InfoView Formula Reference and Training Document

BY MAGGIE FIELDEN

HTTP://DATAMGMT.UCSC.EDU

Table of Contents

Formula Definition

A formula is used to display information that is not retrieved by existing objects in the universe.
The formula is the definition of how to produce that information.

Formulas can be static, meaning that the output of your formula does not change.
Or, they can be dynamic, meaning that the output of your formula changes based on other elements in your report, such as objects, prompts, other rows and so forth.

Formulas are used in:

- Function Rows
- Pre-defined Cells
- Normal cells via the Function Bar
- Custom Variables

Formulas are pre-set in Function Rows and Pre-defined Cells
Formulas are adjustable with the Function Bar and in Custom Variables

Formulas can be used to achieve the following:

- Cosmetic preferences, such as concatenating two text fields together into one field
- Math, including number crunching and aggregating functions (e.g. maximums, minimums, counts, totals)
- Altering existing data, such as renaming fields, Date & Time conversions and Character functions (e.g. length and position)
- Unique sorting and grouping of data
- Metadata, which is data about your data, such as Document information (e.g. author, run date) and Logic functions (e.g. is this condition true?)
- Conditional statements, such as "If/Then" statements, which can apply all of the above only when certain conditions are met

Formula Uses

Function Rows

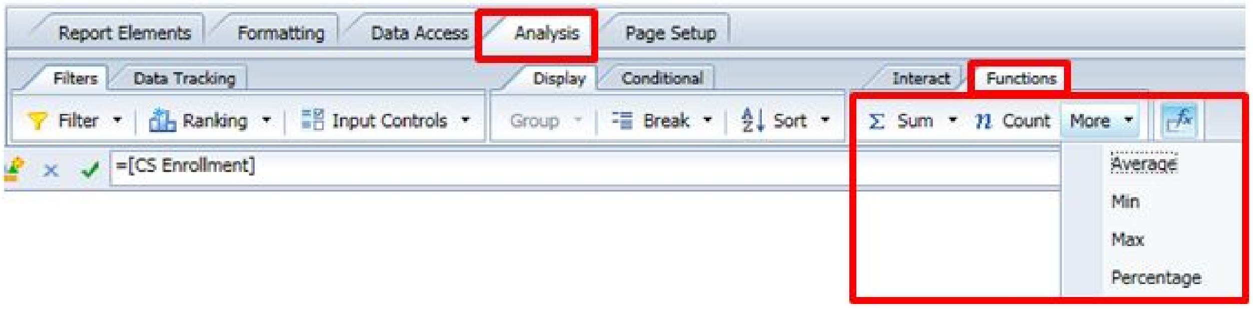
Definition

Adding function rows means adding a row at the bottom of a break, section or table that automatically contains aggregated information.

The quick access aggregate functions are: Min, Max, Sum, Count, Average and Percentage. Please note that some of these functions only work on numbers.

When added after applying a break, adding a function row automatically includes two levels of aggregation, one for the grouped data and one for the entire table. Sections only add one for the grouped data.

How-To
1. To insert a function, click on the content (not the header) that you would like to aggregate.
2. Navigate to the Analysis menu and the Functions submenu.



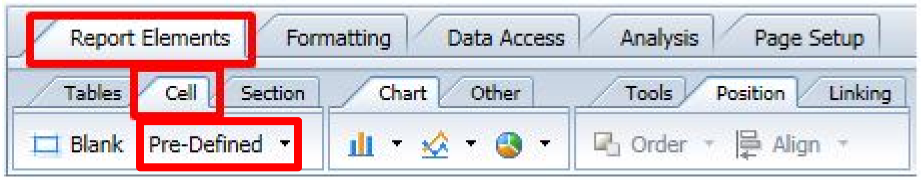
3. Click on the function that you'd like to add.

Alternately, you can short-cut to your options with a right-click (command click on a mac). Then, select "Insert" and choose your desired function.

Pre-defined Cells

Definition
Pre-defined cells are single cells that have a function pre-programmed in to make inserting them quick and easy. These are the most-used functions for single table cells, including last refresh dates and prompt or filter values. See below for a list of options.

How-To
1. Navigate to the menu "Report Elements", then the submenu "Cell"



2. Click on "Pre-defined" and select from the list of options.

The most used pre-defined cells are:
- "Document Name" shows the file name of the entire report
- "Last Refresh Date" shows the last time the report was refreshed
- "Drill Filter" shows the combined values of all of the drill filter selections and changes based on differing selections
- "Prompt" shows the combined values of all of the prompt inputs (individually or as a single summary field)
- Several options for displaying page numbers

3. After choosing the desired cell, click on your report where the cell should go.

Formula Bar

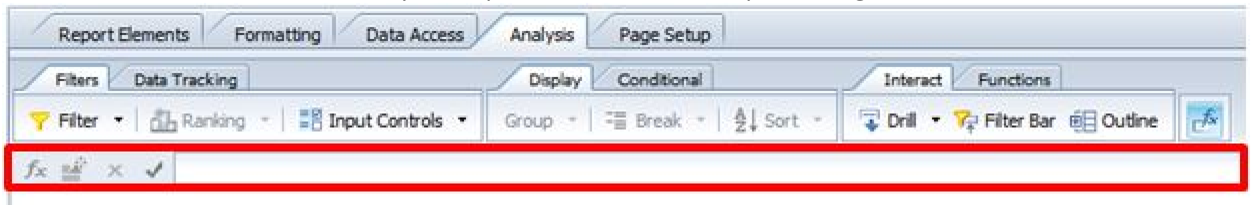
Definition

The Formula Bar shows the contents of cells in your report. It can also be used to enter new information, especially formulas via the formula editor, without the use of a variable. This can be useful as a quick way to complete the same functions, but it can only be used in the one field and can't be transferred to other fields or tables or used in other variables.

How-To

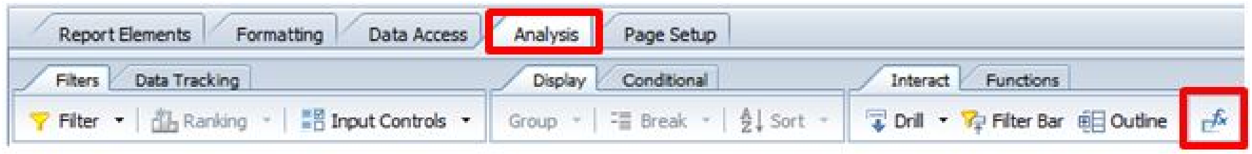
<u>Formula Bar</u>

The formula bar is located above your report and below the report-design menus.



If you do not see this bar already after switching into design mode, complete the following:

1. Navigate to the menu Analysis.

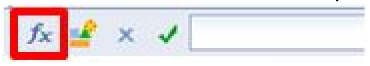


2. Click on the right-most button with the "fx" symbol on it.

The bar should appear and the “fx” button should highlight in blue.

<u>Formula Editor</u>

The formula editor can be opened by selecting a cell and clicking the first button on the formula bar.



