4.1
3.10

eMARS Reporting

FORMULAS & VARIABLES OVERVIEW

NOVEMBER 3, 2017

Formulas & Variables

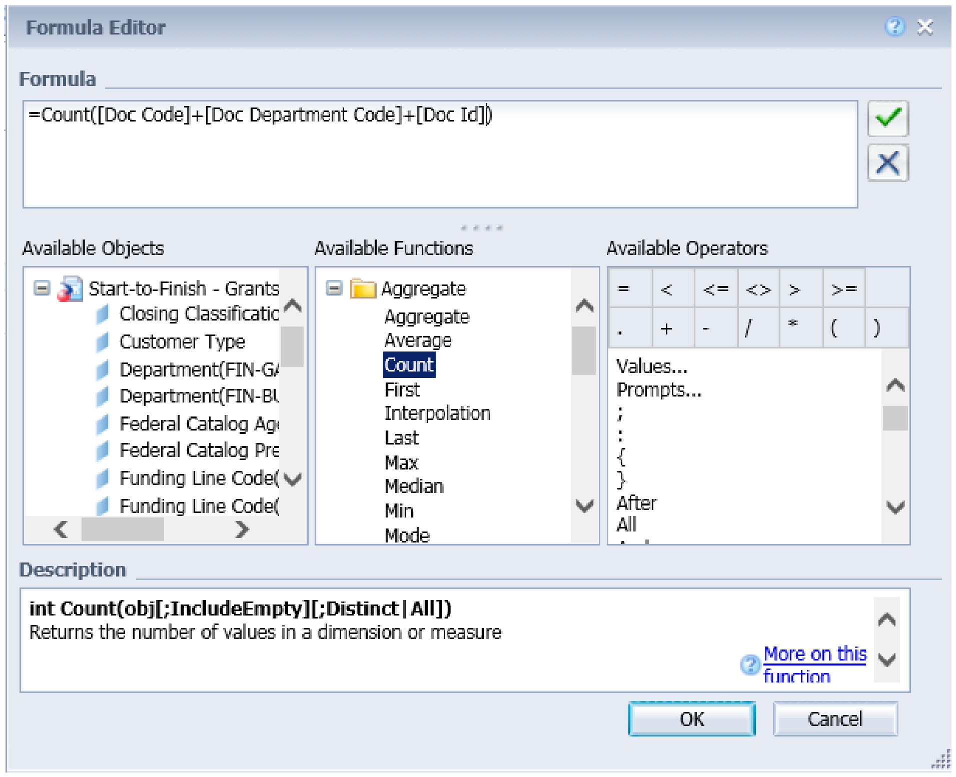
Each time you put an object into your report, you create a *cell* (or a block of cells, which is a *table*) containing *formulas*. Formulas are simply equations which tell the system how to display the contents of the cell. You can also create *variables* to store formulas in an easily accessible way, rather than editing them directly in the cells in your report.

Formulas

As mentioned, *formulas* are equations. Formulas are comprised of:

- **"=" –** Every formula begins the same way – with an equals sign ("="). This tells the system that the cell contains a formula, rather than only text to display.
- **Equation** – As mentioned, there will always be an equation after the equals sign. It could be as simple as the name of an object, a string of text, or a constant. It could also be very complex, including *functions* or even other variables.
- **Objects** – Most equations will include at least one object. Objects can also be concatenated or otherwise manipulated using *functions* or *operators*.
- **Functions** – Many equations include *functions* to manipulate the data contained in objects. There are a wide range of functions pre-defined in WebIntelligence to perform a variety of actions.
- **Operators** – Conditional formulas frequently use operators to relate the results of functions to other objects or function results.

Here are the components of a formula as displayed in the Formula Editor window:



```
=Count([Doc Code]+[Doc Department Code]+[Doc Id])
```

int Count(obj[;IncludeEmpty][;Distinct|All])
Returns the number of values in a dimension or measure

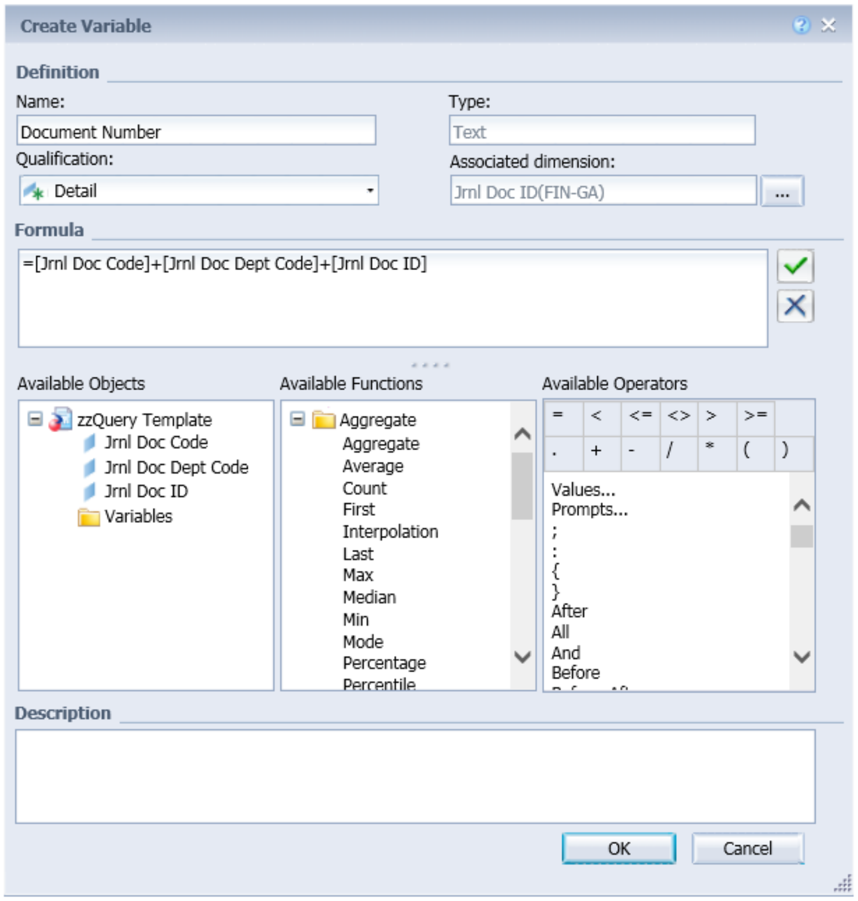
This formula concatenates the three objects making up a Document Number, and displays a count showing how many Document Numbers appear in the report:

=Count([Doc Code]+[Doc Dept Code]+[Doc Id])

Of course, you see the equals sign at the beginning. Next is the Count() function (with a description appearing in the Description section of the window, toward the bottom). Below the formula section are lists of the available Objects, Functions, and Operators that can be used to build the formula, including buttons for the most common Operators.

