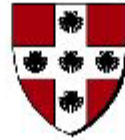




W E S L E Y A N
U N I V E R S I T Y



Student/Faculty Information System

PeopleSoft Query

Part III – Advanced Concepts



Student/Faculty Information System

PeopleSoft Query

Part III – Advanced Concepts

- This is the third module of instructions in the use of PeopleSoft Query. The first two modules that introduce you to PeopleSoft Query are:
 - [Part I – Basic Query Concepts and Query Viewer](#) (Lessons 1 – 4).
 - [Part II – Query Manager](#) (Lessons 5 – 10)
- **Part III** explores more complex Query usage topics and expands on subjects covered in Parts I and II. Before proceeding with this module, you should have a solid understanding of the processes shown in the first two sections.
- This module includes the following Lessons:

11. Improving Query Performance
12. Join Variations
13. Expressions – More Examples
14. Wes Functions Review
15. Prompts and Table Edits
16. Counts and Sums in Query
17. Mastering Outer Joins
18. Subqueries - Introduction
19. Creating Union Queries
20. Supplementary Information





Lesson 11: Improving Query Performance

Overview

When a query is executed, it can sometimes seem to take an fairly long time before you see results. When you are building and testing a query, a similar situation can arise. This lesson contains guidelines on how to fine tune query performance as to response time, through an understanding of the underpinnings of the query process. With careful planning and a systematic approach, you can simplify query development and improve the overall performance of queries you create.

Below are a few approaches that you might want to consider in relation to the smooth execution of a query. Some of this information may be familiar from previous lessons.

1. Run Time
2. Columns
3. Preventing a Runaway Query
4. Joins
5. Duplicates
6. Views
7. Table Ordering
8. PS Query and Security
9. Criteria – Constants and Prompts
10. Multiple Criteria Values
11. Return a Limited Number of Rows

1. Run Time:

After initially building a query, the first time it is run it will be slower than on later tries. The reasons for this are:

- The query may be saved, but is not compiled until the first time it is run
- Behind the scenes, Oracle maps out an execution plan and saves it in memory, so it will usually run faster on subsequent attempts.

2. Columns:

- Only pull the columns into the query that you need. Not all tables are populated in PeopleSoft, and not all fields in all records/tables are populated.

Lesson 11: Improving Query Performance



3. Preventing a Runaway Query

- Be sure to join all appropriate key fields to ensure that the query does not time out.

4. Joins: *(See [Lesson 12](#) for more details on Joins)*

- Because PeopleSoft is a relational database, data is stored in multiple tables. Consequently, a record may consist of several tables. A single page (panel) does not necessarily imply that the data on the page is stored in a single table. If at all possible, join on related records.
- **Related Records** are records/tables that have their relationships/joins established on the database side, rather than defined in the query itself. While query does a good job of detecting the appropriate join conditions, it is not perfect, so be sure the joins make sense.
 - The procedure for joining tables differs depending on how the tables that are being joined are related to each other.
 - The fewer the tables that need to be joined, the better. Reducing the number of tables will reduce the time it takes for the query to run.
 - Adding a description: Note that asking for Translate Table values or description fields from other tables will add more joins and more overhead. Where possible, use an **Expression** or a **Wes Function** to supply a description.
 - *Not all tables can be successfully joined.* If there are no fields that match up, you will return either no data or a very large amount of meaningless data – a [Cartesian join](#). A Cartesian join is a join of every row of one table to every row of another table.

Lesson 11: Improving Query Performance



One very useful Wesleyan View is named **WES_STUDENT** (Wesleyan Student Status). It contains extensive information about each student and, when queried effectively, can be a valuable tool.

5. Duplicates:

(See Lesson 20 for more information)

- In order to eliminate duplicates, you may have to try several methods.
- You may have to join on more than one field.
- The existence of many duplicates in the query record set may mean that you inadvertently performed a Cartesian join.
- In general, avoid using the DISTINCT property setting in PS Query.
- Except in the case of a very small record set, the DISTINCT property may dramatically decrease the performance of the query. However, depending upon the makeup of the query's elements, DISTINCT may be necessary.

6. Views:

- As discussed in Part II, Views – which are used like Records - are saved queries that can be referenced as record sets in other queries. They (mostly) end with a suffix of VW as the naming convention. Some are delivered with PS and some are created in ITS.
- Views are available for various data selection and restriction purposes. End users do not have the ability to create views.
- If a View is available as a record source (i.e., you can see it in the PS Query grid), then you can use it as a record source in a query.
- However, because it is basically a saved query, there is a slight performance hit when joining to one.

Lesson 11: Improving Query Performance

7. Table Ordering: *(See the demonstration [Selecting Tables in the Query Page by Size](#))*

- The order in which you bring records or views into the PS Query grid is important.
- PS Query and Oracle have built-in optimizers that make the code more efficient, but in most cases it helps the query run better if tables that are expected to return the fewest rows are added last in the FROM clause.
- The SQL statements that PS Query builds are evaluated from the bottom up. However, it will not correct a bad (or invalid) join.

8. PS Query and Security:

- PS Query passes through row-level security, which restricts whose data you can see.
- If you don't normally have access to view data through the pages/panels, you will be subject to the same restrictions in PS Query.

9. Criteria - Constants and Prompts: *(See the demonstration [Sorting Criteria by Selection Data](#))*

- You can sort criteria on the Criteria page to improve the performance of a query.
- When you are selecting constants for a criteria grid, and you don't know the values for that field, click the magnifying glass to obtain a list of valid values.
- When creating a Prompt, you will recall that if you select a Prompt Table Edit Type, the correct record ***should*** appear by default. However, that is not always the case, so you must verify that the Prompt Table field displays the record that stores the values that you want users to see.

Lesson 11: Improving Query Performance

10. Multiple Criteria Values:

Multiple criteria values would be a case where you need a query that pulls, for example, more than a single major, but not all of them.

- Build your queries incrementally.
 - If dealing with multiple criteria, pull them in one at a time, and rerun the query after each criterion is added.
 - Set temporary criteria of a single City, rather than an entire State. This allows you to control the output and identify whether the query is working correctly before you run it against your entire target population.
- When querying for multiple values, there are several ways to do it:
 - The **in list** operator – recommended ([See Lesson 8, Page 45, in Part II – Query Manager](#))
 - Multiple OR statements

- The UNION operator (usually fastest, but not recommended because of increased coding complexity ([See Lesson 19](#)))
- The **like** operator may be appropriate if a large number of values are targeted. *ACAD_PLAN like E%* will return ECON and ENGL, etc. ([See Lesson 8, Page 49, in Part II – Query Manager](#))
- A Subquery is usually the slowest. The subqueries have to be evaluated first, and must be compiled before the parent query runs. ([See Lesson 18](#))

11. Return a Limited Number of Rows:

([See the demonstration *Expediting the Response by Returning Partial Results*](#))

- When testing, in order to see an abbreviated set of data, you can direct the query to return a set number of rows.

Lesson 11: Improving Query Performance

Below are demonstrations of three methods to aid in improving query response time.

1. **Selecting tables in the Query page by size**
2. **Sorting criteria in the Criteria page by selection of data**
3. **Expediting the response by returning partial results**



1. Selecting Tables in the Query Page by Size

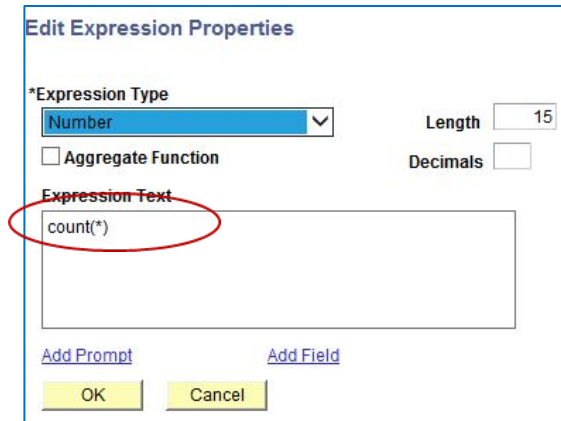
Selecting tables in order by size will have a big impact on query performance. Query runs using Oracle, a relational database, which executes a query from the bottom up - the smaller the table, the quicker the results.

- The first factor in determining the size of a table is simple – the fewer the fields, the smaller the table.
- If a table has a “_TBL” on the end of its name, it is also normally small.
- By using an Expression, the amount of data stored in a table can be easily determined. Here's how:
- Counting Rows in a Query:
 - a. In a new custom query add the first record you would like to use in your query. For this example, the record is the PeopleSoft delivered view named EXT_ORG_PRI_VW, but this can be applied to any record to which you have access.
 - b. Create a new Expression. On the **Expressions** page, click on the Add Expression button.

Lesson 11: Improving Query Performance

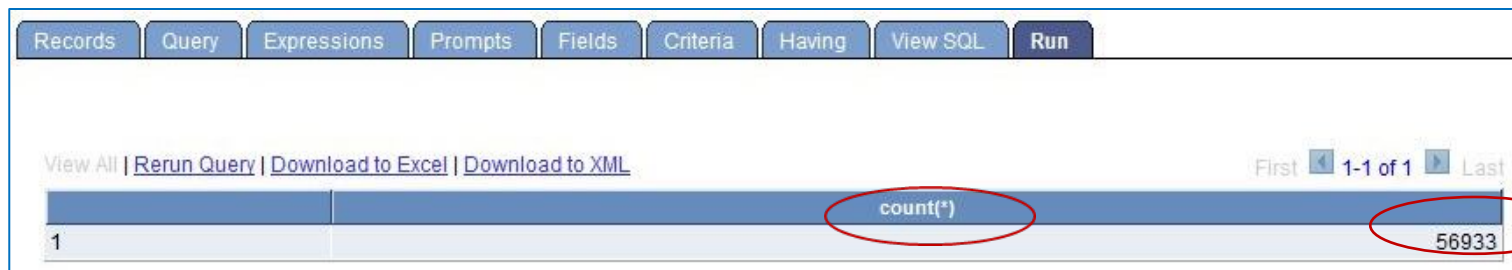
1. Selecting Tables in the Query page by Size (continued)

- c. Using the following sample, select the Expression Type of “Number,” change the Length value to “15,” and enter “count(*)” to count the total number of rows.
- d. Press OK. Click on **Add as Field** link.
- e. Run the query to view results. The results indicate the number of rows in the record. In this example, at the time of running the query, the total count is 56933 rows.
- f. Make a note of the name of the record and its size.



Edit Expression Properties

*Expression Type: Number (dropdown menu)
Length: 15
☐ Aggregate Function
Decimals: ☐
Expression Text: count(*)
Add Prompt Add Field
OK Cancel



Records	Query	Expressions	Prompts	Fields	Criteria	Having	View SQL	Run
View All Rerun Query Download to Excel Download to XML								
First 1-1 of 1 Last								
1	count(*) 56933							

Lesson 11: Improving Query Performance

1. Selecting Tables in the Query page by Size (continued)

- g. For each of the records you're considering for the query, you would repeat this exercise as applicable, deleting the record and selecting the next record name.
- h. Once you have chosen the records to utilize, you would create the new query, selecting the tables from largest to smallest, i.e., they will appear on the Query page in that order.

2. Sorting Criteria by Selection of Data

- Sorting criteria in the Criteria tab will also improve performance of a query. (This example is from the query named **WES_TRAIN_SORT_CRITERIA**.) Once the specific criteria required have been determined, the criteria should resemble this:

Query Name WES_TRAIN_SORT_CRITERIA Description Training sort criteria

[Add Criteria](#) [Group Criteria](#) [Reorder Criteria](#)

Criteria					
Personalize Find First 1-7 of 7 Last					
Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	A.STRM - Term	equal to	B.STRM - Term	Edit	-
AND	A.SESSION_CODE - Session	equal to	B.SESSION_CODE - Session	Edit	-
AND	B.INSTITUTION - Academic Institution	equal to	A.INSTITUTION - Academic Institution	Edit	-
AND	B.ACAD_CAREER - Academic Career	equal to	A.ACAD_CAREER - Academic Career	Edit	-
AND	A.SESSION_CODE - Session	not equal to	OMT	Edit	-
AND	A.STRM - Term	greater than	1131	Edit	-
AND	A.ACAD_CAREER - Academic Career	equal to	:1	Edit	-

[Save](#) [Save As](#) [New Query](#) [Preferences](#) [Properties](#) [Publish as Feed](#) [New Union](#) [Return T](#)

Lesson 11: Improving Query Performance



2. Sorting Criteria by Selection of Data (continued)

- All “table.field” joins should be located in the upper portion of the Criteria tab, with the first criteria row relating to the largest table, etc.

- All outer joins (reviewed later in this module) should be in the middle portion of the Criteria tab.
- Most importantly, all constants and prompts should be in the lower portion of the criteria.

Query Name WES_TRAIN_SORT_CRITERIA Description Training sort criteria

[Add Criteria](#) [Group Criteria](#) [Reorder Criteria](#)

Criteria	Expression1	Condition Type	Expression 2	Edit	Delete
Local	A.STRM - Term	equal to	B.STRM - Term	Edit	-
AND	A.SESSION_CODE - Session	equal to	B.SESSION_CODE - Session	Edit	-
AND	B.INSTITUTION - Academic Institution	equal to	A.INSTITUTION - Academic Institution	Edit	-
AND	B.ACAD_CAREER - Academic Career	equal to	A.ACAD_CAREER - Academic Career	Edit	-
AND	A.SESSION_CODE - Session	not equal to	OMT	Edit	-
AND	A.STRM - Term	greater than	1131	Edit	-
AND	A.ACAD_CAREER - Academic Career	equal to	:1	Edit	-

[Save](#) [Save As](#) [New Query](#) [Preferences](#) [Properties](#) [Publish as Feed](#) [New Union](#) [Return T](#)

Lesson 11: Improving Query Performance



3. Expediting the Response by Returning Partial Results (ROWNUM)

- If you would like to return a limited number of rows, you can use ROWNUM in an Expression. This method could be used when testing to see a small sampling of what would be returned.
- A ROWNUM value is not permanently associated with a row.
 - Using the query **WES_TRAIN_SORT_CRITERIA**, add an Expression with just the word “rownum” (upper or lower case, no quotes) and a Length of 3. Click the *Use as Field* link.
 - From the **Fields** page, set the criteria for the new field of **rownum** as “less than 16” (to limit the number of rows to 15).
 - Click Run. When prompted for the Career, type UGRD.
 - The output looks like this. Note only 15 rows are displayed.