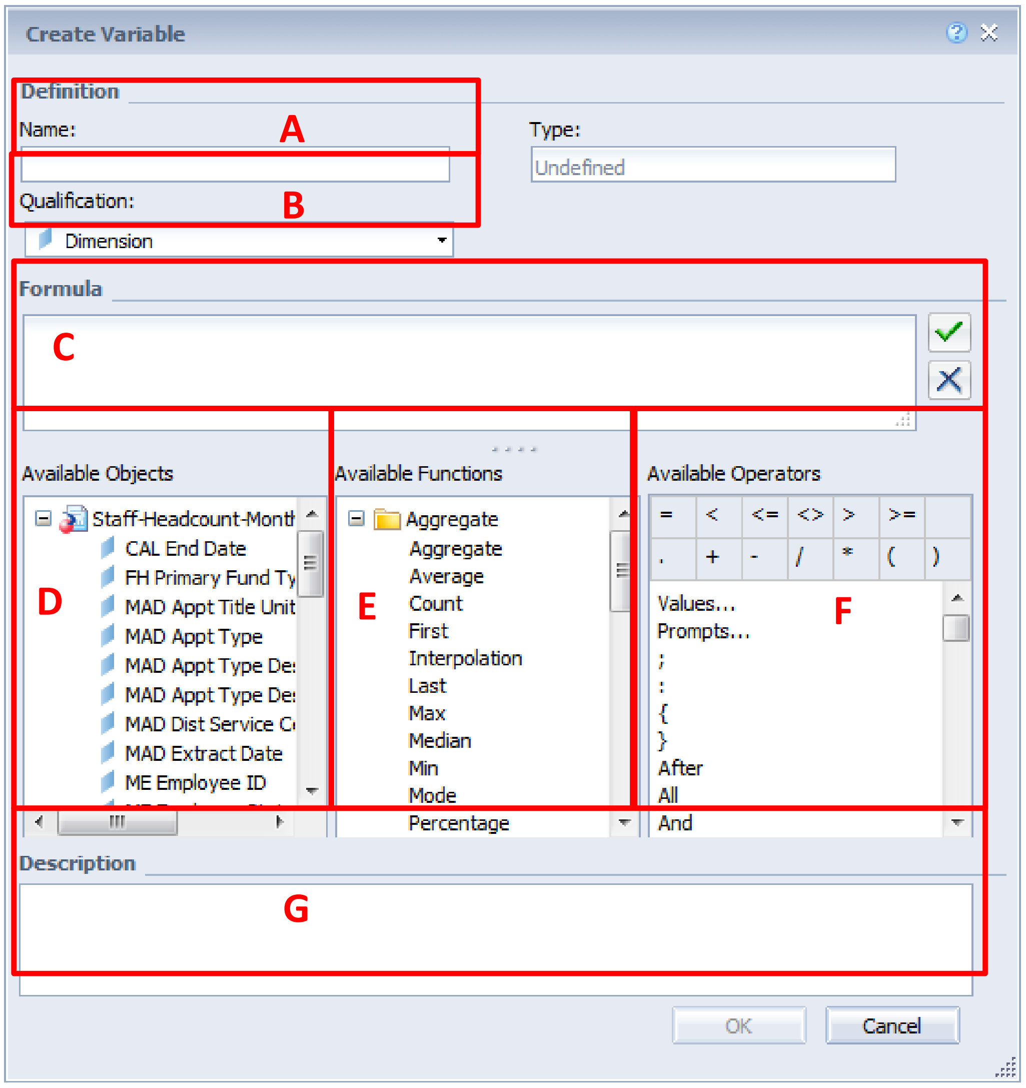
The formula editor is the same panel as the Create or Edit Variable panel (see below), but missing the “Definition” section. The definition section contains the details that define the formula as an object (name, qualification, etc.). Besides this, the formula panel acts in exactly the same way. See below for a detailed map of the Create Variable Panel.

Custom Variables

Definition

Variables are a way to use formulas as if they were objects. To become objects, variables get a distinct name, a data type and an object type in addition to the formula itself. This allows them to be used repeatedly across report tabs as well as any other way that an object could be used (e.g. in headings, in other formulas, in breaks, in sorts, etc.)

Create Variable Panel Guide



A. Name: Type in a unique name for your variable
   - We recommend using all capital letters to make variables instantly recognizable

B. Qualification: This is the data type of your new variable
   - Most of the time, InfoView decides this automatically.

C. Formula: Where you type in your formula

D. Available Objects: These are the result objects from your report (including other variables)

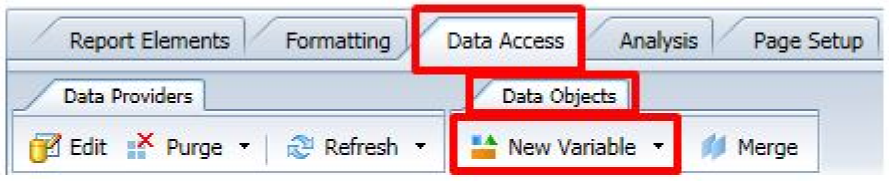
E. Available Functions: Functions perform a particular task on the data available

F. Available Operators: These operators allow the use of multiple objects and functions

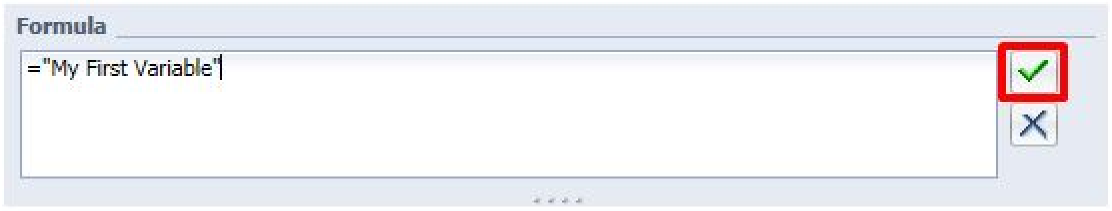
G. Description: When you select any function or operator (Sectors E and F), a description of that function or operator will appear here. Also, a helpful link to more information on the SAP site: More on this function

How-To

1. Navigate to the menu "Data Access", then the sub-menu "Data Objects"



2. Click "New Variable"
3. In the "Create Variable" pop-up panel, give your variable a unique name.
   Note: To improve recognizability, we recommend giving variables names in all capital letters, for example: TEST VARIABLE
4. Change the data type, if desired. InfoView usually can tell automatically.
   Changing the option from the default "Dimension" is rarely necessary.
5. Type in your formula, always starting with a leading equals sign.



="My First Variable"

6. Optional step: click the green check mark to check that your formula is correctly formulated.
7. Click "OK" at the bottom of the Create Variable panel.
8. Your variable will now appear in the "Available Objects" Data Palette in the folder at the bottom entitled "Variables".

Formula Components

Formula can include the following components:

- Leading = (This is required!)
- Text
- Operators
- Objects
- Functions

Leading =

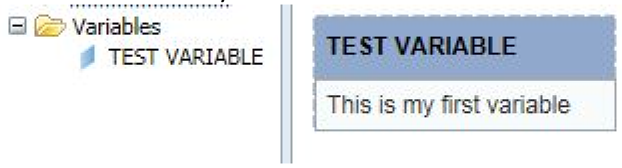
Formulas always start with an equals sign. If you try to save your formula without one, the system will either automatically apply one or give you an error message (usually the latter).

Text

Formulas can contain plain text content. All plain text must be enclosed in double quotation marks.

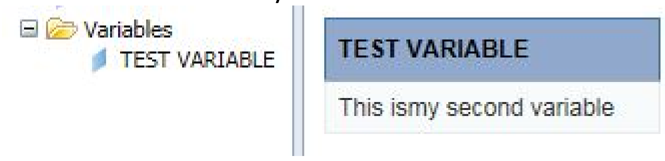
Example: Text

="This is my first variable"



Operators
Operators are the glue that holds all of the other contents together. They go in between text, objects and functions to help the function make sense when more than one thing is present.

Example: The Operator "+"
The plus-sign can be used to glue two things together.
    ="This is" + "my second variable"

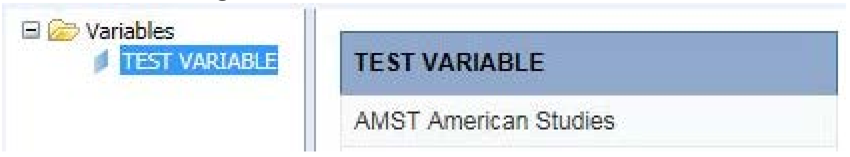


This ismy second variable

Note the lack of space between “is” and “my” because we neglected to add one in our variable.

