:00 PM.

Expressions can also contain *functions*, which take some input values and return an output value. Functions operate similar to how they do in mathematics. The familiar square root operation can be considered a function; it takes an input value and returns the value that, when multiplied by itself, produces the original value. Some common mathematical functions are even written similar to how functions appear in queries; the sine function in trigonometry is written as *sin(x)*, indicating that it accepts one value.

Functions in expressions are of the form FUNCTION_NAME(arg1, arg2, …, argN), where FUNCTION_NAME is the name of the function and arg1, arg2, etc. are its arguments, which are the input values. When using a function in an expression, the name must be spelled exactly as expected, the argument list must be enclosed in parentheses, all arguments must be separated by commas, and all required arguments must be given. (There are sometimes optional arguments. In documentation these are frequently indicated by enclosing the argument in [square brackets].)

******* Note on functions: the functions listed below are for use in Oracle databases version 11 and higher; BGSU is currently using Oracle 11g. Other databases, such as DB/2, Informix, and Sybase, may support different functions.

String Functions

Here are some functions that can be used on strings:

CONCAT(str1, str2)

The CONCAT function concatenates str1 and str2, returning str1 followed immediately by str2. Equivalent to str1 || str2.

LENGTH(str1)

LENGTH returns the length of str1 in number of characters. For example, LENGTH('ABC DEF') returns 7 (the space is included).

LOWER(str1)

LOWER returns a copy of str1 with all uppercase letters changed to lowercase.

SUBSTR(str1, start[, how_many])

SUBSTR returns the substring – a part of a string – of str1, starting at position start. If how_many is not provided, all characters from start through the end of the string are returned. If how_many is provided, the substring beginning at position start and having a length of how_many are returned.

Examples:

- SUBSTR('ABCDEFG', 1, 3) returns 'ABC'
- SUBSTR('ABCDEFG', 3, 4) returns 'CDEF'
- SUBSTR('ABCDEFG', 4) returns 'DEFG'

UPPER(str1)

UPPER returns a copy of str1 with all lowercase letters changed to uppercase.

Numeric Functions

Here are some functions that can be used on numeric values:

ABS(n)

ABS returns the absolute value of n. If n is positive or zero, the result is n. If n is negative, the result is n without the negative sign. For example, ABS(-5) returns 5.

CEIL(n)

CEIL returns the ceiling of n, which is the smallest integer equal to or greater than n. For example, CEIL(5.2) is 6, since 5 is not equal to or greater than 5.2, but 6 is greater than 5.2. Note that CEIL(-4.4) is -4. CEIL is equivalent to rounding up.

FLOOR(n)

FLOOR returns the floor of n, which is the largest integer equal to or less than n. For example, FLOOR(8.7) is 8, since 9 is not equal to or less than 8.7, but 8 is less than 8.7. Note that FLOOR(-3.2) is -4. FLOOR is equivalent to rounding down.

ROUND(n[, dec])

ROUND returns n rounded to dec decimal places, or to the nearest integer if dec is not provided. If dec is negative, the value is rounded to dec powers of 10 (-1 to the nearest 10, -2 to the nearest 100, etc.).

Examples:

- ROUND(12.3) returns 12

- ROUND(12.8) returns 13
- ROUND(-12.3) returns -12
- ROUND(-12.8) returns -13
- ROUND(12345.6789, 2) returns 12345.68
- ROUND(12345.6789, 1) returns 12345.7
- ROUND(12345.6789, 0) returns 12346
- ROUND(12345.6789, -1) returns 12350
- ROUND(12345.6789, -2) returns 12300

SQRT(n)

SQRT returns the square root of n. The square root is the number, which multiplied by itself, results in n.

TRUNC(n[, dec])

TRUNC returns n truncated to dec decimal places. If dec is zero or not provided, all digits after the decimal are dropped. If dec is negative, ABS(dec) digits to the left of the decimal are replaced by zero, and all digits after the decimal are dropped.

Examples:

- TRUNC(12.3) returns 12
- TRUNC(12.8) returns 12
- TRUNC(-12.3) returns -12
- TRUNC(-12.8) returns -12
- TRUNC(12345.6789, 2) returns 12345.67
- TRUNC(12345.6789, 1) returns 12345.6
- TRUNC(12345.6789, 0) returns 12345
- TRUNC(12345.6789, -1) returns 12340
- TRUNC(12345.6789, -2) returns 12300

Date Functions

Here are some functions that can be used on dates:

ADD_MONTHS(dt, m)

ADD_MONTHS returns the date that is m months in the future of dt.

SYSDATE

SYSDATE returns the current system date and time. Note that it does not take any arguments.

Conversion Functions

Here are some functions that can be used to convert values from one type (date, number, string) to another:

TO_CHAR(dt[, fmt])

TO_CHAR converts a date or part of a date into a string. If fmt is not provided, the date is returned in the default format for the system, usually 'DD-MON-YY'. (In this format, 2/13/2009 would be '02-FEB-09'.)