Edit Expression Properties

*Expression Type
Character Length

☐ Aggregate Function Decimals

Expression Text
rownum

[Add Prompt](#) [Add Field](#)

Career = UGRD

[View All](#) | [Rerun Query](#) | [Download to Excel](#) | [Download to XML](#) First -15 of 15

	Course ID	Offer Nbr	Term	Session	Section	Career	Begin Date	End Date	rownum
1	013416	1	1136	JA	01	UGRD	07/01/2013	07/31/2013	1
2	009466	1	1136	JN	01	UGRD	05/29/2013	06/27/2013	2
3	011038	3	1136	JN	01	UGRD	05/29/2013	06/27/2013	3
4	011038	4	1136	JN	01	UGRD	05/29/2013	06/27/2013	4
5	011038	2	1136	JN	01	UGRD	05/29/2013	06/27/2013	5
6	004930	1	1136	JN	01	UGRD	05/29/2013	06/27/2013	6
7	002858	2	1136	JN	01	UGRD	05/29/2013	06/27/2013	7
8	010772	2	1136	JA	01	UGRD	07/01/2013	07/31/2013	8
9	010773	2	1136	JA	01	UGRD	07/01/2013	07/31/2013	9
10	010775	2	1136	JN	01	UGRD	05/29/2013	06/27/2013	10
11	010776	2	1136	JN	01	UGRD	05/29/2013	06/27/2013	11
12	012485	2	1136	JN	01	UGRD	05/29/2013	06/27/2013	12
13	012485	3	1136	JN	01	UGRD	05/29/2013	06/27/2013	13
14	012485	4	1136	JN	01	UGRD	05/29/2013	06/27/2013	14
15	012485	5	1136	JN	01	UGRD	05/29/2013	06/27/2013	15



Lesson 12: Join Variations

(You may want to review Lesson 7, Mastering Record Joins and Selection Criteria, in Part II)



Overview

- It is important to remember the basic rules of table joins:
 - Join records on their common keys.
 - If you have an **Any Record Join***, you will add to your criteria, unless there is only one row of data in the record to be joined.
- A join enables you to retrieve from two or more records or specify criteria from more than one record. In Query, *predefined joins* can be generated as a **Hierarchical join** or a **Related Record join**. Since these types of joins are already predefined, you don't have to add criteria to manually link the records.
- Besides **Auto Join**, other ways to seek and join records are:
 - **Record Hierarchy Joins**
 - **Related Record Joins**
 - **Manual Joins and Unjoins.**

Tables and Views

- A record shown in your list of records may be either a table or a view.
- A table physically stores data. A view is a logical representation of data and may consist of data from multiple tables depending on how the record was defined.
- Additionally, views may already have criteria associated with them. Therefore, it may be easier for users to create a query from a view rather than a table.
- If an appropriate view is not provided and requires data from multiple tables, the Query user must know in which tables the data is stored, and how to join those tables.

* An **Any Record Join** is made by selecting your initial base record, then returning to select another record.

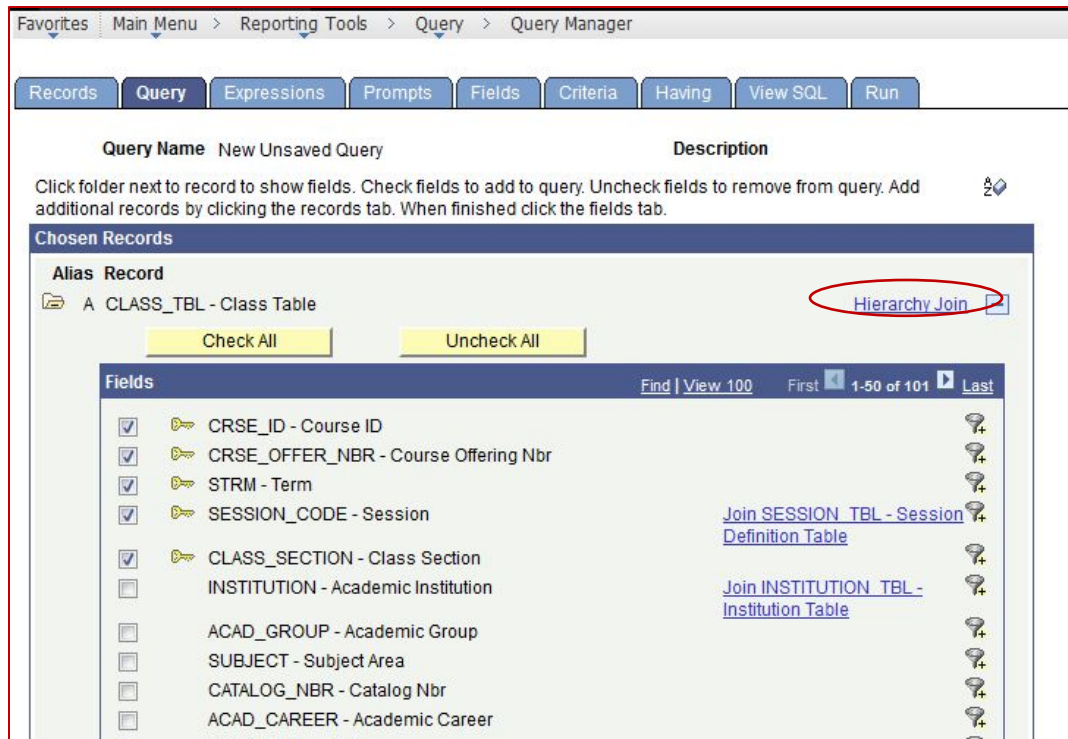
Lesson 12: Join Variations

Record Hierarchy Joins

A **Hierarchical join** uses records that are parents or children of each other. The hierarchical relationship is defined by the Parent Record. (A *child table* is a table that uses all the same key fields as its parent, *plus* one or more additional keys.)

To create a Record Hierarchy Join:

- Select the base record for your query and select the appropriate fields and criteria (in this example, A.CLASS_TBL).
- From the Query page, click the Hierarchy Join link in the upper part of the Chosen Records area.



Lesson 12: Join Variations

To create a Record Hierarchy Join (continued)

- c. When you click the Hierarchy Join link, a new page opens displaying all of the records that have a parent/child relationship with your selected record of CLASS_TBL.

Select record for hierarchy join

Left | Right

- CLASS_TBL - Class Table
- CLASS_ATTRIBUTE - Class Attribute Table**
- CLASS_CHRSTC - Class Characteristics Table
- CLASS_EXAM - Class Exam
- CLASS_MTG_PAT - Class Meeting Pattern Table
- CLASS_INSTR - Class Instructor Table
- CLASS_MTG_VW - Class Meeting Pattern View
- CLASS_NOTES - Class Notes
- CLASS_RSRV_CAP - Class Reserve Capacity Table
- CLASS_RSRV_GRP - Class Reserve Capacity Table
- WES_CLASS_BINS - Wes Limit/Enrollment Bins
- WES_INSTR_CLASS - Wes Instructor Classes

Cancel

- d. Select the desired record for the join (in this case, CLASS_ATTRIBUTE).
- e. The join is reflected on the Query page.

Records Query Expressions Prompts Fields Criteria Having View SQL Run

Query Name New Unsaved Query Description

Click folder next to record to show fields. Check fields to add to query. Uncheck fields to remove from query. Add additional records by clicking the records tab. When finished click the fields tab.

Chosen Records

Alias	Record	
A	CLASS_TBL - Class Table	Hierarchy Join -
B	CLASS_ATTRIBUTE - Class Attribute Table joined with A	Hierarchy Join -

Check All Uncheck All

Fields Find View All First 1-7 of 7 Last

<input type="checkbox"/>	CRSE_ID - Course ID	Join TERM_TBL - Term Definition Table
<input type="checkbox"/>	CRSE_OFFER_NBR - Course Offering Nbr	Join SESSION_TBL - Session Definition Table
<input type="checkbox"/>	STRM - Term	
<input type="checkbox"/>	SESSION_CODE - Session	
<input type="checkbox"/>	CLASS_SECTION - Class Section	
<input type="checkbox"/>	CRSE_ATTR - Course Attribute	Join CRSE_ATTR_TBL - Course Print Attribute Table
<input type="checkbox"/>	CRSE_ATTR_VALUE - Course Attribute Value	Join CRSE_ATTR_VALUE - Course Attribute Value Tbl

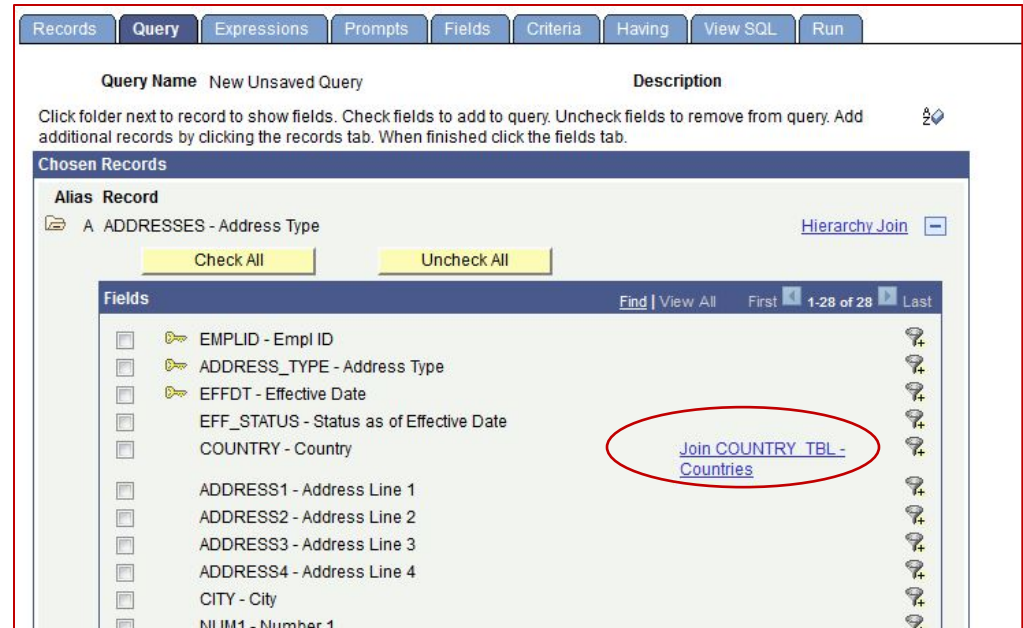
Lesson 12: Join Variations

Related Record Joins

Related Record joins use records from non-hierarchical records that are related by common fields. An example is a description table for common codes. On the Query page shown below the related record is COUNTRY_TBL.

To create a Related Record Join:

- Select the base record for your query and select the appropriate fields and criteria. In this example ADDRESSES.
- From the Query page, click the **Related Record Join** link (in this case [Join COUNTRY_TBL – Countries](#))
- From the Select Join Type page, select the standard join option and click OK.



Records Query Expressions Prompts Fields Criteria Having View SQL Run

Query Name: New Unsaved Query Description

Click folder next to record to show fields. Check fields to add to query. Uncheck fields to remove from query. Add additional records by clicking the records tab. When finished click the fields tab.

Chosen Records

Alias	Record
A	ADDRESSES - Address Type

Check All Uncheck All

Fields Find | View All First 1-28 of 28 Last

<input type="checkbox"/>	EMPLID - Empl ID	
<input type="checkbox"/>	ADDRESS_TYPE - Address Type	
<input type="checkbox"/>	EFFDT - Effective Date	
<input type="checkbox"/>	EFF_STATUS - Status as of Effective Date	
<input type="checkbox"/>	COUNTRY - Country	Join COUNTRY_TBL - Countries
<input type="checkbox"/>	ADDRESS1 - Address Line 1	
<input type="checkbox"/>	ADDRESS2 - Address Line 2	
<input type="checkbox"/>	ADDRESS3 - Address Line 3	
<input type="checkbox"/>	ADDRESS4 - Address Line 4	
<input type="checkbox"/>	CITY - City	
<input type="checkbox"/>	NUM1 - Number 1	

Select join type

Join Type

☒ Join to filter and get additional fields (Standard Join)

☐ Join to get additional fields only (Left outer join)

OK Cancel

Lesson 12: Join Variations

To create a Related Record Join (continued)

- d. The new join is reflected on the Query page.



Auto Joins

Query's Auto Join feature, which is enabled by default, is a good tool for getting your joins started. Query automatically looks for common keys when a new record is added and creates a criteria row for each common key combination.

In the (Auto) Join Criteria window, you can modify the criteria row to be added, add additional joining criteria, remove the joining criteria, or accept the criteria presented. Once you are satisfied with the joining criteria to be added, click the Finish button.

Manual Joins/Un-joins

In some cases, you may need to join or un-join records regardless of the auto joins Query has selected for you. Some PeopleSoft-delivered tables use a different name to further distinguish fields. In addition, custom tables may contain institution-specific field names.

Lesson 13: Expressions – More Examples

(To review, see [Lesson 8](#): Adding Expressions and Using Functions in Part II)



OVERVIEW

The main points to know about Expressions:

1. Expressions are calculations that PS Query performs as part of a query.
2. They are created on the Expressions Page.
3. Use them when you must calculate a value that PeopleSoft Query does not provide by default. The expression can become an additional field.
4. An expression can be used like a field.
 - a. If you use an expression as a field, the expression can be used like any other field in a query.
 - b. When you preview the query, the expression name appears as a column heading in the query.
 - c. When selected for output, you can change its column heading or sort it.
5. Normally, data that is the result of a calculation is produced when the query is run in real time.
6. You use expressions to display a field value differently from the way you store the value. Examples would be an expression that displays the appearance of a date as July 4, 2014, or an expression that inserts a space or hyphen in a field, or an expression that displays characters all in upper case.

Lesson 13: Expressions – More Examples



Reasons to use Expressions:

1. As columns in the query output
2. As comparison values in selection criteria
3. To create outer joins
4. To translate coded values
5. To use SQL commands

Using the SFIS Blog

- To create an expression, you need to know the SQL specific syntax. Illustrations can be found in a number of pages referenced on the [Expressions](#) pages in the SFIS Blog.
- In the SFIS Blog you will find specific documentation related to Wesleyan with many ways to utilize the expression tool.
- In addition, there are links in the SFIS Blog which will introduce you to websites and presentations with valuable information on working with this feature.

Additional Notes:

- On the following slide are further details on working with expressions which highlight:
 - Aliases
 - Working with Dates
 - Literals and Concatenation

Expressions Demonstrations

- Following the notes are examples that will give you a brief glimpse into what you can do with Expressions.
 - Dates
 - formatting
 - calculating
 - Case Statement - to apply if-then-else logic
 - SUBSTR() – to extract part of a string
 - Expressions and Prompts
 - Expressions and mathematical calculations

Lesson 13: Expressions – More Examples

Additional Notes on Expressions



Aliases and Expressions

1. You will recall that PeopleSoft names the first record you select as “A” and labels it as an **Alias**.
 - a. With the exception noted below regarding dates, the Alias *must* be included with the field name in an expression, such as A.EMPLID.
2. Dates: For expressions with *dates*, use the date field *without* an **Alias**.
 - a. Typically you would add a date to an expression so that the format can be changed or to perform a mathematical function. Unless reformatted, dates in PS appear in the form of yyyy-mm-dd (2014-05-01).
 - b. The expression should **not** look like this:
to_char((A.ACTION_DT), 'fmMonth DD, YYYY')
This would cause an error.
 - c. The expression **should** appear as:
to_char((ACTION_DT), 'fmMonth DD, YYYY')
 - d. Embedded spaces can be removed by placing the ‘fm’ prefix – as in ‘fmMonth DD, YYYY’
 - e. In order for the month to appear as **May** (mixed case), rather than **MAY** (upper case), type it as **Month** rather than **MONTH**.
 - f. SYSDATE is the term used for today’s date or the current date.
 - g. See the examples below showing expressions expression for formatting a date.