Objects
Objects are the fields that you have in your report. It is the same list as is available in the "Available Objects" data palette. All objects will appear in your formula with the name of the object between two corner brackets [ ] .

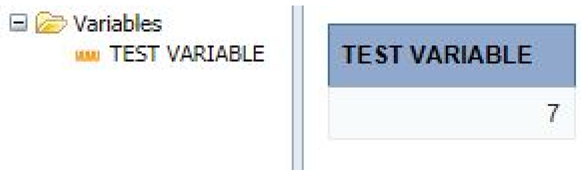
Example: Objects Plus Other Objects
When applied between two objects, the plus sign will glue the contents of the two fields together into one field for all values.
    =[D Dept Code] + " " + [D Dept Name] will show:
        e.g. AMST American Studies



AMST American Studies

Functions
Functions perform a task on the data fed into it, usually producing some other sort of data or output. They introduce the most complexity and flexibility to variables. Each function will have a different effect based on what it has been programmed to do. Functions can be combined with any number of objects, text and operators, depending on the need.

Example: Count()
The function Count() is an example of a function that requires an object. When combined with an object, the function Count() will count the number of rows in that field.
    =Count([DV Division Name])



7

If-Then-Else Formulas

If-Then Formulas are the most popular type of variable. They create the opportunity to change the output of your variable based conditionally on other objects, variables, prompt values and factors.

Some important information:

- If, Then and Else are all functions that work together.
- If-Then statements often use the operators "AND" and "OR". AND, OR and NOT are called Boolean Operators and will also be covered in this section.
- If-Then statements also demonstrate the use and value of parentheses () in formulas

Functions: If, Then, Else

Syntax

If-Then statements are a type of formula logic that allows the output of the variable to be conditionally determined. For all If-Then statements, the conditions must be defined as well as the action or information that should be displayed when those conditions are met.

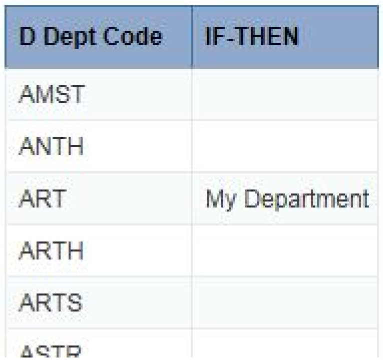
If condition_one (is_met) **Then** do_this

"Condition_one" and "do_this" are defined using our available objects, functions and operators. In many cases, we want to add or change how a particular value in the data base appears in our report. In such a situation, our "condition_one" might look like this: [Object]="value" and our do_this might look like this: "text_here".

Example: **If** condition_one **Then** do_this

=If [D Dept Code]="ART" Then "My Department"

This If-Then statement is saying that when the field Department Code is the three letters "ART" (exact matches only), then the variable should display the text "My Department".



D Dept Code	IF-THEN
AMST	
ANTH	
ART	My Department
ARTH	
ARTS	
ASTR	

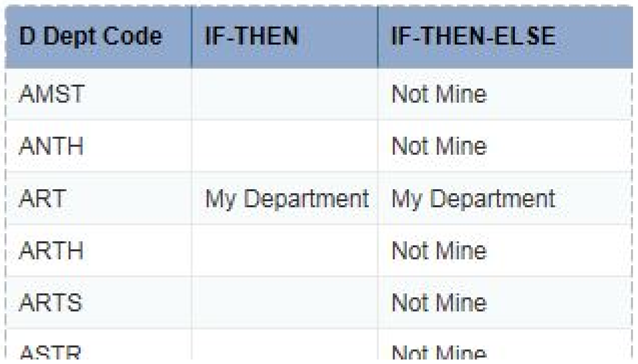
Notice that the other fields are blank. Our variable only said to do something when the condition is met.

Else
Adding "**Else** do_this" at the end of an If-Then statement tells the variable to do something or display information in the situation where the condition is <u>not</u> met.

<u>Example</u>: **If** condition_one **Then** do_this **Else** do_this

```
=If [D Dept Code]="ART" Then "My Department"
  Else "Not Mine"
```

This If-Then statement means that if the Department Code is "ART" then show "My Department". However, if the Department Code is <u>not</u> "ART" then show "Not Mine".



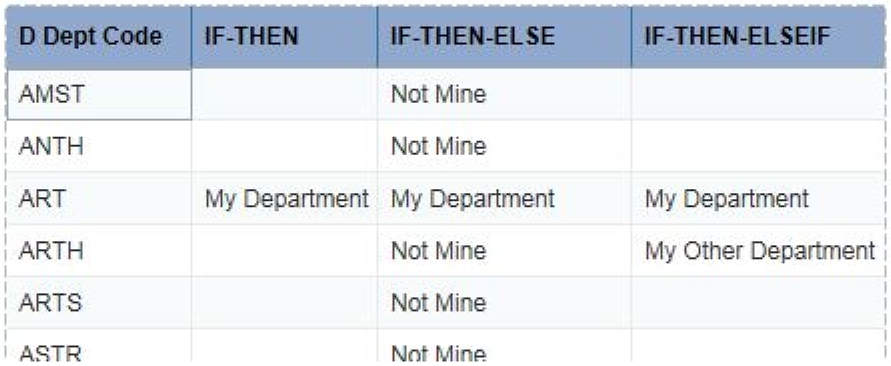
D Dept Code	IF-THEN	IF-THEN-ELSE
AMST		Not Mine
ANTH		Not Mine
ART	My Department	My Department
ARTH		Not Mine
ARTS		Not Mine
ASTR		Not Mine

ElseIf
After the first "Then" statement, another If-Then statement can be added on to the end to create a second conditional statement. Since the new If-Then statement will come after an existing If-Then statement, the "If" turns into "ElseIf"

<u>Example</u>: **If** condition_one **Then** do_this **ElseIf** condition_two **Then** do_this

```
=If [D Dept Code]="ART" Then "My Department"
  ElseIf [D Dept Code]="ARTH" Then "My Other Department"
```

This variable states that if the Department Code shows "ART", the variable should show "My Department". If the Department Code shows "ARTH", the variable should show "My Other Department".

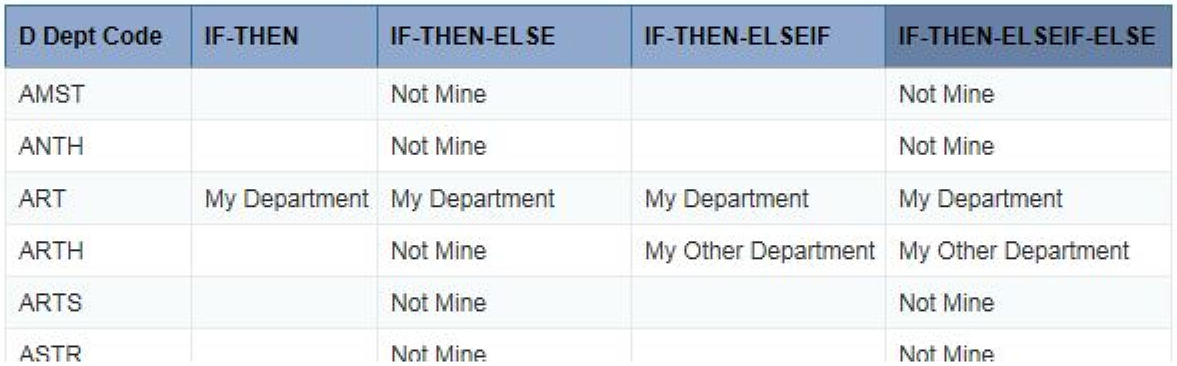


D Dept Code	IF-THEN	IF-THEN-ELSE	IF-THEN-ELSEIF
AMST		Not Mine	
ANTH		Not Mine	
ART	My Department	My Department	My Department
ARTH		Not Mine	My Other Department
ARTS		Not Mine	
ASTR		Not Mine	

If-Then-ElseIf statements can also always have an ending "Else" statement.
Example: **If** condition_one **Then** do_this **ElseIf** condition_two **Then** do_this **Else** do_this

=If [D Dept Code]="ART" Then "My Department"
ElseIf [D Dept Code]="ARTH" Then "My Other Department"
Else "Not Mine"

This variable will show:



D Dept Code	IF-THEN	IF-THEN-ELSE	IF-THEN-ELSEIF	IF-THEN-ELSEIF-ELSE
AMST		Not Mine		Not Mine
ANTH		Not Mine		Not Mine
ART	My Department	My Department	My Department	My Department
ARTH		Not Mine	My Other Department	My Other Department
ARTS		Not Mine		Not Mine
ASTR		Not Mine		Not Mine

If-Then statements can be combined as many times as needed via "ElseIf".

