

Oracle 11g BI Reports and Dashboards

# Column Formulas

**PEAK**  
indicators

# Column Formulas

## *Agenda*

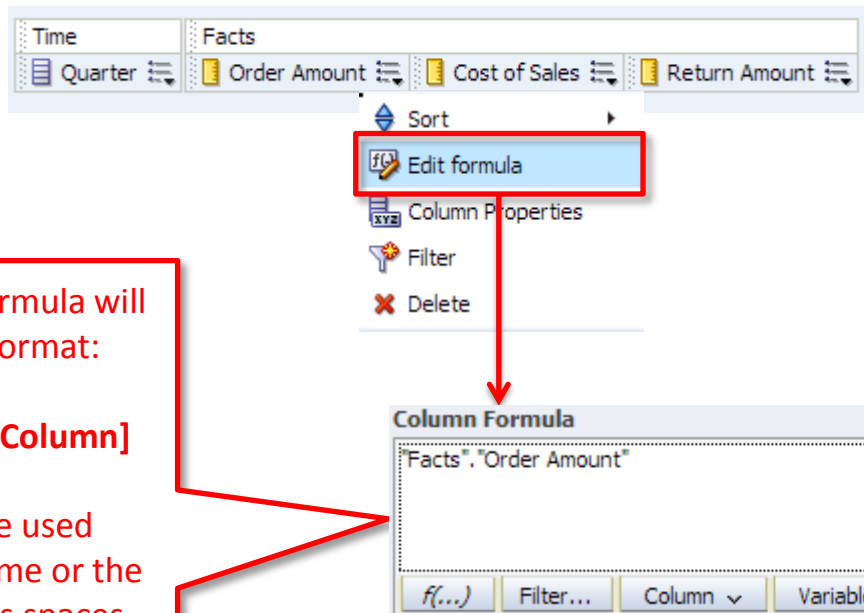
- **Column Formulas Overview**
  - Custom Headings
  - Editing Formulas
  - Adding Variables
  - Aggregation Rule
  - Functions Button
  - Further Notes
- **Examples**
  - CASE Expression
  - IFNULL Expression
  - Aggregate Functions
  - Aggregate BY Functions
  - Running Aggregate Functions
  - Aggregate by Dimension
  - Percentage
  - Filter
  - Bins
  - CAST (Conversion)
  - TimeStampDiff
  - TimeStampAdd
  - Time Series (ToDate, Ago, PeriodRolling)



## Column Formulas Overview

# Overview

- Sometimes it is necessary to perform additional calculations or formulas to the objects present on your request
- In order to achieve this, you can edit a column's formula by choosing the “Edit Formula” menu option on the “Criteria” tab:



By default, a column formula will be in the following format:

**[Presentation Table].[Column]**

“Double quotes” are used whenever the table name or the column name contains spaces

# Overview

## *Custom Headings*

- The “Edit Formula” options allow you to override the default Table / Column Heading
  - NOTE: This facility is exactly the same as the option available on the “Column Format tab within “Column Properties”

**Edit Column Formula**

**Column Formula** | Bins

Folder Heading: Facts

Column Heading: Order Amount (Dollers)

☒ Custom Headings

☐ Contains HTML Markup

Aggregation Rule (Totals Row): Default (Sum)

