Advanced Query for Query Developers

This is a training guide to step you through the advanced functions of Query in NUFinancials. Query is an ad-hoc reporting tool that allows you to retrieve data that is stored in the NUFinancials application. You use a series of query pages to specify what data to retrieve and the format in which results will be displayed. Query translates the information that you enter into SQL code to produce results which can be displayed in a browser window or downloaded to your workstation in an Excel spreadsheet.

There are two query access methods. **Query Viewer** allows execution of existing queries. Casual query users will only have access to queries via Query Viewer. **Query Manager** is used to create new queries or edit existing ones. It is also possible to execute queries from Query Manager. Query developers will have access to both query viewer and query manager pages.

FMS121Introduction to Query course, materials and online demo are recommended as a prerequisite to this guide.

Before you begin . . .

Access to **Query Viewer** and **Query Manager** can be requested using the Central Security Access Form located on the FFRA website:
http://ffra.northwestern.edu/documents/security/CafeCentralSecurityForm.pdf

Access Query via NUFinancials

- Log in to NUPortal* https://nuportal.northwestern.edu
- Select the Financial section
- Click **Login to NUFinancials** in System Login
- Click **Reporting Tools → Query**

Where to get help? For assistance, contact NUIT Support Center at 847-491-HELP (4357) or email consultant@northwestern.edu

Table of Contents

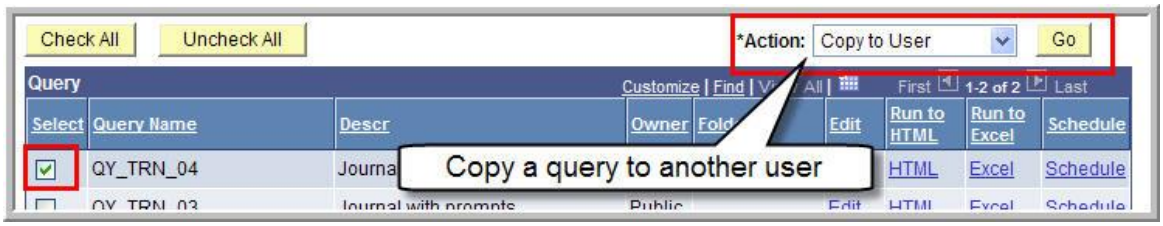
Query Manager Reminders

Navigation: Reporting Tools > Query > Query Manager

IMPORTANT NOTE: Query Manager is used for tailoring queries to specific result sets of data. In NUFinancials, there are many existing public queries. These queries are for use by all Query users. When using public queries, do not save any modifications you make as they will become part of the public query. If you want to capture specific modifications to a query, select **Edit** to open the query and then click **Save As** to save the modified query with a new name. You can choose to make the new query public or private.

HOT Tip! Save your queries into a Favorites folder by selecting them in the check box at the front of the row and using the **Action** drop down menu.

HOT Tip! Only Private queries can be saved to another user. You should save a Public query as a Private one and then can save/share it with another user using the **Action** drop down menu and User ID.



Basics of SQL syntax

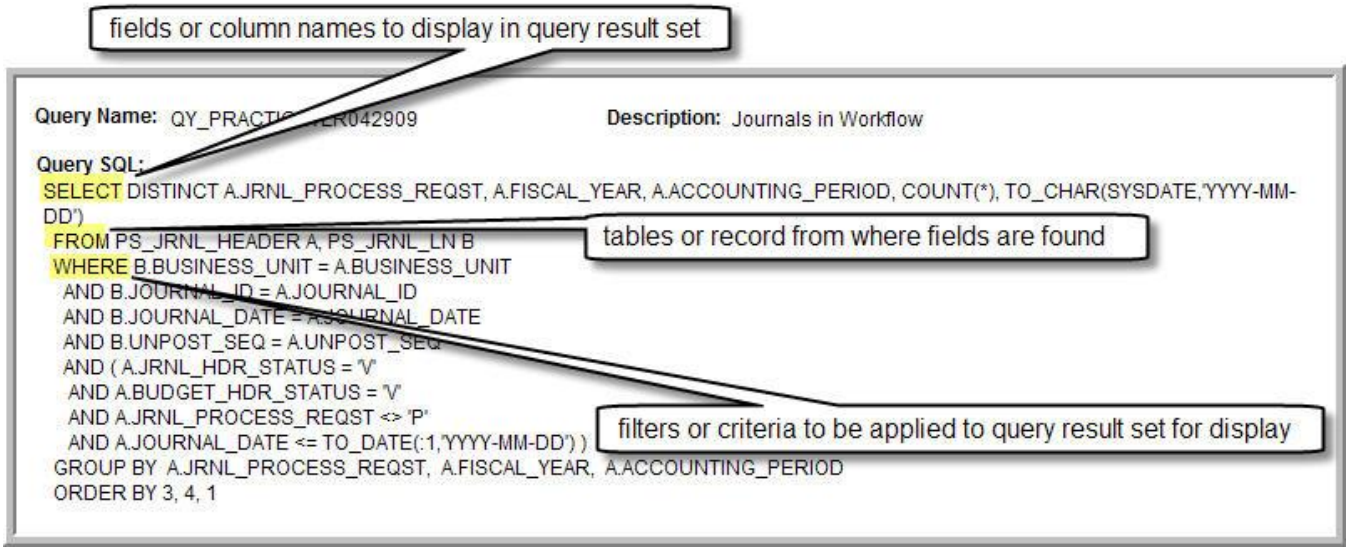
Note: Oracle/PeopleSoft utilizes a subset of SQL standard functions

> Simply, SQL is a set of instructions listing the **SELECT**ion of specific fields, **FROM** specific tables **WHERE** certain circumstances exist.

SELECT: lists fields or column names to be used in query

FROM: lists tables or records to be used in query

WHERE: lists filters to include or exclude data



NUFinancials Query Terminology review:
A query contains records.

A record is a table.

A record contains fields or columns.

Primary Key: a field or group of fields that uniquely defines each row in a table or record
ex. Business Unit/Journal ID field values uniquely identify each row in the Journal header table

NUFinancials query business process flow[1]

[1] Business process flow from **Enterprise PeopleTools 8.48 PeopleBook: PeopleSoft Query, Defining Selection Criteria**, September 8, 2009.

LEFT OUTER JOIN

Let's review first! A standard (or INNER or simple) JOIN is a linking of two tables on selected field(s) and returns only those rows where linking values match in both tables. Two tables are presented as one table.

A LEFT OUTER JOIN first is thinking about the two tables laid out from left to right in the order that they are added to the query. A LEFT OUTER JOIN will return ALL the rows from the first table and any matching rows from the second table. If there isn't a match in the second table, a blank or NULL is returned for fields that are pulled from that table.

Examples:

Show me ALL the Purchase Orders that have vouchers (or are missing vouchers).

Show me ALL the Purchase Orders with their voucher match statuses.

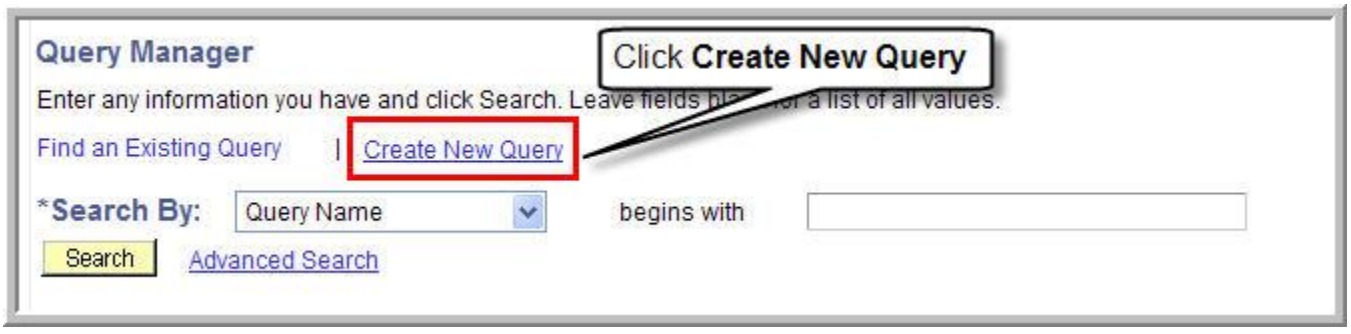
Show me ALL the Requisitions with an error in budget check.

Show me ALL the Projects/Grants with outstanding invoices to sponsors.

Note: SQL standard also offers RIGHT OUTER JOINS and FULL OUTER JOINS, but those are not offered in the NUFinancials Query application.

Step 1: Create a new query

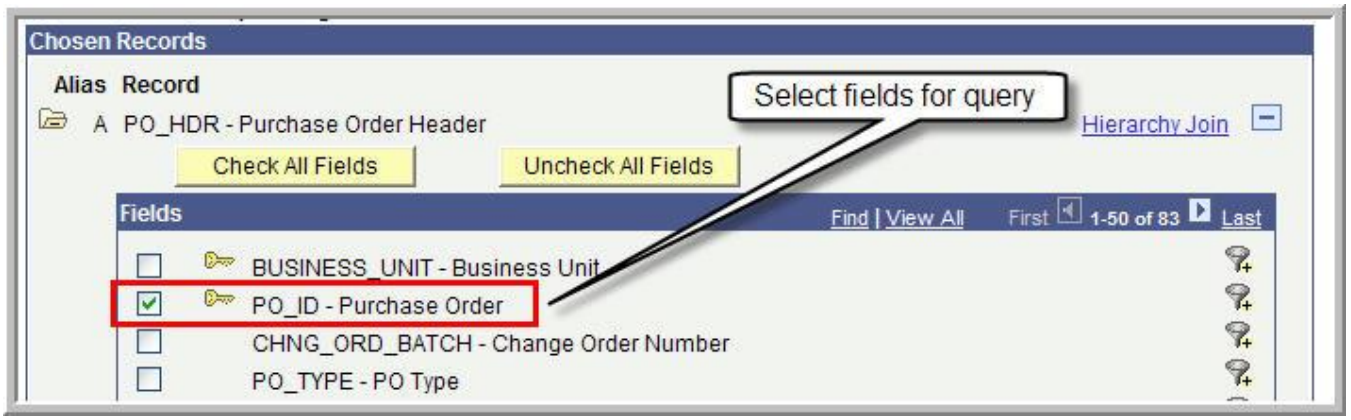
Navigation: Reporting Tools → Query Manager → Create New Query



Step 2a: Select first table (left) in Records tab

The first table will be the Left Table of the query. This is the table that will show all results based on the criteria.

Step 2b: Select field(s) for query

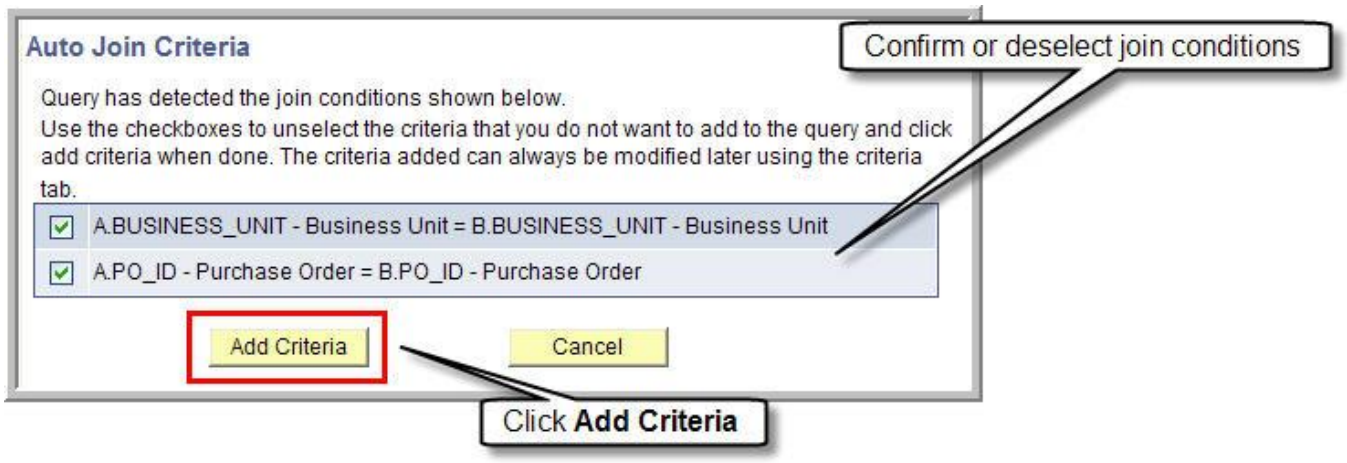


Step 3a: Select second table in Records tab and join

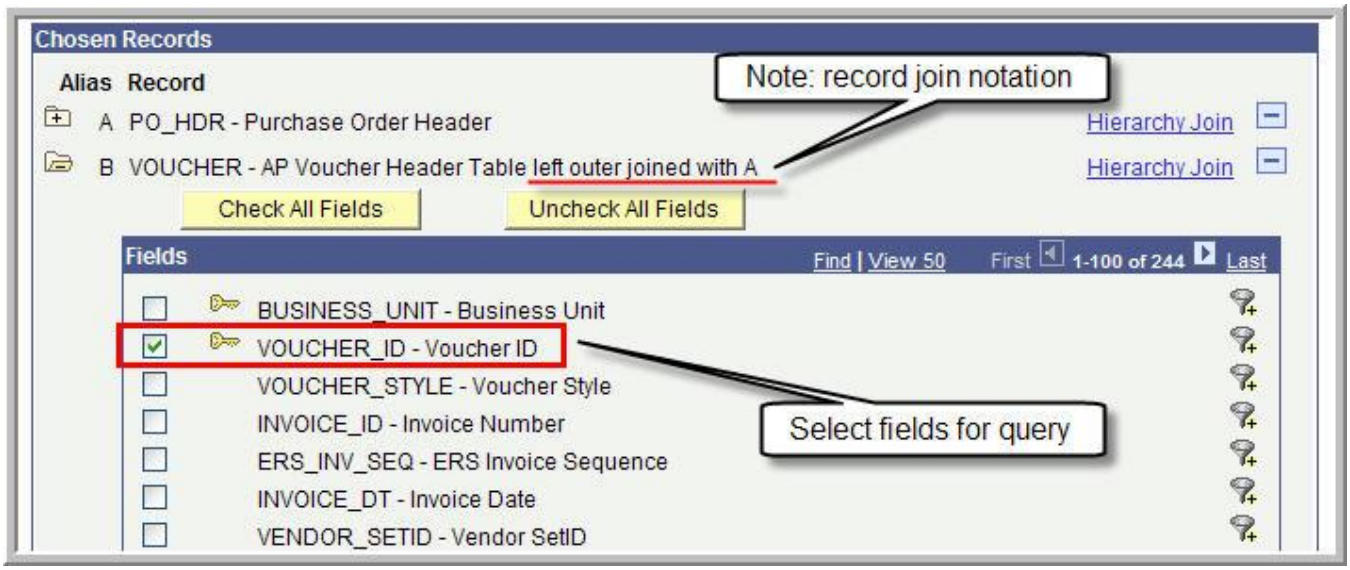
The second table will be the table used to compare values to the first table. Any matching rows will be in the result set. If there isn't a match in the second table a blank or NULL is returned.