Cautions
If-Then statements can get tricky. Here's a few cautions when using them:

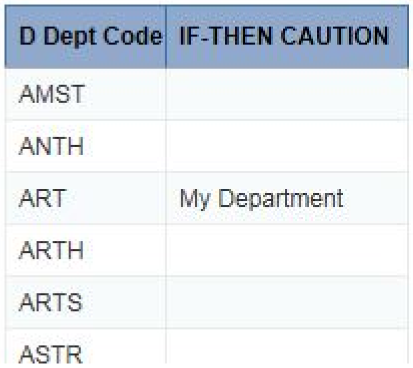
1. If-Then statements can become very complicated very quickly. One If-Then statement written by a UCSC colleague was over two printed pages long! In such cases, logic errors are easy to make. We recommend testing your variable out on a small amount of data in order to verify the accuracy.

2. If you have one row that meets more than one condition in your If-Then-ElseIf string, it will always do what the first "Then" says and ignore the second.

Example:

=If [D Dept Code]="ART" Then "My Department"
ElseIf [D Dept Code]="ART" Then "My Other Department"

This variable will show:



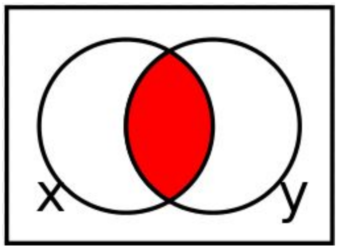
D Dept Code	IF-THEN CAUTION
AMST	
ANTH	
ART	My Department
ARTH	
ARTS	
ASTR	

Operators: AND, OR and NOT

AND, OR and NOT are called Boolean Operators, which are very important and powerful database tools that you can use in InfoView.

As operators, they don't do anything by themselves. Instead, they must be applied to one or more things (text, database objects and functions) to help define a more specific formula or query.

AND

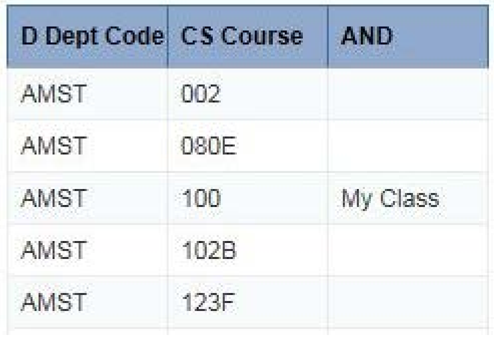


A Boolean AND returns only those results that satisfy <u>both</u> conditions at the same time.

<u>Example</u>: **If** condition_one **AND** condition_two (are both true) **Then** do_this

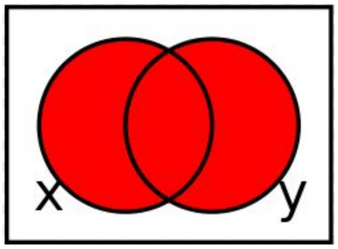
=If [D Dept Code]="AMST" And [CS Course]="100" Then "My Class"

This variable states that in the situation where the department code is equal to "AMST" while at the same time the course code is "100", then it should display "My Class".



D Dept Code	CS Course	AND
AMST	002	
AMST	080E	
AMST	100	My Class
AMST	102B	
AMST	123F	

OR

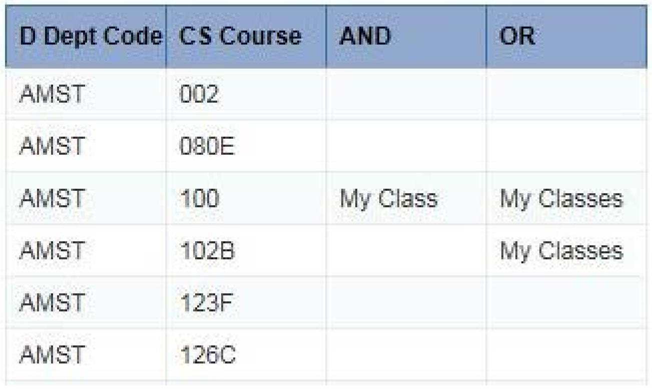


A Boolean OR states that all results need only satisfy one of the two conditions.

<u>Example</u>: **If** condition_one **OR** condition_two **Then** do_this

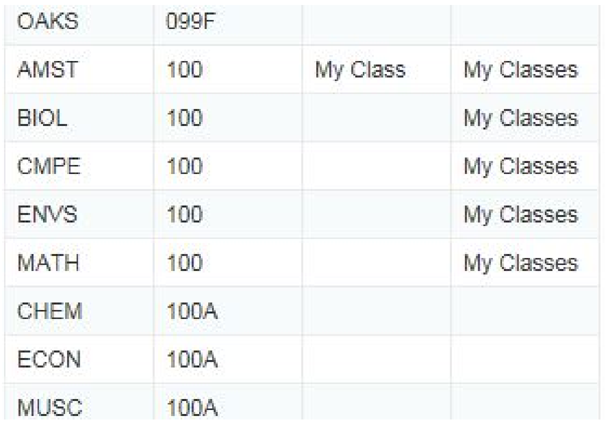
=If [CS Course]="102B" Or [CS Course]="100" Then "My Classes"

This variable states that classes that have either Course Code "102B" or Course Code "100" should display "My Classes"



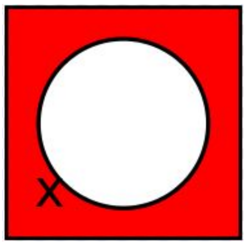
D Dept Code	CS Course	AND	OR
AMST	002		
AMST	080E		
AMST	100	My Class	My Classes
AMST	102B		My Classes
AMST	123F		
AMST	126C		

Note: In this variable, we have not defined if the Department Code should be "AMST"; All classes, regardless of department that have either of those numbers will show "My Classes". To prevent this sort of mix-up, check out the next section on Parentheses.