Variables

As mentioned, *variables* provide a means of storing formulas in an easily accessible way. Variables in your report are always shown at the bottom of the list of Available Objects (in the left panel). Although variables are defined using a separate Create Variable window, it is very similar to the Formula Editor:



Name: Document Number
Type: Text
Qualification: Detail
Associated dimension: Jrnl Doc ID(FIN-GA)

```
=[Jrnl Doc Code]+[Jrnl Doc Dept Code]+[Jrnl Doc ID]
```

Available Objects: zzQuery Template — Jrnl Doc Code, Jrnl Doc Dept Code, Jrnl Doc ID; Variables

As you can see, there are a few additional components to the definition of a *variable*:

- **Name –** Provides the name of the variable, as displayed in the list of Available Objects and in cell formulas. It is recommended that you adopt naming conventions for your variables – see the *Naming Conventions* section for details.
- **Qualification –** Specifies whether the variable should be treated as a dimension, a detail, or a measure object. The differences between these are described later in this section.
- **Type –** Indicates whether the variable will be treated as text or as a number (for aggregation purposes).
- **Associated Dimension** – When Qualification is "Detail", indicates the dimension object for which the variable will be treated as a detail. It should be an object from the query holding the desired values, and should be unique to each record (or as close to that as possible). See "Details from Dimensions" later in this section for more information.

Once a variable is defined, you can use it just like any other object in your report. Variables are preferable over entering formulas directly into cells, as you will see.

Benefits of Variables over Formulas

As mentioned, it is preferable to define variables for your formulas rather than entering them directly into cells. There are several reasons for this:

- **Easier to locate formulas for update –** When formulas are entered directly into the cells in the report, you would have to click on each cell that appears to have similar data in order to locate it and update it. For instance, suppose you had a total formula entered in a cell as “=Sum([Jrnl Posting Amt]+[Intercept Amt])” and you wanted to update the formula to “=Sum([Jrnl Posting Amt])+Sum([Intercept Amt])”, but you couldn’t remember which amount cells in your report contain the formula. If a variable was used, you could easily locate and update the variable.
- **A single definition for use in multiple cells –** If you enter your formulas directly into cells, you have to re-enter the formula for each cell where it is needed. Then if you need to update the formula, you not only have the difficulty of locating those cells, but you also have to re-enter the changes in each cell containing the formula. It is therefore very easy to overlook a cell when making formula updates. With a variable, you edit the variable and all cells containing it are automatically updated.
- **Easy column naming (and renaming) –** Suppose you had a column in your report called “Fixed Asset Amount” but you later learned that “Vehicle Value” would be a better name. If the column contained a formula, you would update the column header in a report tab, but if the same formula was in another report tab, it would still have the old column name. When you use a variable, you can take advantage of WebIntelligence automatically naming the column using the name of the variable. Then, if you update the variable’s name, it automatically updates all of the column headings for cells containing the variable.
- **Fewer accidental deletions –** It is easy to accidentally delete a variable without realizing that it will affect results in your report. Using variables consistently can reduce this possibility, in that WebIntelligence will not permit you to delete a variable if there are other variables which depend upon it.
- **Fewer accidental formula changes –** It is also easy to remove an object from a query, not realizing that it will affect results in your report. For example, suppose you removed [Closing Classification] from your query because it did not appear in your report. Where formulas were directly entered in the cells, the amounts will change – which you may notice, but you may not. Meanwhile, when formulas have been placed in variables, the columns containing the variables display “#ERROR” (since the object on which the formulas depend is no longer available). Additionally, the missing object in the variable formula is replaced with a placeholder (such as [??DP4.DO60536??]). You can then add the object back (and the formula is automatically corrected) or edit the formula to use a different object. (Formulas entered directly into cells do not indicate missing objects using placeholders; the objects are simply removed from the formula.)

Prompt Variables

It is recommended that the very first thing you do when creating a new report is to create variables for each of the prompts in your query. The recommended naming convention for prompt variables is to prefix them with “Prompt-“. So, for instance, if your query has a prompt for Department, you would define a new variable named “Prompt-Dept” as follows:

=UserResponse(“Enter Department:”)

Tip: The UserResponse() function requires that the text for the prompt *exactly* match what is entered for the Query Filter. Any difference (including a missing space at the end) will result in the variable not displaying the prompt value.

If your report has more than one query, your prompt variable definition may need to specify the query name. For example:

=UserResponse([FIN-GA];”Enter Department:”)

Formatting Prompt Variable Values

The resulting value of the UserResponse() function is always text. Sometimes you need the user's response to be interpreted as a number – for example, so that "10" sorts after "9" instead of between "1" and "2". Text is converted to a number as follows:

=ToNumber(UserResponse([FIN-GA];"Enter Department:");"###")

Note that in the ToNumber() function, you must specify the value to be converted as well as the format in which the user would have entered the number. "###" means the user most likely entered three digits.

Now, if you want the numerical value displayed in a particular way, there are a couple of ways to accomplish that: you can use Number Formatting within your report (just like any measure object), or you can specify the format in the variable definition, as follows:

=FormatNumber(ToNumber(UserResponse([FIN-GA];"Enter FY:");"####");"####")

The interpretation of this formula (in English) is something like: "The user will enter a FY as 4 characters, but in the report we want to treat that value as a number with 4 digits (no commas)." The first "####" tells the ToNumber() function how the user is entering the prompt value, while the second "####" tells the FormatNumber() function how to display the value.

Prompt Variables for "InList" Query Filters

Note: When defining prompt variables for Query Filters using the InList operator, define the "Prompt-" variable without converting the value from text. Then define a other variables to pull the values and convert them. For instance:

Fiscal Year InList "Enter FY(s):" (prompt; required; keep values)

		Result for Variable in Report
Prompt-FY	**= UserResponse("Enter FY(s):")**	2017;2014;2015;2016
Text-First FY	**= Left([Prompt-FY];4)**	2017
Text-Last FY	**= Right([Prompt-FY];4)**	2016
Text-Second FY	**= Substr([Prompt-FY];6;4)**	2014
Text-Third FY	**= Substr([Prompt-FY];11;4)**	2015
Num-Last FY	**= ToNumber([Text-Last FY];"####")**	2,016
FNum-Last FY	**= FormatNumber([Num-Last FY];"####")**	2016