Creating Expressions Using Literals

1. Literals are text placeholders. They are useful for combining text from two or more columns in a query.
2. The Oracle concatenation operator is: | |
3. See **Lesson 8** for a detailed example.

Using Pipes and Literals

1. After having added B.CITY in the Expression Type box

Enter one space

Type | | ' | |

2. Click the **Add Field** link and select B.STATE

3. After having added B.STATE

Enter one space

Type | | ' | |

4. Click the **Add Field** link and select B.POSTAL

5. The line in the Expression Text box should now appear as follows:

B.CITY | | ' | | B.STATE | | ' | | B.POSTAL

6. Click OK.

Notes:

- There is a space between each set of pipes, i.e. | |
- There is a space after B.CITY and B.STATE
- Between the first two quote marks are a comma and one space.
- Between the second two quote marks is one space.
- Between each pipe and quote is a space.

Lesson 13: Expressions – More Examples



1. Expression with a formatted date

- This example shows a date that is formatted with the month spelled out (using “to_char”)
- The length is 18 characters to account for the length of “September.”
- As noted above, “fm” is added to remove any embedded spaces.
- “Month” is in mixed case so that the result is in mixed case.
- When the query is run the output resembles the following:

May 1, 2014
May 2, 2014

Edit Expression Properties

*Expression Type

Character

Length

18

☐ Aggregate Function

Decimals

Expression Text

to_char((ACTION_DT),'fmMonth DD, YYYY')

Add Prompt

Add Field

OK

Cancel

2. Other Expression Text with formatted dates and results

Length	Expression Text	Usage	Sample Output
10	to_char((SYSDATE),'mm/dd/yyyy')	TODAY'S DATE	05/05/2014
10	ACTION_DT - 7	Date minus 7 days (no formatting)	2014-04-24
18	to_char((ACTION_DT - 7),'fmMonth dd, yyyy')	Date minus 7 days	April 24, 2014

- As you can see, dates can be used in calculations with constants.
- You can also use dates in comparisons, to subtract two date values, to derive time values; and you can also convert a string value to a date value.

18	to_char((SYSDATE + 14),'fmMONTH dd, yyyy')	TODAY'S DATE plus 14 days	MAY 19, 2014
----	--	---------------------------	--------------

Lesson 13: Expressions – More Examples



3. Expression Using Case

- This expression uses the Case statement which is a method for applying if-then-else logic in an expression.
- The expression is used to create a new field that will display “Undergraduate Student” if the Academic Career is UGRD, “Graduate Student” if the Career is GRAD, “GLSP Student” if the Career is GLSP; and if the Academic Career is none of these, the text that will appear in the new field is “Check Career.”
- The Case statement appears as follows on the Edit Expression Properties page:
- A sample of the output would be:

Edit Expression Properties

*Expression Type
Character Length

☐ Aggregate Function Decimals

Expression Text

```
CASE  
WHEN A.ACAD_CAREER = 'UGRD' THEN 'Undergraduate  
Student'  
WHEN A.ACAD_CAREER = 'GRAD' THEN 'Graduate Student'  
WHEN A.ACAD_CAREER = 'GLSP' THEN 'GLSP Student'  
ELSE 'Check career'  
END
```

[Add Prompt](#) [Add Field](#)

{	UGRD	Undergraduate Student
}	UGRD	Undergraduate Student
{	UGRD	Undergraduate Student
}	UGRD	Undergraduate Student
{	UGRD	Undergraduate Student
}	GLSP	GLSP Student
{	UGRD	Undergraduate Student
}	GRAD	Graduate Student
{	GRAD	Graduate Student
}	UGRD	Undergraduate Student
{	UGRD	Undergraduate Student
}	UGRD	Undergraduate Student
{	GRAD	Graduate Student
}	DCST	Check career
{	UGRD	Undergraduate Student
}	UGRD	Undergraduate Student

Lesson 13: Expressions – More Examples



4. Expression Using substr()

- This expression uses the substr() method which extracts parts of a string, beginning at the character at the specified position, and returns the specified number of characters.
- This is a sample of the data returned by a query displaying the fields of EXT_ORG_ID and ATP_CD (Note that the ATP_CODE field is 6 characters long):

	Org ID	ATP
9146	225566	060671
9147	126769	800146
9148	335037	0230cc
9149	129593	004950
9150	931002	051082
9151	425530	233120
9152	849500	450249
9153	100782	005139
9154	296630	0314CW
9155	106145	110950

- Some of the items in the ATP_CD field end in one or two letters rather than numbers. This Query will use this expression to select just those items: **substr(A.ATP_CD,5,2)**
- The expression searches for and displays only the 5th and 6th characters in that field.

Edit Expression Properties

*Expression Type: Character Length: 2

☐ Aggregate Function Decimals:

Expression Text: SUBSTR(A.ATP_CD,5,2)

[Add Prompt](#) [Add Field](#)

OK Cancel

- The “5” represents the position of the fifth character, and “2” represents the number of characters to be displayed. (You would use a minus to count from the end, e.g. “-5.”)
- The Expression Length is set at “2.”
- The Use as Field link is activated.

Lesson 13: Expressions – More Examples

4. Expression Using substr() (continued)

- h. Once the query is run to verify the new field output, criteria can be associated with the field.
- i. On the **Fields** page, click on the funnel next to the new field.

Fields									
Personalize Find View All [Grid Icon] First 1-4 of 4 Last									
Col	Record.FieldName	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.EXT_ORG_ID - External Org ID	Char11	2			Org ID	[Funnel Icon]	Edit	[Minus Icon]
2	A.ATP_CD - ATP Code	Char6				ATP	[Funnel Icon]	Edit	[Minus Icon]
3	SUBSTR(A.ATP_CD,5,2)	Char2				SUBSTR(A.ATP_CD,5,2)	[Funnel Icon]	Edit	[Minus Icon]

- j. On the **Edit Criteria Properties** page, select Condition Type of “greater than” and Constant of “99.” This will limit the output to items in the new, two-character field that are greater than 99, i.e. those items that have letters rather than numbers.

Edit Criteria Properties

Choose Expression 1 Type

☐ Field

☒ Expression

Expression 1

Define Expression

Expression: SUBSTR(A.ATP_CD,5,2)

[Search Icon] [New Expression](#) [Edit the Expression](#)

*Condition Type: greater than

Choose Expression 2 Type

☐ Field

☐ Expression

☒ Constant

☐ Prompt

☐ Subquery

Expression 2

Define Constant

Constant: 99

OK Cancel

- k. When the query is run, only those ATP Codes with letters in 5th or 6th position show up in the query.

- l. Sample rows appear as follows:

	Org ID	ATP	SUBSTR(A.ATP_CD,5,2)
1	313808	0227cc	cc
2	323707	9490CW	CW
3	787514	1976CW	CW
4	323709	9492CW	CW
5	206887	0126CW	CW
6	206903	0130CW	CW
7	354301	0381cw	cw
8	296664	0304CW	CW
9	323731	9495CW	CW
10	323732	90029D	9D
11	198069	0113CW	CW



Expressions and Prompts

1. Using an Expression with a Prompt was demonstrated in [Lesson 9, page 70](#). Also see [Lesson 15](#) about Prompts. The following is another approach using Expressions and Prompts together.

2. The sample query has six existing expressions. All are from Wes Functions:

Expressions List
Expression Text
WES_GET_CLASS_YEAR(A.EMPLID)
WES_GET_NAME (A.EMPLID)
WES_GET_MAJORS(A.EMPLID)
WES_GET_ACAD_PROG_DESCR(A.ACAD_PROG)
WES_GET_ADDR_FROM_ADDR_USAGE(A.EMPLID,'BASIC')
WES_GET_MAJOR_DESCRS(A.EMPLID)

3. Under **Criteria** are currently two prompts

- Prompt :1** relates to a field in the record.
- Prompt :2** is associated with an Expression:
WES_GET_CLASS_YEAR(A.EMPLID)

Criteria	Expression1	Condition Type	Expression 2
Logical	WES_GET_CLASS_YEAR(A.EMPLID)	equal to	:2
AND	A.STRM - Term	equal to	:1

- Another Wes Function, named **WES_GET_MAJORS (A.EMPLID)**, displays the four-letter code for each Major linked with a student. If there is more than one major, they are all displayed with commas between.
- A prompt will be created using this expression along with the LIKE Condition Type and the % wildcard (if desired).
- To start, click on the **Criteria** tab.
- On the **Edit Criteria Properties** page:
 - Under **Choose Expression 1 Type**, select the **Expression** radio button.
 - When you do, the contents under the **Expression 1** header change.
 - Click on the lookup button to see the available expressions.

Edit Criteria Properties

Choose Expression 1 Type

☐ Field
☒ Expression

Expression 1

Define Expression

Expression:

[New Expression](#)
[Edit the Expression](#)

Lesson 13: Expressions – More Examples

Expressions and Prompts (continued)

Select an expression

Select an expression Personalize Find First 1-6 of 6 Last

- WES_GET_CLASS_YEAR(A.EMPLID)
- WES_GET_NAME(A.EMPLID)
- WES_GET_MAJORS(A.EMPLID)
- WES_GET_ACAD_PROG_DESCR(A.ACAD_PROG)
- WES_GET_ADDR_FROM_ADDR_USAGE(A.EMPLID,BASIC)
- WES_GET_MAJOR_DESCRS(A.EMPLID)

Cancel

- d. Click on the third link, **WES_GET_MAJORS(A.EMPLID)**. You are returned to the **Edit Criteria Properties** page. The expression now appears under **Expression 1**.

Edit Criteria Properties

Choose Expression 1 Type

☐ Field

☒ Expression

Expression 1

Define Expression

Expression: WES_GET_MAJORS(A.EMPLID)

New Expression Edit the Expression

*Condition Type: equal to

Choose Expression 2 Type

☐ Field

☐ Expression

☒ Constant

☐ Prompt

☐ Subquery

OK Cancel

Expression 2

Define Constant

Constant:

- e. Change **Condition Type** to like.
- f. Under **Choose Expression 2 Type** change **Constant** to **Prompt**.

*Condition Type: like

Choose Expression 2 Type

☐ Constant

☒ Prompt

Expression 2

Define Prompt

Prompt: New Prompt Edit Prompt

- g. Click **New Prompt** to open the **Edit Prompt Properties** page.
- h. The only change is to add **Heading Text** with a notation about the wildcard - **Enter Major (% to see all)**
- i. Click OK twice.

Edit Prompt Properties

Field Name:

*Type: Character

*Format: Upper

Length: 11

Decimals:

*Edit Type: No Table Edit

*Heading Type: Text

Heading Text: Enter Major (% to see all)

*Unique Prompt Name: BIND3

Prompt Table:

OK Cancel

Lesson 13: Expressions – More Examples



Expressions and Prompts (continued)

9. Run the query.

10. When the prompt box opens, after filling the first two boxes, you can populate the third box in a few ways:

- The four-letter code for one major
 - MATH
- Just the % wildcard to see all majors
 - %
- The % wildcard at the beginning and/or end with four or fewer letters entered to see double or triple majors with one particular one designated.
 - %CSS
 - ECON%
 - %ENGL%

Expressions and Mathematical Calculations

- You can use an expression to calculate a value mathematically and then display the result as a new column or field.
- The assumption is that an existing field, MAX_AMOUNT, is to be multiplied by 125. The expression would appear as follows.

- The **Expression Type** is changed to **Number**.
- The total **Length** is changed to 17 and **Decimals** are changed to 2 (to match the field).
- And MAX_AMOUNT * 125 is entered into **Expression Text**.
- Use as Field** is selected.

Lesson 13: Expressions – More Examples

Expressions and Mathematical Calculations (continued)

- The **Fields** page now includes the new field (you may change the column header as appropriate).

Fields									
Personalize Find View All First 1-2 of 2 Last									
Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	A.MAX_AMOUNT - Maximum Amount	Num15.2	1			Max Amt		Edit	
2	A.MAX_AMOUNT * 125	Num15.2				A.MAX_AMOUNT * 125		Edit	

- When the query is run, the output for the two fields is as follows:

	Max Amt	A.MAX_AMOUNT * 125
1	8.00	1000.00
2	8.00	1000.00
3	8.00	1000.00
4	8.00	1000.00
5	8.00	1000.00
6	10.00	1250.00
7	10.00	1250.00
8	15.00	1875.00
9	15.00	1875.00
10	15.00	1875.00
11	15.00	1875.00



Lesson 14: Wes Functions Review



Overview:

1. A Wes Function can be used to get a piece of data that you continually need to extract for your output. For example, rather than joining a table to get a student's class year, you can use a Wes Function in your Query to get this information.
2. Since the term "function" as used with PeoplesSoft Query can have a number of definitions, to differentiate this use of the term, we refer to it as a Wes Function.
3. *A Wes Function is a method created by the Wesleyan ITS staff that enables the PeopleSoft user to exploit the benefit that Expressions can lend to a query. It is pre-written text created specifically for inclusion in Query Expressions for Wesleyan users.*
4. You may incorporate functions into a query:
 - a. As a field
 - b. As part of the Where clause, which is created in the Criteria tab
5. You can make a Wes Function part of any expression, or you can combine more than one Wes Function into a query.
6. Most Wes Functions start with "WES." Those created for a specific office may start with other letters (WRG, WRL, WSF, etc.). If you have security access to the pertinent tables, you can use those functions. (Note: Admission related queries may not work properly since that office no longer uses PeopleSoft for recruiting and admitting purposes. They mostly start with WAD.)

Locating Wes Functions:

1. **Information on using Wes Functions is in the [SFIS Blog](#).** There you will find -
 - a. A number of Wes Function demonstrations.
 - b. [A list that contains tested functions](#). The entries contain the description, sample output, text, source fields, field type, length and notes for each function.
2. In addition, ***in PeopleSoft there is a menu with a searchable listing*** that provides a quick glimpse of functions, as well as a brief description of each. (All these functions contain the word "GET.")
 - a. The navigation is: **Main Menu > Wesleyan Menu > Campus Community > Database Function List.**
 - b. You can view the entire list or search for a specific function. If you don't know the full name, you can search with the % wildcard.
3. You can copy the text from either list into your expression, being sure to change the Alias as appropriate. To avoid errors, observe and enter the length of the output that is required, and be sure to click on the Use as Field link if you want to use it as a field in your query.

Lesson 14: Wes Functions Review

Wesleyan Function Examples:

EMPLID

Many of the Wes Functions are based on the EmplID field; that is, if the table/ record you are querying contains the EmplID field, you can create new fields with EmplID related information.