The fmt argument is a string containing codes that instruct TO_CHAR how to format the date. There are many different format codes and several can be used at one time. Some of the format codes are:

- AM – the AM/PM indicator
- DD – day of the month
- HH – hour of the day (1-12)
- HH24 – hour of the day (0-23)
- MI – minute of the hour (0-59)
- MM – month (1-12, with 1 = January)
- MON – three-character abbreviation of the month
- SS – second of the minute (0-59)
- YY – two-digit year
- YYYY – four-digit year

Punctuation such as dashes, colons, and slashes are included in the result string in the positions given in the fmt.

Assume the current date is March 15, 2009 and the time is 4:25 PM. Here are several examples of how TO_CHAR would display this date:

- TO_CHAR(SYSDATE, 'MM/DD/YYYY') returns 03/15/2009
- TO_CHAR(SYSDATE, 'DD/MM/YYYY') returns 15/03/2009
- TO_CHAR(SYSDATE, 'YYYYMMDD') returns 20090315
- TO_CHAR(SYSDATE, 'HH:MI') returns 04:25
- TO_CHAR(SYSDATE, 'HH24:MI') returns 16:25
- TO_CHAR(SYSDATE, 'MM/DD/YYYY HH24:MI:SS') returns 03/15/2009 16:25:00
- TO_CHAR(SYSDATE, 'MM') returns 03 (this could be used for grouping in order to aggregate results by month!)

TO_CHAR(n[, fmt])

TO_CHAR can also convert a number to a string. If fmt is not provided, a default format will be used. This function is typically used to format numbers with commas, periods, currency symbols, and leading zeroes.

The fmt argument is a string containing codes that instruct TO_CHAR how to format the number. There are many different format codes and several can be used at one time. Some of the format codes are:

- $ – display a dollar sign

- 0 – display a leading zero
- 9 – display a digit
- , – display a comma
- . – display a period
- FM – disable leading and trailing spaces

Here are some examples of using TO_CHAR to format a number:

- TO_CHAR('1234567.89') returns 1234567.89
- TO_CHAR('1234567.89', '9,999,999') returns 1,234,568 (note the automatic rounding!); there is a leading space in front of the number to leave room for a negative sign
- TO_CHAR('123', '099999') returns 000123; there is a leading space in front of the number to leave room for a negative sign
- TO_CHAR('123', 'FM099999') returns 000123; there is no leading space
- TO_CHAR('12345.60', '$999,999.99') returns $12,345.60; there are two leading spaces, one to leave room for the sign and one because there is no hundred-thousands digit

Condition Functions

These are functions that return different values based on a set of conditions that you specify.

DECODE(x, val1, [val2, result2, …, valN, result] [, default])

DECODE is used to produce a result that is based on the result of x. The x, val1, result1, etc. arguments can be numbers or strings. Up to 127 comparisons (value/result pairs) are allowed if there is not default value and up to 136 comparisons are allowed if there is a default.

The value of x is compared against val1, val2, etc. through valN, one at a time. When a match occurs, the corresponding result argument is returned. If x matches val1, DECODE returns result1, if x matches val2, DECODE returns result2, and so on. If there is no match, the default argument is returned, provided it is given; otherwise the special value NULL is returned.

Consider the following examples:

- DECODE(A.COLOR, 'R', 'Red', 'G', 'Green', 'B', 'Blue')
  - If A.COLOR is “R” then DECODE returns “Red”
  - If A.COLOR is “G” then DECODE returns “Green”
  - If A.COLOR is “B” then DECODE returns “Blue”
  - If A.COLOR is “Y” then DECODE returns NULL
- DECODE(A.DIRECTION, 0, 'North', 90, 'East', 180, 'South', 270, 'West', 'Unknown')
  - If A.DIRECTION is 0 then DECODE returns “North”
  - If A.DIRECTION is 90 then DECODE returns “East”

- If A.DIRECTION is 180 then DECODE returns “South”
- If A.DIRECTION is 270 then DECODE returns “West”
- If A.DIRECTION is 360 then DECODE returns “Unknown”

You can also use the DECODE function to substitute a value when a field is blank. For date fields, an empty value is the special value NULL. To replace NULL values from the field A.DATE_FIELD with a default of date of 5/22/2012, use DECODE(A.DATE_FIELD, NULL, TO_DATE('05/22/2012', 'MM/DD/YYYY'), A.DATE_FIELD). Note the second A.DATE_FIELD as the last argument; this indicates that the original value of DATE_FIELD whenever DATE_FIELD is not NULL.

For text fields, an empty value is a single space. To replace a single space from the field A.TEXT_FIELD with the default string “N/A,” use DECODE(A.TEXT_FIELD, ' ', 'N/A', A.TEXT_FIELD). As with the previous example, the original value of the field is used when the field is not empty.

CASE Expression

A CASE expression is a powerful tool for producing one of several possible results based on conditions. It is more flexible than DECODE, which just maps input values to output values. CASE expressions allow *ranges* to be mapped to output values and for multiple input fields to influence the output.