OAKS	099F		
AMST	100	My Class	My Classes
BIOL	100		My Classes
CMPE	100		My Classes
ENVS	100		My Classes
MATH	100		My Classes
CHEM	100A		
ECON	100A		
MUSC	100A		

NOT

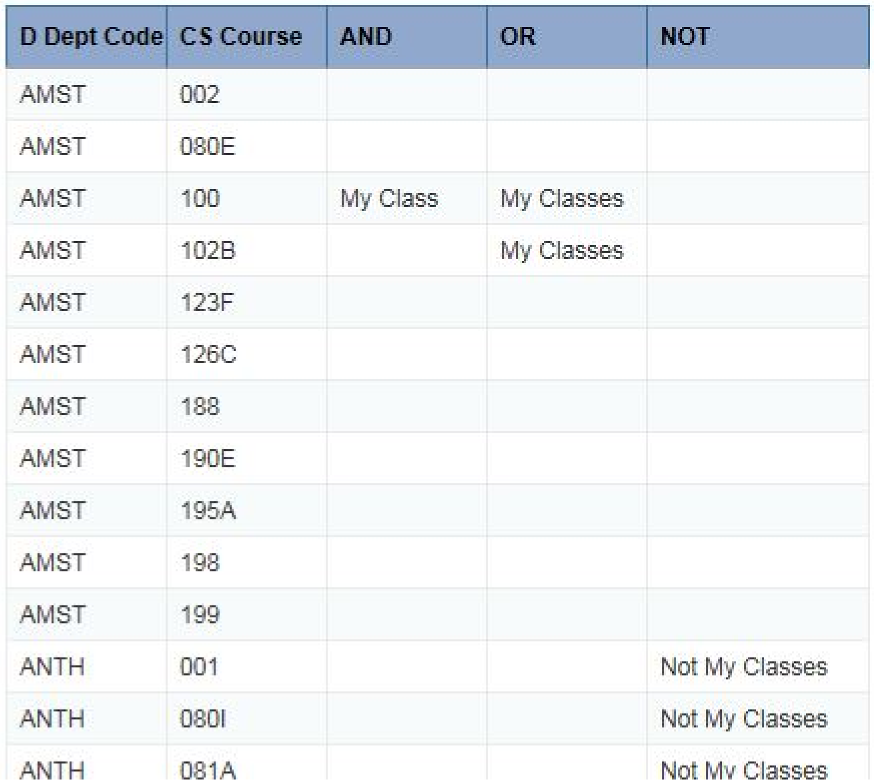


A Boolean NOT states that all results returned will not satisfy the condition defined.

<u>Example</u>: **If NOT(**condition_one**) Then** do_this

=If Not([D Dept Code]="AMST") Then "Not My Classes"

This variable states that all classes that are not coded with Department Code "AMST" will show the text "Not My Classes"



D Dept Code	CS Course	AND	OR	NOT
AMST	002			
AMST	080E			
AMST	100	My Class	My Classes	
AMST	102B		My Classes	
AMST	123F			
AMST	126C			
AMST	188			
AMST	190E			
AMST	195A			
AMST	198			
AMST	199			
ANTH	001			Not My Classes
ANTH	080I			Not My Classes
ANTH	081A			Not My Classes

Combining ANDs, ORs and NOTs
ANDs, ORs and NOTs are infinitely combinable with each other. Exactly how they execute can be tightly controlled with parentheses. If you do not use parentheses, the variable will execute left-to-right, which might not be what you mean.

Parentheses ( )

Definition
Parentheses in this case refer to the round brackets (and). Many functions use these to delineate all of the content that they should process. They are very powerful tools in InfoView variable writing, specifically in any mathematical transactions and in logical transactions. Parentheses in math or logic work as a way to direct InfoView to do groups of things in a particular order.

General note: All open parentheses, e.g. (, **must** be closed inside the formula, otherwise InfoView will not be able to process your variable. Parentheses can be nested within each other, e.g. (()).

Functions
Functions use parentheses to delineate exactly what should be processed and what should not. In the parentheses that follow a function, everything between them will be processed by the function. See the aggregate functions below for examples.

Math
InfoView completes mathematical transactions according to "Operator Precedence", which is another word for "Order of Operations" from high school algebra. The Operator Precedence dictates that certain

mathematical calculations happen before others. For example, multiplication and division happen before addition and subtraction. However, parentheses contents always calculate first.

The formula =2+3*4 will compute as follows: 3 times 4 is 12, then add two to make 14

The formula =(2+3)*4 compute as follows: 2 plus 3 is 5, then multiplied by 4 will make 20.

Both formulas are completely legitimate in InfoView, either computed with typed-numbers or replaced by objects, but the two formulas will produce different numeric results.

Logic
InfoView reads logical transactions in your variable formula from left to right, unless parentheses group the transactions differently.

Example: **If** condition_one **AND** (condition_two **OR** condition_three) **Then** do_this

Without parentheses:

=If [D Dept Code]="AMST" And [CS Course]="102B" Or [CS Course]="100" Then "My Classes"

This will label the following classes as "My Classes"
- The Course 102B from the department AMST
- All courses that that have the number 100

With parentheses:

=If [D Dept Code]="AMST" And ([CS Course]="102B" Or [CS Course]="100") Then "My Classes"

This will label the following classes as "My Classes"
- The course 102B from the department AMST
- The course 100 from the department AMST

Functions
Aggregate
All functions in this area must have an object from your report inside the parentheses.

Count()
Syntax: Count([Object_name])
Explanation: This function counts the rows returned for the selected object. In a table by itself, it will display the total number of rows for the given object. When added to a table, it will split the total into the appropriate number for each row.

Example:

=Count([DV Division Name])

This count in a course report will show the numeral 7 because seven divisions give classes at UCSC

Troubleshooting: If you notice that your count function is returning a number that is too high, try adding "distinct" to the syntax to make sure it is not counting duplicate values.
Syntax: Count([Object_name];Distinct)

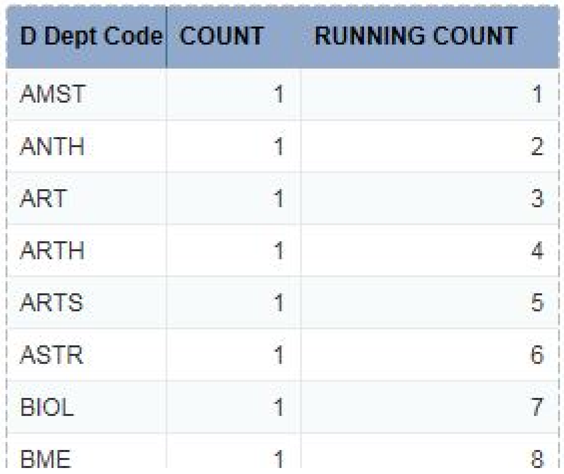
Max()
Syntax: Max([Object_name])
Explanation: This function returns the highest value of the set retrieved for your report.
Note: This function will also work on letters. For example if you have rows returned that either contain the letter "N" for no and the letter "Y" for yes, a variable asking for the max value in this field will return a "Y" because "Y" comes after "N" in the alphabet.

Min()
Syntax: Min([Object_name])
Explanation: This function returns the lowest value of the set retrieved for your report.
Note: This function will also work on letters (see explanation in Min, above).

Sum()
Syntax: Sum([Object_name])
Explanation: This function adds numeral values together.

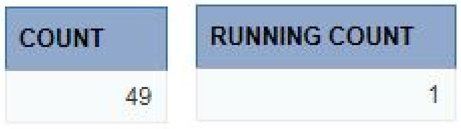
Running-
Explanation: All of the functions prefaced by the word "Running" will do the same action to the chosen field as the original function, but it will complete the action continually along the side of a table with the counted field in it.

Example Count() vs. RunningCount()
This table shows both variable functions counting D Dept Code while D Dept Code is in the table:



D Dept Code	COUNT	RUNNING COUNT
AMST	1	1
ANTH	1	2
ART	1	3
ARTH	1	4
ARTS	1	5
ASTR	1	6
BIOL	1	7
BME	1	8

This table shows both variable functions counting D Dept Code but without any other fields in each table:



COUNT
49

RUNNING COUNT
1

Character

Length

Syntax: Length(string) where

- "string" can be an object [Object_name] or text between quotation marks

Explanation: This function returns the number of characters in an object or text.
Examples:
=Length("no") would return 2, because the word "no" has two characters in it.
=Length([Employee_ID]) would return 7 for every single row, because Employee IDs are always seven numbers long.

Pos

Syntax: Pos(test_string;pattern) where

- "test_string" can be an object [Object_name] or text between quotation marks
- "pattern" can be an object [Object_name] or text between quotation marks.