Common Prompt Variable Definitions

The most commonly used prompt variables are as follows:

```
Prompt-Cab      = UserResponse("Enter Cabinet:")
Prompt-Dept     = UserResponse("Enter Department:")
Prompt-FY       = FormatNumber(ToNumber(UserResponse("Enter FY:");"####");"####")
Prompt-APD      = ToNumber(UserResponse("Enter Accounting Period:");"##")
Prompt-Date     = FormatDate(ToDate(
                        UserResponse("Enter Record Date:");
                        "MM/dd/yyyy hh:mm:ss A");"M/dd/yyyy")


Hdr-Page Numbers        = Page()+" of "+NumberOfPages()
Hdr-Run Date            = FormatDate(LastExecutionDate();"MM/dd/yyyy")
Hdr-Run Time            = FormatDate(LastExecutionDate();"hh:mm:ss A")
Hdr-Execution           = FormatNumber(LastExecutionDuration();"###0.00")+" sec"
```

Measure Variables

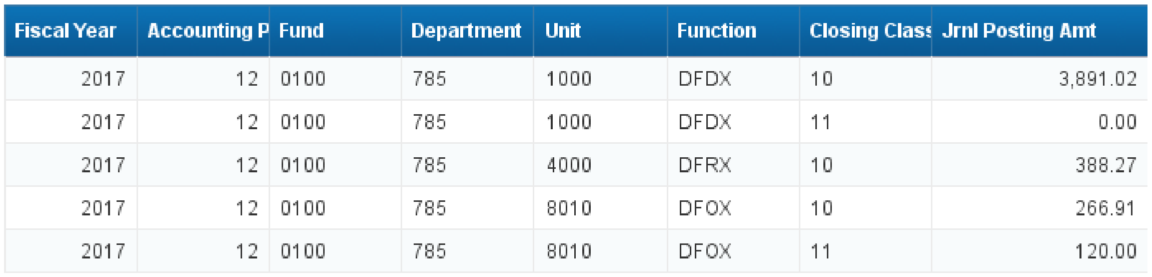
As mentioned, there are three types of objects: dimensions, details, and measures. *Measure objects* represent amounts, can be manipulated as numbers. For example, they can be summed or multiplied, and they can be formatted as numbers using FormatNum().

Measure variables are simply variables defined as measures (by setting Qualification="Measure" in the variable definition); that is, they are expected to contain numbers, and can also be manipulated and formatted as numbers.

Flag and Amount Variables

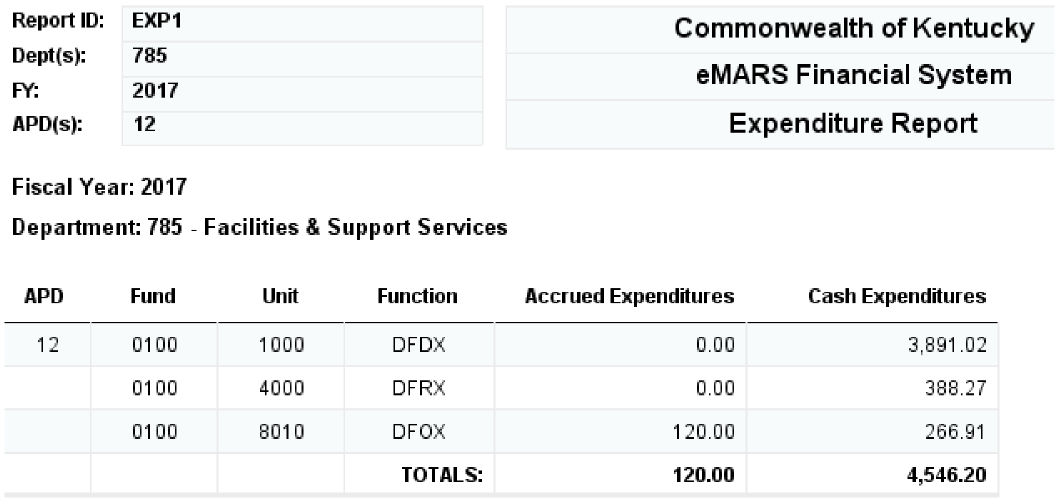
Although not a requirement of WebIntelligence, Report Developers in the Commonwealth have been instructed to create "flag variables" and "amount variables" when they need to include Posting Amounts in reports. For example, a report containing encumbrances and expenditures would have two flag variables (i.e., an Encumbrance Flag and an Expenditure Flag) and two amount variables (i.e., an Encumbrance Amount and an Expenditure Amount). This section provides an explanation for this practice.

Suppose you have a query against the Accounting Journal that returns the following records:



Fiscal Year	Accounting P	Fund	Department	Unit	Function	Closing Class	Jrnl Posting Amt
2017	12	0100	785	1000	DFDX	10	3,891.02
2017	12	0100	785	1000	DFDX	11	0.00
2017	12	0100	785	4000	DFRX	10	388.27
2017	12	0100	785	8010	DFOX	10	266.91
2017	12	0100	785	8010	DFOX	11	120.00

As you can see, there are records for both Accrued Expenditures ([Closing Classification]=11) and Cash Expenditures ([Closing Classification]=10). Now, for your report, you would like to see one column showing the total for each amount:



Report ID:	EXP1
Dept(s):	785
FY:	2017
APD(s):	12

Commonwealth of Kentucky
eMARS Financial System
Expenditure Report

Fiscal Year: 2017

Department: 785 - Facilities & Support Services

APD	Fund	Unit	Function	Accrued Expenditures	Cash Expenditures
12	0100	1000	DFDX	0.00	3,891.02
	0100	4000	DFRX	0.00	388.27
	0100	8010	DFOX	120.00	266.91
			TOTALS:	**120.00**	**4,546.20**

Here are the flag and amount variables you would define for this report, if you choose to take that approach:

Flag-Acc Exp	**= If([Closing Classification]=11;"Yes";"No")**
Amt-Acc Exp with Flag	**= [Jrnl Posting Amt] Where ([Dim-Acc Exp Flag]="Yes")**
Flag-Cash Exp	**= If([Closing Classification]=10;"Yes";"No")**
Amt-Cash Exp with Flag	**= [Jrnl Posting Amt] Where ([Dim-Cash Exp Flag]="Yes")**

Comparison with Other Approaches

Let’s compare those results to other approaches. For example, we could define the flags as Booleans:

Bool-Is Acc Exp =([Closing Classification]=11)
Amt-Acc Exp with Bool =[Jrnl Posting Amt] Where ([Bool-Is Acc Exp])

Bool-Is Cash Exp =([Closing Classification]=10)
Amt-Cash Exp with Bool =[Jrnl Posting Amt] Where ([Bool-Is Cash Exp])