Step 3b: Select Left outer join

Step 3c: Confirm or deselect criteria join conditions



Step 3d: Select field(s) on second table



Once you have joined the two records, you can use expressions (discussed later in this guide), prompts, field options, criteria, and HAVING (discussed later in this guide) to fine tune query to desired result set. Use the **Distinct** check box under the **Properties** link to exclude duplicates from the result set.

Run query.

Option: Review SQL on **View SQL** tab.

UNION

Let's review first! A standard (or INNER or simple) JOIN is a linking of two tables on selected column(s) and returns only those rows where linking values match in both tables. Two tables are presented as one table.

A UNION selects all rows from two tables for the result set, but has several restrictions. (note: some of these restrictions are specific to NUFinancials Query application)

1. Same number of output columns must be specified from each table.
2. Each corresponding column must have the same data type (ex. character to character, or numeric to numeric, or date/time to date/time).
3. Columns must be in the same display order.

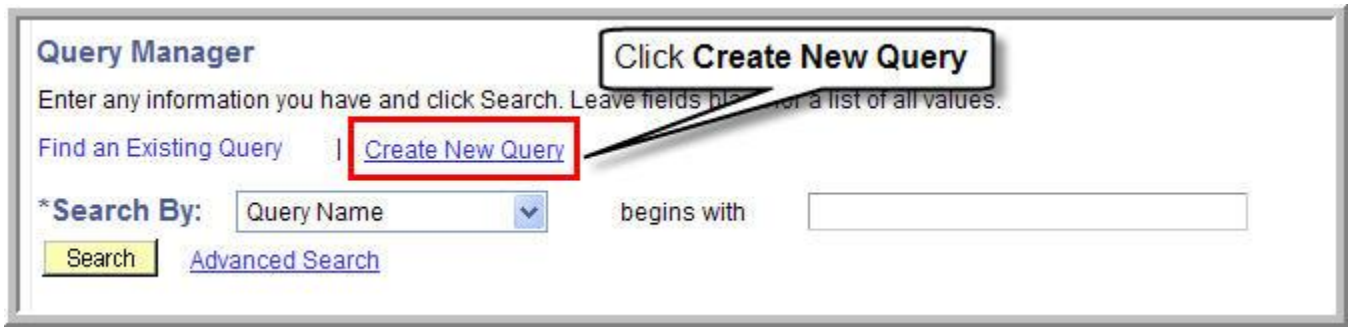
A UNION can also be a result set of results sets of two queries. ALL in the SELECT statement includes duplicate rows. Column name comes from first table column name

Example:

Show me all Requisitions and Purchase Orders and their statuses.

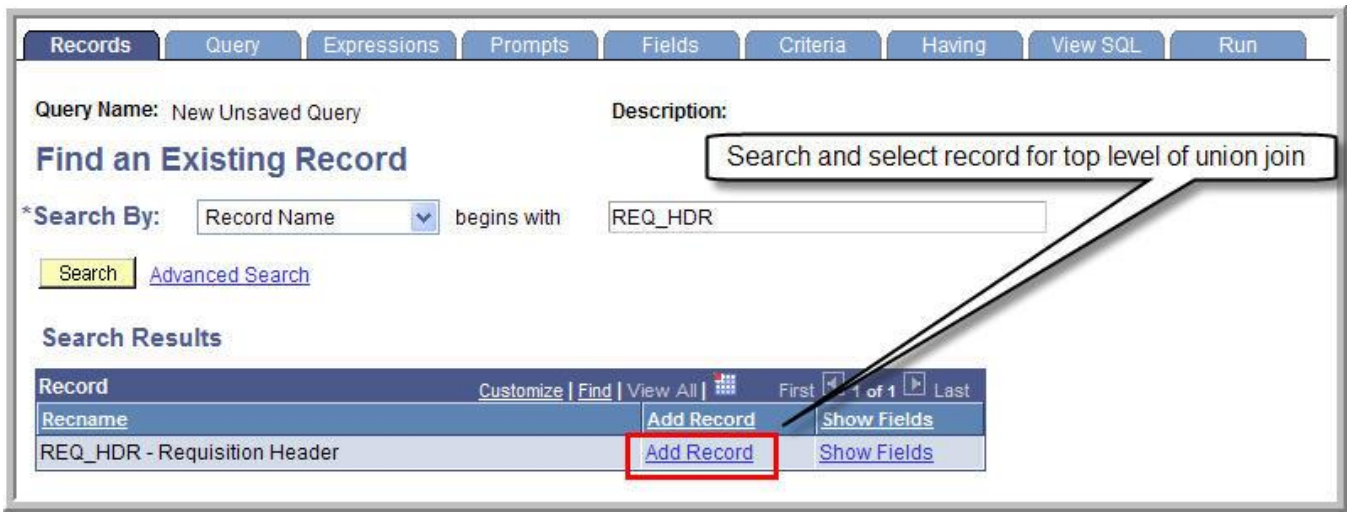
Step 1: Create a new query

Navigation: Reporting Tools → Query Manager → Create New Query

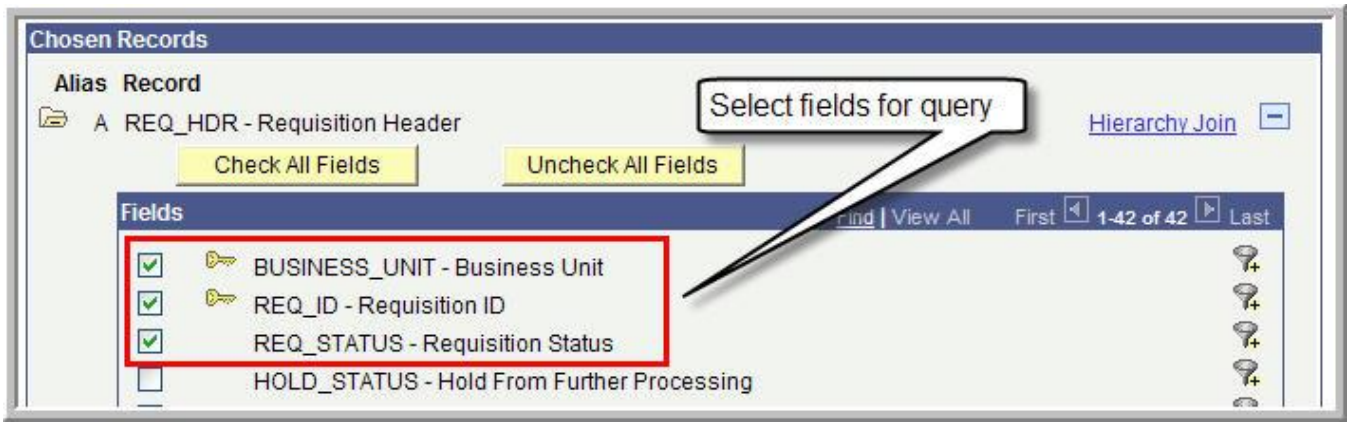


Step 2a: Select first table in Records tab

The first table will be designated the *Top Level of Query* when toggling between the two tables.



Step 2b: Select field(s) for query

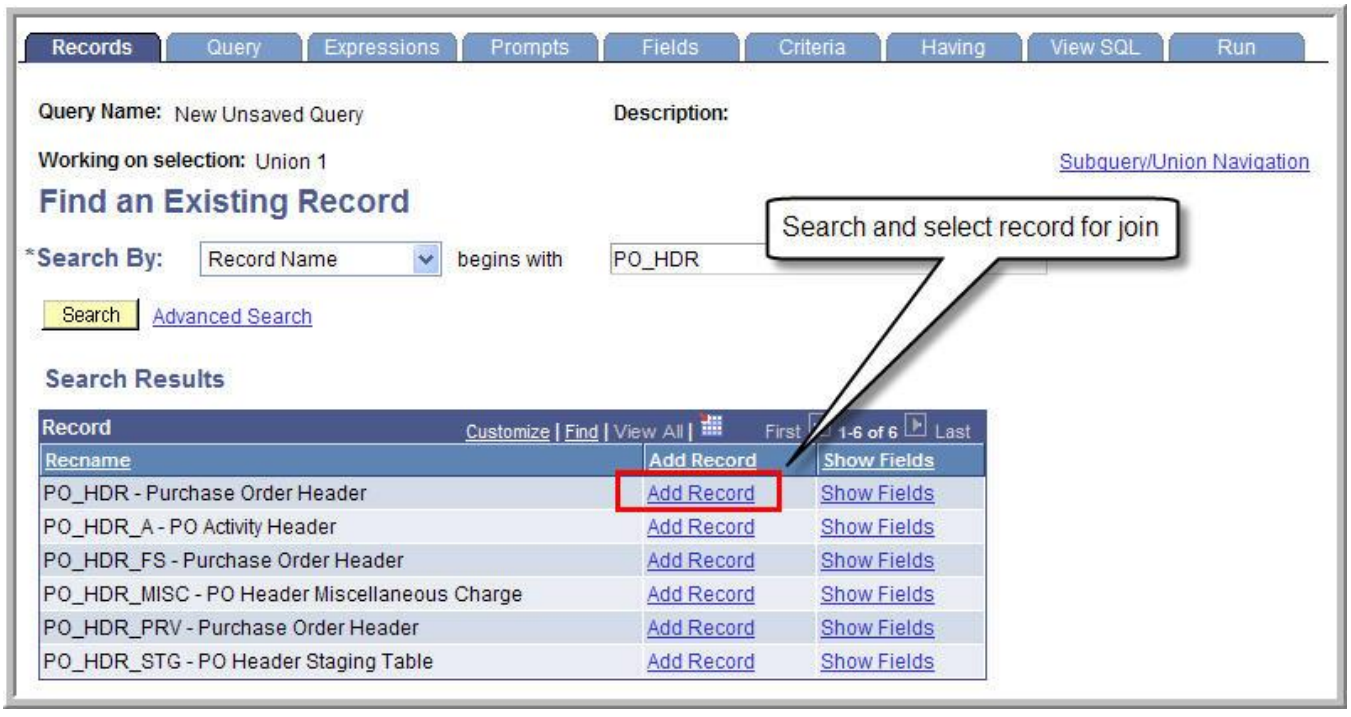


Step 3a: Create UNION join

To create a union, click **New Union** at the bottom of any tab except the Run tab. The second table will be designated as *Union 1*.

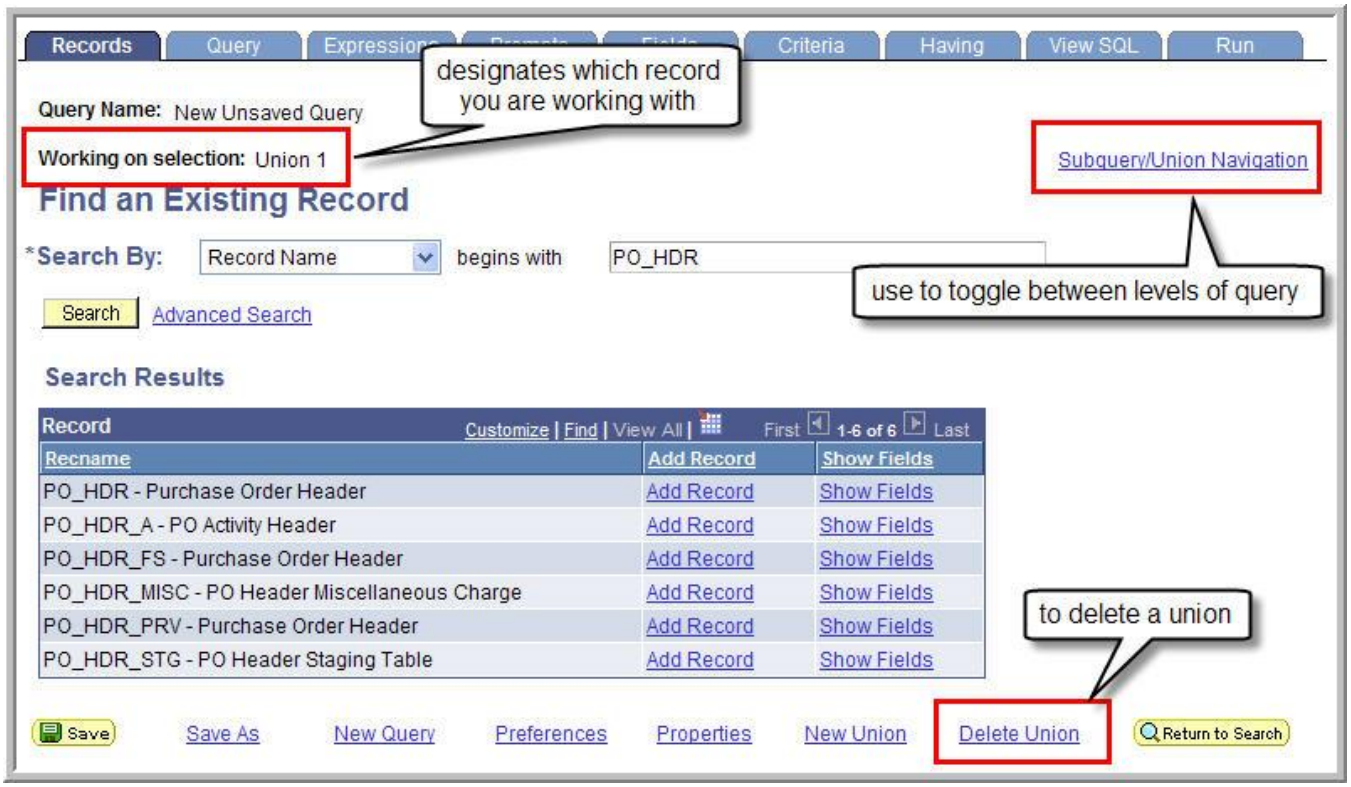
Step 3b: Select second table

In Records tab, search and select table to be joined by UNION.