Explanation: This function returns the position of a particular pattern of numbers and letters inside an object or text between quotation marks. It is particularly helpful in combination with other functions, such as Left and Right noted below.
Examples:
=Pos("How are you?";"?") would return 12, because the question mark appears in the twelfth position in the string.
=Pos("Santa Cruz";"Sa") would return 1, because the pattern starts on the first character.
=Pos([Employee_ID];"7") would return 1 for people at UCSC because 7 is the first digit of employee IDs here. Note that anyone from a different campus would have a different number returned based on where the first 7 actually appears, if it does at all. Note also that if someone has more than one 7 in their ID number, the position number returned will denote the very first instance of 7 only.

Left

Syntax: Left(string;number_of_characters) where

- "string" can be an object [Object_name] or text between quotation marks
- "number_of_characters" can be a simple number (e.g. 7) or a second function, such as Pos(), which will output a number.

Explanation: This function returns the first number of characters of a string, starting at the left.
Examples:
=Left("How are you?";5) would return "How a" because those are the first five characters of the string.
=Left([Employee_ID];"2") would return just the first two numbers of a person's employee ID.
Additional: LeftPad and LeftTrim offer options to add or subtract characters to the beginning of a text string.

Right

Syntax: Right(string;number_of_characters) where

- "string" can be an object [Object_name] or text between quotation marks
- "number_of_characters" can be a simple number (e.g. 7) or a second function, such as Pos(), which will output a number.

Explanation: This function returns the last number of characters of a string, starting at the right.
Examples:
=Right("How are you?";6) would return "e you?" because those are the last six characters of the string.
=Right([Employee_ID];"2") would return just the last two numbers of a person's employee ID.
Additional: RightPad offers the option to add or subtract characters to the beginning of a text string.

Match

Syntax: Match(test_string;pattern) where

- "test_string" can be an object [Object_name] or text between quotation marks
- "pattern" can be an object [Object_name] or text between quotation marks

Explanation: This formula returns a "true" or "false" based on the presence or absence of the pattern in the test_string. "True" is denoted by the numeral 1, "False" by the numeral 0.

When the pattern is shorter than the test_string, you will have to use a wildcard as a placeholder for the extra characters on either side.

- * is the wildcard for one or more characters
- ? is the wildcard for one and only one character

Examples:

=Match("How are you?";"How are you?") would return 1 because those two strings match exactly.
=Match("How are you?";"are") would return 0 because the string and the pattern do not match exactly.
=Match("How are you?";"*are*") would return 1 because the string contains the pattern.

WordCap

Syntax: WordCap(string) where

- "string" can be an object [Object_name] or text between quotation marks

Explanation: WordCap will capitalize the first letter of every word in your string.

Example:

=WordCap("how are you?") would return "How Are You?"

Additional: The functions Upper and Lower make all the letters in a string or object upper or lower case.

Date & Time

Note: Many of these functions require the use of parentheses, but the parentheses remain empty. This is normal.

CurrentDate

Syntax: CurrentDate()

Explanation: This function returns the current date at the time of refreshing. This is useful for noting on your reports when the report was refreshed.

CurrentTime

Syntax: CurrentTime()

Explanation: This function returns the current time at the time of refreshing. This is useful for noting on your reports when the report was refreshed.

DayNumberOfMonth, DayNumberOfWeek, DayNumberOfYear

Syntax: e.g. DayNumberOfMonth(date)

- date is an object [Object_name] containing only a date or a function which will output a date

Explanation: This function returns the number day number in a month/week/year.

LastDayOfMonth, LastDayOfYear

Syntax: e.g. LastDayOfMonth(date)

- date is an object [Object_name] containing only a date or a function which will output a date

Explanation: This function returns the last day of in a month/year. This can be useful when trying to match up extract dates from two queries or data sets.

DaysBetween, MonthsBetween
Syntax: e.g. MonthsBetween(first_date;last_date)

- first_date and last_date are objects [Object_name] containing only a date or a function which will output a date

Explanation: This function allows date mathematics within InfoView, calculating the days or months between two input dates.

Month, MonthNumberOfYear
Syntax: e.g. Month(date)

- date is an object [Object_name] containing only a date or a function which will output a date

Explanation: This function pulls the month out of a given date. Month will pull the name of the month, MonthNumberOfYear will pull the number of the month.
Example: 12/31/2015

- Month will show "December"
- MonthNumberOfYear will show "12"

Year, Week
Syntax: e.g. Year(date)

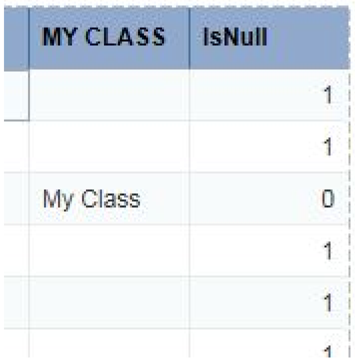
- date is an object [Object_name] containing only a date or a function which will output a date

Explanation: This function pulls the year or the number of the week out of the date.
Example: 1/01/2016

- Year will show "2016"
- Week will show "1" because it is the first week of the year

Logical

IsNull
Syntax: IsNull([Object_name])
Explanation: This function returns true ("1") or false ("0") based on whether or not the object selected is empty of all data. This can be particularly helpful in If-Then statements.
Example: =IsNull([MY CLASS])



MY CLASS	IsNull
	1
	1
My Class	0
	1
	1
	1

Note that the output is always 0 or 1, which is true (1) or false (0). All logical functions give this numbered output, though additional parts of the variable or other variables can change the ones and zeros into language.

Even
Syntax: Even([Object_name])
Explanation: This function will return true ("1") or false ("0") based on whether the object selected is an even number.

Odd
Syntax: Odd([Object_name])
Explanation: This function will return true ("1") or false ("0") based on whether the object selected is an odd number.

Document

DocumentAuthor
Syntax: DocumentAuthor()
Explanation: This will show the username of the document author.

DocumentName
Syntax: DocumentName()
Explanation: This function will show the name of the document.

Operators

Between, InList
Between and InList can select multiple values more easily. Both can be used in situations where the formula would include multiple "[Object_name]=value" statements would be strung together with ANDs.
Between Syntax: [Object_name] **Between(**first_value**;**last_value**)**

- first_value and last_value can be numbers, text in quotation marks or objects

Notes: Between includes the first value, the last value and everything in between. The first and last values must be separated by a semicolon with no spaces.
InList Syntax: [Object_name] **InList(**first_value**;**second_value**;**etc_value**)**

- first_value, second_value and etc_value can be numbers, text in quotation marks or objects

Notes: InList includes all of the values individually written out and each separated from the next by a semicolon and no spaces.
Example: [Number]=1 AND [Number]=2 AND [Number]=3 AND [Number]=5 AND [Number]=6
Using Between: [Number] Between (1;3) AND [Number] Between(5;6)
Using InList: [Number] InList(1;2;3;5;6)

Distinct, All
Syntax: e.g. Count([Object_name];**Distinct**)
Note: Distinct and All usually come after a semicolon within a function.
Explanation: The operators Distinct and All can be helpful in situations where a function must handle multiple rows with the same value.
Example: [Revenue] has the following values: 5, 5, 6 and 4.
Count([Revenue];Distinct) returns 3 because it has counted the 5 only once
Count([Revenue];All) returns 4 because it has counted the 5 twice

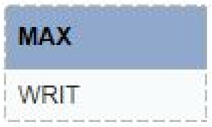
Contexts

Definition

A "context" is a command that can be given in a variable formula after some functions that tells the function to work on report data according to the limitations set by the context rather than the table structure. The context command includes a definition of the way the data should be grouped.