Examples include:

- WES_ACTIVE_STUDENT(A.EMPLID)
- WES_GET_ADMIT_TERM(A.EMPLID)
- WES_GET_CITIZENSHIP (A.EMPLID)
- WES_GET_CLASS_YEAR (A.EMPLID)
- WES_GET_CONTACT_CODES(A.EMPLID)
- WES_GET_EMAIL_USERNAME(A.EMPLID)
- WES_GET_ENRL_STATUS(A.EMPLID)
- WES_GET_EXP_GRAD_TERM(A.EMPLID)
- WES_GET_MAJORS(A.EMPLID)
- WES_GET_NAME (A.EMPLID)
- WES_GET_PREFERRED_EMAIL(A.EMPLID)
- WES_GET_STUDENT_TYPE(A.EMPLID)
- WES_GET_STUDENTS_ADVISORS(A.EMPLID)
- WES_GET_WESPO(A.EMPLID)

EMPLID Plus Other Fields

There are Wes Functions that are based on the EmplID field plus other fields. The other fields can be from the same record or different records:

- WES_GET_ABSENCE_REASON
(A.EMPLID,A.STRM,A.FORM_OF_STUDY)
- WES_GET_FORM_OF_STUDY
(A.STRM,A.EMPLID,A.ACAD_CAREER)
- WES_GET_FOS_WITH_AGREEMENT
(A.STRM,A.EMPLID,A.ACAD_CAREER)
- WES_GET_SRVC_IND(A.EMPLID,A.SRVC_IND_CD,A.SRVC_IND_ACT_TERM)
- WES_GET_TERM_LEVEL(A.EMPLID,A.STRM)
- WRG_GET_ADMIT_TERM(A.EMPLID,A.ACAD_CAREER)

Full Descriptions of Fields:

Some Wes Functions produce the description of a given field, such as:

- WES_GET_ACAD_ORG_DESCR(A.ACAD_ORG)
- WES_GET_ACAD_PLAN_DESCR(A.ACAD_PLAN)
- WES_GET_ACAD_PROG_DESCR(A.ACAD_PROG)
- WES_GET_COUNTRY_NAME(A.COUNTRY)
- WES_GET_PROG_REASON_DESCR(A.PROG_ACTION, A.PROG_REASON)
- WES_GET_TERM_DESC(A.STRM)

Lesson 14: Wes Functions Review

Today's Date and Term

Today's Date is referenced in some Wes Functions. You can include the designation of SYSDATE between the parentheses or you can leave the area empty as shown in the examples:

- WES_GET_ADVISEE_TERM()
- WES_GET_CURRENT_TERM()
- WES_GET_FALL_SPRING_TERM()
- WES_GET_MILL_TERM()
- WES_GET_NEXT_TERM()
- WES_GET_UGRD_TERM ()
- WRG_GET_REGISTRATION_STRM()
- WRL_GET_CURR_TERM()

Variations in Results

You can apply variations to some functions in how the result is presented. For example, to specify a particular kind of **Phone** - cell, home, or local - these options are available:

- WES_GET_PHONE(A.EMPLID,'CELL')
- WES_GET_PHONE (A.EMPLID,'HOME')
- WES_GET_PHONE (A.EMPLID,'LOC1')

The function related to **Citizenship** can return a number of possible displays, such as:

- WES_GET_CITIZEN_STATUS_DESCR
(A.CITIZENSHIP_STATUS,A.COUNTRY)
- WES_GET_CITIZENSHIP (A.EMPLID,'CODE')
- WES_GET_CITIZENSHIP (A.EMPLID,'DESCR')
- WES_GET_CITIZENSHIP_STATUS(A.EMPLID)
- WES_GET_CITIZENSHIP_STATUS(A.EMPLID,'LONG
'')
- WES_GET_CITIZENSHIP_STATUS(A.EMPLID,'SHORT')

The **Name** field display can also be presented in a few different ways:

- WES_GET_NAME (A.EMPLID,'N')
- WES_GET_NAME_PARTS(A.EMPLID,'PRI',SYSDATE,
'F L')
- WES_GET_NAME_PARTS(A.EMPLID,'PRI',SYSDATE,
'F')
- WES_GET_NAME_PARTS(A.EMPLID,'PRI',SYSDATE,
'L, F M')

Lesson 15: Prompts and Table Edits



Overview:

1. Prompts extend the life of a query and make the query more flexible for future requests. For example, instead of hard-coding a course ID value in the criteria, you can prompt the user to enter a course ID.
2. The query becomes more flexible, and you do not have to create multiple queries with hard-coded constant values for each course ID. You run the query, and the query prompts you for the course ID.

Restricting User Input with Table Edits

The **Edit Type** drop down on the **Edit Prompt Properties** page is where you can restrict user input by using table edits. There are three types of restricted edits:

1. **Prompt Table Edit**

- a. This type of edit restricts selection to only data that is in a Prompt Table.
- b. A lookup button (magnifying glass) under the Prompt Table header displays the source for the prompt.
- c. You click the lookup icon and then search and select a value for the Prompt Table.

- d. Note: If you select a Prompt Table edit type, the correct record *should* appear by default. However, that is sometimes not the case, so you should verify that the Prompt Table field displays the record that stores the values you want users to see. *(See the demonstration beginning on Slide 30 about this topic.)*

2. **Translate Table Edit**

- a. This type of edit restricts selection to only data that is in the Translate table (PSXLATITEM). The Translate Table is a PeopleTools table with predefined values that are associated with a particular field.
- b. A drop down list box indicates a Translate Table runtime prompt.

3. **Yes/No Table Edit**

- a. This type of edit restricts you to selecting only yes (Y) or no (N) values.

Lesson 15: Prompts and Table Edits

Unrestricted User Input

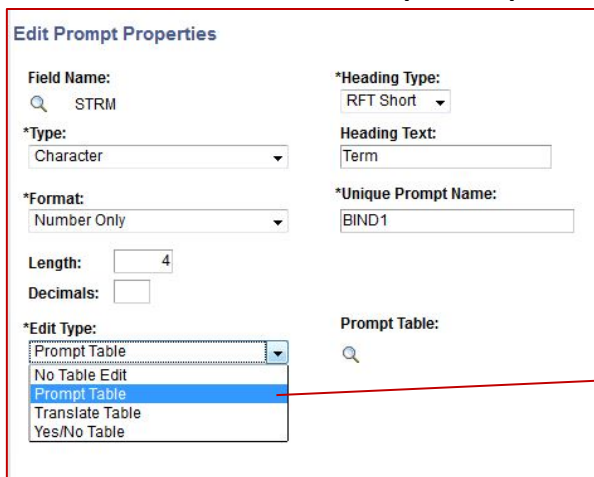
In addition, you can create a prompt without a table edit.

1. No Table Edit

- a. Some fields are unrestricted and have no edits. A likely field for no table edit is Date.
- b. A prompt *text box* (rather than a drop down) indicates that there is no table edit.
- c. Another example of a no edit prompt is using a [wild card for prompt criteria](#).
 - a. Make sure that for the prompt definition the edit type is 'No Table Edit.'
 - b. Also while using a wildcard prompt in the Criteria definition, remember to make sure the operator is 'Like' and not 'Equal to'

No Table Edit Example (Edit Prompt Properties page)

1. Change **Edit Type** from **Prompt Table** to **No Table Edit**
2. If necessary, remove **Prompt Table** name. There can be no **Prompt Table** name if the **Edit Type** is not also a **Prompt Table**.
3. Note that the prompt is a text box.



Edit Prompt Properties

Field Name:

*Heading Type:

*Type:

Heading Text:

*Format:

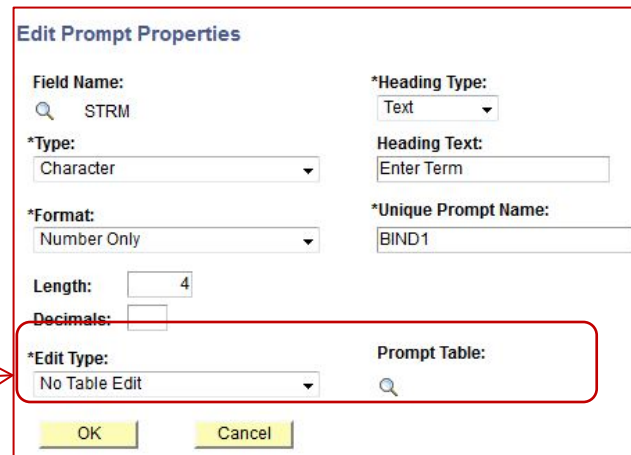
*Unique Prompt Name:

Length:

Decimals:

*Edit Type:

Prompt Table:



Edit Prompt Properties

Field Name:

*Heading Type:

*Type:

Heading Text:

*Format:

*Unique Prompt Name:

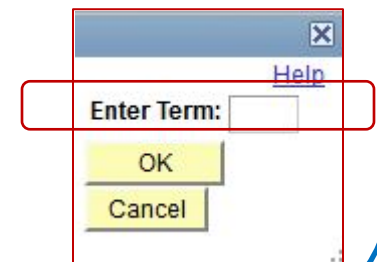
Length:

Decimals:

*Edit Type:

Prompt Table:

OK Cancel



Enter Term:

OK Cancel

Lesson 15: Prompts and Table Edits



Demonstration – Making a Change to the Prompt Table Edit Type (Edit Prompt Properties page)

1. In the **Introduction to Query Class Part II, Lesson 9: Adding a Runtime Prompts, Slide 66**, a prompt was created to ask the user to enter a two-letter state abbreviation. (The original query is named WES_TRAINING_STATE_ZZZ_PUBLIC.)
2. When initially observed, these fields on the **Edit Prompt Properties** page for the State appeared as follows:

*Edit Type:	Prompt Table	Prompt Table:	STATE_TBL
	▼	🔍	

3. The **Edit Type** was changed to **No Table Edit** and the **Prompt Table** was changed to **No Value**:

*Edit Type:	No Table Edit	Prompt Table:	
	▼	🔍	

4. The reason for this change is that the table listed under **Prompt Table** - which is **STATE_TBL** - does not render a list that the user can utilize.

5. This example points out one problem with the prompt system. It tries to guess at what you are doing and occasionally gets it wrong.
 - a. What it is trying to do is make sure that the state code you entered is valid by checking it against the items listed in STATE_TBL.
 - b. When you try to run a prompt based on that table, you receive the message “No matching values were found.”
6. You have two options to resolve the situation.
 - a. You can change the **Edit Type** to **No Table Edit** and change the **Prompt Table** to **No Value**.
 - b. Or you can retain the **Edit Type** and change the **Prompt Table** to another table. Either will work. The advantage of the latter option is that it retains the ability to search for valid values.
7. Note that most prompts you set up won't need this extra detail. You will be able to setup a prompt and go.



Demonstration – Find and Use Correct Prompt Table (Edit Prompt Properties page)

1. Click on the magnifying glass under the **Prompt Table** header. (As the cursor rolls over the magnifying glass you may see the words “Select a Prompt Table”).
2. The following page opens

Select a Prompt Table

Search by: Name begins with

Search Cancel No Value

3. In the text box (next to “begins with,” type the word **state** (upper or lower case). A list opens displaying 30 of 50 choices for a prompt table.
4. Click on View All to see the entire list of tables. The **Select a Prompt Table** list appears as follows.

Search by: Name begins with STATE

Search Cancel No Value

Search Results

Select a Prompt Table Personalize Find View 30 First 1-50 of 50 Last

STATE_NAME_SFVW
STATE_NAMES_VW - State Codes/Names (No Country)
STATE_NAMES_VW1 - State Names view - by Country
STATE_NMCAN_LNG - State Names - Country List Box
STATE_NMUSA_LNG - State Names - Country List Box
STATE_NMUS_LNG - State Names - Country List Box
STATE_NM_CAN_VW - State Canada View
STATE_NM_TX_VW - State Codes/Names (No Country)
STATE_NM_USA_VW - State USA View
STATE_NM_US_VW - State USA View without country
STATE_OTH - STATE_OTHER Codes/Names
STATE_SFVW_LNG - STATE_SFVW Codes/Names

5. For purposes of this exercise, we are only looking for addresses in the United States. Scroll through the list of names.
 - a. There are a few tables that appear to be exclusively related to the US. However, unless you know the best table, there is some trial and error involved.
 - b. You will need to select a likely table, and save the prompt on the **Edit Prompt Properties** page.
 - c. Select **STATE_NM_USA_VW**. When you do so, it appears under the **Prompt Table** header.
 - d. Change the **Edit Type** from **No Table Edit** to **Prompt Table**.

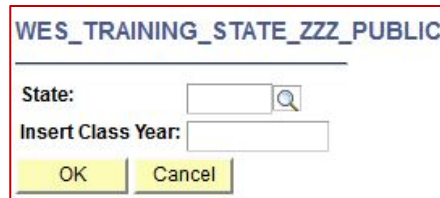
Lesson 15: Prompts and Table Edits

Demonstration – Find and Use Correct Prompt Table (continued)

e. The two fields now look like this:



- f. To test if this is the correct Prompt Table, you can begin to run the query up through the point of observing the list shown for the **State** when prompted.
- g. Note that once you observe the prompt output, you can cancel out without actually running the prompt.



- h. The *Look Up State* prompt output for **STATE_NM_USA_VW** - *State USA View* includes 93 items, many of which are not desired. So another table must be selected.



State	Country	Description
SE	USA	Invalid Code
SU	USA	Invalid Code
SUAS	USA	Invalid Code
SUGU	USA	Invalid Code
SUPR	USA	Invalid Code
SUVI	USA	Invalid Code
AA	USA	APO
AE	USA	APO (Europe)
AK	USA	Alaska
AL	USA	Alabama
AP	USA	APO (Pacific)

6. Return to the **Edit Prompt Properties** page, and click on the **Prompt Table** lookup button to observe the list of tables that begin with **STATE**.
- Within the **Select a Prompt Table** list, there is a table named **STATE_US_VW** - *US States - no country or fed*. This view also contains state names related to the United States but is shorter than the previous one.

Lesson 15: Prompts and Table Edits

Demonstration – Find and Use Correct Prompt Table (continued)

- b. Select **STATE_US_VW** so that it appears under the **Prompt Table** header. The final appearance of the **Edit Prompt Properties** page using this view is as follows:

Edit Prompt Properties

Field Name:

*Heading Type:

*Type:

Heading Text:

*Format:

*Unique Prompt Name:

Length:

Decimals:

*Edit Type:

Prompt Table:

- c. Click OK twice, and run the query.
- d. The *Look Up State* prompt list from **STATE_US_VW** has 66 rows.

Look Up State

Search by: begins with

[Advanced Lookup](#)

Search Results

View 100 First 1-66 of 66 Last

State	Description
AA	APO
AE	APO (Europe)
AK	Alaska
AL	Alabama
AP	APO (Pacific)
AR	Arkansas
AS	American Samoa
AZ	Arizona
CA	California
CO	Colorado
CT	Connecticut
DC	District of Columbia

- e. Choose *State* and *Class Year*, click OK. The output should resemble the following.