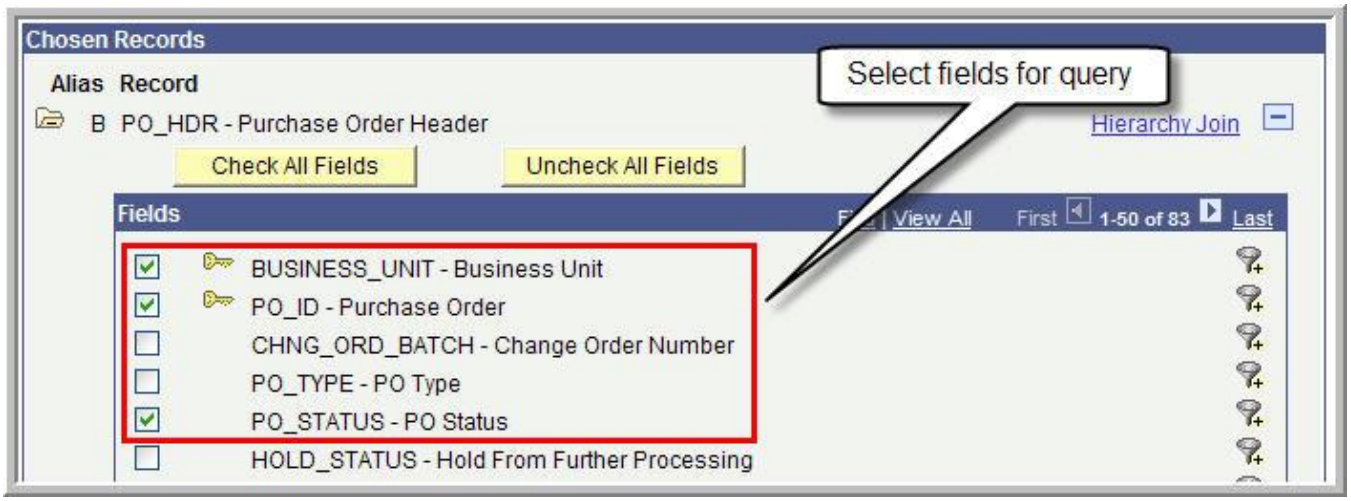


Note additional components of query page:

- **Working on selection** displaying which record/table you are working with
- **Subquery/Union Navigation** link used for toggling between *Top Level of Query* and *Union1* records
- **Delete Union** link used to delete union



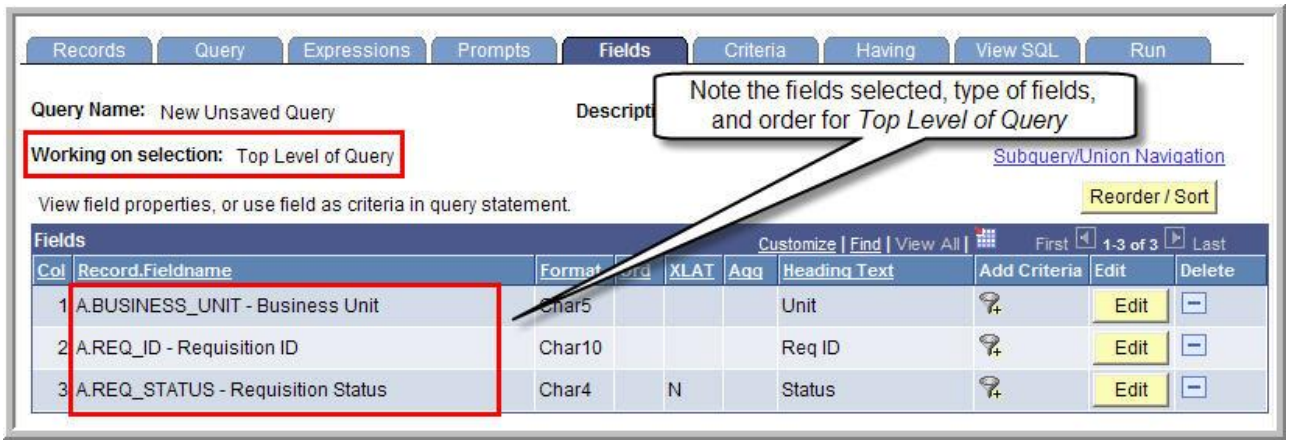
Step 3c: Select field(s) for query



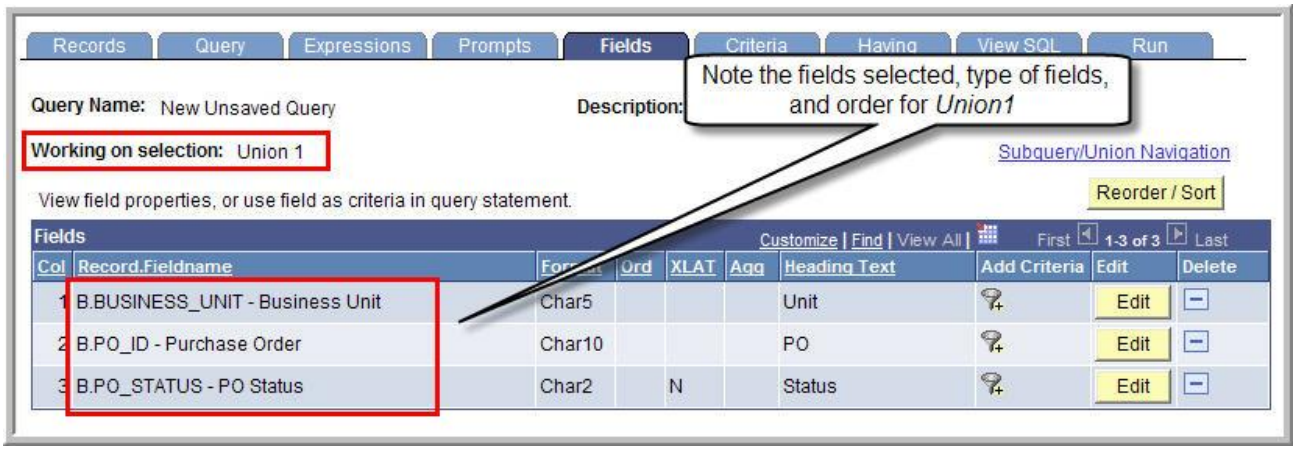
Did you remember the three restrictions when building a UNION join (see below)?

1. Same number of output columns must be specified from each table.
2. Each corresponding column must have the same data type (ex. character to character, or numeric to numeric, or date/time to date/time).
3. Columns must be in the same display order.

Top Level of Query fields



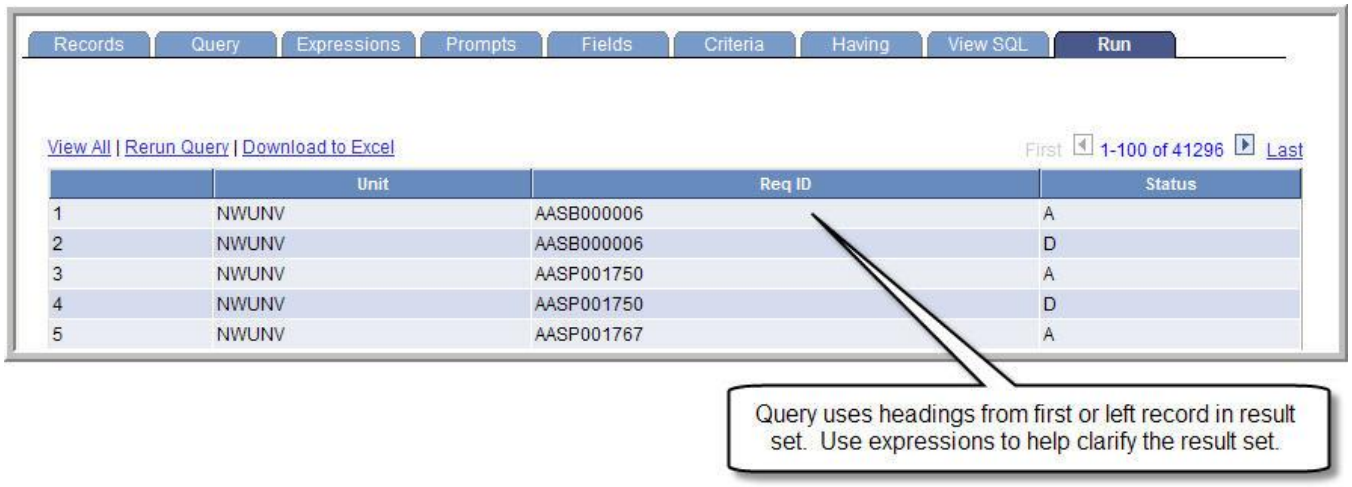
Union 1 fields



Once you have joined the two records, you can use expressions (discussed later in this guide), prompts, field options, criteria, and HAVING (discussed later in this guide) to fine tune query to desired result set. Use the **Distinct** check box under the **Properties** link to exclude duplicates from the result set.

Important note

When you use a UNION join, the query selects the column headings from the first or left record. Use expressions (discussed later in this guide) to clarify the result set.



Run query.

Option: Review SQL on **View SQL** tab.

Subqueries

Let's review first! Remember your algebra equations from school? The expression *(x+1)y* is indicating that *x+1* needs to be solved before multiplying the result by *y*. The equation inside the parentheses needs to be executed, then applied to the operation outside the parentheses.

A Subquery is a query whose results are used by another query. The main query uses the subquery's result set as a comparison value for a selection criterion.

- *Scalar subquery*: a query that returns only one column and no more than one row as a comparison value.

Examples:

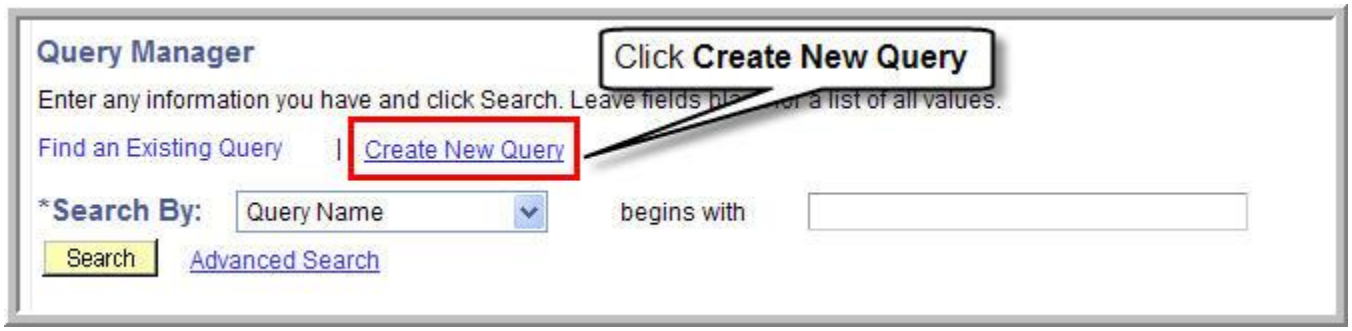
Show me vendors whose address is in New York.

Show me requisitions and their statuses for a specific user or list of users. (List REQs and status where Requestor ID equals xxx).

Show me POs where the voucher's match status is exception.

Step 1: Create a new query

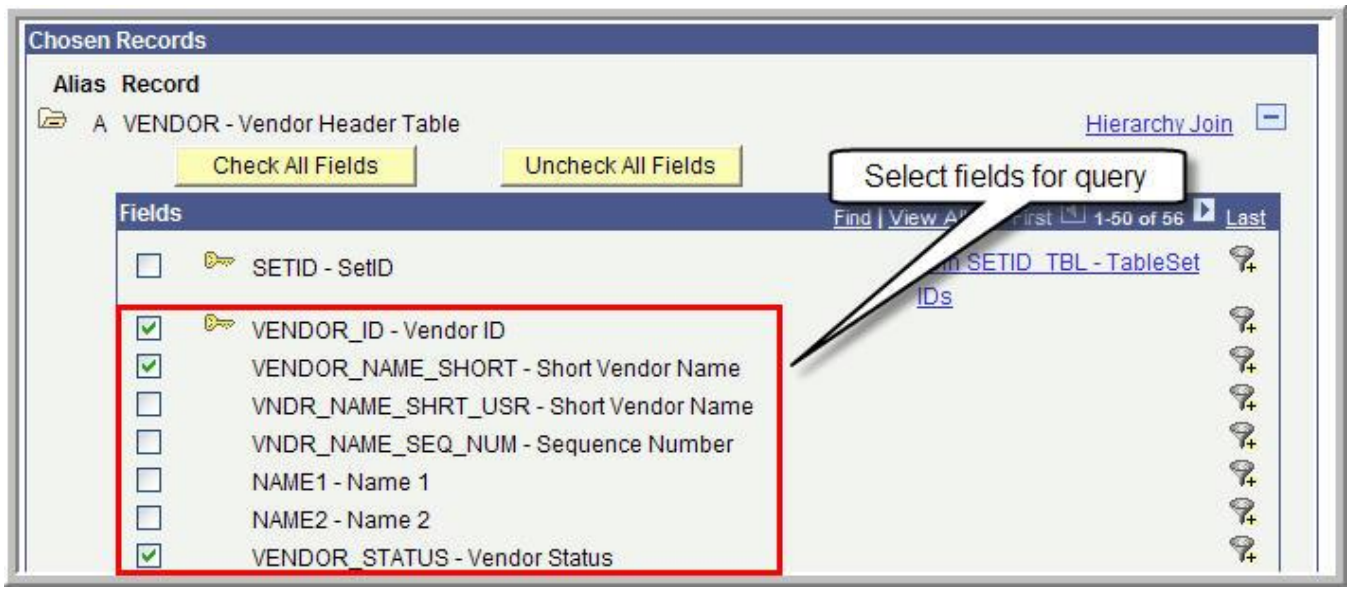
Navigation: Reporting Tools → Query Manager → Create New Query



Step 2a: Select first table in Records tab

The first table will be designated the *Top Level of Query* when toggling between the two tables.

Step 2b: Select field(s) for query



Step 3a: Create subquery as criteria on a field

Remember that a subquery is a query whose values will be used as criteria for the top level query. Select field to apply subquery.