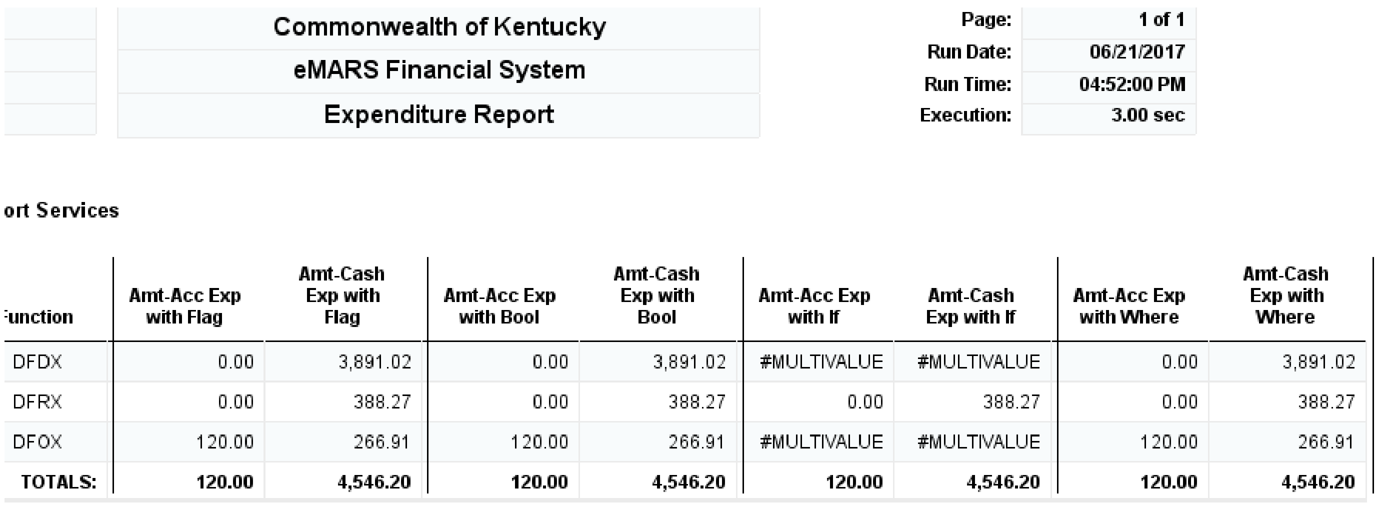
Or, we could try bypassing flag variables entirely, here using “If”:

Amt-Acc Exp with If = If([Closing Classfication]=11;[Jrnl Posting Amt];0)
Amt-Cash Exp with If = If([Closing Classfication]=10;[Jrnl Posting Amt];0)

And, finally, using “Where”:

Amt-Acc Exp with Where =[Jrnl Posting Amt] Where ([Closing Classification]=11)
Amt-Cash Exp with Where=[Jrnl Posting Amt] Where ([Closing Classification]=10)

Here are the results of all four approaches:



Commonwealth of Kentucky	Page: 1 of 1
eMARS Financial System	Run Date: 06/21/2017
Expenditure Report	Run Time: 04:52:00 PM
	Execution: 3.00 sec

ort Services

unction	Amt-Acc Exp with Flag	Amt-Cash Exp with Flag	Amt-Acc Exp with Bool	Amt-Cash Exp with Bool	Amt-Acc Exp with If	Amt-Cash Exp with If	Amt-Acc Exp with Where	Amt-Cash Exp with Where
DFDX	0.00	3,891.02	0.00	3,891.02	#MULTIVALUE	#MULTIVALUE	0.00	3,891.02
DFRX	0.00	388.27	0.00	388.27	0.00	388.27	0.00	388.27
DFOX	120.00	266.91	120.00	266.91	#MULTIVALUE	#MULTIVALUE	120.00	266.91
TOTALS:	**120.00**	**4,546.20**	**120.00**	**4,546.20**	**120.00**	**4,546.20**	**120.00**	**4,546.20**

As you can see, for this report the approaches yield the same results, except the “If” approach which yields some #MULTIVALUE results. **Tip:** #MULTIVALUE can sometimes be corrected by putting a Sum() function around the formula. Generally speaking, you can’t go wrong using the flag variable approach – other approaches require more confirmation.

Filter Variables and “Matches Pattern”

There are limitations on Report Filters. For example, you can use the “Matches Pattern” operator in a Query Filter, but that option isn’t available in the list for Report Filters. One way around such limitations is to use Filter Variables. **Note:** This is not a WebIntelligence feature – it is just a practice which simplifies filtering in reports.

A Filter Variable is nothing more than a variable defined with a formula for a Boolean value – that is, a formula that evaluates to TRUE (=1) or FALSE (=0). For example, suppose we need vendor addresses containing “Street”. In a Query Filter using “Matches Pattern” and the string, “%Street%” works. But how can the same thing be done within a report?

Here is the definition of a Filter Variable – once defined, you would apply a filter for this variable on the report for all values equal to “1” (=TRUE):

Bool-Contains Street = Match([Address Line 1];”*Street*”)

This approach can be used for any Boolean formula; simply filter for values equal to “1” for the Filter Variable.

Dimension and Detail Objects and Variables

As discussed, there are three types of objects: dimensions, details, and measures. *Measure objects* have been discussed; they hold amounts and are manipulated as numbers. *Dimension objects* can hold any value (including numbers). They represent a meaningful data element having its own unique identity, such as User ID.

Detail objects can also hold any value (including numbers), but rather than standing alone, they only have meaning in the context of a specific dimension value. An example of a detail object is User Name. Since multiple User IDs could have the same User Name, we would not want to treat User Name as a dimension (although we could try). User Name is really only meaningful within the context of a specific User ID.

When a system administrator designs a universe, a decision must be made for each object: should it be treated as a dimension or as a detail? Sometimes the decision is easy: Fund should be a dimension; Fund Name should be a detail associated with Fund. Other times it's more difficult: should Vendor Legal Name should be considered a detail (because there can be duplicates) or should it be considered a dimension (because it must be a particular value regardless of Vendor Code)? For these complex scenarios, objects are usually established as dimensions, by default.

Report developers sometimes encounter situations where an object is established in the universe as a detail, but for the purposes of a report it needs to be treated as a dimension (or vice-versa). This section describes these scenarios.

Detail Variables from Dimensions (Incompatible Objects)

Sometimes as a report developer, you will need to create detail variables before you will be able to see the data for a dimension object in your report. The most common scenario for this is "incompatible objects". When you have a report with more than one query, you have to merge objects to be able to use them from both queries in a single table. Even when you have merged all possible objects, you may still have some dimension variables that simply will not go into your table – or which display #DATASYNC when they are included in the table.

Suppose, for example, that you created a report to provide a list of Accounting Templates for each Cabinet, sorted by Department and Fund. The report has an ACTPL Data query (against the Accounting Template and Profile universe) for the Department and Fund values, and a FIN-AP Data query (against FIN-Accounts Payable) for Cabinet. The queries are merged on Fund and Department, but the report needs to include [Accounting Template Id] from ACTPL Data and [Cabinet] from FIN-AP Data. You are able to include one of these objects, but when you attempt to include the other, WebIntelligence prevents it – you see "Incompatible Object" or the object simply won't drop into the table. You manually add a column and type the object name in the formula, which results in a #DATASYNC error.