CASE expressions have the following format:

```
CASE
   WHEN condition1 THEN result1
   [WHEN condition2 THEN result2]
   ...
   [ELSE default_result]
END
```

The CASE and END keywords are required. Each possible case is designated by the WHEN keyword, followed by a condition, the THEN keyword, and the result. The result must be a value or an expression that evaluates to a single value. The condition can be simple or complex, with many Boolean operators (such as AND and OR), as long as it evaluates to a result of true or false.

At least one WHEN clause must be given. All other WHEN clauses and the ELSE clause are optional. However, it is good practice to provide a default case with the ELSE clause in case unexpected values are encountered.

The conditions are examined in order from the first WHEN clause to the last WHEN clause. If condition1 is true, then the result of the CASE expression is set to result1. If condition1 is false, then condition2 is examined (if given); if condition2 is true, then the result of the CASE expression is set to result2. Each condition is examined until one that is true is

encountered. If none of the conditions are true, the result of the CASE expression is set to default_result if an ELSE clause is provided; otherwise, the expression is set to NULL.

Consider this example of determining if a state is a Great Lakes state:

```
CASE
   WHEN A.STATE IN ('OH', 'MI', 'IN', 'IL', 'WI', 'MN', 'NY', 'PA') THEN 'Y'
   ELSE 'N'
END
```

If the STATE field contains one out of several state codes (OH, MI, etc.) then the CASE expression evaluates to “Y”. Otherwise, it evaluates to “N”.

Next, consider a version of this expression that takes into account that the Canadian province of Ontario is considered part of the Great Lakes region:

```
CASE
   WHEN A.COUNTRY = 'CAN' AND A.STATE = 'ON' THEN 'Y'
   WHEN A.COUNTRY = 'USA' AND A.STATE IN ('OH', 'MI', 'IN', 'IL', 'WI', 'MN', 'NY',
'PA') THEN 'Y'
   ELSE 'N'
END
```

Each of the conditions includes an AND operator and examines data in two fields, COUNTRY and STATE. The conditions can include many conditions combined together.

The above could also be expressed as the following:

```
CASE
   WHEN
      (A.COUNTRY = 'CAN' AND A.STATE = 'ON') OR
      (A.COUNTRY = 'USA' AND A.STATE IN ('OH', 'MI', 'IN', 'IL', 'WI', 'MN', 'NY',
'PA')) THEN 'Y'
   ELSE 'N'
END
```

Here is an example with many conditions. This CASE expression translates a temperature in Fahrenheit into a description.

```
CASE
   WHEN A.TEMPERATURE < 30 THEN 'Very cold'
   WHEN A.TEMPERATURE BETWEEN 30 AND 44 THEN 'Cold'
   WHEN A.TEMPERATURE BETWEEN 45 AND 59 THEN 'Cool'
   WHEN A.TEMPERATURE BETWEEN 60 AND 69 THEN 'Mild'
   WHEN A.TEMPERATURE BETWEEN 70 AND 79 THEN 'Warm'
   WHEN A.TEMPERATURE BETWEEN 80 AND 89 THEN 'Hot'
   ELSE 'Very hot'
END
```

Assume that the value in A.TEMPERATURE is 53. The first condition, A.TEMPERATURE < 30, is examined and found to be false. The second condition, A.TEMPERATURE BETWEEN 30

AND 44, is examined and is also false. The third condition, A.TEMPERATURE BETWEEN 45 AND 59, is true. The result of the expression is “Cool”.

If the value of A.TEMPERATURE is above 89, none of the conditions in the WHEN clauses will be true, so the default value “Very hot” will be the result.

Note that since the order of the clauses is significant, the following CASE expression is equivalent:

```
CASE
   WHEN A.TEMPERATURE < 30 THEN 'Very cold'
   WHEN A.TEMPERATURE <= 44 THEN 'Cold'
   WHEN A.TEMPERATURE <= 59 THEN 'Cool'
   WHEN A.TEMPERATURE <= 69 THEN 'Mild'
   WHEN A.TEMPERATURE <= 79 THEN 'Warm'
   WHEN A.TEMPERATURE <= 89 THEN 'Hot'
   ELSE 'Very hot'
END
```

What Else is There?

There are many more functions and formats beyond those listed here. The Oracle Database SQL Language Reference (http://docs.oracle.com/cd/E11882_01/server.112/e41084/toc.htm) is comprehensive, but may be difficult to read for the non-technical user. If you plan to frequently use functions in your queries, you may want to find a book on the Oracle dialect of SQL. There are plenty of Oracle SQL books, both hardcopy and electronic, available at the BGSU Jerome Library and through OhioLINK.