**Available**

**Subject Areas**

▼ Sales Orders


**Column Formula**

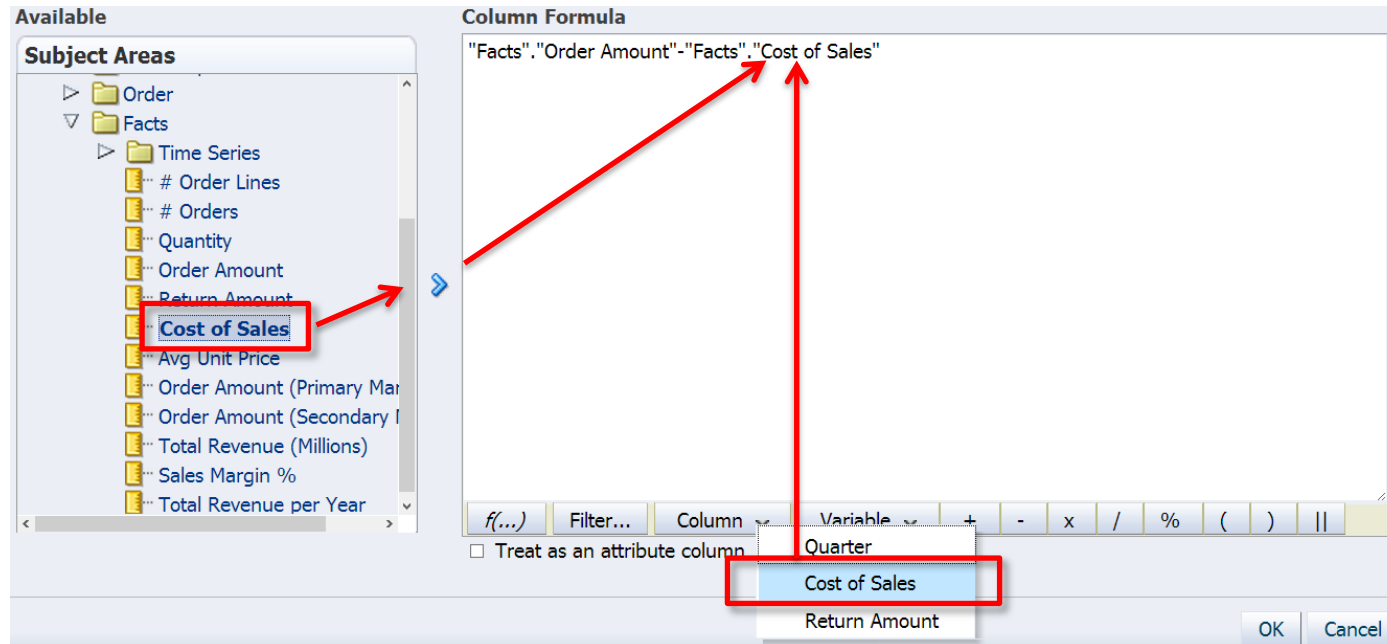
"Facts". "Order Amount"

Quarter	Order Amount	Cost of Sales	Return Amount
2007 Q 1	219,625,779	119,371,008	7,469,436
2007 Q 2	213,137,608	114,455,730	7,122,777
2007 Q 3	218,814,199	117,474,874	7,445,088
2007 Q 4	228,677,416	123,093,662	7,435,507
<b>Grand Total</b>	<b>880,255,002</b>	<b>474,395,273</b>	<b>29,472,808</b>

# Overview

## *Editing Formulas*

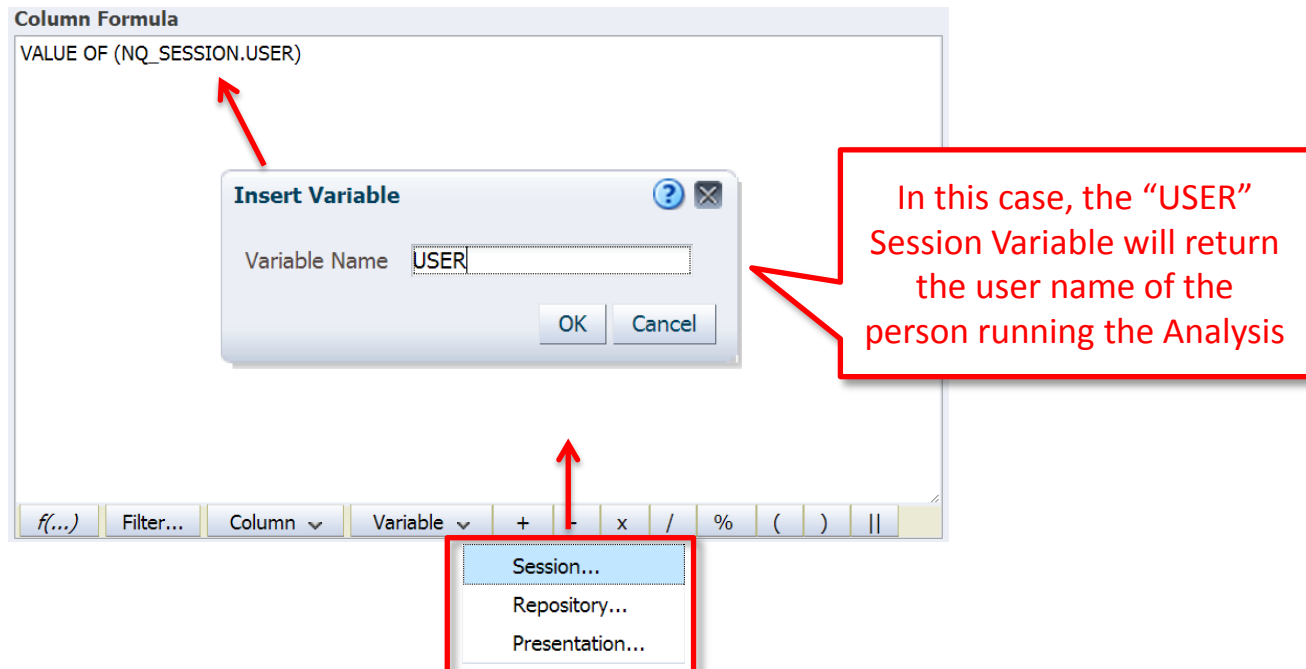
- It is possible to enter new expressions and formulas manually by typing.
- There are also shortcuts for adding new columns into the expression:
  - You can select a column within the Subject Area and then click on 
  - or
  - Use the “Column” button at the bottom to select columns already on the Analysis