With merged queries, you can only include unmerged dimension objects from one of the queries at a time. If you need unmerged dimension objects from the other query, you must create detail variables for them. In our example, we would include [Accounting Template Id] in the report table, then establish a detail variable for [Cabinet] as follows:

Dtl-Cab = [Cabinet] (Qualification = Detail; Type = Text; Associated Dimension = [FIN-AP].[Department])

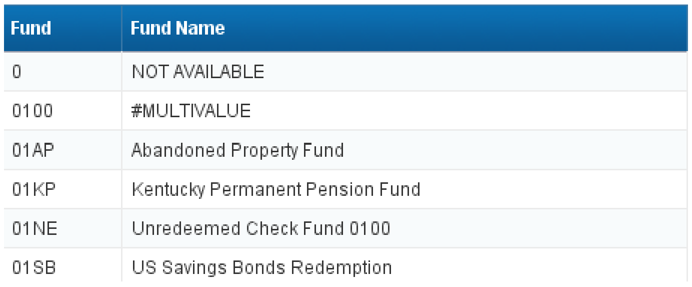
When establishing a detail variable, consider these guidelines when choosing the Associated Dimension:

- Object must be a dimension (i.e., not a detail or measure),
- Object must be present in both queries and merged,
- Detail variable must have a unique value in the context of the object.

Tip: The most commonly selected Associated Dimension is [Doc ID], since it is usually merged and unique. Sometimes, however, you may have to created a concatenated dimension variable to be your Associated Dimension, because none of the concatenated values are unique when they stand alone. An example of this might be [Dim-Full Name] (=[First Name]+[Middle Initial]+[Last Name]) or [Dim-Complete Phone Number] (=[Area Code]+[Phone Number]+[Extension]).

Dimension Variables from Details (#MULTIVALUE Error)

Occasionally you may come across an object that is established as a detail, but you need to see it in your report as a dimension. (It is rare to have a report requiring this, but it can be helpful in troubleshooting #MULTIVALUE errors.) For example, suppose you created a query for [Fund] and [Fund Name]. Here are the first few results:



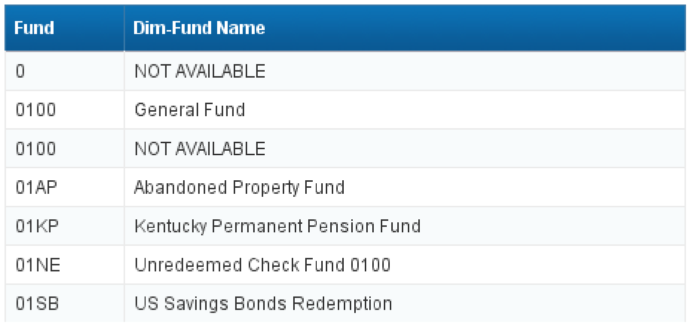
Fund	Fund Name
0	NOT AVAILABLE
0100	#MULTIVALUE
01AP	Abandoned Property Fund
01KP	Kentucky Permanent Pension Fund
01NE	Unredeemed Check Fund 0100
01SB	US Savings Bonds Redemption

#MULTIVALUE means there is more than one value for [Fund Name] associated with Fund code 0100. Detail objects are supposed to have a single value in the context of a specific value for the associated dimension; WebIntelligence won't display multiple values for a detail object. So how can we see the values represented by #MULTIVALUE?

The simplest way is to create a dimension variable for detail object [Fund Name]:

Dim-Fund Name = [Fund Name] (Qualification = Dimension)

Replacing [Fund Name] with [Dim-Fund Name] in the report yields the following results:



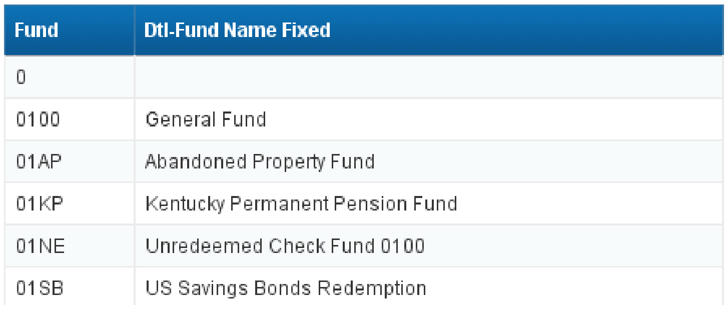
Fund	Dim-Fund Name
0	NOT AVAILABLE
0100	General Fund
0100	NOT AVAILABLE
01AP	Abandoned Property Fund
01KP	Kentucky Permanent Pension Fund
01NE	Unredeemed Check Fund 0100
01SB	US Savings Bonds Redemption

Since [Dim-Fund Name] is a dimension, it is treated as though it stands alone (instead of being interpreted in the context of [Fund]). Thus, the individual [Fund Name] values for Fund 0100 are displayed: "General Fund" and "NOT AVAILABLE". Since "NOT AVAILABLE" is not a meaningful value as a Fund Name, we can work around this issue as follows:

Dtl-Fund Name Available = If([Fund Name]="NOT AVAILABLE";"";[Fund Name])
(Qualification = Detail; Type = Text; Associated Dimension = [Fund])

Dtl-Fund Name Fixed = Max([Dtl-Fund Name Available])
(Qualification = Measure due to Max() function; Type = Text; Associated Dimension = [Fund])

Replacing [Dim-Fund Name] with [Dtl-Fund Name Fixed] in the report yields the following results:



Fund	Dtl-Fund Name Fixed
0	
0100	General Fund
01AP	Abandoned Property Fund
01KP	Kentucky Permanent Pension Fund
01NE	Unredeemed Check Fund 0100
01SB	US Savings Bonds Redemption

Encumbrance & Expenditure Report / Header Formatting (Letter, Landscape) – Start to Finish

This section follows the standard approach to report design detailed in "Basic Concepts" to create a report showing Encumbrance and Expenditure amounts for the accounting strings defining a budget line. Here is summary information about this report (following the conventions detailed in "Basic Concepts":