	State	ID	Name	Addr Type	City	Postal	City State Zip	Class Year
1	CA			HOME	Los Angeles	90004	Los Angeles, CA 90004	2017
2	CA			HOME	Los Angeles	90020	Los Angeles, CA 90020	2017
3	CA			HOME	Los Angeles	90020	Los Angeles, CA 90020	2017
4	CA			HOME	Los Angeles	90024	Los Angeles, CA 90024	2017
5	CA			HOME	Los Angeles	90025	Los Angeles, CA 90025	2017
6	CA			HOME	Los Angeles	90027	Los Angeles, CA 90027	2017
7	CA			HOME	Los Angeles	90027	Los Angeles, CA 90027	2017

- f. To preserve these changes, save the query under a new name. You can see a version in PS named WES_TRAINING_STATE_ZZZ_PUBPRM

Lesson 16: Counts and Sums in Query (Aggregate Functions and Having Criteria)



Overview

1. In this lesson, you will learn how to use predefined aggregate functions in a query and create Having Criteria for an aggregate field.
2. Instead of returning many rows of data, you may only be interested in a count of rows or a sum of a numeric field. You can produce these results using Query.

Aggregate Functions are created on the **Edit Field Properties** page.

Describing Aggregate Functions and Having Criteria

1. In Query Manager, you use:
 - a. **Aggregate functions** to associate query fields with predefined calculations
 - b. **Aggregate functions** to return a single value for multiple rows of output
 - c. The **Having** page to access fields that use aggregate functions in selection criteria

Using Aggregate Functions

1. You can use the aggregate function to group data and perform calculations on a field that is within the group.
2. For instance, instead of viewing all rows of data, you want to view only a count of rows; or you may want to have an average or sum by a given category.

Having Criteria

1. When you associate a field with an aggregate, you cannot use that field in selection criteria.
2. Structured Query Language (SQL) supports the use of aggregate functions in the WHERE clauses, but PeopleSoft applications don't.
3. Because the **Criteria** page corresponds to a SQL statement's WHERE clause, PeopleSoft Query provides the **Having** page.

Lesson 16: Counts and Sums in Query (Aggregate Functions and Having Criteria)

Having Criteria (continued)

4. The **Having** page enables you to add criteria on the aggregate instead of on the field generating the aggregate.
5. The **Having** page criteria appear in a SQL statement's HAVING clause.
6. If you experience problems with a query that is associated with aggregates, keep in mind that aggregate functions are not supported in the WHERE clause. The **Criteria** page creates the WHERE clause. Remember that you can use fields to join tables.

Using Predefined Aggregate Functions

1. When you apply an aggregate function to a field, PeopleSoft Query *replaces the field*, wherever it occurs, with the *results* of the function.
2. The table to the left lists the aggregate functions in Query Manager and their uses.

Uses of Aggregate Functions	
Aggregate Function	Use
Sum	Adds the values from each row and displays the total.
Count	Counts the number of rows.
Min (Minimum)	Checks the value from each row and returns the lowest one.
Max (Maximum)	Checks the value from each row and returns the highest one.
Average	Adds the values from each row

Lesson 16: Counts and Sums in Query (Aggregate Functions and Having Criteria)



Demonstration – Using Aggregate Functions

1. In this example, a query retrieves one row for every active Academic Plan. The total number of rows returned in this query is 254.

[View All](#) | [Rerun Query](#) | [Download to Excel](#) | [Download to XML](#) First 1-100 of 254 Last

	Acad Plan
1	ENVS
2	W-CSED
3	W-REES
4	MECO
5	REES
6	FRST-MN
7	FILM-MN
8	NS&B-MA
9	PRECOL

2. If the goal is simply to count the number of Acad Plans, a better way is to use the Count function, on the **Edit Field Properties** page, as this example illustrates.

- a. Under **Aggregate**, select the **Count** radio button.
- b. Change the radio button under **Heading** from **RFT Short** to **Text**.
- c. Change **Heading Text** from **Acad Plan** to **Count Acad Plan**.
- d. Click OK

Edit Field Properties

Field Name: A.ACAD_PLAN - Academic Plan

Heading	Aggregate
<input type="radio"/> No Heading	<input type="radio"/> None
<input checked="" type="radio"/> Text	<input type="radio"/> Sum
Heading Text: Count Acad Plan	<input checked="" type="radio"/> Count
	<input type="radio"/> Min
*Unique Field Name: A.ACAD_PLAN	<input type="radio"/> Max
	<input type="radio"/> Average
OK	Cancel

Lesson 16: Counts and Sums in Query (Aggregate Functions and Having Criteria)

Demonstration – Using Aggregate Functions (continued)

3. When you run the query, one row displays the number of plans that are in the record:



The screenshot shows a query result interface. At the top, there are links: 'View All', 'Rerun Query', 'Download to Excel', and 'Download to XML'. On the right, there are navigation controls: 'First', '1-1 of 1', and 'Last'. Below these is a table with one row. The first column contains the number '1'. The second column contains the text 'Count Acad Plan', which is circled in red. The third column contains the number '254', which is also circled in red.

	Count Acad Plan	
1		254



- When you aggregate a field, you cannot use Query to manipulate the individual field values.
- In the previous example, if you try to add criteria on the ACAD_PLAN field, you find that the ACAD_PLAN field no longer exists; there is only a count of the field.
- For this reason, standard criteria do not work on an aggregated field. You must use **Having** criteria.
- **Note that you cannot use the Sum or Average function with character fields.**

Lesson 16: Counts and Sums in Query (Aggregate Functions and Having Criteria)

Using Having Criteria in Queries

1. When you create a Row of Having Criteria, note that you can add the criteria from the **Fields** page, and then the system populates Expression 1 of the Having criteria. You do not have to access the **Having** page or the **Edit Having Criteria Properties** page. Those steps are optional.
2. The **Edit Having Criteria Properties** page is identical to the **Edit Criteria Properties** page, *except* that when you select the field for Expression 1, Query Manager lists only the fields that are associated with an aggregate function on the **Select a field** page. Those pages are shown in the following example.

Demonstration – Select a Field and Edit Having Criteria Properties Pages

1. In this example, based on a different table, two fields are selected, EMPLID and COUNTRY.
2. First, apply the Count aggregate function.
 - a. Select the **Fields** page, and then click on the Edit button for the EMPLID field.
 - b. Select the **Count** option in the **Aggregate** group box, and then click the OK button.

Edit Field Properties

Field Name: A.EMPLID - Empl ID

Heading	Aggregate
<input type="radio"/> No Heading	<input type="radio"/> None
<input checked="" type="radio"/> Text	<input type="radio"/> Sum
Heading Text: Count ID	<input checked="" type="radio"/> Count
	<input type="radio"/> Min
	<input type="radio"/> Max
	<input type="radio"/> Average
*Unique Field Name: A.EMPLID	
OK	Cancel

Lesson 16: Counts and Sums in Query (Aggregate Functions and Having Criteria)

Demonstration – Select a Field and Edit Having Criteria Properties Pages (continued)

- c. If you run the query at this time, the output would appear as follows showing how many EMPLIDs there are for each Country. These are the first few entries.

	Count ID	Country
1		2 AFG
2		1 ALB
3		1 ANT
4		3 ARE
5		2 ARG
6		23 AUS
7		4 AUT
8		11 BEL

5. To add rows of **Having** criteria:
- Select the **Fields** page, and then click on the Add Criteria button (funnel) for the EMPLID field.
 - The **Edit Having Criteria Properties** page opens.
 - At this point you select a **Condition Type** as you would with regular criteria.

Edit Having Criteria Properties

Choose Expression 1 Type

☒ Field
☐ Expression

Expression 1

Choose Record and Field

Record Alias.Fieldname:
A.EMPLID - Empl ID

*Condition Type: equal to

Choose Expression 2 Type

☐ Field
☐ Expression
☒ Constant
☐ Prompt
☐ Subquery

Expression 2

Define Constant

Constant:

OK Cancel

Lesson 16: Counts and Sums in Query (Aggregate Functions and Having Criteria)

Demonstration – Select a Field and Edit Having Criteria Properties Pages (continued)

- d. Select the **Condition Type** of *less than* and type the **Expression 2 Constant** of **5**. Click the OK button and Save.

Note: If you apply the criteria from the **Fields** page *after* an aggregate is applied (as described here), the system creates the criteria as **Having** criteria. This approach eliminates the need to define Expression 1 for the criteria.

Edit Having Criteria Properties

Choose Expression 1 Type

☒ Field
☐ Expression

Expression 1

Choose Record and Field

Record Alias.Fieldname:

A.EMPLID - Empl ID

*Condition Type: less than

Choose Expression 2 Type

☐ Field
☐ Expression
☒ Constant
☐ Prompt
☐ Subquery

Expression 2

Define Constant

Constant: 5

OK Cancel

- e. When the query is run now, only the Countries with a count of EMPLIDs *fewer than 5* will be displayed:

	Count ID	Country
1		2 AFG
2		1 ALB
3		1 ANT
4		3 ARE
5		2 ARG
6		4 AUT
7		3 BFA
8		2 BHS
9		1 BMU
10		2 BOL
11		2 BRB
12		1 CIV
13		1 COL

Lesson 17: Mastering Outer Joins



NOTE: Under **Join Type**, when you select **Left outer join**, you must join to the most recently added record in the query, or you will receive an error message. If you are creating a query with more than two records, be sure to add the “left” record just before the “right” record.

Join Type	
<input type="radio"/>	Join to filter and get additional fields (Standard Join)
<input checked="" type="radio"/>	Join to get additional fields only (Left outer join)

Overview

1. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition.
2. An outer join forces a row from one of the participating tables to appear in the result if no matching row exists. In an outer join, all rows of data are included from the master table (the first record added to the query). Matching rows from the subordinate table are also included.
3. In a *left outer join*, all rows of the first (left) record are present in the result set, even if there are no matches in the joining record. If there isn't a match in the second table, a blank or NULL is returned for fields that are pulled from that table.
4. Outer joins combine aspects of record-hierarchy joins and subqueries (subqueries are discussed later).
 - a. Remember that a record-hierarchy join retrieves rows for which fields match from different tables, for example, A.Field1 equals B.Field1.
 - b. A subquery may retrieve rows that *don't exist* in a secondary table.
5. As you will see in the first demonstration, the **Edit Criteria Properties** page provides a drop down list box in which you can select the criteria that belongs to the *On clause* for outer joins or the *WHERE clause* for other join types.

Lesson 17: Mastering Outer Joins

(These examples of working with outer joins are based on data from the formerly used Admission Module.)



Demonstration - Outer Join selecting criteria that belongs to On Clause

- This example shows you one way in which an outer join can be used.
- The initial query displays the *ID* of each Prospect's Last School Attended (LAST_SCH_ATTEND).
- If you wanted to see the *name* of the Last School Attended, you could join the EXT_ORG_TBL. *
- However, if it were joined with a Standard Join, those rows that did not have an entry under LAST_SCH_ATTEND would not be included in the query results.
- In order to see those rows, we need to do an Outer Join.
- **Note:** This is an example of joining fields that do not have the same name.

** In real life, relying on the EXT_ORG_TBL is unnecessary. There is a Wesleyan Function that can display the name of an external organization - WES_GET_ORG_NAME(A.EXT_ORG_ID)*

1. A query is created from the table WES_PROAPP containing the following fields:

EMPLID - Empl ID
ADMIT_TERM - Admit Term
LAST_SCH_ATTEND - Last School Attended
APPL_ON_FILE - Application On File

2. The criteria are:

Criteria		Personalize	
Logical	Expression1	Condition Type	Expression 2
	AADMIT_TERM - Admit Term	equal to	1139
AND	AAPPL_ON_FILE - Application On File	equal to	N

3. To join the EXT_ORG_TBL as an outer join, the Join Type page will appear as follows:

Join Type

☐ Join to filter and get additional fields (Standard Join)

☒ Join to get additional fields only (Left outer join)

Join Record

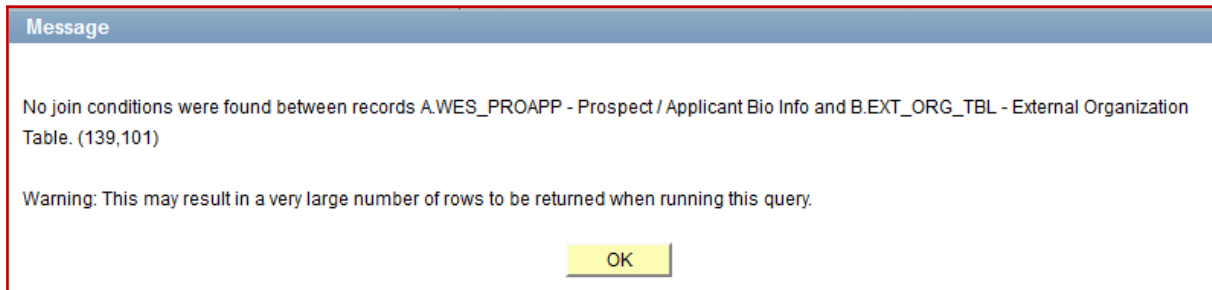
Personalize | Find |

A = WES_PROAPP - Prospect / Applicant Bio Info

Lesson 17: Mastering Outer Joins

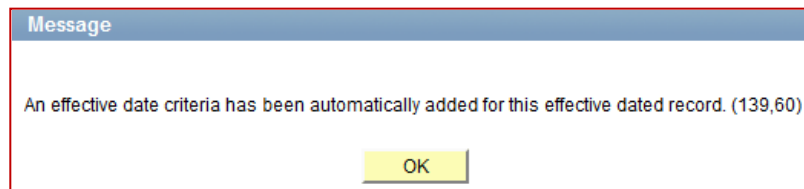
Demonstration - Outer Join selecting criteria that belongs to On Clause (continued)

- When we click the Join Record of “Join to get additional fields only (Left outer join)” we get this warning. This is because the fields we will be joining have different names:



Pay very close attention to this warning! When you see it, either make a join, or remove the table.

- In this case we know that A.LAST_SCH_ATTEND is really the same as B.EXT_ORG_ID. We will be creating criteria to that effect and make it part of the outer join clause. So click OK. When the next popup appears, regarding the Effective Date, click OK again.



- The standard effective date criterion is automatically added.
- Select the field **DESCR – Description** from EXT_ORG_TBL for display in the query.