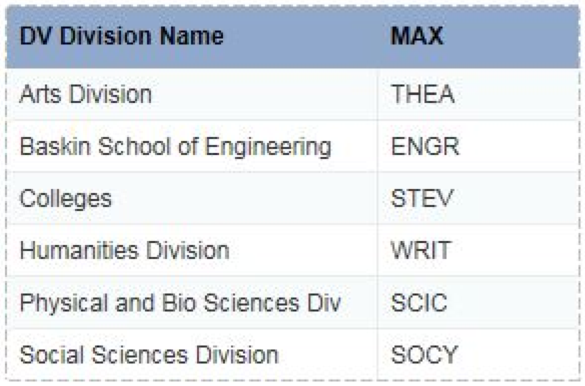
Example: Max Function
A formula looking for maximum values will return a different value depending on the structure of the table that it has been added to.

By itself, it will always return the biggest value for the whole report.
Shown here: the maximum department code (alphabetically last) in the entire report



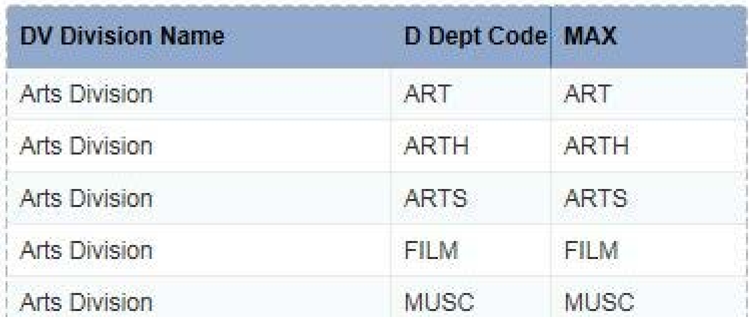
MAX
WRIT

When added to an existing table, each row will show the maximum value for each combination of data.
Shown here: the maximum department code per division



DV Division Name	MAX
Arts Division	THEA
Baskin School of Engineering	ENGR
Colleges	STEV
Humanities Division	WRIT
Physical and Bio Sciences Div	SCIC
Social Sciences Division	SOCY

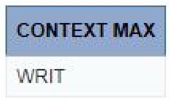
When added to a table containing the object being maxed, the max value will show the same value.
Shown here: the maximum department code per department code/division combination.



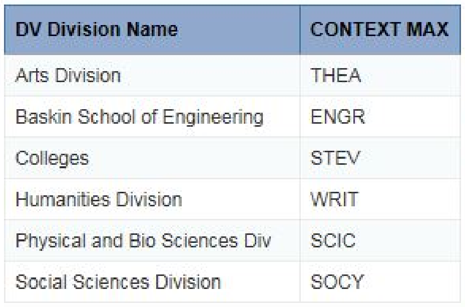
DV Division Name	D Dept Code	MAX
Arts Division	ART	ART
Arts Division	ARTH	ARTH
Arts Division	ARTS	ARTS
Arts Division	FILM	FILM
Arts Division	MUSC	MUSC

Adding a context will change the values given because the max function will ignore the specificity of the table structure.

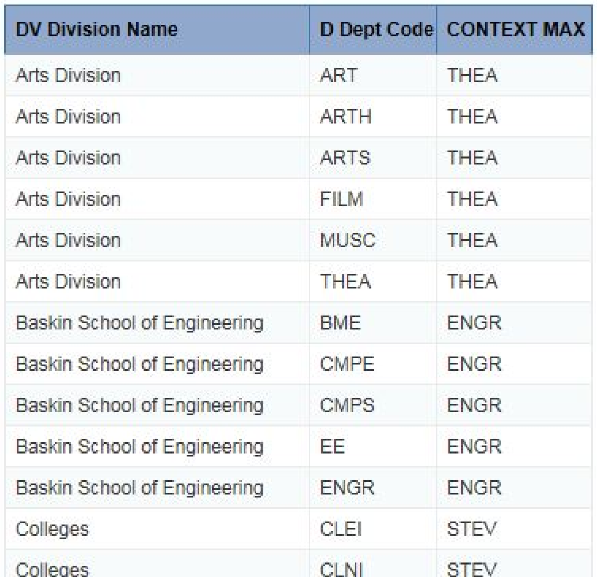
Here, those same tables with the context "DV Division Name" added (note how the last table looks different and shows the highest department code within each division):



CONTEXT MAX
WRIT



DV Division Name	CONTEXT MAX
Arts Division	THEA
Baskin School of Engineering	ENGR
Colleges	STEV
Humanities Division	WRIT
Physical and Bio Sciences Div	SCIC
Social Sciences Division	SOCY



DV Division Name	D Dept Code	CONTEXT MAX
Arts Division	ART	THEA
Arts Division	ARTH	THEA
Arts Division	ARTS	THEA
Arts Division	FILM	THEA
Arts Division	MUSC	THEA
Arts Division	THEA	THEA
Baskin School of Engineering	BME	ENGR
Baskin School of Engineering	CMPE	ENGR
Baskin School of Engineering	CMPS	ENGR
Baskin School of Engineering	EE	ENGR
Baskin School of Engineering	ENGR	ENGR
Colleges	CLEI	STEV
Colleges	CLNI	STEV

Contexts are notoriously slippery and difficult concepts to understand, but also only necessary in very specific situations. If you're having trouble with this one, don't hesitate to talk to us!

Context Commands

The functions "In", "ForEach" and "ForAll" all establish contexts. The function "Where" does not establish contexts, but uses a similar but more limited/precise mechanism.

In

In adds contexts to the function preceding it.

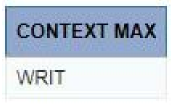
Syntax 1: function() **In(**[Object_name]**)**

Example: =Max([D Dept Code]) **In(**[DV Division Name]**)**

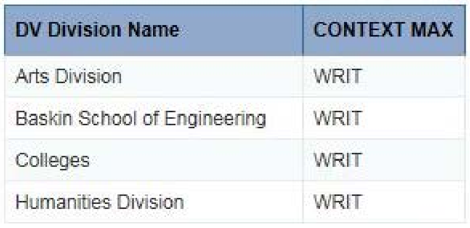
See screenshots above

Syntax 2: function() **In** Operator

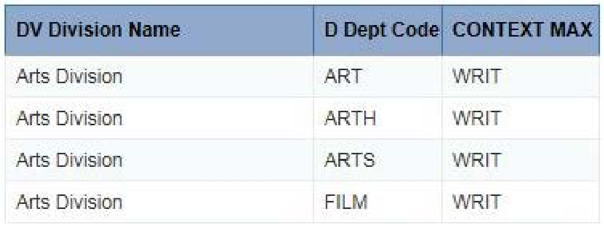
Example: =Max([D Dept Code]) **In** Report



CONTEXT MAX
WRIT



DV Division Name	CONTEXT MAX
Arts Division	WRIT
Baskin School of Engineering	WRIT
Colleges	WRIT
Humanities Division	WRIT



DV Division Name	D Dept Code	CONTEXT MAX
Arts Division	ART	WRIT
Arts Division	ARTH	WRIT
Arts Division	ARTS	WRIT
Arts Division	FILM	WRIT

ForEach and ForAll
ForEach and ForAll are opposites of each other. ForEach adds a dimension to a context modifying a calculation formula.
Syntax: function(**ForEach(**[Object_name]**)**)
Notice the double parenthesis ending!

Syntax: function(**ForAll(**[Object_name]**)**)
Notice the double parenthesis ending!

Where
Syntax: function() **Where(**condition_one**)**

Troubleshooting

Contexts are tricky and depend heavily on the report they are being used in.
Here are some suggestions:

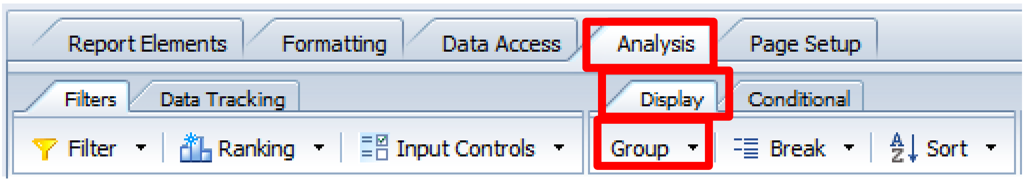
- Try changing the location of your context command: function() context() vs. function(context()). Sometimes moving it inside the function parentheses or outside the parentheses can make a difference.
- Try a different context command. ForEach and ForAll are especially tricky. Sometimes they overlap with In().
- Add more than one context to In() by adding a semicolon and another field In([Object_name1];[Object_name2])
- Use Operators like "Report" or "Document"
- Still having trouble? Come to our Open Labs! Or, contact us via phone 9-5083 or email data-mgmt@ucsc.edu and be ready to send us the report you're working on.