Item	Name	Filters	Variables
Document	Accounting – Encumbrances & Expenditures		
Universe(s)	FIN-General Accounting		
Query(s)	FIN-GA Data	Department InList Prompt	Filter objects (& names)
		FY InList Prompt	Jrnl Doc Code
		APD InList Prompt (optional)	Jrnl Doc Dept Code
		Closing Classification InList (10;11;12)	Jrnl Doc ID
			Jrnl Doc Record Date
			Unit (/Name)
			Fund (/Name)
			Fund Type (for table filter)
			Object (/Name)
			Function (/Name)
			Jrnl Posting Amt
Report(s)	Enc & Exp Summary	Fund Type = "0100" table filter	Prompt variables
	For Export		Flag-Enc dimension
			Flag-Acc Exp dimension
			Flag-Cash Exp dimension
			Amt-Enc measure
			Amt-Acc Exp measure
			Amt-Cash Exp measure

I. Query Design

A. **Name the report and Save it** – The best universe for an Encumbrance and Expenditure report at this time is the FIN-General Accounting universe. Since we need a query with at least one object in it to save our initial report, the first step is to create a query against FIN-General Accounting.
   1. Drag the [Department] and [Department Name] objects (from COA-Organization > Organization-Centralized view) to Result Objects. Rename your query tab to "FIN-GA Data".
   2. Use the "Close" button to "Close and apply changes". (This takes you to the Report Panel.)
   3. Click the "Save" button to save the report to your "My Favorites" folder with a meaningful name (such as "Accounting – Encumbrances and Expenditures").

B. **FIN-General Accounting universe query** – Edit the "FIN-GA Data" query, pulling Chart of Accounts elements corresponding to the budget line you are monitoring. For example, to monitor Encumbrances and Expenditures against an Operational Budget (Budget Structure 3), pull the following objects:
   1. *Chart of Accounts > COA - Fund Accounting class* – Pull the following objects:

Description	Sub-Class	Objects
Fund	Fund	Fund
		Fund Name
Fund Type	Fund	Fund Type
	Fund Hierarchy	
Object	Object	Object
		Object Name

2. *Chart of Accounts > COA – Organization class* – Pull the following objects:

Description	Sub-Class	Objects
Department	Organization - Centralized view	Department Department Name
Unit	Organization – Decentralized view	Unit Unit Name

3. *Chart of Accounts > COA – Detailed Accounting class* – Pull the following objects:

Description	Sub-Class	Objects
Function	Function	Function Function Name

4. *Financial Reporting Periods*– Pull the following objects:

Description	Sub-Class	Objects
Accounting Period	Accounting Period	Accounting Period
Fiscal Year	Fiscal Year	FY

5. *Posting Attributes class* – Pull the following objects:

Description	Sub-Class	Objects
Closing Classification	Posting Code	Closing Classification

6. *Accounting Journal class* – Since our query include Document Number objects, we must use the Accounting Journal class for the amount. Pull the following objects:

Description	Sub-Class	Objects
Document Code	Jrnl Doc Identification	Jrnl Doc Code
Document Department Code	Jrnl Doc Identification	Jrnl Doc Dept Code
Document ID	Jrnl Doc Identification	Jrnl Doc ID
Record Date	Jrnl Doc Dates	Jrnl Doc Record Date
Enc & Exp Amount	Jrnl Doc Amounts	Jrnl Posting Amt

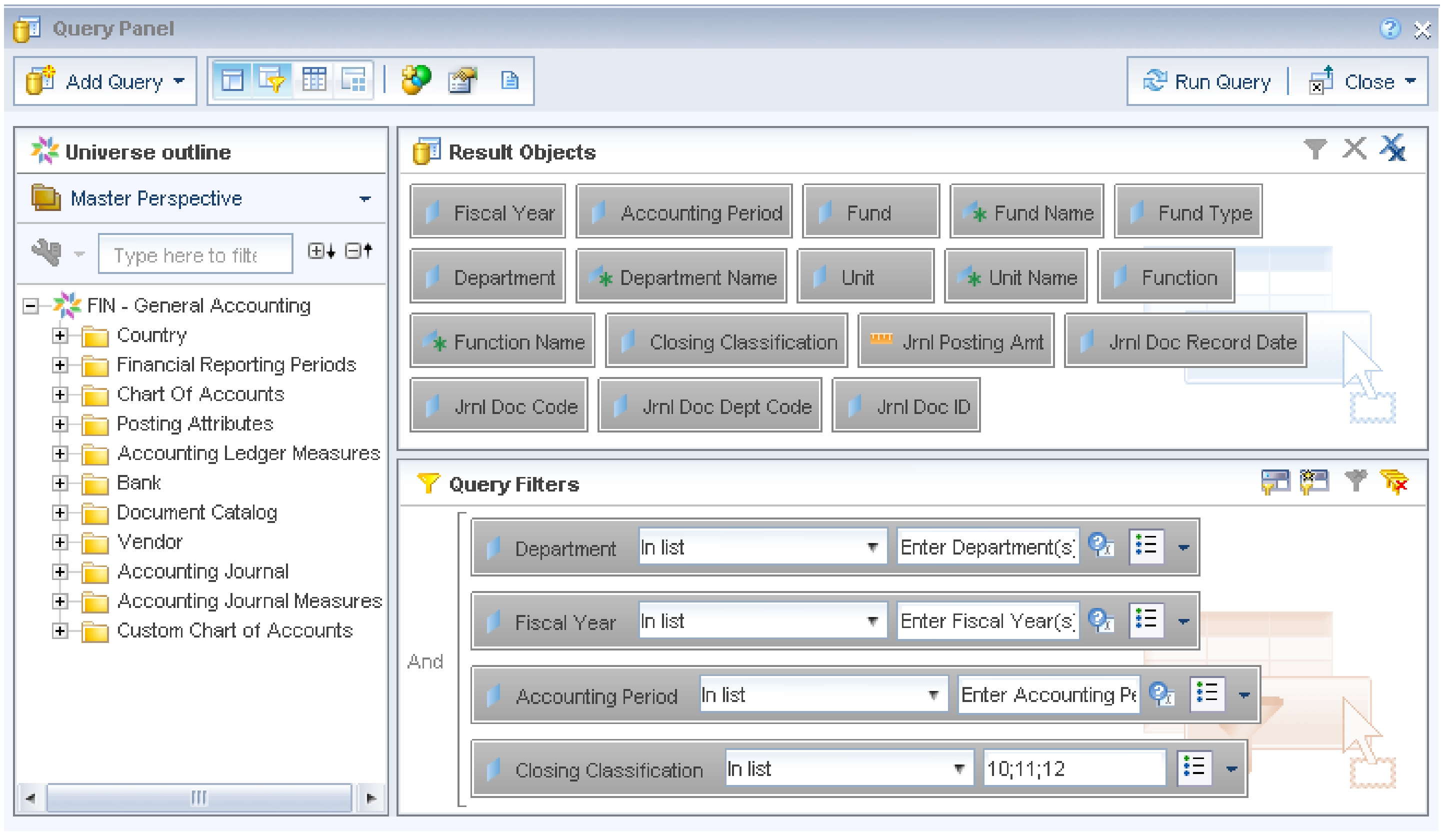
7. *Query Filters* – Drag the following objects from Result Objects to Query Filters to create filters:

Object	Operator	Values	Optional?	Default Values
Department	In List	Prompt: Enter Department:	No	
FY	In List	Prompt: Enter FY(s):	No	2017 (or current FY)
Accounting Period	In List	Prompt: Enter Accounting Period(s) (optional):	Yes	
Closing Classification	In List	10;11;12 (= Encumbrances, Accrued Expenditures, Cash Expenditures)		

8. *Prompt Order* – Open Query Properties and arrange the prompts in the order shown above.