Lesson 17: Mastering Outer Joins

Demonstration - Outer Join selecting criteria that belongs to On Clause (continued)

8. We now add criteria with a Condition Type of **A.LAST_SCH_ATTEND** equal to **B.EXT_ORG_ID**.
9. We make the criteria part of the outer join clause by selecting under “This criteria belongs to” the drop down item “ON clause of outer join B.” *
10. Once added, the **Edit Criteria** Properties page should look as follows:

Edit Criteria Properties

Choose Expression 1 Type
☒ Field
☐ Expression

Expression 1
Choose Record and Field
Record Alias.FieldName:
A.LAST_SCH_ATTEND - Last School

*Condition Type: equal to

Choose Expression 2 Type
☒ Field
☐ Expression
☐ Constant
☐ Prompt
☐ Subquery

Expression 2
Choose Record and Field
Record Alias.FieldName:
B.EXT_ORG_ID - External Org ID

This criteria belongs to
ON clause of outer join B

OK Cancel



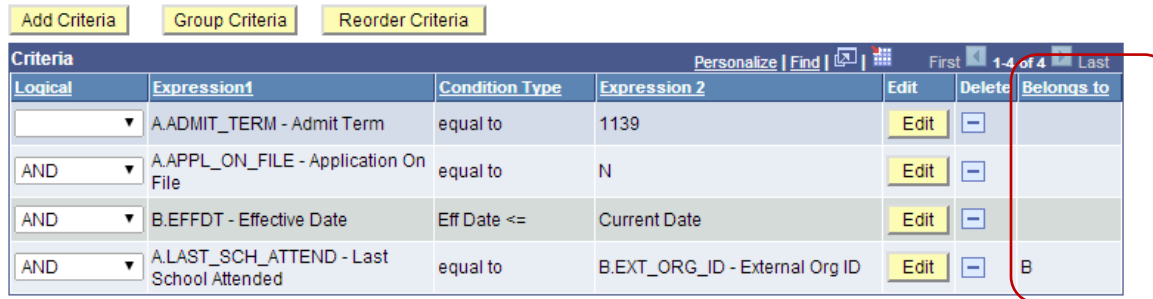
*** Note:** This is an important factor to consider when adding criteria to a query that has an outer join. You must make sure that the criteria “belong” to the correct *clause* of the query. This determines whether the criteria is applied **before** the join is made (*limiting the rows that could be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

See more information on the next slide.

Lesson 17: Mastering Outer Joins

Demonstration - Outer Join selecting criteria that belongs to On Clause (continued)

11. After clicking OK, observe the **Criteria** page:



Logical	Expression1	Condition Type	Expression 2	Edit	Delete	Belongs to
	A.ADMIT_TERM - Admit Term	equal to	1139	Edit		
AND	A.APPL_ON_FILE - Application On File	equal to	N	Edit		
AND	B.EFFDT - Effective Date	Eff Date <=	Current Date	Edit		
AND	A.LAST_SCH_ATTEND - Last School Attended	equal to	B.EXT_ORG_ID - External Org ID	Edit		B

12. Since there is an outer join in the query, the **Belongs to** column is displayed.

- As referenced above, this indicates whether a criterion belongs to the main query (the value is blank), so that it is applied *before* the join,
- or it indicates if the criterion belongs to a specific record (the value is an alias letter – B in this case), so that it is applied *after* the join.

13. 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 (WES_PROAPP)*.

- The two criteria we added before the outer join – ADMIT_TERM and APPL_ON_FILE – belong to the WHERE clause because Admit Term and Application on File are not in EXT_ORG_TBL and have no effect on whether there is a matching row in EXT_ORG_TBL.

Lesson 17: Mastering Outer Joins

Demonstration - Outer Join selecting criteria that belongs to On Clause (continued)

14. Effective Date: The third row on the **Criteria** page – **B.EFFDT <= Current Date** – introduces another element that needs to be addressed before the query is complete.

- If the query were run at this point, it would only return rows where the LAST_SCH_ATTEND is actually populated. That's because the EXT_ORG_TBL is effective dated. We have to add criteria that accounts for **null effective dates** for that field.
- This involves reordering and grouping the rows in **Criteria** that are related to the Effective Date and External Org ID fields.

New Position	Position	Expression1	Condition Type	Expression 2	Belongs to
1		A.ADMIT_TERM - Admit Term	equal to	1139	
2		A.APPL_ON_FILE - Application On File	equal to	N	
3		B.EFFDT - Effective Date	Eff Date <=	Current Date	
4		A.LAST_SCH_ATTEND - Last School Attended	equal to	B.EXT_ORG_ID - External Org ID	B

- On the **Criteria** page, click on the yellow **Reorder Criteria** button to open the **Edit Criteria Ordering** page.
- On the **Edit Criteria Ordering** page move the third row (B.EFFDT) by typing a “3” under the “New Position” column in the fourth row.
- When you click OK, the **Criteria** page reopens and rows 3 and 4 have changed places.

Logical	Expression1	Condition Type	Expression 2	Edit	Delete	Belongs to
	A.ADMIT_TERM - Admit Term	equal to	1139	Edit	-	
AND	A.APPL_ON_FILE - Application On File	equal to	N	Edit	-	
AND	A.LAST_SCH_ATTEND - Last School Attended	equal to	B.EXT_ORG_ID - External Org ID	Edit	-	B
AND	B.EFFDT - Effective Date	Eff Date <=	Current Date	Edit	-	

Lesson 17: Mastering Outer Joins

Demonstration - Outer Join selecting criteria that belongs to On Clause (continued)

- f. Add a new criteria row from EXT_ORG_TBL of EXT_ORG_ID and make the Condition Type "is null."
- g. On the **Criteria** page:
 - i. Change the **AND** on the EXT_ORG_ID row to **OR**.
 - ii. Click on the yellow **Group Criteria** button.
 - iii. On the **Edit Criteria Grouping** page, put opening and closing parentheses at the beginning and end of the criteria related to the Effective Date and External Org fields.

Edit Criteria Properties

Choose Expression 1 Type

☒ Field
☐ Expression

Expression 1

Choose Record and Field

Record Alias.Fieldname:

B.EXT_ORG_ID - External Org ID

***Condition Type:** is null

This criteria belongs to

WHERE clause

OK Cancel


Use the edit boxes to enter parenthesis for each criteria. Use only the '(' and ')' characters.

Edit Criteria Grouping						
Personalize Find First 1-5 of 5 Last						
Logical	Left Paren	Expression1	Condition Type	Expression 2	Right Paren	Belongs to
		A.ADMIT_TERM - Admit Term	equal to	1139		
AND		A.APPL_ON_FILE - Application On File	equal to	N		
AND		A.LAST_SCH_ATTEND - Last School Attended	equal to	B.EXT_ORG_ID - External Org ID		B
AND	(B.EFFDT - Effective Date	Eff Date <=	Current Date		
OR		B.EXT_ORG_ID - External Org ID	is null)	

Lesson 17: Mastering Outer Joins

Demonstration - Outer Join selecting criteria that belongs to On Clause (continued)

15. After clicking OK, the **Criteria** page should now look like this:

Criteria						
		Personalize Find 		First 1-5 of 5 Last		
Logical	Expression1	Condition Type	Expression 2	Edit	Delete	Belongs to
	A.ADMIT_TERM - Admit Term	equal to	1139	Edit	-	
AND	A.APPL_ON_FILE - Application On File	equal to	N	Edit	-	
AND	A.LAST_SCH_ATTEND - Last School Attended	equal to	B.EXT_ORG_ID - External Org ID	Edit	-	B
AND	(B.EFFDT - Effective Date	Eff Date <=	Current Date	Edit	-	
OR	B.EXT_ORG_ID - External Org ID	is null)	Edit	-	

16. And when the query is run, a typical portion of the output will resemble this. The Prospects not associated with a last school each have rows, but have a blank column for Lst School and Descr.

ID	Admit Term	Lst School	Applied	Descr
1139	117200	N		Northwest Guilford Senior High
1139		N		
1139	116384	N		Cardinal Spellman High School
1139	104077	N		Leigh High School
1139	114976	N		John F Kennedy Memorial High S
1139		N		
1139	631087	N		Southridge High School
1139		N		
1139	106358	N		Youth Development Center
1139		N		
1139	124777	N		Lakeside School
1139	111171	N		Northampton High School
1139	105975	N		Greater Atlanta Chr School Arl
1139		N		
1139		N		
1139	927917	N		Summit International Prep

Lesson 17: Mastering Outer Joins

You may on occasion come across an existing query that incorporates a field or fields with the notation of (+) attached. This is another method for creating an outer join which can be used for the rare occasion when you cannot join with the most recently added record in the query.



Demonstration - Outer Join using (+)

1. This demonstration uses the same tables as the first Demonstration, but identifies the join in a different way.
2. To create an outer join with this methodology, you must include a plus sign in parentheses (+) after the key fields of the subordinate record in the criteria that link the records.
3. You will perform an Any Record join and code an expression that contains the (+) instead of a field. The following example uses Define Expression on the Edit Criteria Properties page.
4. This query begins with the same table and fields and criteria as the previous query.

Criteria			Personalize
Logical	Expression1	Condition Type	Expression 2
	A.ADMIT_TERM - Admit Term	equal to	1139
AND	A.APPL_ON_FILE - Application On File	equal to	N

5. To perform an outer join using this approach, when adding the EXT_ORG_TBL select the first option under Join Type (Standard Join).

Select join type and then record to join with EXT_ORG_TBL - External Organization Table.

Join Type
<input checked="" type="radio"/> Join to filter and get additional fields (Standard Join)
<input type="radio"/> Join to get additional fields only (Left outer join)

Join Record	Personalize	Find	First	1 of 1	Last
A = WES_PROAPP - Prospect / Applicant Bio Info					
Cancel					

Lesson 17: Mastering Outer Joins

Demonstration - Outer Join using (+) (continued)

6. The warning message regarding no join conditions being found will appear.
7. The standard effective date criterion is automatically added.
8. The join criteria is modified as follows.
 - a. On the **Criteria** page, add A.LAST_SCH_ATTEND.
 - b. On the **Edit Criteria Properties** page, in the Condition Type operator field, accept the value of **equal to**.
 - c. In the Choose Expression 2 Type field, accept the value of Expression.
 - d. Under Define Expression, click **Add Field**. Select the required field from the second record, i.e. B.EXT_ORG_ID.
 - e. Manually enter **(+)** after the field name in the Expression Box on the **Edit Criteria Properties** page and click OK.

Edit Criteria Properties

Choose Expression 1 Type: ☒ Field ☐ Expression

Expression 1

Choose Record and Field

Record Alias.FieldName: A.LAST_SCH_ATTEND - Last School

*Condition Type: equal to

Choose Expression 2 Type: ☐ Field ☒ Expression ☐ Constant ☐ Prompt ☐ Subquery

Expression 2

Define Expression

Expression: B.EXT_ORG_ID(+)

Add Prompt Add Field

OK Cancel

Choose Expression 1 Type: ☒ Field ☐ Expression

Expression 1

Choose Record and Field

Record Alias.FieldName: A.LAST_SCH_ATTEND - Last School

*Condition Type: equal to

Choose Expression 2 Type: ☐ Field ☒ Expression ☐ Constant ☐ Prompt ☐ Subquery

Expression 2

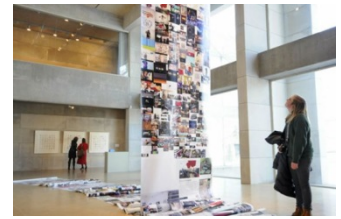
Define Expression

Expression: B.EXT_ORG_ID(+)

Add Prompt Add Field

- f. Follow the instructions from the previous demonstration starting at Step 14 to account for the Effective Date.
- g. Run the query.

Lesson 18: Subqueries - Introduction



Overview

1. A subquery is a query whose results are used within a query.
2. By creating nested queries, the primary query displays the fields necessary for the results and the *subordinate query serves as criteria* that filter the data.
3. Subqueries compare the value for a field in the primary query to the results of a subordinate query. The subordinate query is imbedded in the WHERE clause using the **Criteria** page.
4. The **Condition Type** that you specify in the criteria determines what the subquery returns to the query.
5. A subquery can retrieve only one data field from a single table, and the subquery can contain a join. You can use this feature to specify criteria based on two records.
6. You never see the result of the subquery itself; you see the results of the primary query. The subquery is part of the criteria that limits the return.
7. You can place additional rows of criteria in the primary query or the subquery.



One way to look at a subquery -

- Sometimes, before you can answer a question, you need to answer one or more other questions.
- You could run one query to give you results to use in another query.
- However, rather than creating two queries, you can use a subquery to collect the information. *The result of the subquery can be used like a field or expression in the main query criteria.*

Lesson 18: Subqueries - Introduction

There are two general ways to use a subquery:

Test for existence or non-existence

In this situation you have a subquery that selects rows based on a set of criteria. Then, in your Parent query, you have one criterion that specifies to ***select a row that exists in that subquery result set or does not exist in the subquery result set.*** Examples:

Parent Query	Operator	Subquery
Selects all prospects where...	...a record for that prospect exists in the subquery, which...	...selects all prospects who have submitted test scores.
Selects all prospects where...	...a record for that prospect exists in the subquery, which...	...selects all prospects who have a certain Academic Interest.
Selects all applicants where...	...a record for that applicant does not exist in the subquery, which...	...selects Immunization data for the applicant.

Compare a field to a list

~~In this situation you have a subquery that again selects rows based on a set of criteria. In this case,~~ however, the subquery returns a list of values for one field. Then, in the parent query, you have one criterion that ***compares a field or expression to the subquery to determine if there is (or is not) a matching value in the subquery list.*** Examples:

Parent Query	Operator	Subquery
Selects all prospects where EMPLID...	... is in the list of EMPLIDs returned by the subquery, which...	...selects the EMPLID of all prospects who have submitted test scores.
Selects all prospects where EMPLID...	... is in the list of EMPLIDs returned by the subquery, which...	...selects the EMPLID of all prospects who have a certain Academic Interest.
Selects all applicants where EMPLID...	... is not in the list of EMPLIDs returned by the subquery, which...	...selects the EMPLID of all applicants with Immunization data.

Lesson 18: Subqueries - Introduction

- The following demonstration was adapted from a [presentation on subqueries](#) given at a regional conference by a representative from Northern Illinois University.
- That was based on the STDNT_CAR_TERM table. This example uses the WES_STUDENT table.
- The query displays how to incorporate the same table more than once in the query.
- It also shows the use of a minimum value for the criteria.

Demonstration – Using a subquery to identify first term of enrollment

1. The scenario is that we want to see a list of all *currently enrolled* Undergraduate and Graduate students with their *first term*. *For this example, the current term is 1141.*
2. The query is created using two instances of WES_STUDENT.
 - a. The first instance of WES_STUDENT is used to identify the current term of enrollment (1141).
 - b. The second instance of the table is used to get term data for *all terms* for students from first to most recent, i.e. the current term.
3. From the **Records** page, select WES_STUDENT. On the **Query** page select these fields:
 - i. EMPLID
 - ii. ACAD_CAREER
 - iii. STRM
 - iv. FORM_OF_STUDY



Lesson 18: Subqueries - Introduction

Demonstration – Using a subquery to identify first term of enrollment

4. On the **Criteria** page, set the first three criteria to see all Undergrads and Grads enrolled in Term 1141.

Criteria				Personalize	Find		
Logical	Expression1	Condition Type	Expression 2				
	A.STRM - Term	equal to	1141				
AND	A.ACAD_CAREER - Academic Career	in list	('UGRD','GRAD')				
AND	A.FORM_OF_STUDY - Form of Study	equal to	ENRL				

5. Join the second instance of WES_STUDENT to the first:

- a. On the **Join Type** page, select Standard Join. Click to join the record.
- b. On the **Auto Join Criteria** page, uncheck A.STRM so that the second instance of the table (B) will look at all Terms.
- c. Click Add Criteria.