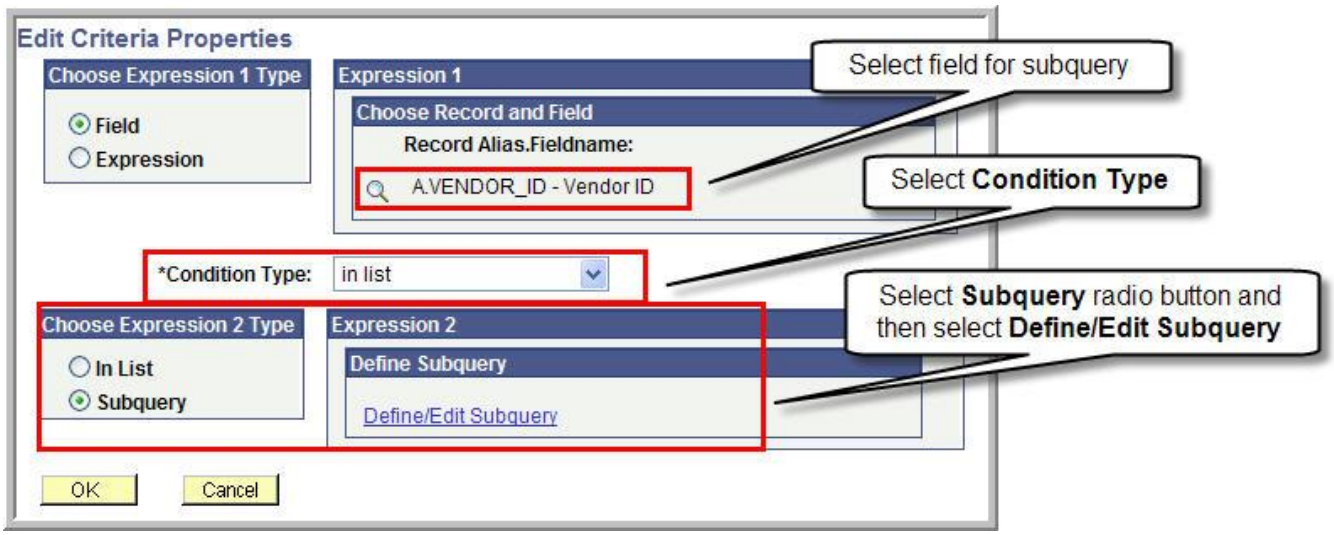
Criteria can be added by clicking  next to a field on the **Fields** tab or by clicking **Add Criteria** and selecting a field on the **Criteria** tab.

When linking tables in a subquery, you need to consider if the relationship is going to be 1 to 1 or 1 to many. This will help you decide the type of link and criteria you want to use.

For **Condition Type** in subqueries, selecting

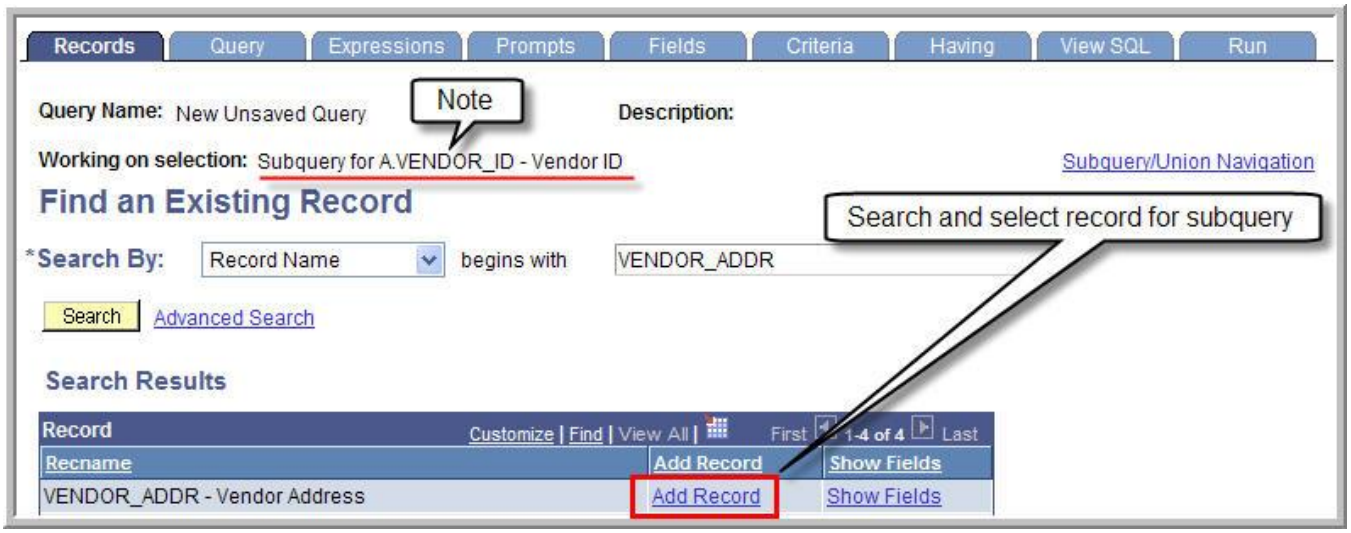
- **equals to** is asking for a 1 to 1 relationship in the comparison,
- **in list** is asking for a 1 to many relationship in the comparison,
- **exists** is asking for a 1 to many comparison where if the condition is satisfied in the subquery, show results in the top level query.

If you are unsure about the relationship, it is recommended to use **in list** as the **Condition Type.**

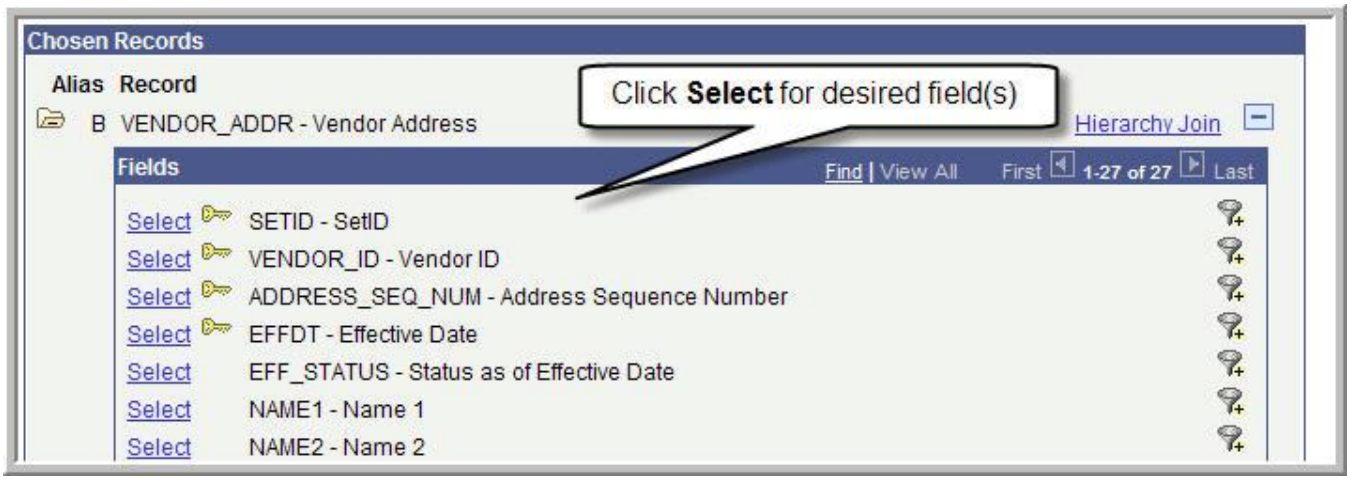


Step 3b: Select subquery table in Records tab

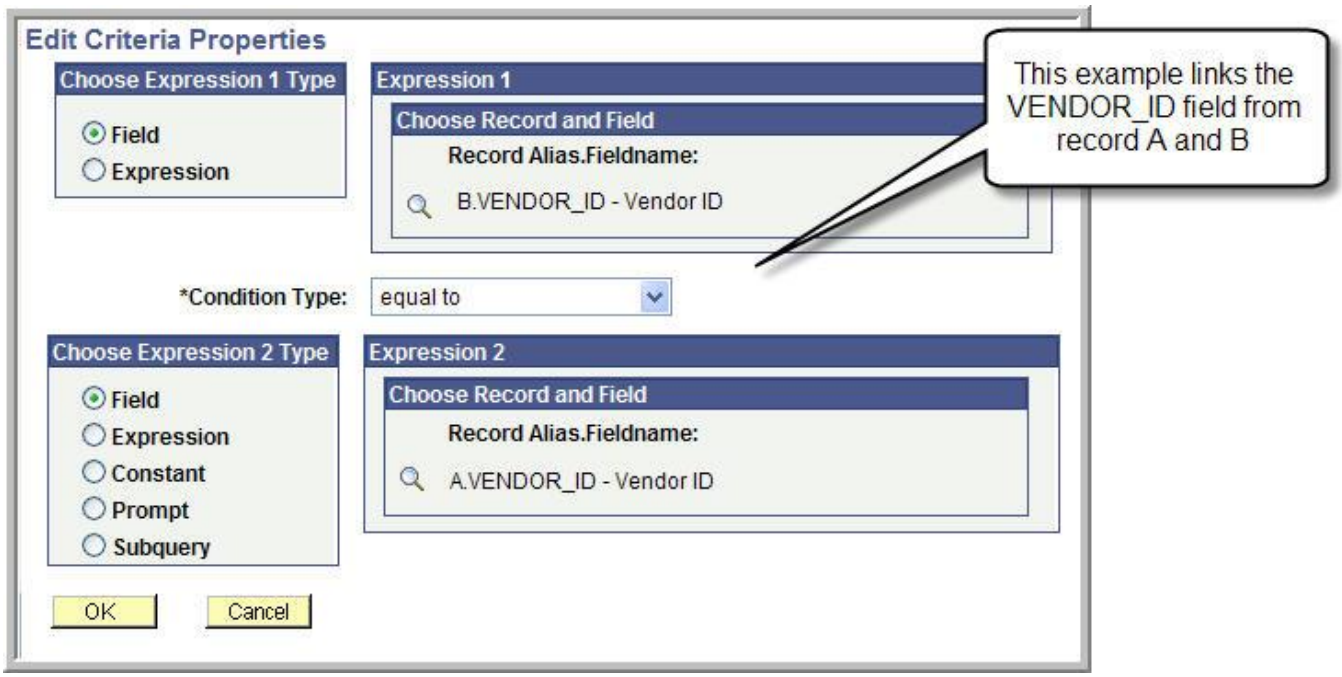
The subquery table will be designated the *Subquery for xxx* when toggling between the two tables.



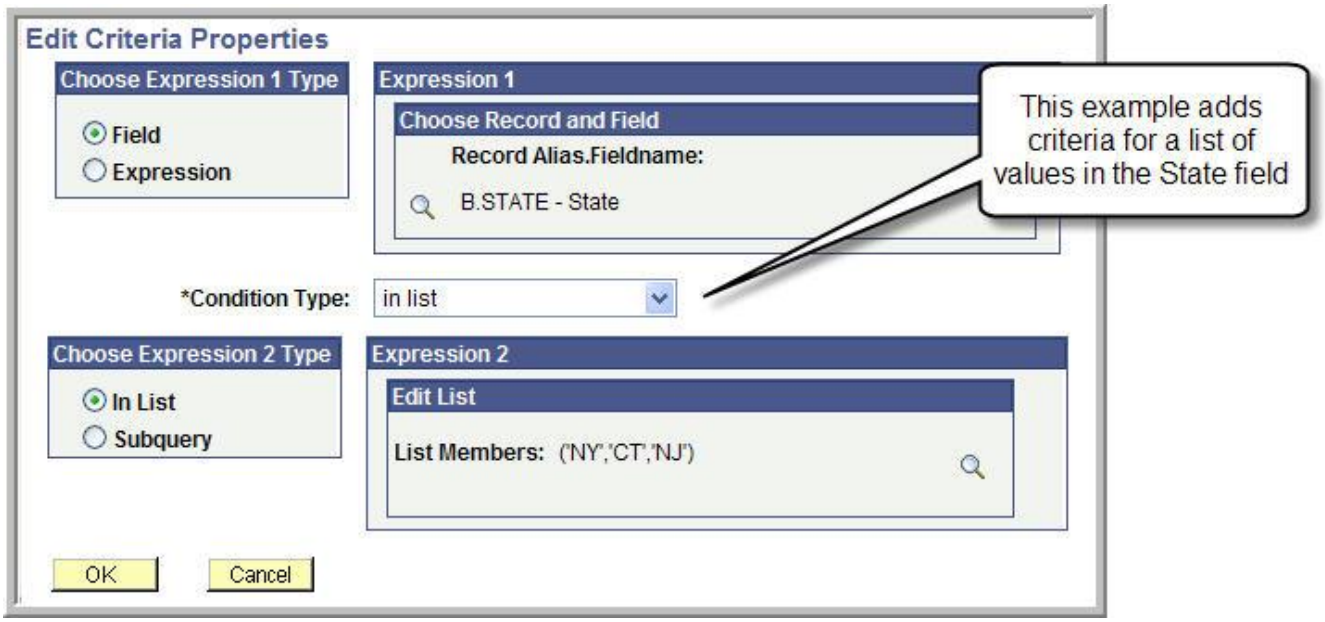
Step 3c: Select field(s) for subquery



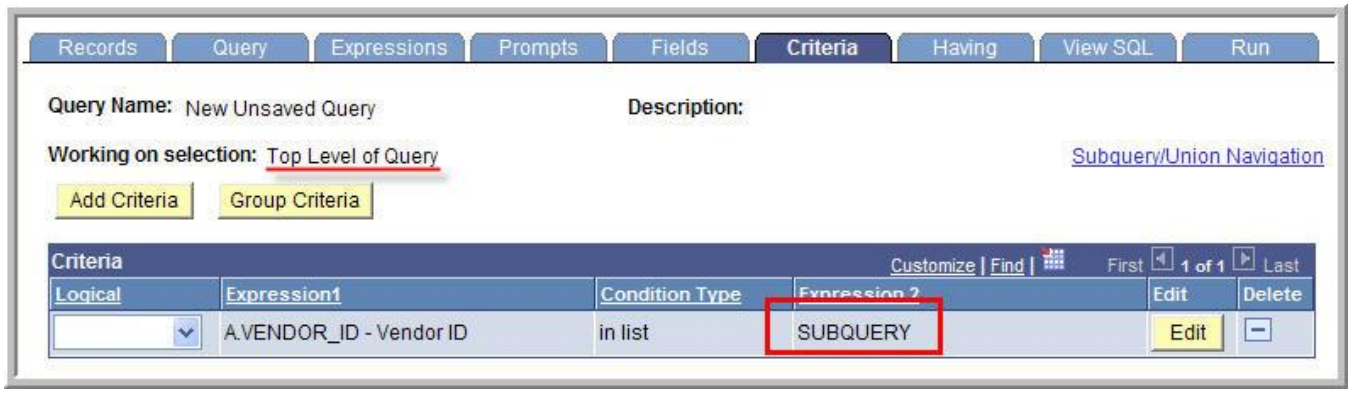
When selecting fields and adding criteria in the subquery, one of criterion needs to create a link between the two records so that there is a basis for comparison. You can do this by selecting a field and creating an **equals to** criteria.