A. **Other universe queries and merging** – No other queries are required for an Encumbrance / Expenditure report.

B. **Save the report** – Once again, use the “Close” button and choose “Apply changes and close”, then save the report with the “Save” button.



Department	In list	Enter Department(s
Fiscal Year	In list	Enter Fiscal Year(s
Accounting Period	In list	Enter Accounting Pe
Closing Classification	In list	10;11;12

I. Report Design – Refresh the report as needed while you design your report.

A. **Default report design** – Make the following changes to the generated report:

1. Delete the title box and the table block.
2. Rename the report tab from “Report 1” to “For Export”.
3. Save the report.

B. **Add table block** – Add the table block back as follows:

1. Hold your “Ctrl” key down while you select the following objects, then drag them into the report and arrange them in the following order (add “Name” objects as desired):

Objects for Table Block
Fiscal Year
Accounting Period
Fund
Department
Unit
Function
Closing Classification
Jrnl Posting Amt

Tip: For reports having only one query, the table is automatically created with the objects in the order in which you select them.

2. Format the table to set the position to the top of the page (i.e., Horizontal Position = 0, Vertical Position = 0).

C. **Report & Table filters** – For simplification, apply the following table filters (if you want other Fund Types, remove this filter once the report design is complete):

Object	Operator	Values
Fund Type	Equal to	0100

D. **Report data & variables** – Make sure the data is complete and totals are correct, as follows:

Notice that the [Jrnl Posting Amt] values currently include Encumbrance amounts ([Closing Classification] = "12"), Accrued Expenditure amounts ([Closing Classification] = "11"), and Cash Expenditure amounts ([Closing Classification] = "10"). These amounts must be broken out separately for inclusion in the report. To accomplish this, create "flag variables" for each type of amount, then measure variables for the amounts themselves.

1. *Create "flag variables"* – Create dimension variables using the following formulas:

Flag-Enc	= If([Closing Classification="12"]; "Yes";"No")
Flag-Acc Exp	= If([Closing Classification="11"]; "Yes";"No")
Flag-Cash Exp	= If([Closing Classification="10"]; "Yes";"No")

2. *Create amount variables* – Create measure variables using the following formulas:

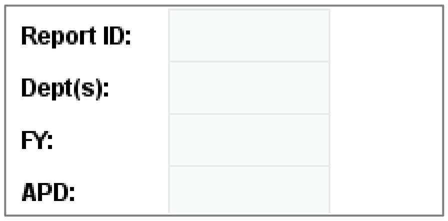
Amt-Enc	= [Jrnl Posting Amt] Where ([Flag-Enc]="Yes")
Amt-Acc Exp	= [Jrnl Posting Amt] Where ([Flag-Acc Exp]="Yes")
Amt-Cash Exp	= [Jrnl Posting Amt] Where ([Flag-Cash Exp]="Yes")

3. *Add amount variables to report* – add [Amt-Enc], [Amt-Acc Exp], and [Amt-Cash Exp] to your table (after [Jrnl Posting Amt]). Notice that each amount is only included in the correct column.
4. *Update column headings* – Remove [Closing Classification] and [Jrnl Posting Amt] from the table, then update the amount column headings (to "Encumbrances", "Accrued Expenditures", and "Cash Expenditures", respectively). Save the report.
5. *Add totals to amount columns* – Use the "Sum" button (located in the "Functions" section of the "Analysis" tab in the toolbars) to create column totals for amount column.
6. *Verify totals* – Spot-check the amounts in the table to ensure they are correct. (You may have to summarize the data in other ways to match inquiries in other systems such as eMARS or FAS3.) **Tip:** To quickly check totals, duplicate the report tab adding breaks and totals as necessary.
7. *Format totals* – As desired, format all numbers consistently (using "Numbers" options on the "Format" tab).
8. Right-click the report tab and choose "Duplicate Report". Rename the new report "Enc/Exp Summary" and make it the first report tab.

E. **Report headers and footers** – Follow these steps for the "Enc/Exp Summary" report tab header:

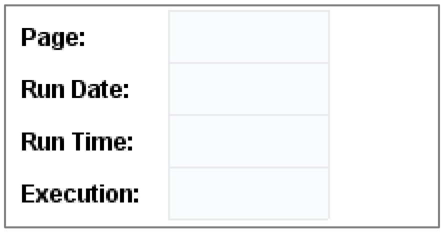
1. *Set Page Orientation and Margins* – Follow these steps to set up your page:
    a. In the "Display" section of the "Page Setup" tab in the toolbars, choose the "Page" button.
    b. In the "Page" section of the "Page Setup" tab, choose "Landscape" as the orientation and set the paper size to "Letter".
    c. In the "Margins" section, set the margins to 0.25" (= 0.6 cm) on each side. **Tip:** When typing in changes to settings in the toolbars, the changes will not take effect until the "Return" key on the keyboard is pressed while the cursor is in the field.
    d. Hover over the Header area of the page, then drag the bottom of the Header box down to create enough space in which to work on the header.
2. *Left Table of Header* – Follow these steps to create the left table of the header:
    a. From the "Report Element" tab in the toolbars, under "Table", select the "Insert Horizontal Table" button.
    b. Click in the upper left corner of the header to create a new horizontal table.
    c. Right-click the edge of the table to select it, then choose "Format Table…".
    d. In the "Layout" section, set the Relative Position of the table to the upper left corner of the page: Horizontal Position = 0, Vertical Position = 0.

e. Enter labels and format your table to appear as follows(there are two columns, but the second column is empty at this stage):



Report ID:	
Dept(s):	
FY:	
APD:	

3. *Right Table of Header* – Follow these steps to create the left table of the header:
   a. Select the left table, then click the "Copy" button (or right-click and choose "Copy").
   b. Right-click in the middle of the header and select "Paste" (or click the "Paste" button).
   c. Enter labels and format your table to appear as follows (there are two columns, but the second column is empty at this stage; the font is Ariel 9pt, bolded):