Auto Join Criteria

Query has detected the join conditions shown below.

Use the checkboxes to unselect the criteria that you do not want to add to the query and click add criteria when done. The criteria added can always be modified later using the criteria tab.

<input checked="" type="checkbox"/>	A.EMPLID - Empl ID = B.EMPLID - Empl ID
<input type="checkbox"/>	A.STRM - Term = B.STRM - Term

Add Criteria

Cancel

6. On the **Query** page, select STRM for the just added B table.
7. Add two more criteria to the **Criteria** page to link fields between A and B – ACAD_CAREER and STDNT_CAR_NBR:

Criteria				Personalize	Find			First	1-6 of 6	Last
Logical	Expression1	Condition Type	Expression 2							
	A.STRM - Term	equal to	1141					Edit		
AND	A.ACAD_CAREER - Academic Career	in list	('UGRD','GRAD')					Edit		
AND	A.FORM_OF_STUDY - Form of Study	equal to	ENRL					Edit		
AND	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.STDNT_CAR_NBR - Student Career Nbr	equal to	B.STDNT_CAR_NBR - Student Career Nbr					Edit		

Lesson 18: Subqueries - Introduction

Demonstration – Using a subquery to identify first term of enrollment (continued)

8. Run the query. There are multiple rows for each student who meets the criteria, i.e. a row for each term the student has been enrolled. For example, the first eight rows represent one student's terms.
9. To limit the output to only the **first term** for each student, you can create a subquery on STRM from the second instance of WES_STUDENT.
10. Click *Add Criteria* for B.STRM. On the **Edit Criteria Properties** page:
 - a. Leave the Condition Type as “equal to.”
 - b. Under Choose Expression 2 Type, select “Subquery.”
 - c. Click on the link *Define/Edit Subquery*.

	ID	Career	Term	Study Form
1	8	UGRD	1141	ENRL
2	8	UGRD	1141	ENRL
3	8	UGRD	1141	ENRL
4	8	UGRD	1141	ENRL
5	8	UGRD	1141	ENRL
6	8	UGRD	1141	ENRL
7	8	UGRD	1141	ENRL
8	8	UGRD	1141	ENRL
9	9	UGRD	1141	ENRL
10	9	UGRD	1141	ENRL
11	9	UGRD	1141	ENRL
12	9	UGRD	1141	ENRL
13	9	UGRD	1141	ENRL
14	9	UGRD	1141	ENRL
15	9	UGRD	1141	ENRL
16	9	UGRD	1141	ENRL
17	9	UGRD	1141	ENRL
18	9	UGRD	1141	ENRL
19	9	UGRD	1141	ENRL
20	9	UGRD	1141	ENRL
21	9	UGRD	1141	ENRL
22	9	UGRD	1141	ENRL
23	7	UGRD	1141	ENRL
24	7	UGRD	1141	ENRL
25	7	UGRD	1141	ENRL
26	7	UGRD	1141	ENRL

Edit Criteria Properties

Choose Expression 1 Type

☒ Field
☐ Expression

Expression 1

Choose Record and Field

Record Alias.Fieldname:
B.STRM - Term

*Condition Type: equal to ▼

Choose Expression 2 Type

☐ Field
☐ Expression
☐ Constant
☐ Prompt
☒ Subquery

Expression 2

Define Subquery

[Define/Edit Subquery](#)

Lesson 18: Subqueries - Introduction

Demonstration – Using a subquery to identify first term of enrollment (continued)

10. The following page opens. Do a search for the table which is to be used for the subquery. In this example it is the same table as used in the main query, i.e. WES_STUDENT.

The screenshot shows a query editor interface. At the top, there are two tabs: 'Query Name' and 'Description'. The 'Query Name' tab is active, showing 'Working on selection' and 'Subquery for B.STRM - Term'. To the right of the 'Description' tab is a 'Feed' icon and a dropdown menu. Below the tabs, there is a search section with a '*Search By' dropdown set to 'Record Name', a 'begins with' label, and an empty text input field. There are two buttons: 'Search' and 'Advanced Search'. At the bottom, there is a row of buttons: 'Save', 'Save As', 'New Query', 'Preferences', 'Properties', 'Publish as Feed', 'New Union', and 'Return To Search'. A link 'Subquery/Union Navigation' is also visible.

11. You'll see that in some ways the page resembles the **Records** page. However, it also has:
- The additional identifier of **Working on selection Subquery for B.STRM – Term** and
 - The link **Subquery/Union Navigation** to enable toggling between the subquery and the main query.

This screenshot shows the same query editor interface as the previous one, but with the search results displayed. The search criteria are 'Record Name' and 'begins with WES_STUDENT'. The search results are shown in a table with the following data:

Record	Personalize	Find	View All	First	1-2 of 2	Last
Recname						
WES_STUDENT - Wesleyan Student Status						
WES_STUDENT_ADV - WES Student Advisor Table						

Each row in the results table has 'Add Record' and 'Show Fields' links. The 'Subquery/Union Navigation' link is also present at the top right of the interface.

Lesson 18: Subqueries - Introduction

Demonstration – Using a subquery to identify first term of enrollment (continued)

12. When you click on Add Record, the subquery **Query** page opens.
 - a. It resembles a regular **Query** page, but it has the two elements described above, and
 - b. Next to each field is a Select link.
13. Click on the Select link next to STRM. The subquery **Fields** page opens.

Working on selection Subquery for B.STRM - Term [Subquery/Union Navigation](#)

Add additional records by clicking the records tab. When finished select a single field for this subquery and you will be transferred to the fields tab.

Chosen Records

Alias Record [Hierarchy Join](#)

C WES_STUDENT - Wesleyan Student Status

Fields	Find	View All	First	1-50 of 78	Last
Select EMPLID - Empl ID					
Select NAME - Name					
Select FIRST_NAME - First Name					
Select MIDDLE_NAME - Middle Name					
Select LAST_NAME - Last Name					
Select COUNTRY - Country					
Select WES_CNTRY_TRANS - Wes Country Trans					
Select ADDRESS1 - Address Line 1					
Select ADDRESS2 - Address Line 2					
Select ADDRESS3 - Address Line 3					
Select ADDRESS4 - Address Line 4					
Select CITY - City					
Select STATE - State					
Select POSTAL - Postal Code					
Select WES_HOME_PHONE - Home Phone					
Select WESPO - Wesleyan PO Box					
Select PHONE - Telephone					
Select BIRTHDATE - Date of Birth					

Working on selection Subquery for B.STRM - Term [Subquery/Union Navigation](#)

View field properties, or use field as criteria in query statement. [Reorder / Sort](#)

Col	Record.Fieldname	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	C.STRM - Term	Char4				Term		Edit	Delete

[Save](#) [Save As](#) [New Query](#) [Preferences](#) [Properties](#) [Publish as Feed](#) [New Union](#) [Return To Search](#)

14. Click on the Edit button to open the **Edit Field Properties** page.

Lesson 18: Subqueries - Introduction

Demonstration – Using a subquery to identify first term of enrollment (continued)

15. Under Aggregate, select Min, i.e. you are looking for the first Term in the subquery for each student.

Edit Field Properties

Field Name: C.STRM - Term

Heading

- ☐ No Heading
 ☒ RFT Short
 ☐ Text
 ☐ RFT Long

Heading Text:

Term

*Unique Field Name:

C.STRM

OK

Cancel

Fields									
Col	Record.FieldName	Format	Ord	XLAT	Agg	Heading Text	Add Criteria	Edit	Delete
1	C.STRM - Term	Char4			Min	Min Term		Edit	

16. Open the **Criteria** page and add the following:

- Join the subquery instance of WES_STUDENT (C) to the second top-level instance (B) of WES_STUDENT. The three fields to be joined are EMPLID, ACAD_CAREER and STDNT_CAR_NBR.
- You will recall that in the top-level instance of WES_STUDENT, the FORM_OF_STUDY was equal to ENRL. The same field needs to be selected in the subquery.
- The **Criteria** page under subquery should now have these rows:

Criteria					
Logical	Expression1	Condition Type	Expression 2	Edit	Delete
	C.EMPLID - Empl ID	equal to	B.EMPLID - Empl ID	Edit	
AND	C.ACAD_CAREER - Academic Career	equal to	B.ACAD_CAREER - Academic Career	Edit	
AND	C.STDNT_CAR_NBR - Student Career Nbr	equal to	B.STDNT_CAR_NBR - Student Career Nbr	Edit	
AND	C.FORM_OF_STUDY - Form of Study	equal to	ENRL	Edit	

Lesson 18: Subqueries - Introduction

Demonstration – Using a subquery to identify first term of enrollment (continued)

17. Click on the *Subquery/Union Navigation* link.
18. And then click on the *Top Level of Query* link.
19. Run the query. You'll see that the number of rows has been significantly decreased so that only one row appears for each student. The last column in each row is the first term for that student.

[Subquery/Union Navigation](#)

Select subquery or union to navigate to

Left | Right

 [Top Level of Query](#)

 [Subquery for B.STRM - Term](#)

As mentioned in the section on improving query performance, when querying for multiple values, a subquery is usually the slowest in that the subqueries have to be evaluated first, and must be compiled before the parent query runs.

	ID	Term	Career	Study Form	Term
1	8	1141	UGRD	ENRL	1129
2	9	1141	UGRD	ENRL	1129
3	9	1141	UGRD	ENRL	1129
4	7	1141	UGRD	ENRL	1129
5	5	1141	UGRD	ENRL	1129
6	8	1141	UGRD	ENRL	1129
7	6	1141	UGRD	ENRL	1129
8	7	1141	UGRD	ENRL	1129
9	6	1141	UGRD	ENRL	1129
10	4	1141	UGRD	ENRL	1129
11	2	1141	UGRD	ENRL	1139
12	4	1141	UGRD	ENRL	1129
13	4	1141	UGRD	ENRL	1129
14	3	1141	GRAD	ENRL	1121
15	7	1141	UGRD	ENRL	1129
16	5	1141	UGRD	ENRL	1129
17	2	1141	UGRD	ENRL	1129
18	6	1141	UGRD	ENRL	1129
19	8	1141	UGRD	ENRL	1129
20	5	1141	UGRD	ENRL	1139
21	4	1141	UGRD	ENRL	1129
22	3	1141	UGRD	ENRL	1129
23	4	1141	UGRD	ENRL	1129



Lesson 19: Creating Union Queries



Overview

1. 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.
2. 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.
3. Unions can be used to combine records that have no fields in common and to retrieve similar data from unrelated records in one query. And this can be done without creating a Cartesian product.

A Union is two (or more) separate select statements (queries)* that are brought together in the same query.

There are three rules you must follow when using a Union:

1. Each query must consist of the same number of fields/columns.
2. Each query must consist of the same data types
3. The field data types in each query should correspond, i.e. be in the same order.

For example, if the third column in the first query is a date, the third column in the second query must be a date as well.

* The term “statement” or “select statement” is sometimes used to designate the queries.

Lesson 19: Creating Union Queries

3. Note: You do not have to use fields of like data type in one statement (query). That is, you can mix field data types within each statement as long as the data types correspond between the two statements.
4. Keep in mind the following points when using Unions:
 - a. The sorting and headings are established in the first select statement
 - b. The table with the largest field sizes must be chosen as the top level of the query.
 - c. You cannot retrieve the long or short translate description in a Union
 - d. Unions are automatically Distinct

Using Literals in Unions

1. To execute properly, the three rules listed earlier must be followed. The two queries must have the same number of fields, in the same order, like to like (field type and length). The field type must be exact and length similar.
2. In order to have this necessary equivalence, use literal expressions as placeholders or as pieces of text.
3. *When a Union is created, it is required that each SELECT statement have the same number of fields (the fields don't have to be identical).* This is where the practice of applying literals as placeholders comes in handy.
4. Literals are created on the **Expressions** Page.
5. Examples of literals:
 - a. Character: two single quotes with a space between – ‘ ’
 - b. Number: zero - 0
 - c. A word between two single quotes – ‘Complete’
 - d. Integer between two single quotes – ‘5’

Lesson 19: Creating Union Queries



Steps for Using Literals in Unions

1. Click the Add Expression button on the **Expressions** page
 - a. Select Character as the expression
 - b. Enter 1 (or whatever is appropriate) as the length
 - c. Enter two single quotes with a space (' ') as the expression text
 - d. Click the OK button.
 - e. Note: Enclose the literal value between the single quotes. Change the length according to the text requirements.
2. Click the *Use as Field* link to use the expression in the query.
3. The name of the new field is ' ' (two single quotes).

Edit Expression Properties

*Expression Type
Character Length

☐ Aggregate Function Decimals

Expression Text



The basic steps for creating a union query are:

1. Create a query, including selecting the fields and criteria
2. Click the *New Union* hyperlink at the bottom of the Query Manager pages
3. Select the record to use in the union
4. Select the same number of fields and the same field types for each field as in the original query and arrange in the same order
5. Save and Preview the query

Lesson 19: Creating Union Queries



Demonstration – Creating Unions That Use Literals

In this demonstration the output will be a listing of students for a given career and term. Displayed will be their:

- Academic Plan
 - Form of Study
 - Primary Academic Program
 - Academic Org Owner associated with the Academic Plan
- This example uses two queries (statements) that have these three records in common. They are added to the query in this order:
 - STDNT_CAR_TERM
 - ACAD_PROG
 - ACAD_PLAN
- However, the second query (Union 1) has one additional record that the first query (Top Level of Query) does not:
 - ACAD_PLAN_OWNER
- This is because the first query includes the ACAD_PROG_PRIMARY field of GRNON (Graduate Non-Degree) as criteria, which has no Academic Plan Owner associated with it.

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals

1. The **first query** is created from the three records listed above; six fields are selected. Note that the columns need to be re-positioned exactly as shown in order the union query to work properly.

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	C.ACAD_PLAN - Academic Plan	Char10				Acad Plan		Edit			
2	A.EMPLID - Empl ID	Char11				ID		Edit			
3	A.FORM_OF_STUDY - Form of Study	Char4		N		Study Form		Edit			
4	A.STRM - Term	Char4				Term		Edit			
5	A.ACAD_PROG_PRIMARY - Primary Academic Program	Char5				Prim Prog		Edit			
6	B.PROG_STATUS - Academic Program Status	Char4		N		Status		Edit			

2. The criteria in the **first query** should appear as follows; most of the criteria were automatically set, except for the last three rows which were manually created.