Other criteria filter for desired result set.



Note: You can see that the subquery has been established for the *Top Level of Query* by looking at the **Criteria** tab for the *Top Level of Query*. See below.



Once you have created the subquery, you can use expressions (discussed later in this guide), prompts, field options, criteria, and HAVING (discussed later in this guide) to fine tune query to desired result set. Use the **Distinct** check box under the **Properties** link to exclude duplicates from the result set.

Run query.

Option: Review SQL on **View SQL** tab.

Query SQL:

```
SELECT A.VENDOR_ID, A.VENDOR_NAME_SHORT, A.VENDOR_STATUS,A.SETID
  FROM PS_VENDOR A
  WHERE A.VENDOR_ID IN (SELECT B.VENDOR_ID
  FROM PS_VENDOR_ADDR B
  WHERE B.EFFDT =
      (SELECT MAX(B_ED.EFFDT) FROM PS_VENDOR_ADDR B_ED
      WHERE B.SETID = B_ED.SETID
        AND B.VENDOR_ID = B_ED.VENDOR_ID
        AND B.ADDRESS_SEQ_NUM = B_ED.ADDRESS_SEQ_NUM
        AND B_ED.EFFDT <= SYSDATE)
    AND B.VENDOR_ID = A.VENDOR_ID
    AND B.STATE IN ('NY','CT','NJ'))
```

there is criteria linking the 2 tables by VENDOR_ID
and
there is criteria detailing the list of values to include

Advanced Selection Criteria

In **FMS121 Introduction to Query** materials, we discussed selection criteria by defining a constant or a list of constants or a range of constants with condition types of *equal to*, *in list*, and *between*. Condition types of *greater than* and *less than* return values larger or small than the comparison value. These condition types can also be used in the negative to return the opposite rows. There are other condition types that can be used to further include or exclude date from the result set. See Condition Types table.[2]

Condition Types	Return Values
between	The value in the selected record field falls between two comparison values. The range is inclusive.
equal to	The value in the selected record field exactly matches the comparison value.
exists	This operator is different from the others, in that it does not compare a record field to the comparison value. The comparison value is a **subquery**. If the **subquery** returns any data, PeopleSoft Query returns the corresponding row.
greater than	The value in the record field is greater than the comparison value.
in list	The value in the selected record field matches one of the comparison values in a list.
in tree	The value in the selected record field appears as a node in a tree created with PeopleSoft Tree Manager. The comparison value for this operator is a tree or branch of a tree that you want PeopleSoft Query to search.
is null	**Note:** Many fields in NUFinancials are assigned a default value (0 for numerical fields and a single space for character fields) when an actual value is not specified. Is Null searches for fields that have no value. Null fields are not the same as zeros or blanks. Null fields have no data, whereas zeros and blanks are considered data. The selected record field does not have a value in it. You do not specify a comparison value for this operator. Key fields, required fields, character fields, and numeric fields do not allow null values.
less than	The value in the record field is less than the comparison value.
like	The value in the selected field matches a specified string pattern. The comparison value may be a string that contains wildcard characters. The wild-card characters that PeopleSoft Query recognizes are % and _. % matches any string of zero or more characters. For example, C% matches any string starting with C, including C alone. _ matches any single character. For example, _ones matches any five-character string ending with *ones,* such as Jones or Cones. To use one of the wild-card characters as a literal character (for example, to include a % in your string), precede the character with a \ (for example, percent\%).

[2] Condition Types table from **Enterprise PeopleTools 8.48 PeopleBook: PeopleSoft Query, Defining Selection Criteria**, September 8, 2009.

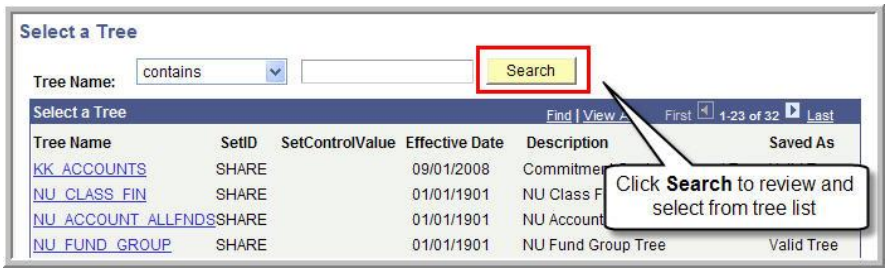
Selection criterion is a standard for comparison of values. The steps for creating criteria differ depending on the type of value: field, expression, constant, prompt, subquery, list, effective date, tree option. Not all value types are available for all condition types; for example, the *exists* condition type is only available for subquery. Query will only show you the value types for the selected condition type. See Value Type table.[3]

Value Type	Action
Field	The value in the selected field is compared to the value in another field, usually a field in another record component. When you select *Field* as the comparison value, the Choose Record and Field page appears. The Record Alias field lists all the records that are part of the current query. Select the record and the field. The selected field name appears in the second Expression column of that field's row.
Expression	The value in the selected field is compared to an expression that you enter, which PeopleSoft Query evaluates once for each row before comparing the result to the value in the selected field. When you select Expression as the comparison value, the Define Expression page appears. In the text box, enter a valid SQL expression. To add a field or user prompt to the expression, click the Add Field or Add Prompt link, respectively. These links display the same pages that you see when adding a field or prompt as a comparison value: Add Prompt displays the Run-time Prompt page; Add Field displays the Select a Field page. The only difference is that PeopleSoft Query adds the field or prompt to your expression rather than using it directly as the comparison value.
Constant	The value in the selected field is compared to a single fixed value. When you select Constant as the comparison value, the Define Constant page appears. In the text box, enter the value that you want to compare the first expression to. To add a value by selecting it from a list, click the Look Up button to display the Select a Constant page.
Prompt	The value in the selected field is compared to a value that you enter when running the query. When you select *Prompt* as the comparison value, the Define Prompt page appears. Click the New Prompt link to move to the Edit Prompt Properties page. To modify an existing prompt, you can click the Edit Prompt link.
Subquery	The value in the selected field is compared to the data that is returned by a subquery. When you select Subquery as the comparison value, the Define Subquery page appears. Click the Define/Edit Subquery link to move to the Records tab to start a new query.
In List	The value in the selected field is compared to a list of values that you enter. This value type is available only when the selected operator is *in list* or *not in list.* When you select *In List* as your comparison value, the Edit List page appears. Use the Look Up button to display the Edit List page and search for the desired values. Note that values will appear here only if you have entered them previously.
Current Date	The value in the selected field is compared to the current date on the database server.

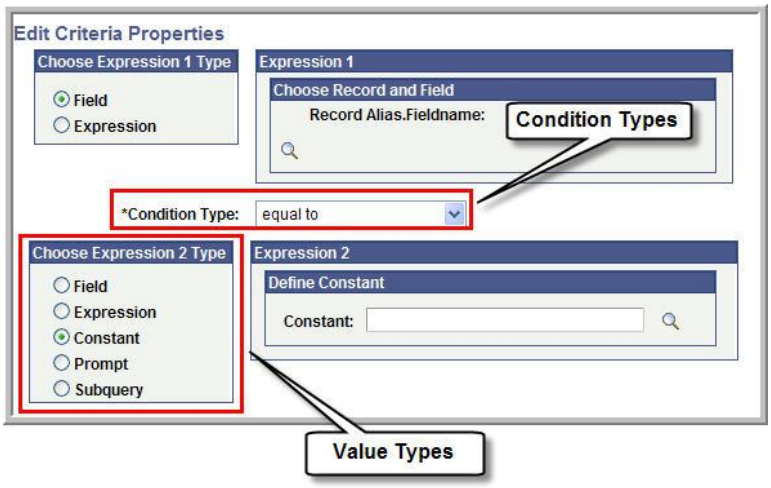
[3] Value Type table from **Enterprise PeopleTools 8.48 PeopleBook: PeopleSoft Query, Defining Selection Criteria**, September 8, 2009.

Tree Option	The value in the selected field is compared to a selected set of tree nodes. This value type is available only when the selected operator is *in tree* or *not in tree.* When you select *Tree Option* as the comparison value, the Select Tree Node List page appears. Use this page to create a list of values for PeopleSoft Query to compare to the value from the first expression.
Effective Seq (effective sequence)	Used on some effective-dated records, the effective sequence is a sequencing number that is provided to further refine the effective date.

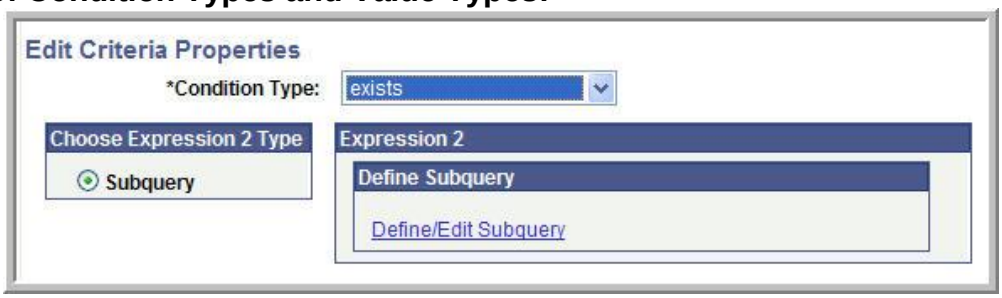
Note: use **in tree** criteria is an alternative method for filtering the DEPTID field.

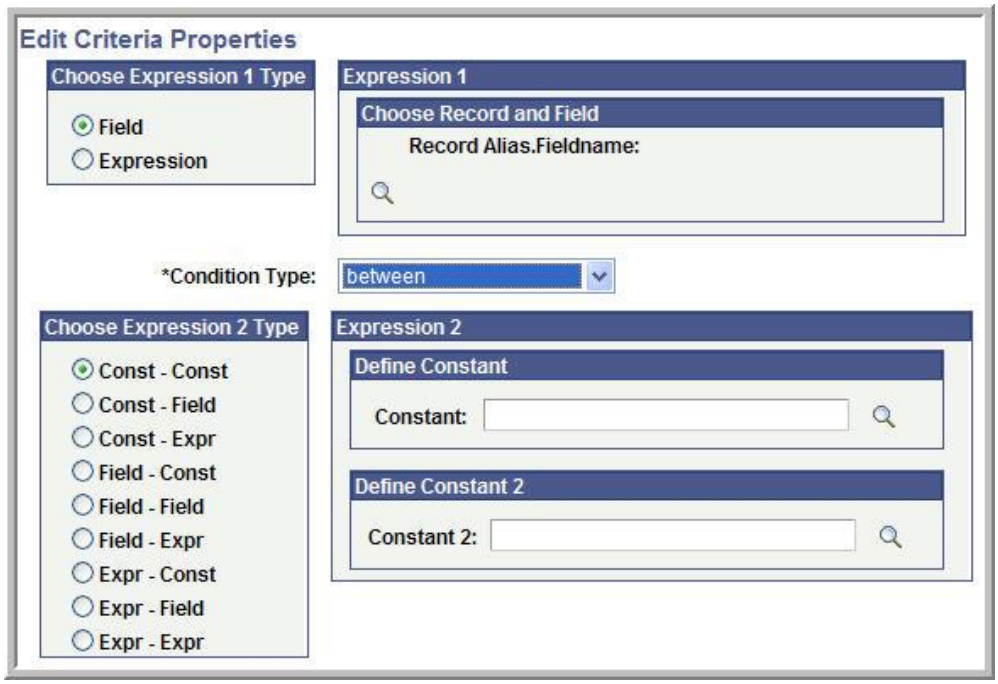


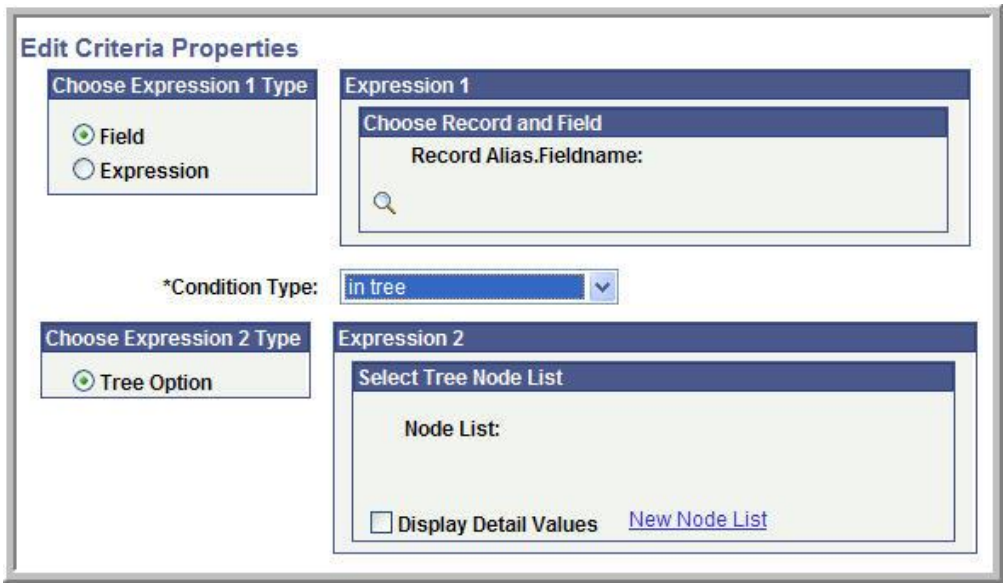
The **Edit Criteria Properties** page will reset to the appropriate choices when value or condition types are selected. See examples below.