Blank and Unknown Values

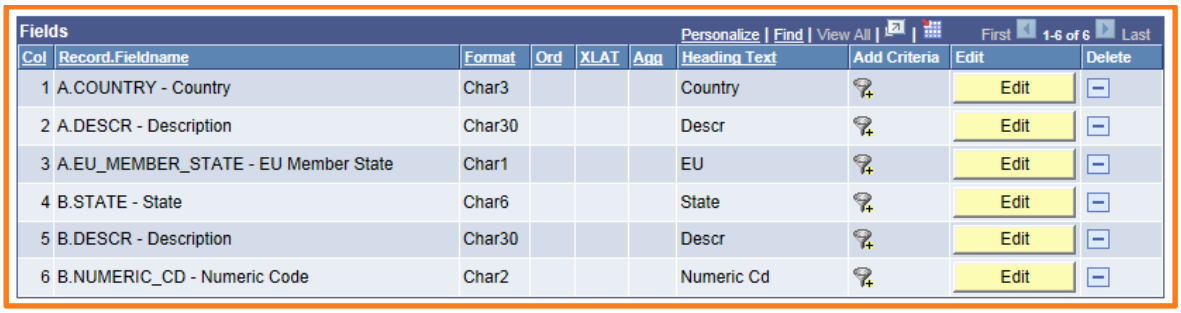
In many instances, not every field in a row will have a value. This may occur because a value is not required and one was not available to be entered or the user chose not to enter one. The special keyword NULL is used in SQL to designate that a field has no value or an unknown value.

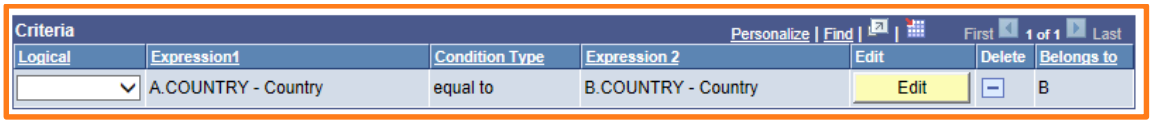
However, in PeopleSoft the convention for blank and unknown values is different. If a field is optional, then a blank field has a different representation based on its data type: date fields use NULL, numeric fields use zero, and text fields use a single space. This has the disadvantage of being unable to distinguish between a truly unknown value and a default value; for instance, if the value of a numeric field is zero, there is no way to know whether the value is actually zero or if the value is not known.

Outer joins introduce further complexity in dealing with blank fields. When two tables are connected in a left outer join, if there is no match in the right table for the join fields in the left table, then all fields selected from the right table are given a value of NULL. This occurs because the join takes place at the database level, outside of PeopleSoft, whereas indicators

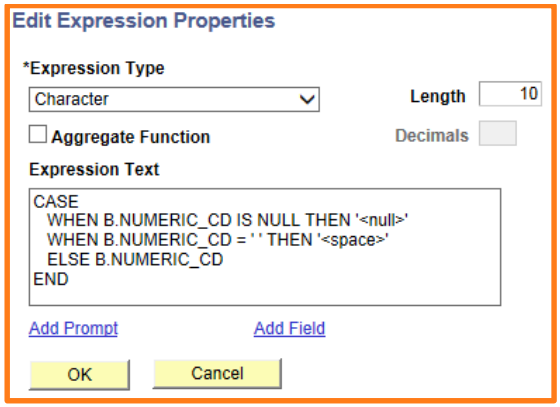
of unknown values like a single space and zero are applied within the PeopleSoft application. This means that the results of the outer join might have cases of both a NULL and a space in the same field in two different rows; the NULL indicates no matching row was found while a space indicates a match was found but the field value is unknown or empty.

Consider a case in which COUNTRY_TBL is outer-joined to STATE_TBL on the COUNTRY field. The COUNTRY, DESCR, and EU_MEMBER_STATE fields of COUNTRY and the STATE, DESCR, and NUMERIC_CD fields of STATE_TBL are displayed. The results will show all defined countries, along with each country's states or provinces; if there are no states or provinces, the base information about the country is still shown.





To demonstrate that a text field that is empty from a matched row has a single space but a text field from an unmatched row has NULL, an expression is built using CASE and this expression is used as a field for display.



Fields — Personalize | Find | View All | First 1-7 of 7 Last

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.COUNTRY - Country	Char3	1			Country		Edit	
2	A.DESCR - Description	Char30				Descr		Edit	
3	A.EU_MEMBER_STATE - EU Member State	Char1				EU		Edit	
4	B.STATE - State	Char6	2			State		Edit	
5	B.DESCR - Description	Char30				Descr		Edit	
6	B.NUMERIC_CD - Numeric Code	Char2				Numeric Cd		Edit	
7	CASE WHEN B.NUMERIC_CD IS NULL THEN '<null>' WHEN B.NUMERIC_CD = ' ' THEN '<space>' ELSE B.	Char10				Numeric Cd Edit		Edit	

The CASE expression will substitute "<null>" whenever the NUMERIC_CD field is NULL (from an unmatched row) and "<space>" whenever the NUMERIC_CD field is a single space (from an empty field of a matched row).

The screen shot below is an excerpt from the results of the query.