Extras

Grouping

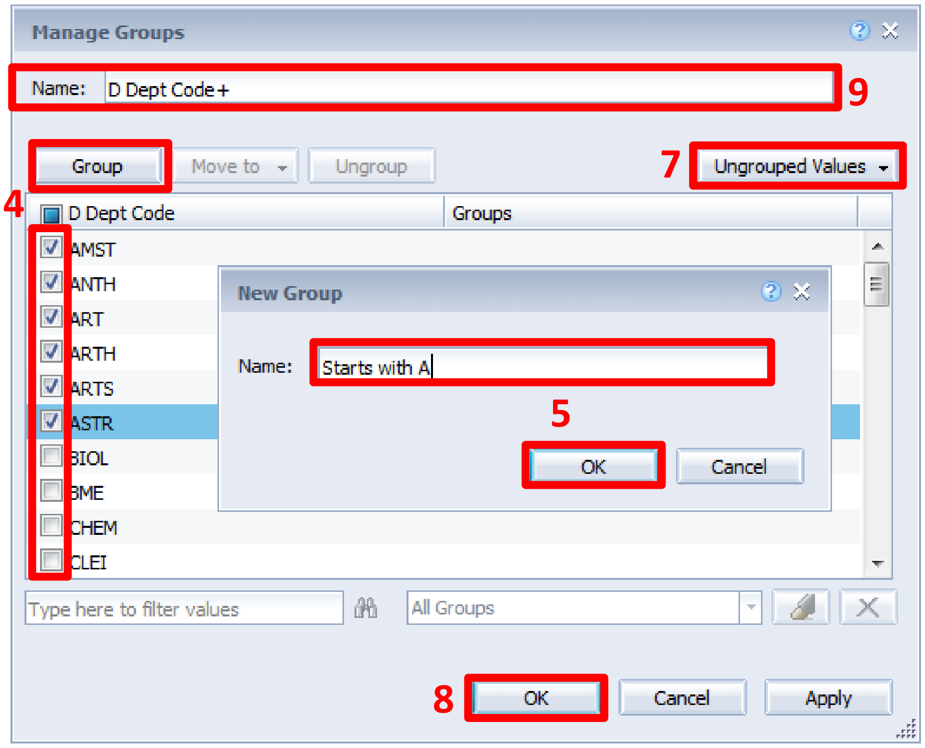
Definition
When grouping values, If/Then statements can get really lengthy and difficult to read. An alternative is to build a custom variable using the Analysis/Group feature.

How-To
1. Click on the content that you want to group
2. Navigate to the Analysis Tab, and the Display subtab.



3. Click "Group" or, click the small triangle next to the group button and select "Manage Groups".

4. In the Manage Groups pop-up window, click the check box next to the fields of the first group that you want to put together. Then click “Group”.



5. Name the group and click OK.
6. Complete step 4 and 5 for each group that you want to define.
7. If you have any values left over that are not grouped, you can group them all automatically by click on “Ungrouped Values” and selecting “Automatically Grouped”. You’ll be prompted to name this group, but it will default to “Others”.
8. If desired, change the name of the grouping rule (it will default to “field_name +”)
9. When you are satisfied with your name and the groups, click “OK”
10. Your grouped field will appear in the variables section under the name defined in step 8.
11. To change an existing rule, complete one of the following to get back to the Groups pop-up:

-Double click on the group label in your variables folder

-Navigate back to the small triangle next to the “Group” button and select “Manage Groups…”

12. To un group, navigate back to the “Group” button. Click on the small triangle next to the button and select “Ungroup” in the menu.

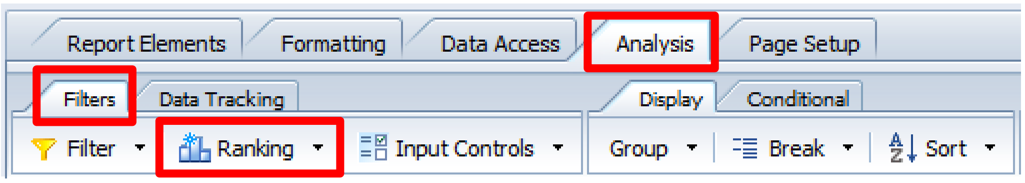
Ranking

Definition

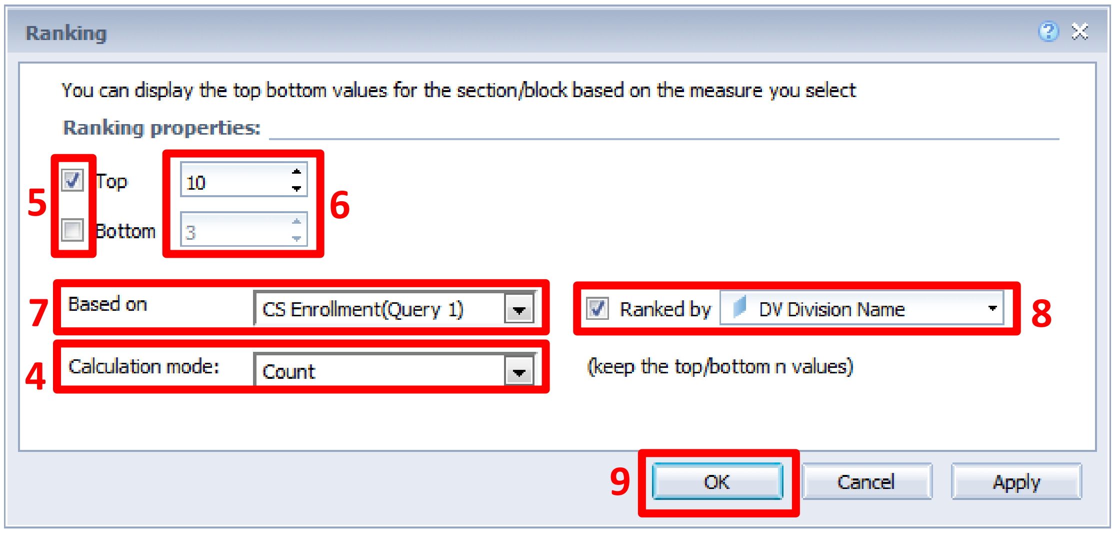
The Ranking button can be used to limit information to a “Top XXX” or “Bottom XXX” set based on an internal formula that is generated to set the ranking.

How-To

1. Click anywhere inside your table (not on the headers).
2. Navigate to the Analysis menu and the Filters submenu.
3. Select “Ranking” and then “Add New”.



4. In the ranking pop-up window, you can change the mechanism by which InfoView picks the top or bottom, if desired, by changing “Count” to something else, e.g. “percentage”.


Ranking

You can display the top bottom values for the section/block based on the measure you select

Ranking properties:

[✓] Top 10
[ ] Bottom 3

Based on: CS Enrollment(Query 1)

[✓] Ranked by: DV Division Name

Calculation mode: Count

(keep the top/bottom n values)

5. Next, decide if you want the Top or Bottom group.
6. Choose the number of rows (count), or the % number (percentage), that you would like to see.
7. Pick a field from the drop-down menu following “Based on”. The fields presented are all measures.
8. You can order your results by checking “Ranked by” and picking a field from your report.
9. Click OK and your table will limit itself accordingly.