Examples of Condition Types and Value Types:







Expressions

An expression is some form of operation involving numbers, character strings, or dates and times. It can use values drawn from specific fields in a table, constant values or a combination. Expressions can define constants (perhaps, in a separate column) or can derive calculations. Character string can be concatenated (linked together, as in a chain) into a field. Numerical or date columns can have mathematical operations applied.

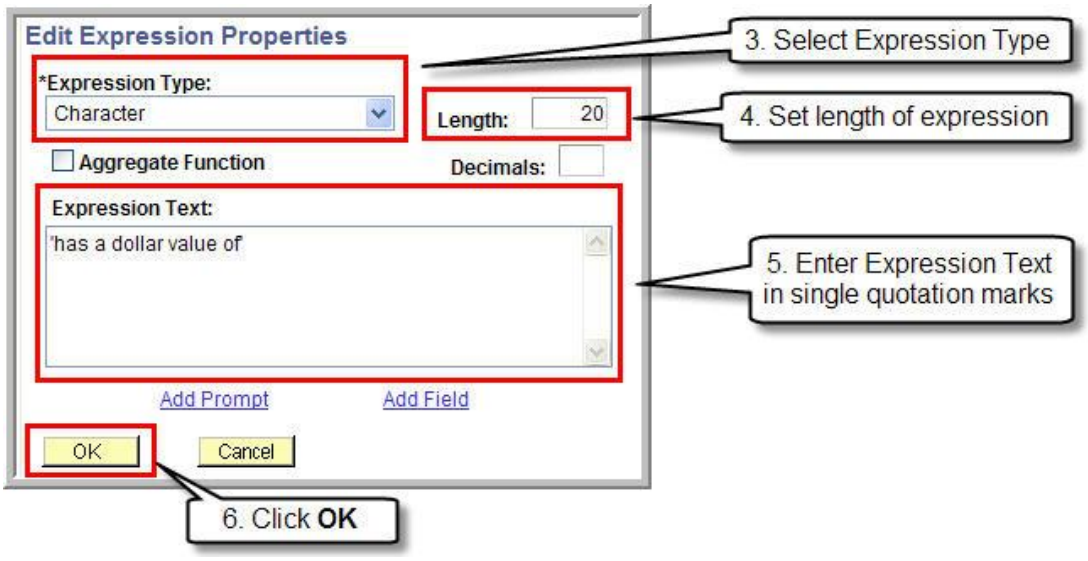
- *Character:* a literal value used as a column, ex. “The Vendor name is.”
- *Numeric:* can be a literal value or used as a mathematical expression, ex. numeric field *2 to multiply a field by 2, (+,-,*,/)
- *Date and Time*: can be a literal value or used as an interval expression, ex. date field +6mos (+,-,*,/)

Character Expressions

Step 1: Create literal value as a column

1. After selecting record(s) and field(s), select Expressions tab
2. Click **Add Expression**

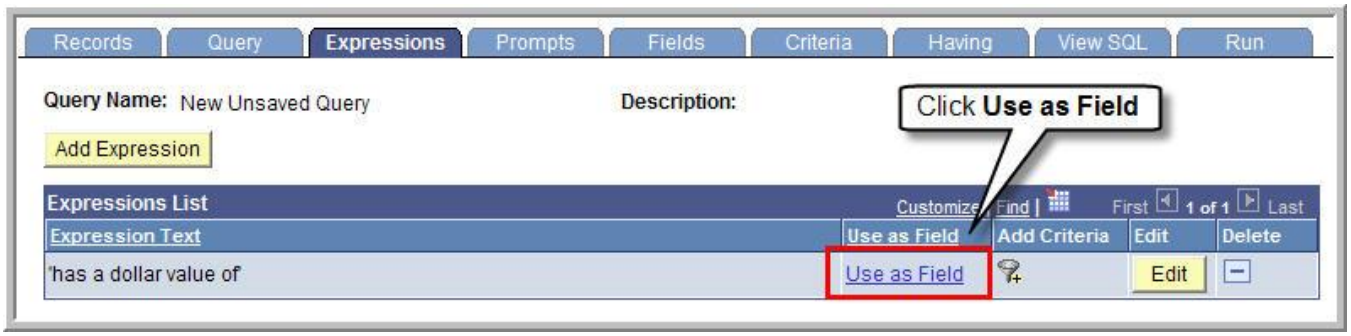
3. Select Expression Type as **Character**
4. Set the length of the column
5. Type in character string inside single quotation marks
6. Click **OK**



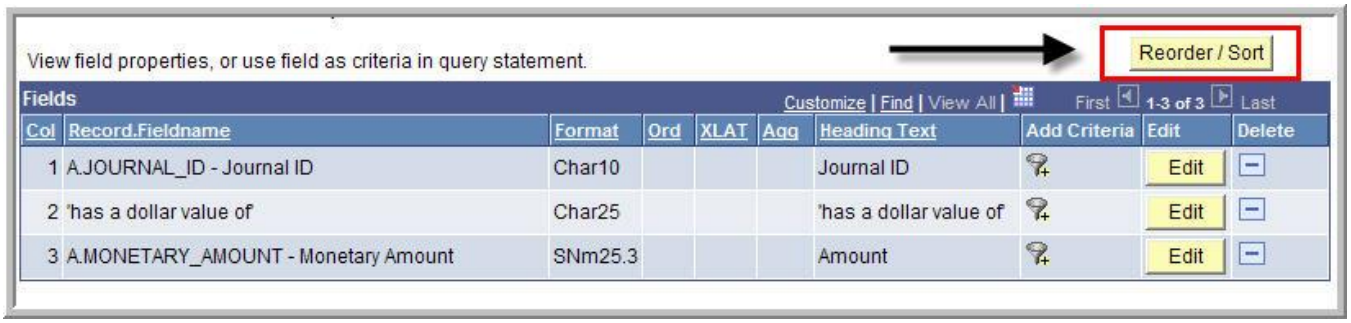
Character Expressions

Step 2: Use field as a column in the result set

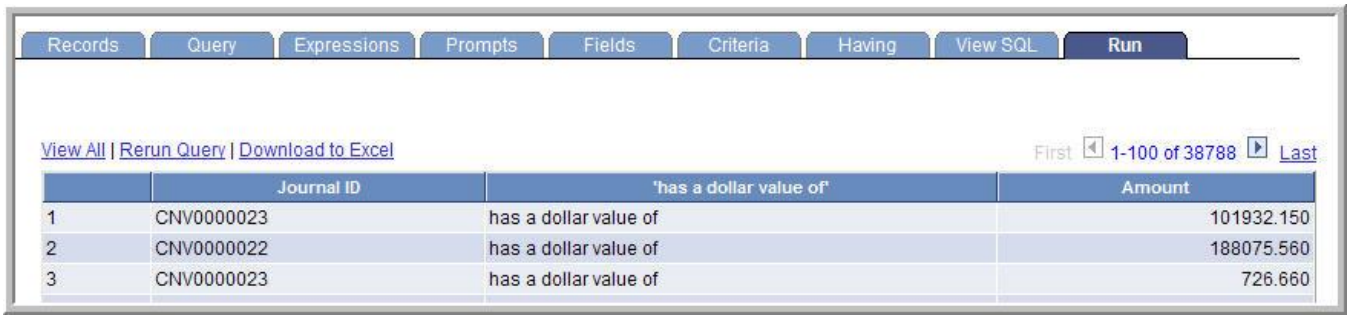
1. Click **Use as Field** if desired



2. Reorder fields and edit Header Text if desired



3. Run query to see character string in result set

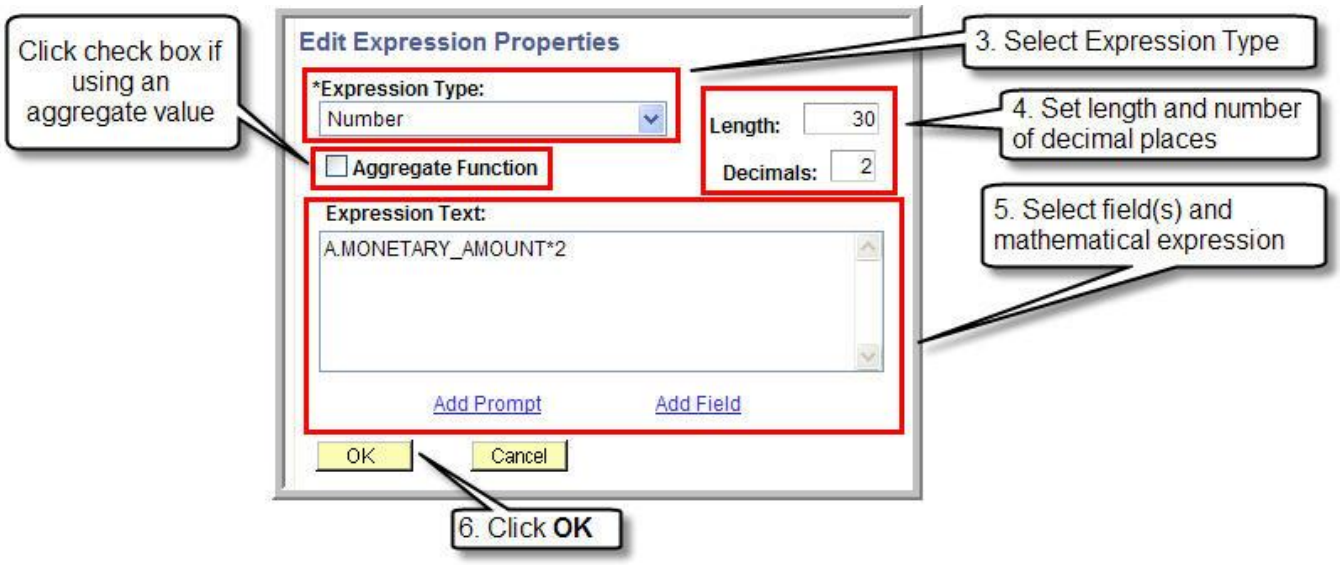


Numeric Expressions

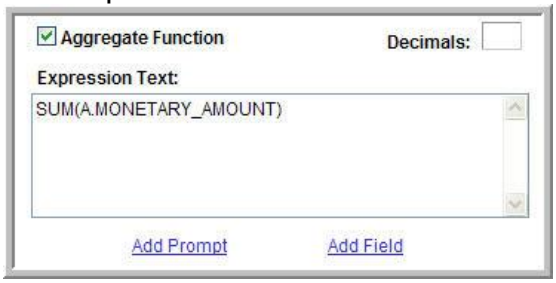
Step 1: Create a numeric expression

1. After selecting record(s) and field(s), select Expressions tab
2. Click **Add Expression**

3. Select Expression Type as **Number**
4. Set the length of the column
5. Type in operation in the Expression Text box
6. Click **OK**



If you are entering an aggregate value, such as SUM, AVG or COUNT, select the **Aggregate Function** check box and type in Expression Text box.

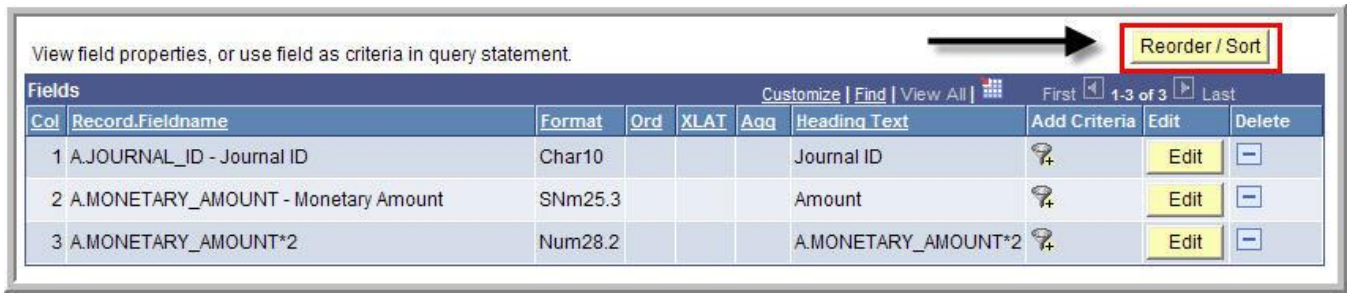


Numeric Expressions

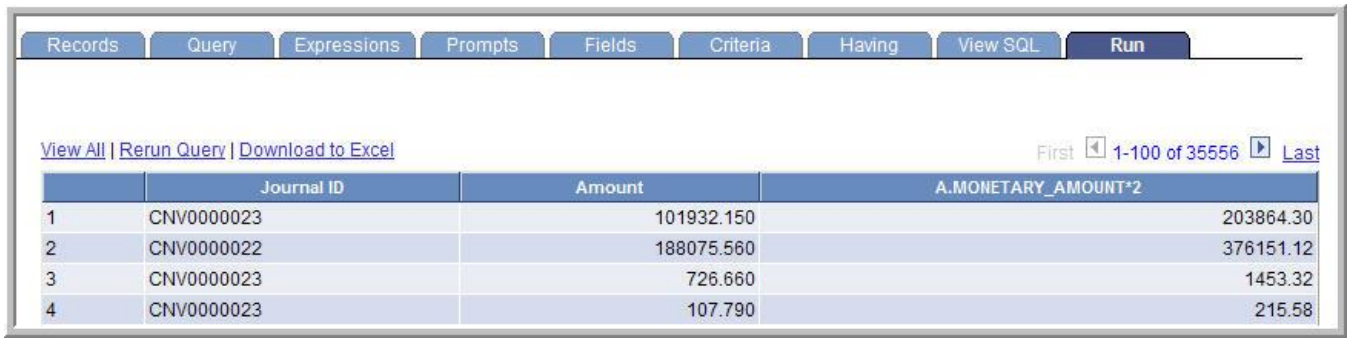
Step 2: Use field as a column in the result set