View All | Rerun Query | Download to Excel | Download to XML — First 1-100 of 1733 Last

	Country	Descr	EU	State	Descr	Numeric Cd	Numeric Cd Edit
1	ABW	Aruba	N				<null>
2	AFG	Afghanistan	N				<null>
3	AGO	Angola	N				<null>
4	AIA	Anguilla	N				<null>
		Aland Islands	N				
30	ARG				San Juan		18
31	ARG	Argentina	N	SL	San Luis	19	19
32	ARG	Argentina	N	TF	Tierra del Fuego	24	24
33	ARG	Argentina	N	TU	Tucuman	23	23
34	ARM	Armenia	N				<null>
35	ASM	American Samoa	N				<null>
36	ATA	Antarctica	N				<null>
37	ATF	French Southern Territories	N				<null>
38	ATG	Antigua and Barbuda	N				<null>
39	AUS	Australia	N	ACT	Austl. Cap. Terr.		<space>
40	AUS	Australia	N	NSW	New South Wales		<space>
41	AUS	Australia	N	NT	Northern Territory		<space>

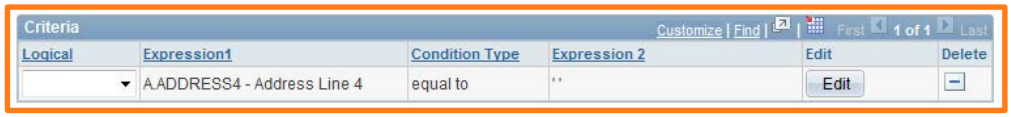
Note how Antarctica has the value "<null>" in the Numeric Cd Edit field. This indicates that no rows were found in STATE_TBL meeting the outer join conditions A.COUNTRY = B.COUNTRY.

The New South Wales region of Australia has the value "<space>" in the Numeric Cd Edit field. This indicates that there is a row in STATE_TBL meeting the outer join conditions, but that the NUMERIC_CD field has no data.

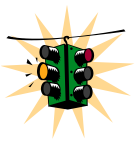
The Argentinian state of Tierra del Fuego has an actual value in the Numeric Cd Edit field because there is a matching row in STATE_TBL and the NUMERIC_CD field is not blank.

The following lists summarize the handling of blank and unknown values in PeopleSoft:

- When you see an empty cell in the results of a query
  - If the field is a date or a number, the value is the default NULL assigned in an outer join
  - If the field is text, the value is either the default NULL assigned in an outer join or the single space that represents an empty text value
- To substitute alternate values when a field is empty or NULL, create an expression using CASE; see the *Expressions* section of the Supplemental Material for examples.
- To write criteria to find blank/unknown values
  - For dates, use the Condition Type "is null."
  - For numbers, use the Condition Type "equal to," select Constant for Expression 2 Type, and enter the number 0 in the Constant text box. (There is no way to distinguish a true zero from a missing value.)
  - For text, use the Condition Type "equal to," select Constant for Expression 2 Type, and enter nothing in the Constant text box. When you see this criterion in the Criteria page, it will display Expression 2 as ' '.



- To write criteria to find rows from the left table of an outer join that have no match in the right table, use any field from the right table and the Condition Type "is null." Place this criterion on the WHERE clause, *not* the ON clause, even though it involves the table being outer-joined. This is because the criterion is intended to find *results* in which a field has the NULL value, not rows in the table being outer -joined in which that field has the NULL value.
- The "is null" Condition Type can also be used to find rows in which the result of a DECODE function is NULL because none of the values in it were matched; see the DECODE function in the *Expressions* section of the Supplemental Material for more information.



Make sure that you use the special Condition Type "is null" rather than "equal to" the value NULL. If you do the latter, you are telling the database system you want all rows in which the field is the *word* "NULL" rather than the *indicator* NULL.

Structured Query Language (SQL)

The standard language for writing queries to retrieve and manipulate the data in a relational database is the Structured Query Language, abbreviated SQL. When you use Query Manager to construct a query, a SQL statement is built automatically, and it is this statement that the database management system executes to return data to you.

While it is not necessary to be able to read or write SQL statements to use Query Manager, a basic understanding of SQL can help with troubleshooting queries by revealing the details that Query Manager hides.

Basic Query

A basic query in SQL has the following form:

```
SELECT <display field/expression list>
  FROM <table/record list>
 WHERE <criteria>
 ORDER BY <sort field/expression list>
```

The keywords in all capital letters indicate the general structure of a query. You *select* the data that you want to retrieve *from* one or more tables *where* certain conditions are met and *order* the results *by* particular fields.

Consider the following example query that retrieves the codes and descriptions of all countries and states in which the country is a member of the European Union and sorts the results by country name and state name.

```
SELECT A.COUNTRY, A.DESCR, B.STATE, B.DESCR
  FROM PS_COUNTRY_TBL A, PS_STATE_TBL B
 WHERE A.COUNTRY = B.COUNTRY
   AND A.EU_MEMBER_STATE = 'Y'
 ORDER BY A.DESCR, B.DESCR
```

The FROM clause lists the tables (corresponding to *records* in PeopleSoft terminology) from which the data is being retrieved. The two tables in this case are PS_COUNTRY_TBL and PS_STATE_TBL. These have been given *aliases* of A and B, respectively. The tables are referred to by alias throughout the rest of the query. Note that all tables in the FROM clause are joined together, with the join conditions given in the WHERE clause. The joins are inner joins unless otherwise specified; this is explained below.

Most tables in PeopleSoft have the same name as the corresponding record, but the name is preceded by "PS_." Thus, the table corresponding to ADDRESSES is PS_ADDRESSES. Tables that contain internal PeopleSoft information do not necessarily follow this convention.