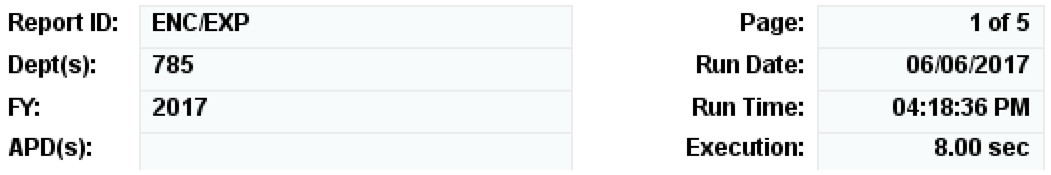


Page:	
Run Date:	
Run Time:	
Execution:	

   d. If necessary, drag the table inside the Header boundaries, then right-click it and select "Format Table…".
   e. In the "Layout" section, set the Relative Position of the table to the top center of the page: Horizontal Position = 5.25" (= 13.3 cm), Vertical Position = 0. (This will allow you to adjust the columns before setting the permanent Horizontal Position.)
4. *Formulas for Tables* – Follow these steps to populate the second column cells in each table:
   a. Create the following variables:

Report ID	= "ENC/EXP"
Prompt-Dept	= UserResponse("Enter Department(s):")
Prompt-FY	= UserResponse("Enter Fiscal Year(s):")
Prompt-APD	= UserResponse("Enter Accounting Period:")
Page Numbers	= Page()+" of "+NumberOfPages()
Run Date	= FormatDate(LastExecutionDate();"MM/dd/yyyy")
Run Time	= FormatDate(LastExecutionDate();"hh:mm:ss A")
Execution	= FormatNumber(LastExecutionDuration();"###0.00")+" sec"

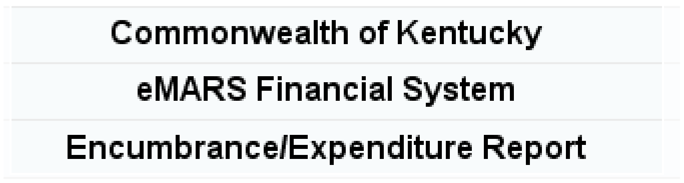
   b. Drag each of the variables into the second column cell corresponding to its name.



Report ID:	ENC/EXP	Page:	1 of 5
Dept(s):	785	Run Date:	06/06/2017
FY:	2017	Run Time:	04:18:36 PM
APD(s):		Execution:	8.00 sec

5. *Set Column Widths* – In each table, use the "Size" section of the "Format" tab in the toolbars to set the height and width for the rows and columns. The rows should be 0.2" (= 0.55 cm). Each first column should be 0.75" (= 1.9 cm). The second column of the first table should be 2.25" (= 5.72 cm). The second column of the second table should be 1.2" (= 3 cm). **Tip:** Right-justifying the columns of the second table will make your report more visually appealing.

6. *Position Second Table* – Now that the column widths are set, you can move the second table to its permanent position. In the “Layout” section, set the Relative Position of the table to the top right of the page: Horizontal Position = 8.5” (= 22 cm), Vertical Position = 0.
7. *Report Title Block* – Follow these steps to create a title for your report:
   a. From the “Report Element” tab in the toolbars, under “Table”, select the “Insert Vertical Table” button.
   b. Click in the upper left corner of the header to create a new vertical table.
   c. Right-click the edge of the table to select it, then choose “Format Table…”.
   d. In the “Layout” section, set the Relative Position of the table to the upper left corner of the page: Horizontal Position = 3.15” (= 8 cm), Vertical Position = 0.
   e. Enter labels and format your table to appear as follows(there is only one column in this table, and the font is Ariel 12pt, bolded):



Commonwealth of Kentucky
eMARS Financial System
Encumbrance/Expenditure Report

   f. Use the “Size” section of the “Format” tab in the toolbars to set the height of the rows to 0.25” (= 0.64 cm) and the width of the column to 4.25” (= 11 cm).
8. *Set Header Height* – In the “Header” section of the “Format” tab in the toolbars, set the Height for the header to 1” (= 2.54 cm).

F. **Sections** – Follow these steps to create sections on [Fiscal Year] and [Department]:
1. Right-click the Fiscal Year column and choose “Set as section”.
2. Replace the function in the Fiscal Year section cell with:
   =“Fiscal Year: ”+[Fiscal Year]
3. Right-click the Department column and choose “Set as section”.
4. Replace the function in the Department section cell with
   =“Department: ”+[Department]+ “ – ”+[Department Name]
5. Format the cells for each section using Ariel 10pt font, black and bolded, and adjust the widths. **Tip:** You can format both cells at once by holding down your “Ctrl” key while selecting them.
6. Use the Document Structure and Filters tab in the left panel to navigate to the Fiscal Year section. Right-click it and choose “Format section…”.
7. In the “Layout” section of the “Format Section” window, set the Relative Position of the table to the upper left corner of the section: Horizontal Position = 0, Vertical Position = 0.
8. Repeat the previous two steps for the Department section, the Department cell, and the Fiscal Year cell. **Tip:** When setting the position of the cells, choose “Format cell…” instead of “Format section…”.

G. **Breaks** – Create a break on [Accounting Period] by right-clicking the object column and choosing “Break > Add Break”. Use the “Sum” button (under “Functions” on the “Analysis” tab) to add break totals. Use the “Numbers” options (on the “Format” tab) to format the totals consistently with the other numbers.

H. **Totals** – If desired, add totals for the Cabinet section and Grand Totals for the report. Refer to the “Capital Project Report – Start to Finish” section for details.

I. **Other reports** – Follow these steps to update the "For Export" report tab:
   1. Add the following objects to the "For Export" report tab (after [Jrnl Posting Amt]):

Objects for Table Block
Jrnl Doc Code
Jrnl Doc Dept Code
Jrnl Doc ID
Jrnl Doc Record Date

   2. Update column headings as desired.
   3. Create an "Enc/Exp Documents" report tab if desired. Refer to the "Start to Finish: Allotment Balance Report" section for details.

J. **Save, review, correct, purge and publish**– Complete formatting of your report according to the standards and conventions for your organization. Normally, you would be designing this report for other users, so in addition to keeping a copy of the report in your "My Favorites" folder (or a sub-folder), you would want to put a purged copy in your Agency folder (or sub-folder). Remember, your Agency folder is under your cabinet number on the "Folders" tab under Public Folders > eMARS Financial > eMARS Agency Reports.

II.For Review and Discussion

A. What Chart of Accounts elements would you need to add to the query and report tabs to better meet the needs of your Department or Cabinet?

B. When formatting report tabs to create printed reports, how do you know when to use a Section, a Break, or a Report Tab to organize your data?

C. If you wanted an "Enc/Exp Documents" report tab containing document data, how would you organize that report tab?