1. Click **Use as Field** if desired

2. Reorder fields and edit Header Text if desired



3. Run query to see numeric string in result set

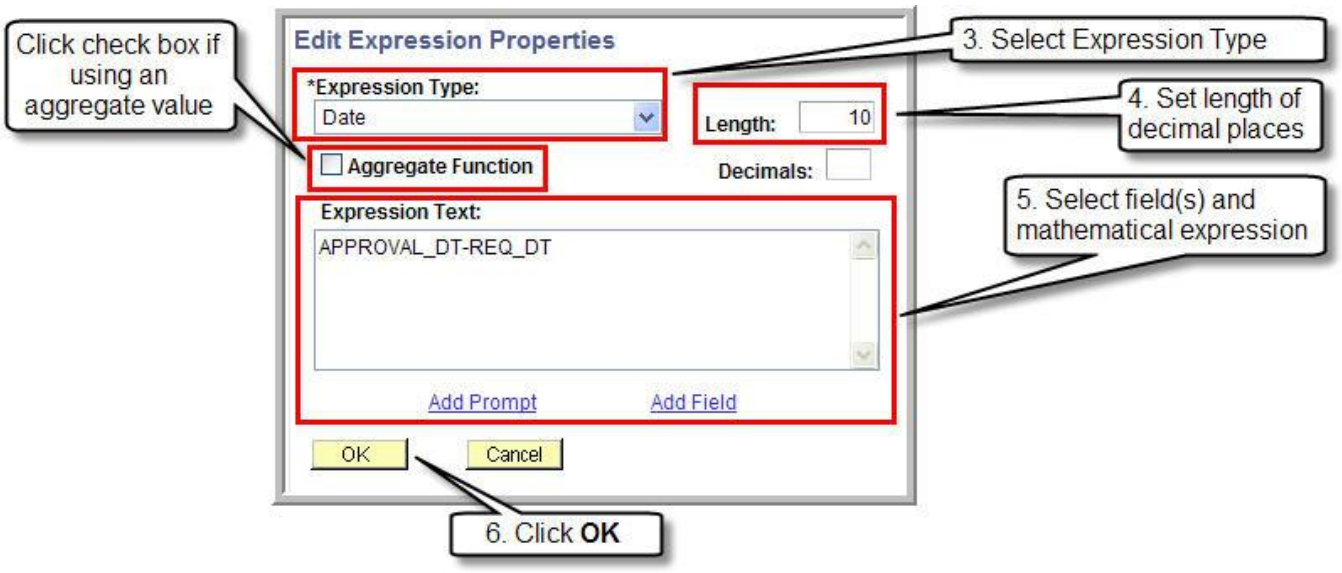


Date/Time Expressions

Step 1: Create a Date/Time expression

1. After selecting record(s) and field(s), select Expressions tab
2. Click **Add Expression**

3. Select Expression Type as **Date, Datetime or Time**
4. Set the length of the column
5. Type in operation in the Expression Text box
6. Click **OK**



IMPORTANT NOTE

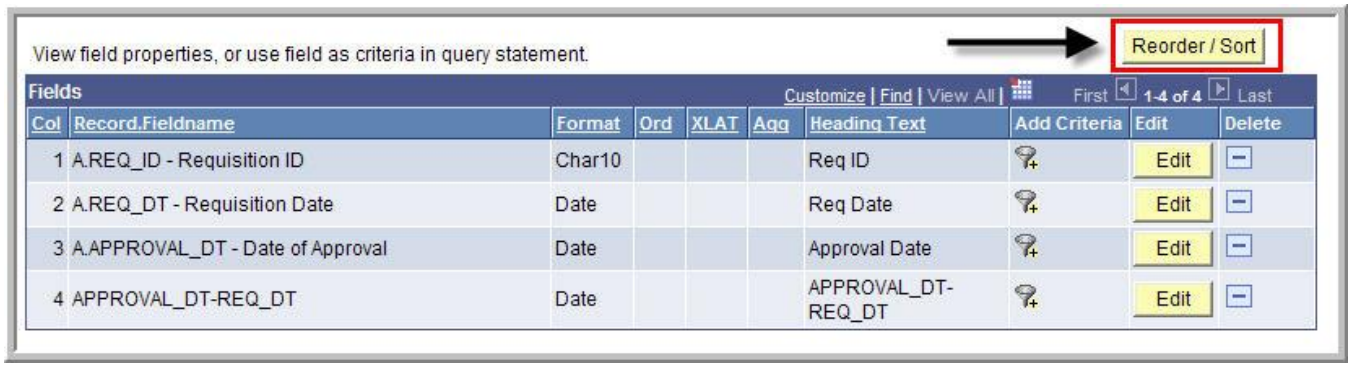
When you use **Add Field**, the application will insert the field's Alias name A. APPROVAL_DT. You need to either delete the "A." characters or enter the field freeform to make the field operate as a date field and not a character field.

Date/Time Expressions

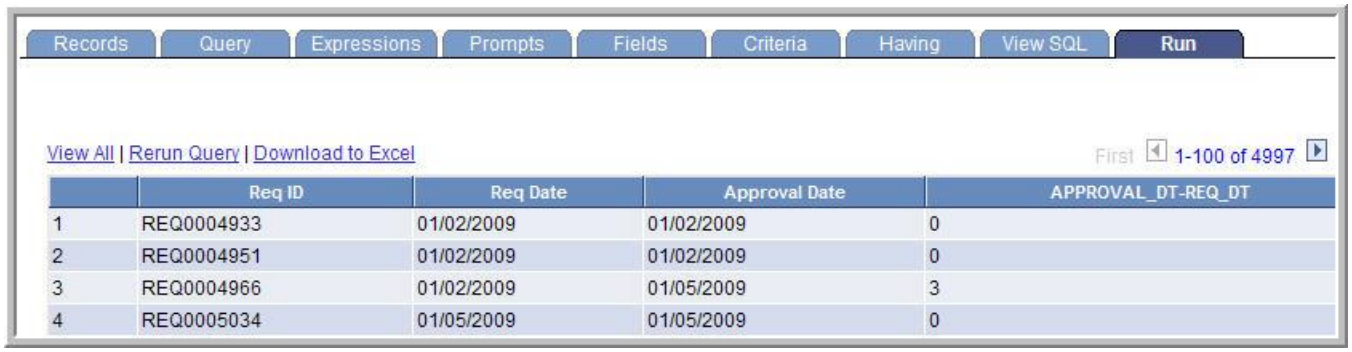
Step 2: Use field as a column in the result set

1. Click **Use as Field** if desired

2. Reorder fields and edit Header Text if desired



3. Run query to see date in result set



HAVING

A HAVING clause is similar to a WHERE clause, but used for rows of data that have been aggregated. A WHERE clause looks at the individual table rows before they are grouped in an aggregate function. A HAVING clause evaluates the results of the aggregation with further criteria. Essentially, use HAVING to select rows based on results of an aggregated function.

Examples:

After creating a query of purchase orders summed by vendor, you could add a HAVING clause to filter for vendor sums greater than $10,000.

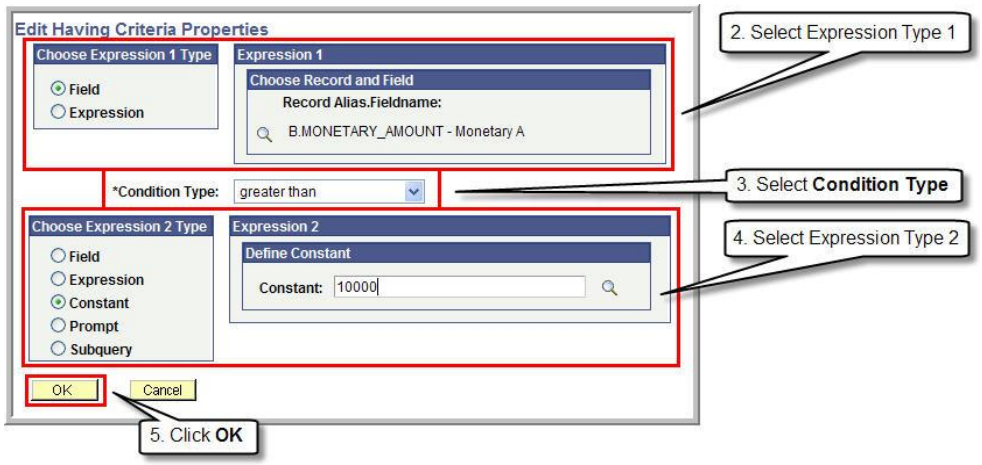
Show me the number of journal entries by source between 7/01/2009 and 7/31/2009. Show only sources where there have been more than 20 journals in total during this time period.

Step 1: Create a having expression

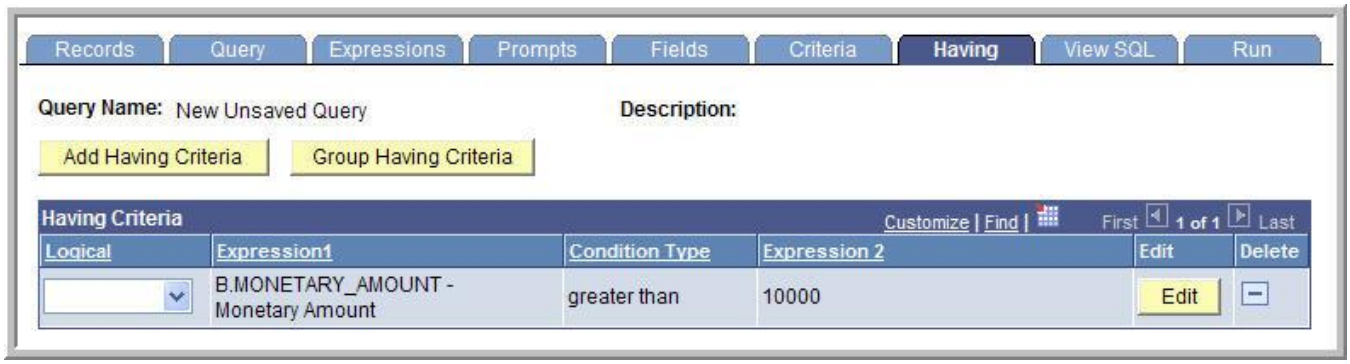
HAVING clauses can be added from the **Having** tab or from the **Criteria** or **Query** tabs, by clicking on the  next to aggregated field.

1. Click **Add Having Criteria**

2. Select field in Expression 1 Type for having clause
3. Select **Condition Type**
4. Select Expression 2 Type and define, as applicable
5. Click **OK**



Step 2: Confirm or create additional having clause



Logical Operators AND/OR

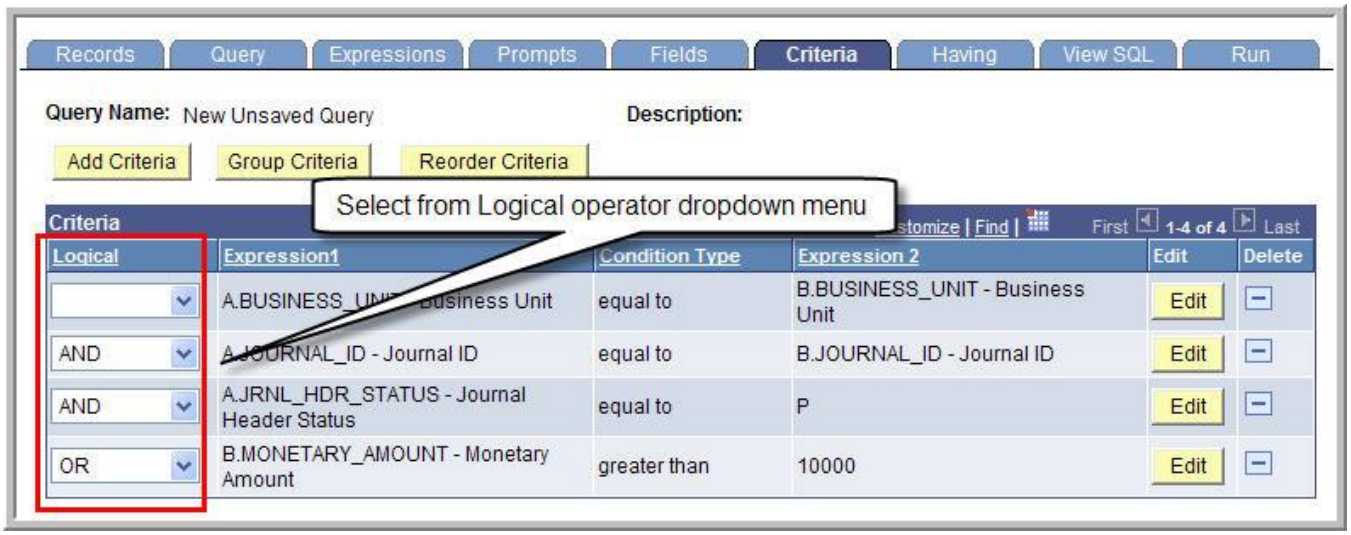
Logical Operators (AND, AND NOT, OR, OR NOT) allow you to relate multiple criteria in specific ways. When you specify two or more selection criteria, you need to coordinate the criteria. You can use the Logical column to further define the rows of criteria. The Logical Operator always defaults to AND. The first row of criteria will have a blank operator. When you use multiple criteria, rules of logic apply. The query evaluates criteria that are linked by AND before those linked by OR.

Examples:

Show me a list of Vendors in Chicago and Vendors who have previously provided lab supplies. The AND operator returns results that meet *both* conditions.

Show me a list of vendors located in Chicago OR New York. The OR operator returns results that meet one or the other condition.

Step 1: Select operator from Logical menu on Criteria tab



Grouping Criteria with Parentheses

Group Criteria controls the order in which query executes the criteria rows. The query evaluates the criteria inside the parentheses before the criteria outside the parentheses. The group criteria *(x+1)y* is indicating that *x+1* needs to be solved before multiplying the result by *y*. Use grouping to funnel data from largest to smallest to expedite search.