Aliases do not have to be single letters (A, B, C, etc.). They can be abbreviations. Query Manager uses single letters; you are therefore limited to a maximum of 26 records in a query.

The list of fields being displayed is given as a series of field names, separated by commas, in the SELECT clause. The field names are *fully qualified* because they are preceded by a table alias (A or B) and a period. For instance, B.STATE indicates that STATE should be taken from table PS_STATE_TBL, which has the alias B. Using fully qualified names is not required except when a field being referenced exists in multiple tables used within the query, in which case it is needed to distinguish between such fields. In the query above, a

reference to COUNTRY must be fully qualified so that the database system knows whether to take it from PS_COUNTRY_TBL (A.COUNTRY) or PS_STATE_TBL (B.COUNTRY).

The WHERE clause contains the criteria used to limit results in the query. In this example there are two criteria. The first, A.COUNTRY = B.COUNTRY, is the join criterion between PS_COUNTRY_TBL and PS_STATE_TBL on the COUNTRY field. The second, A.EU_MEMBER_STATE = 'Y,' is what restricts the results to those countries that are members of the European Union.

The ORDER BY clause designates how the results are sorted. The first field given is the primary sort field; the second field given is the secondary sort field, and so on. To have the rows sorted by a field in descending order, follow the field name with the DESC keyword. In the example, an ORDER BY clause of A.DESCR, B.DESCR directs the database server to return the results in order by first by country name, then by state name within each country.

You can use numbers instead of fields and expressions in the ORDER BY clause. Each number corresponds to the *position* of a field or expression in the SELECT clause, with 1 representing the first field or expression. This is not recommended for manually-written SQL since if you change the order of items in the SELECT clause; you will have to change the position numbers as well. In the given example, the ORDER BY clause could be rewritten as ORDER BY 2, 4. Note that the queries generated by Query Manager use position numbers.

Whitespace is ignored; tabs, spaces, and carriage returns are legal except in the middle of field names and keywords (except for multi-word keywords such as ORDER BY). When using a fully-qualified name, there cannot be a space between the alias and the period or the period and the field name. "A.DESCR" is allowed but "A . DESCR" is not. However, "A.COUNTRY=B.COUNTRY" and "A.COUNTRY = B.COUNTRY" are the same.

Outer Joins

The previous example involves two tables connected using an inner join. The syntax for an outer join is a little more complex, involving giving the join conditions within the FROM clause. The following example obtains a list of History classes held in Fall 2006 at PeopleSoft University and their exam times; when an exam time is not known, the class is still listed. (This is the SQL equivalent of exercise TRNG_QM##_E14 as generated by Query Manager.)

```
SELECT A.CRSE_ID, A.CRSE_OFFER_NBR, A.STRM, A.SESSION_CODE, A.SUBJECT, A.CATALOG_NBR,
       A.CLASS_SECTION, A.DESCR, A.SSR_COMPONENT, B.CLASS_EXAM_SEQ,
       TO_CHAR(B.EXAM_DT,'YYYY-MM-DD'),
       TO_CHAR(CAST((B.EXAM_START_TIME) AS TIMESTAMP),'HH24.MI.SS.FF'),
       TO_CHAR(CAST((B.EXAM_END_TIME) AS TIMESTAMP),'HH24.MI.SS.FF')
  FROM (PS_CLASS_TBL A LEFT OUTER JOIN  PS_CLASS_EXAM B ON  A.CRSE_ID = B.CRSE_ID AND
        A.CRSE_OFFER_NBR = B.CRSE_OFFER_NBR AND A.STRM = B.STRM AND
```

```
        A.SESSION_CODE = B.SESSION_CODE AND A.CLASS_SECTION = B.CLASS_SECTION )
  WHERE ( A.STRM = '0570'
    AND A.SUBJECT = 'HISTORY' )
 ORDER BY 5, 6, 7
```

In the FROM clause, the description of the outer join is enclosed in parentheses. The LEFT OUTER JOIN keyword indicates that an outer join will be used to combine data from the two tables. The ON clause designates the criteria that are used to perform the outer join; this is built by Query Manager when "this criteria belongs to" is set to "ON clause of outer join B."

There is alternate outer join syntax for various database systems. For instance, with Oracle 11g, the table names can be a regular list in the FROM clause, and the criteria that would normally be in the ON clause would have the symbol "(+)" following them. The following Oracle-specific query is equivalent to the above example:

```
SELECT A.CRSE_ID, A.CRSE_OFFER_NBR, A.STRM, A.SESSION_CODE, A.SUBJECT, A.CATALOG_NBR,
       A.CLASS_SECTION, A.DESCR, A.SSR_COMPONENT, B.CLASS_EXAM_SEQ,
       TO_CHAR(B.EXAM_DT,'YYYY-MM-DD'),
       TO_CHAR(CAST((B.EXAM_START_TIME) AS TIMESTAMP),'HH24.MI.SS.FF'),
       TO_CHAR(CAST((B.EXAM_END_TIME) AS TIMESTAMP),'HH24.MI.SS.FF')
  FROM PS_CLASS_TBL A, PS_CLASS_EXAM B
 WHERE A.CRSE_ID = B.CRSE_ID(+)
   AND A.CRSE_OFFER_NBR = B.CRSE_OFFER_NBR(+)
   AND A.STRM = B.STRM(+)
   AND A.SESSION_CODE = B.SESSION_CODE(+)
   AND A.CLASS_SECTION = B.CLASS_SECTION(+)
   AND ( A.STRM = '0570'
   AND A.SUBJECT = 'HISTORY' )
 ORDER BY 5, 6, 7
```