➤ *Note that this is the query under which the criterion is set for the ACAD_PROG_PRIMARY field to equal GRNON.*

Criteria						Personalize	Find	View All	First	1-12 of 12	Last
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.STDNT_CAR_NBR - Student Career Nbr	equal to	A.STDNT_CAR_NBR - Student Career Nbr	Edit							
AND	B.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	Edit							
AND	A.EMPLID - Empl ID	equal to	C.EMPLID - Empl ID	Edit							
AND	A.ACAD_CAREER - Academic Career	equal to	C.ACAD_CAREER - Academic Career	Edit							
AND	C.STDNT_CAR_NBR - Student Career Nbr	equal to	A.STDNT_CAR_NBR - Student Career Nbr	Edit							
AND	C.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	Edit							
AND	A.ACAD_CAREER - Academic Career	equal to	GRAD	Edit							
AND	A.STRM - Term	equal to	1139	Edit							
AND	A.ACAD_PROG_PRIMARY - Primary Academic Program	equal to	GRNON	Edit							

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals (continued)

3. This is a sample of output from the **first query** at this point. *Make a note of how many records are returned for verification after the union query is completed.*

	Acad Plan	ID	Study Form	Term	Prim Prog	Status
1	OTHR-GR		ENRL	1139	GRNON	AC
2	OTHR-GR		ENRL	1139	GRNON	AC
3	VINT-GR		ENRL	1139	GRNON	DC
4	VINT-GR		ENRL	1139	GRNON	DC
5	VINT-GR		ENRL	1139	GRNON	DC
6	VINT-GR		ENRL	1139	GRNON	DC

4. Now it is time to create the union. At the bottom of any page (except the Run page), click on the **New Union** link which opens the following page. Note the **Subquery/Union Navigation** link in the upper right that enables you to toggle between the **Top Level of Query** and **Union 1**.

The screenshot shows a query builder interface with several tabs at the top: Records, Query, Expressions, Prompts, Fields, Criteria, Having, View SQL, and Run. The 'Query' tab is selected. Below the tabs, there are fields for 'Query Name' and 'Description'. A red box highlights the 'Working on selection' tab, which is currently selected, and the 'Union 1' label next to it. To the right of this, there is a 'Feed' icon and a link labeled 'Subquery/Union Navigation'. Below these, there is a search section with a '* Search By' dropdown menu set to 'Record Name', followed by the text 'begins with' and an empty text input field. At the bottom of the search section, there is a yellow 'Search' button and a link for 'Advanced Search'.

(Tip: Once you're satisfied with the query output, it's a good idea to save the query at this point and often during creation.)

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals (continued)

5. The three records (STDNT_CAR_TERM, ACAD_PROG and ACAD_PLAN) that will be the same as the **first query (Top Level of Query)** need to be added to the **second query (Union 1)**. And the way the records are joined to the **second query** needs to mimic the creation of the **first query**.
 - a. Add the record STDNT_CAR_TERM
 - b. Once you click the record name, the **Query** page for the **Union 1** opens displaying an Alias of D for STDNT_CAR_TERM

Working on selection Union 1 [Subquery/Union Navigation](#)

Click folder next to record to show fields. Check fields to add to query. Uncheck fields to remove from query. Add additional records by clicking the records tab. When finished click the fields tab.

Chosen Records

Alias	Record
D	STDNT_CAR_TERM - Student Career Term Table

[Hierarchy Join](#) [-](#)

[Check All](#) [Uncheck All](#)

Fields [Find](#) [View 50](#) First 1-100 of 124 Last

Field	Join
<input type="checkbox"/> EMPLID - Empl ID	Join PEOPLE_SRCH - People Search View
<input type="checkbox"/> ACAD_CAREER - Academic Career	Join STDNT_CAREER - Student Career
<input type="checkbox"/> INSTITUTION - Academic Institution	Join INSTITUTION_TBL - Institution Table
<input type="checkbox"/> STRM - Term	Join TERM_TBL - Term Definition Table
<input type="checkbox"/> REG_CARD_DATE - Registration Card Date	
<input type="checkbox"/> WITHDRAW_CODE - Withdrawal \ Cancel	
<input type="checkbox"/> WITHDRAW_REASON - Withdrawal \ Cancel Reason	
<input type="checkbox"/> WITHDRAW_DATE - Withdrawal \ Cancel Date	
<input type="checkbox"/> LAST_DATE_ATTENDED - Last Date of Attendance	
<input type="checkbox"/> STDNT_CAR_NBR - Student Career Nbr	
<input type="checkbox"/> ACAD_PROG_PRIMARY - Primary Academic Program	
<input type="checkbox"/> ACAD_LOAD_APPR - Approved Academic Load	

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals (continued)

- c. To **Union 1** add the other two records, ACAD_PROG and ACAD_PLAN, using the same joins as in the **Top Level of Query**. They will have aliases of E and F. Arrange the fields in **Union 1** so the columns are the same as in the **Top Level of Query**.
- d. Reorder by EMPLID.

Col	Record.Fieldname	Format	Ord	KLAT	Agg	Heading Text	Add Criteria
1	F.ACAD_PLAN - Academic Plan	Char10				Acad Plan	
2	D.EMPLID - Empl ID	Char11	1			ID	
3	D.FORM_OF_STUDY - Form of Study	Char4		N		Study Form	
4	D.STRM - Term	Char4				Term	
5	D.ACAD_PROG_PRIMARY - Primary Academic Program	Char5				Prim Prog	
6	E.PROG_STATUS - Academic Program Status	Char4		N		Status	

6. Now add the record that is not in the **first query**, ACAD_PLAN_OWNER.
 - a. Join it on the Auto Join page to the field ACAD_PLAN in the ACAD_PLAN record.
 - b. On the Query page, select the field ACAD_ORG.
 - c. The Query page for **Union 1** will now have these records:

Working on selection Union 1
[Subquery/Union](#)

Click folder next to record to show fields. Check fields to add to query. Uncheck fields to remove from query. Add additional records by clicking the records tab. When finished click the fields tab.

Alias	Record	
	D STDNT_CAR_TERM - Student Career Term Table	Hierarchy Join
	E ACAD_PROG - Student Academic Program	Hierarchy Join
	F ACAD_PLAN - Student Academic Plan Table	Hierarchy Join
	G ACAD_PLAN_OWNER - Academic Plan Owner Table	Hierarchy Join

Lesson 19: Creating Union Queries



You will note a number of differences on the **Query** page when you are working with union queries, some of which are similar to those with subqueries.

- On the upper left, the **Working on selection** field displays information text to help you keep track of whether you are working on the **Top Level of Query** (first query) or **Union 1** (second query).
- As you work with the union query, you can switch between the two queries.
 - To navigate between the top level of the query and the Union 1 query, click on the *Subquery/Union Navigation* link in the upper right.
 - Note: The Subquery/Union Navigation link appears on all pages except the Run page.

Demonstration – Creating Unions That Use Literals (continued)

7. To continue with the union query, both queries need to be set up with the same number of fields containing comparable data. The seven field header names for the final output are:
 - a. Acad Org
 - b. Acad Plan
 - c. ID
 - d. Term
 - e. Prim Prog
 - f. Study Form
 - g. Status

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals (continued)

8. These fields are identical to those in the *first query*, except that the new field of Acad Org (ACAD_ORG) has been added. In the *second query*, re-position the columns so that the **Fields** page now looks like this, in this exact order, i.e. ACAD_ORG is now the first column:

Fields					Personalize Find View A	
Col	Record.FieldName	Format	Ord	XLAT	Agg	Heading Text
1	G.ACAD_ORG - Academic Organization	Char10				Acad Org
2	F.ACAD_PLAN - Academic Plan	Char10				Acad Plan
3	D.EMPLID - Empl ID	Char11	1			ID
4	D.FORM_OF_STUDY - Form of Study	Char4		N		Study Form
5	D.STRM - Term	Char4				Term
6	D.ACAD_PROG_PRIMARY - Primary Academic Program	Char5				Prim Prog
7	E.PROG_STATUS - Academic Program Status	Char4		N		Status

Criteria				Personalize Find First
Logical	Expression1	Condition Type	Expression 2	
	D.EMPLID - Empl ID	equal to	E.EMPLID - Empl ID	
AND	D.ACAD_CAREER - Academic Career	equal to	E.ACAD_CAREER - Academic Career	
AND	D.INSTITUTION - Academic Institution	equal to	E.INSTITUTION - Academic Institution	
AND	E.STDNT_CAR_NBR - Student Career Nbr	equal to	D.STDNT_CAR_NBR - Student Career Nbr	
AND	E.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	
AND	D.EMPLID - Empl ID	equal to	F.EMPLID - Empl ID	
AND	D.ACAD_CAREER - Academic Career	equal to	F.ACAD_CAREER - Academic Career	
AND	F.STDNT_CAR_NBR - Student Career Nbr	equal to	D.STDNT_CAR_NBR - Student Career Nbr	
AND	F.EFFDT - Effective Date	Eff Date <=	Current Date (EffSeq = Last)	
AND	F.ACAD_PLAN - Academic Plan	equal to	G.ACAD_PLAN - Academic Plan	
AND	G.EFFDT - Effective Date	Eff Date <=	Current Date	
AND	D.ACAD_CAREER - Academic Career	equal to	GRAD	
AND	D.STRM - Term	equal to	1139	

9. Add the following two criteria from STDNT_CAR_TERM (Record Alias D), being sure they are the same as those in the first query:

a. ACAD_CAREER equal to GRAD

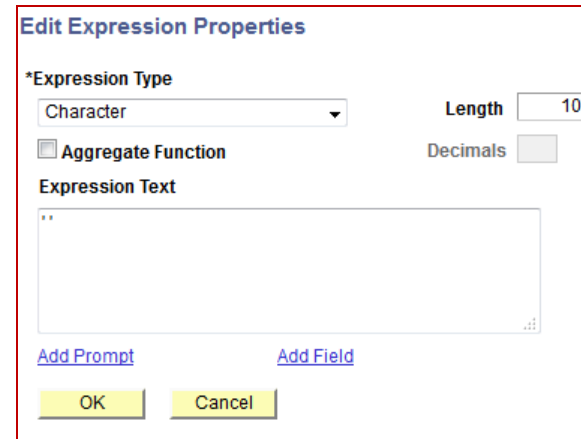
b. STRM equal to 1139

10. The *second query* Criteria page should now appear like the image to the left.

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals (continued)

11. Navigate back to the *first query (Top Level of Query)*.
 - a. If you look at the **Fields** page, you'll see that one field needs to be added to make both queries have the same output. That field is ACAD_ORG.
 - b. Since the *first query* does not include the record containing ACAD_ORG, this is where a literal placeholder is used.
 - c. As shown in the section above entitled **Steps for Using Literals in Unions**, create an expression with a literal placeholder.
 - d. Make it 10 characters long.
 - e. Be sure to click on *Use as Field* link.



Edit Expression Properties

*Expression Type
Character Length 10

☐ Aggregate Function Decimals

Expression Text
''

[Add Prompt](#) [Add Field](#)

OK Cancel

12. The newly created expression is now a field on the **Fields** page with a name of ' ' (two single quotes with a space).
 - a. Rearrange the columns so that they are in the same order as the *second query* with the new field as the first column (see below).
 - b. Change the Heading Text for the new field to **Acad Org**.
 - c. Sort by EMPLID.

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals (continued)

- c. The **first query's** Fields page should now look like this:

Working on selection Top Level of Query [Subquery/Union Navigation](#)

View field properties, or use field as criteria in query statement. [Reorder / Sort](#)

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	'	Char10				Acad Org		Edit	-
2	C.ACAD_PLAN - Academic Plan	Char10				Acad Plan		Edit	-
3	A.EMPLID - Empl ID	Char11	1			ID		Edit	-
4	A.FORM_OF_STUDY - Form of Study	Char4		N		Study Form		Edit	-
5	A.STRM - Term	Char4				Term		Edit	-
6	A.ACAD_PROG_PRIMARY - Primary Academic Program	Char5				Prim Prog		Edit	-
7	B.PROG_STATUS - Academic Program Status	Char4		N		Status		Edit	-

- d. Contrast with the **second query's** Fields page. Compare Row 1.

Working on selection Union 1 [Subquery/Union Navigation](#)

View field properties, or use field as criteria in query statement. [Reorder / Sort](#)

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	G.ACAD_ORG - Academic Organization	Char10				Acad Org		Edit	-
2	F.ACAD_PLAN - Academic Plan	Char10				Acad Plan		Edit	-
3	D.EMPLID - Empl ID	Char11	1			ID		Edit	-
4	D.FORM_OF_STUDY - Form of Study	Char4		N		Study Form		Edit	-
5	D.STRM - Term	Char4				Term		Edit	-
6	D.ACAD_PROG_PRIMARY - Primary Academic Program	Char5				Prim Prog		Edit	-
7	E.PROG_STATUS - Academic Program Status	Char4		N		Status		Edit	-

Lesson 19: Creating Union Queries

Demonstration – Creating Unions That Use Literals (continued)

13. Save and run the query. Below is a sample of the output for the fields through the Prim Prog (Primary Academic Program) or ACAD_PROG_PRIMARY field. You will see that those rows with the Primary Academic Program of GRNON have a blank field under Acad Org.

Acad Org	Acad Plan	ID	Study Form	Term	Prim Prog	Status
MUSC	ETHN-PHD		ABD	1139	PHD	AC
MUSC	ETHN-PHD		ENRL	1139	PHD	AC
	OTHR-GR		ENRL	1139	GRNON	AC
MB&B	MB&B-PHD		ENRL	1139	PHD	AC
MB&B	MB&B-PHD		ENRL	1139	PHD	AC
	OTHR-GR		ENRL	1139	GRNON	AC
PHYS	PHYS-PHD		ENRL	1139	PHD	AC
MB&B	MB&B-PHD		ENRL	1139	PHD	AC
	VINT-GR		ENRL	1139	GRNON	DC
	VINT-GR		ENRL	1139	GRNON	DC
CHEM	CHEM-PHD		ABD	1139	PHD	AC
MUSC	ETHN-PHD		ABD	1139	PHD	AC
MATH	MATH-PHD		ABD	1139	PHD	AC
	VINT-GR		ENRL	1139	GRNON	DC

As noted in the section on improving query performance, when querying for multiple values, the UNION operator is one possible method. As mentioned, it is not recommended because of increased coding complexity

Lesson 20: Supplementary Information



There are documents and links in the SFIS Blog that relate to the above topics and other areas related to PS Query and PeopleSoft in general. Here is a sampling.

PeopleSoft Query Training:

- [PS Query I and II – Introduction](#)
- [Part I – Basic Query Concepts and Query Viewer](#)
- [Part II – Query Manager](#)
- [Grouping Criteria and Boolean Expressions](#)
- [Annotated Visual Demonstration of Query Process](#)
- [Sample Relational Database](#)

Query Presentations:

- [Criteria and Having Pages](#)
- [Dates in Query](#)
- [Effective Dates in Query](#)
- [Expressions and Wesleyan Functions](#)
- [Prompts in Query](#)

Query Tools and References

- [Criteria Properties](#)
- [Query Manager Pages](#)
- [Query to Excel \(Pivot Table Intro\)](#)
- [Records Commonly Used](#)
- [Tip Sheet](#)
- [Encyclopedia](#)
- [Glossary](#)