Examples:

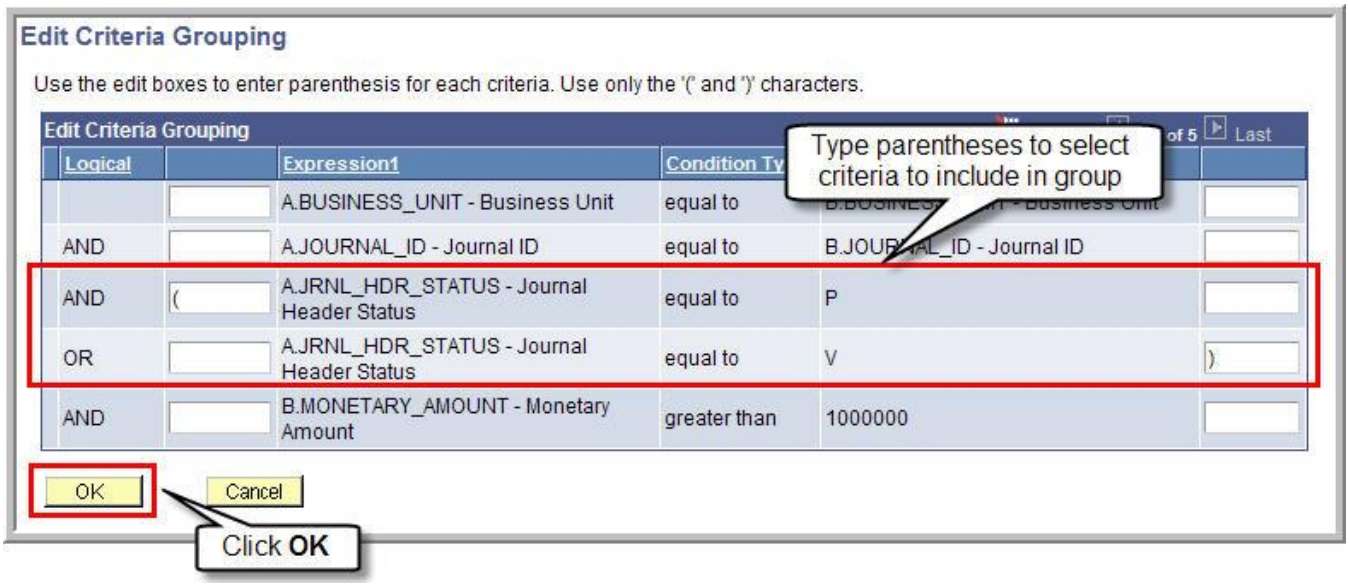
Show me a list of Vendors in Chicago or New York City who have provided lab supplies (filter for vendors by location in parentheses and then filter for the category lab supplies).

Show me posted or valid journals with credits that are greater than $10,000 (filter for journal status, then filter for greater than $10,000).

Step 1: Create Criteria groupings

1. Click **Group Criteria**

2. Type an open and close parenthesis around the criteria rows to be grouped
3. Click **OK**



Creating a Query

What is your question?

Given that question, what fields on what tables will show you the results?

How do they need to be joined together?

Do you need expressions, prompts, sorts, criteria to include/exclude data?

Example: Given a specific accounting period and fiscal year, show me a list of greater than $0 journal lines for a specific tree node, aggregated on the monetary amount, and sorted by Account code and then journal ID.

What tables/records and fields do you need for this?
A.JRNL_HEADER record with the fields:
JOURNAL_ID,
JOURNAL_DATE,
FISCAL_YEAR,
ACCOUNTING_PERIOD
DESCR
B.JRNL_LN record with the fields:
ACCOUNT,
FUND_CODE,
MONETARY_AMOUNT
C.GL_ACCOUNT_TBL record with the fields:
DESCR

How should these tables be joined?

Standard join on business unit, journal ID, journal date, unpost_seq, setid, account

What expressions, prompts, sorts, criteria do you need?

Create criteria for the tree node.

Create a prompt for fiscal year and accounting period.

Create a sort on Account and then journal ID.

Create SUM aggregation on monetary amount.

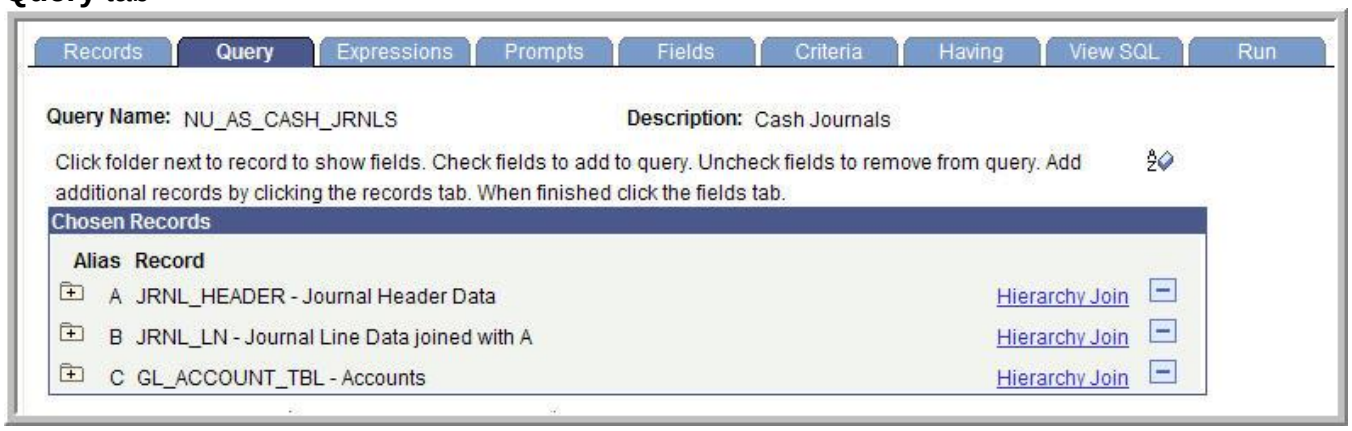
Add HAVING criteria to monetary amount for values greater than 0.

Run Query!

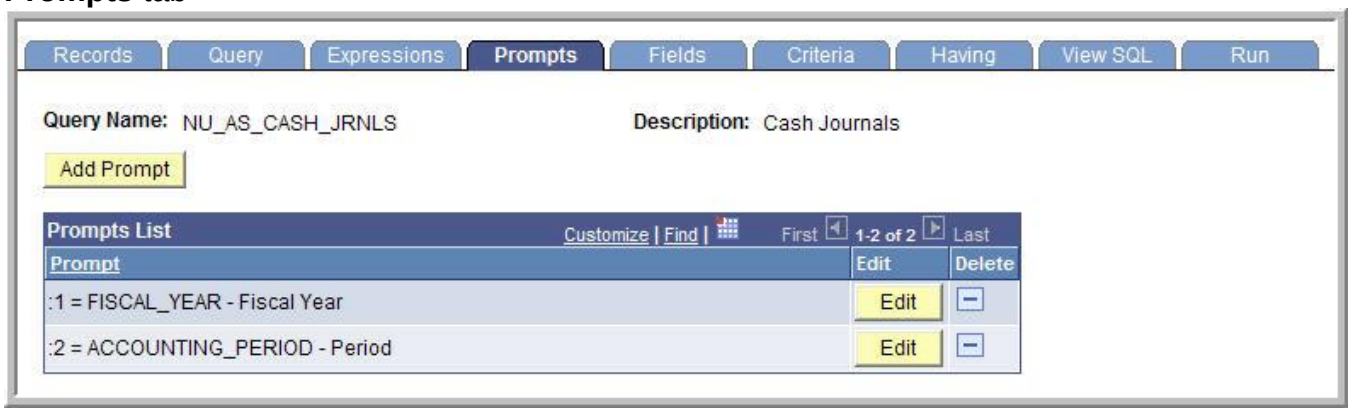
See screenshots of this query below.

Screenshots for example:

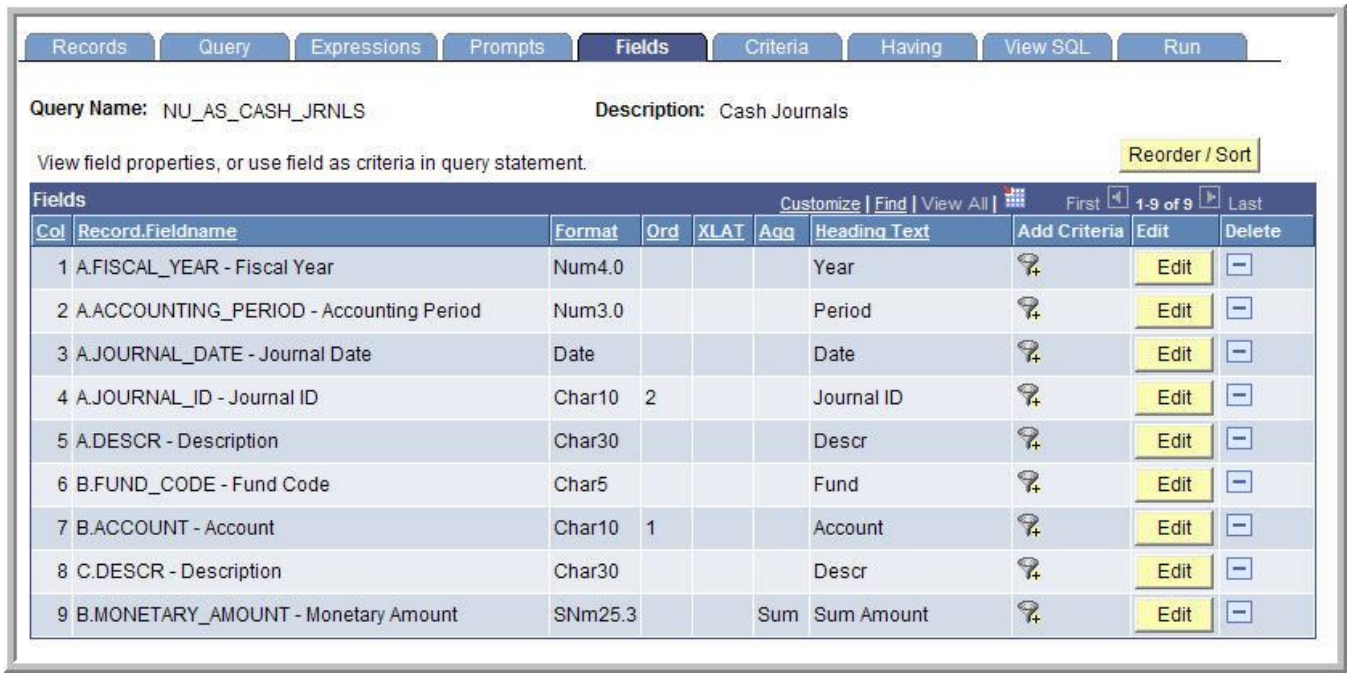
Query tab



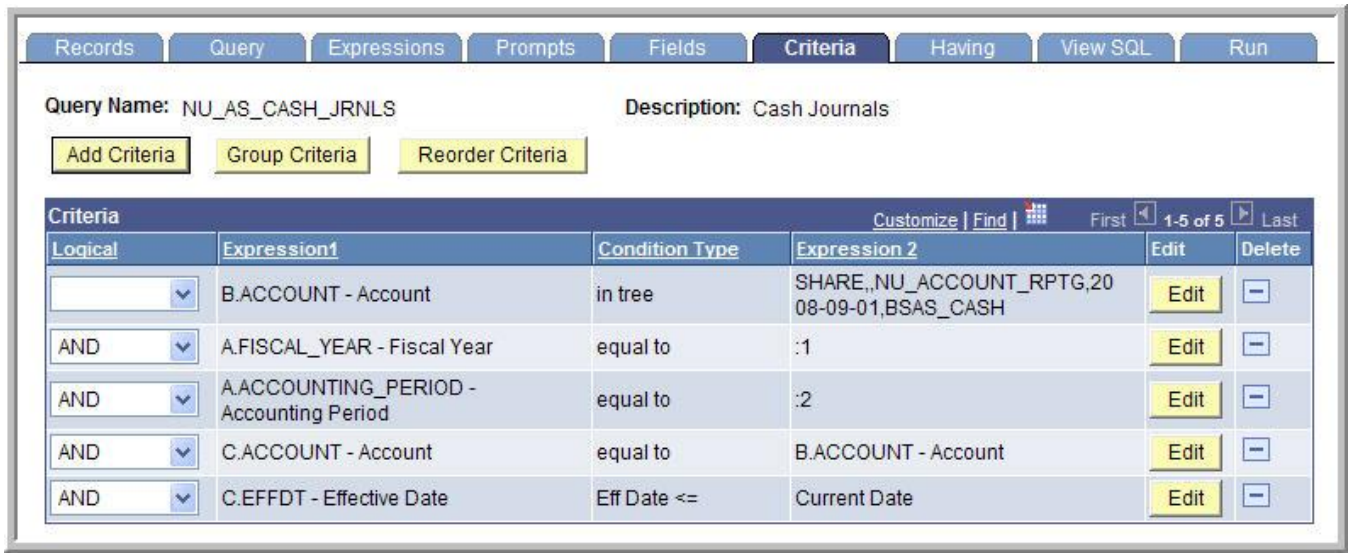
Prompts tab



Fields tab



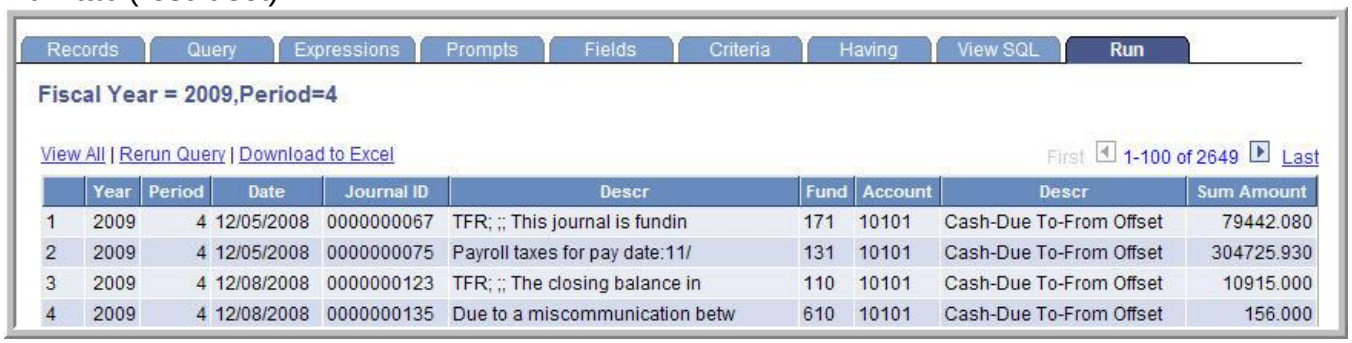
Criteria tab



Having tab

View SQL tab

Run tab (result set)



	Year	Period	Date	Journal ID	Descr	Fund	Account	Descr	Sum Amount
1	2009	4	12/05/2008	0000000067	TFR; ;; This journal is fundin	171	10101	Cash-Due To-From Offset	79442.080
2	2009	4	12/05/2008	0000000075	Payroll taxes for pay date:11/	131	10101	Cash-Due To-From Offset	304725.930
3	2009	4	12/08/2008	0000000123	TFR; ;; The closing balance in	110	10101	Cash-Due To-From Offset	10915.000
4	2009	4	12/08/2008	0000000135	Due to a miscommunication betw	610	10101	Cash-Due To-From Offset	156.000

Tips on the data structure

- There is a general hierarchy of tables: Header, Line, Distribution
- Think about where data might be to make an educated guess if it will be in a Header, Line or Distribution table.