Aggregate Query

A query that involves aggregate functions has a couple of additional clauses:

```
SELECT <display field/expression list>
  FROM <table/record list>
 WHERE <criteria>
 GROUP BY <grouping field/expression list>
HAVING <having criteria>
 ORDER BY <sort field/expression list>
```

Consider the following example in which the query gets a count of active plans per program, ordered by program code, showing only those programs for which there are more than 50 plans.

```
SELECT A.ACAD_PROG, COUNT(*)
  FROM PS_ACAD_PLAN_TBL A
 WHERE A.EFFDT = (SELECT MAX(A_ED.EFFDT)
                    FROM PS_ACAD_PLAN_TBL A_ED
                   WHERE A.INSTITUTION = A_ED.INSTITUTION
                     AND A.ACAD_PLAN = A_ED.ACAD_PLAN
                     AND A_ED.EFFDT <= SYSDATE)
   AND A.EFF_STATUS = 'A'
 GROUP BY A.ACAD_PROG
HAVING COUNT(*) > 50
 ORDER BY A.ACAD_PROG
```

The GROUP BY clause lists the fields that are used to group rows together for the aggregate function to work upon. In the above example, the GROUP BY clause has the field A.ACAD_PROG, so the counts are of rows having the same academic program code.

If the results will be limited based on the result of the aggregate function, there must be a HAVING clause. In this clause, there are criteria that use the aggregate function to restrict the *groups* that are shown. The above example has one criterion in its HAVING clause – COUNT(*) > 50 – which instructs the database server to only return groups having a count of rows greater than 50. The results will therefore be limited to programs that have more than 50 plans.

The asterisk * is a shortcut representing all fields. When used in the SELECT clause, it represents all fields from all tables in the query if it is not preceded by an alias (SELECT *) or all fields from one table if it is preceded by an alias (A.*). When used in the COUNT(*) function, in represents all fields in the GROUP BY clause.

Query Manager and SQL

The tabs in Query Manager correspond roughly to the clauses of a query in SQL as follows:

- Records – no equivalent, since this is where tables are searched for and added or joined
- Query – the FROM clause
- Expressions – no equivalent; can appear where fields and values are allowed
- Prompts – no equivalent; can appear where fields and values are allowed
- Fields
    - The SELECT clause since each field listed in this tab is displayed in the results
    - The ORDER BY clause if the sort order is changed using the Edit Field Ordering page (Reorder/Sort button)
    - The GROUP BY clause since if there is a field with an aggregate function, those fields *not* being aggregated comprise the groupings
- Criteria
    - The WHERE clause of the main query
    - The FROM clause when “this criteria belongs to” is set to “ON clause of outer join X,” where X is the alias of the record on the right side of the join
- Having – the HAVING clause
- View SQL – no equivalent, but shows the SQL generated to execute the query
- Run – no equivalent, since it shows the results of the query

Note that the SQL that Query Manager generates may not exactly match the SQL that you or someone else writes. There are multiple ways to express the same query in SQL, just as there are multiple ways to express the same concept or instructions in human languages. This does not mean that your query is incorrect or that the query from Query Manager is incorrect; they may be equivalent.

Troubleshooting

Suggestions of what to check when a query produces incorrect results and/or errors:

- If the query produces many more rows than are expected
  - Are there any criteria missing?
    - Were you asked to limit the results to a particular term but there is no criteria on STRM?
    - Should you be showing only "active" rows, such as those with EFF_STATUS = 'A', PROG_STATUS = 'AC', etc.?
    - Are there criteria on the appropriate key fields, such as INSTITUTION = BGSUN', SETID = 'BGSUN' or 'BGHCM', ACAD_CAREER = 'UGRD' or 'GRAD', etc.?
  - Are the joins between records correct?
    - If a join is missing, each row from the first table is combined with each row from the second table.
    - If you added the join yourself by creating criteria, did you join on all appropriate key fields? For example, if you are querying on SRVC_IND_DATA and join to SRVC_IND_RSN_TBL to get the description of the reason code, did you join on INSTITUTION and SRVC_IND_CD as well as SRVC_IND_REASON?
    - Did you use a left outer join instead of a standard join? Left outer joins are used only when you need rows from the first table to be shown regardless of whether there is a match in the second table. If a match is required, use a standard join.
- If the query produces many fewer rows than are expected
  - Did you create a criterion on the wrong field? For example, to find a class by course ID, did you put a criterion on CATALOG_NBR instead of CRSE_ID?
  - Did you compare against the wrong value? For instance, if you want active programs, did you check for PROG_STATUS = 'AC' or PROG_STATUS = 'ACTV'? (In this case, the former is correct; 'ACTV' is a valid value for PROG_ACTION.)
  - Did you manually join two records on the wrong fields? The values in each field are not like each other, so there is unlikely to be a match, and if there is one, it will be by coincidence. For instance, the field ACAD_PLAN should not be joined to the field ACAD_PROG since they represent different things. (Hint: fields that can be joined will likely have identical or similar names.)
  - Is it impossible to meet the conditions? If you wrote a query to find cases where PROG_STATUS is either 'AC' or 'LA', did you use PROG_STATUS = 'AC' AND PROG_STATUS = 'LA' or PROG_STATUS = 'AC' OR PROG_STATUS = 'LA'? Since it is not possible for a field to have two values that the same time, a query with X = <value1> AND X = <value2> will always return no rows.

- Problems with expressions
  - "Invalid identifier" error
    - Field names referenced in an expression must be spelled correctly (SRVC_IND_RSN is incorrect; SRVC_IND_REASON is correct)
    - Function names must also be spelled correctly (TRUNCATE is incorrect; TRUNC is correct)
    - Strings (literal text included in the expression) must be enclosed in single quotes
  - Value displayed has too few characters or digits
    - Check the Length property of the expression.
      - Character: If Length is 10 but the result of the expression is 20 characters long, only the first 10 characters are shown.
      - Number: Values are shortened by trimming digits off the end of the integer portion of the number (1234567.890 when displayed with a Length property of 5 is shown as 12345).
  - Missing parentheses
    - Each opening parenthesis (must have a corresponding closing parenthesis)
    - Errors can vary depending on how the query engine interprets the generated SQL
  - Functions
    - Did you include all required arguments? For example, SUBSTR takes two or three arguments, and the first two are required. Using SUBSTR('ABC') results in a "not enough arguments for function" error.
    - Did you include too many arguments? For example, LOWER takes only one argument. Using LOWER('ABC', 5) results in a "too many arguments for function" error.
    - Did you use the right types of arguments? MONTHS_BETWEEN takes two date arguments results in an "inconsistent datatypes" error. If a date is expected, a date must be given, if a number is expected, a number (or string that can be converted to a number) must be given, etc.
    - Did you use the correct function? CEIL, FLOOR, ROUND, and TRUNC all round numbers, but only ROUND and TRUNC can round to multiples other than 1, and all four round slightly differently (CEIL rounds up, FLOOR rounds down, ROUND follows regular rounding rules, and TRUNC drops the digits after the decimal point).
- Problems with unions
  - Each query that is part of a union must have the same number of fields. If there is a difference, this error is displayed: "A UNION requires the same number of fields for each SELECT. Correct and retry."
- Problems with prompts
  - A prompt that uses a prompt table may show no values when using the lookup feature. If the prompt table has a key with more than one field, there must be prompts for all fields before that field in the field order. For instance, if there is a prompt for Program (ACAD_PROG) using ACAD_PROG_TBL as a

prompt table, since the key of ACAD_PROG_TBL is INSTITUTION + ACAD_PROG + EFFDT, there must also be a prompt for INSTITUTION. In addition, the prompts must be specified in the order of the fields of the key; in the example, the prompt on INSTITUTION must come before the prompt on ACAD_PROG.

- Problems with aggregates
  - You cannot use fields that are being aggregated as regular criteria. If you apply the Count aggregate function to STRM, you cannot also add the criterion STRM = '2095'. When you aggregate, you are no longer dealing with individual rows, but groups. (You can still add criteria on the fields that are not being aggregated. In the example on STRM, perhaps you could aggregate on ACAD_CAREER or ACAD_PROG.)
  - Remember that once you add an aggregate to a displayed field, the aggregation is done on groups defined by the other fields. For example, if you want to count how many students there are per program, don't include EMPLID in the fields being displayed, or else the count will be per combination of program and EMPLID.
  - If you put an aggregate function into an expression, you must check the Aggregate Function checkbox, or you will receive the error "not a single-group group function." This is indicating that you are trying to use an aggregate function on results that are not being aggregated.
- Unable to find a query
  - Make sure you are on the Query Manager search page (titled "Query Manager") rather than the Records tab (titled "Find an Existing Record"). Since both have search capabilities, one can mistake one of these search pages for the other page.
  - If you tried searching for the full name of the query, you may have misspelled the name either when searching for it or saving it. Try entering less of the name. For instance, search for "TRNG_QM##" or even "TRNG" instead of "TRNG_QM##_Q15." You may get several extraneous results but have a better chance of getting the one that you need.
- Learn SQL!
  - The query is not checked for errors before it reaches the database. The error messages reported by Query Manager are native to the database and refer to the generated SQL. To debug the query, it is sometimes easiest to look at the generated SQL. This will show which records are included, how they are joined (even if you didn't create the join conditions yourself); subqueries, effective date logic, and so on, as a legal SQL query. It can also show a little more clearly which criteria are for which part of the query in the case of large queries.