# Overview

## *Adding Variables*

- The “Variables” button can be used to add Session, Repository or Presentation variables to your Analysis
- Variables can simplify your expressions and make them more dynamic (less hard-coding) by making use of pre-calculated variables such as “CURRENT\_YEAR”
  - NOTE: Your BI Developers are responsible for advising End Users on all the custom variables that have been created



# Overview

## *Adding Variables*

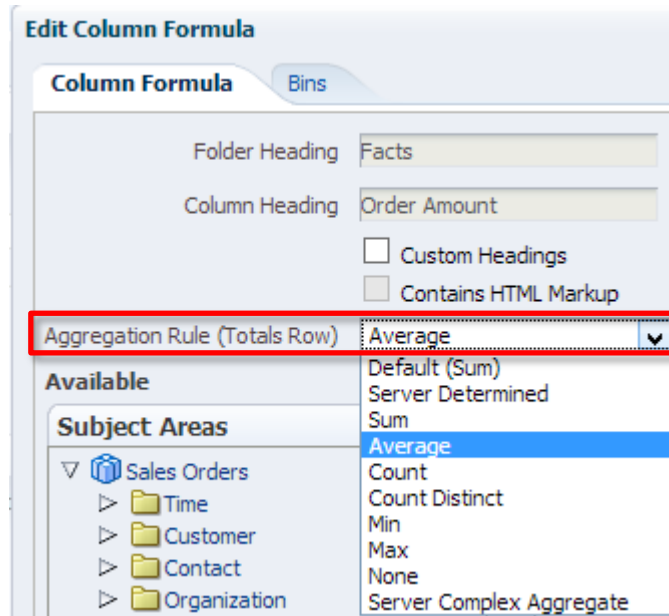
- There are 3 types of variable which can be created by Developers:
  - **Session Variables** are variables that are unique to each user's session. When a user logs in, a process takes place to “initialise” all their Session Variables. For example:
    - **USER:** Your OBIEE user name e.g. ASMITH
    - **MY\_ORG:** The name of your own specific Organization
  - **Repository Variables** are global variables where all users see the same value (they are not unique to each user's session). For example:
    - **YEAR\_AGO\_CAL:** The calendar date 1 year ago
    - **CURRENT\_YEAR:** The current year e.g. 2011
  - **Presentation Variables** are variables that can be created by report developers and be referenced in their reports. Users are able to change the value assigned to a Presentation Variable using a “Dashboard Prompt”. For example, a user could select the Year “2010” in a Dashboard Prompt and the text “Year 2010” would appear in the report's subtitle.
    - **NOTE:** Presentation Variables are covered in a later topic



# Overview

## Aggregation Rule

- You can override the default “Aggregation Rule” which determines how the grand total and sub totals are calculated



Quarter	Order Amount
2007 Q 1	219,625,779
2007 Q 2	213,137,608
2007 Q 3	218,814,199
2007 Q 4	228,677,416
<b>Grand Total</b>	<b>220,063,751</b>

Within the “Table” view editor, it is possible to change the “Grand Total” text to something different

- NOTE: The aggregation rule does not actually modify the values displayed within a table, it only affects the grand total and subtotal

# Overview

## *Server Complex Aggregate*

- Sometimes it is not possible or logical to calculate the grand total by summarising the values displayed within the Analysis
- If you find that grand total has not been calculated correctly, then you should try setting the aggregation rule to “**Server Complex Aggregate**”
- The “Server Complex Aggregate” setting will force OBIEE to issue a separate query to calculate the grand total, instead of trying to calculate it by summarising the end results returned the Analysis

Quarter	YTD Total Revenue (Millions)
2007 Q 1	213.08
2007 Q 2	419.77
2007 Q 3	632.15
2007 Q 4	853.90
<b>Grand Total</b>	<b>2118.90</b>

The grand total on the left for this “Year-to-Date” column is wrong.

It has been calculated simply by summing up the request’s values which is incorrect.

A “**Server Complex Aggregate**” aggregation rule as shown on the right can often fix this type of issue

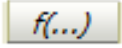
Quarter	YTD Total Revenue (Millions)
2007 Q 1	213.08
2007 Q 2	419.77
2007 Q 3	632.15
2007 Q 4	853.90
<b>Grand Total</b>	<b>853.90</b>

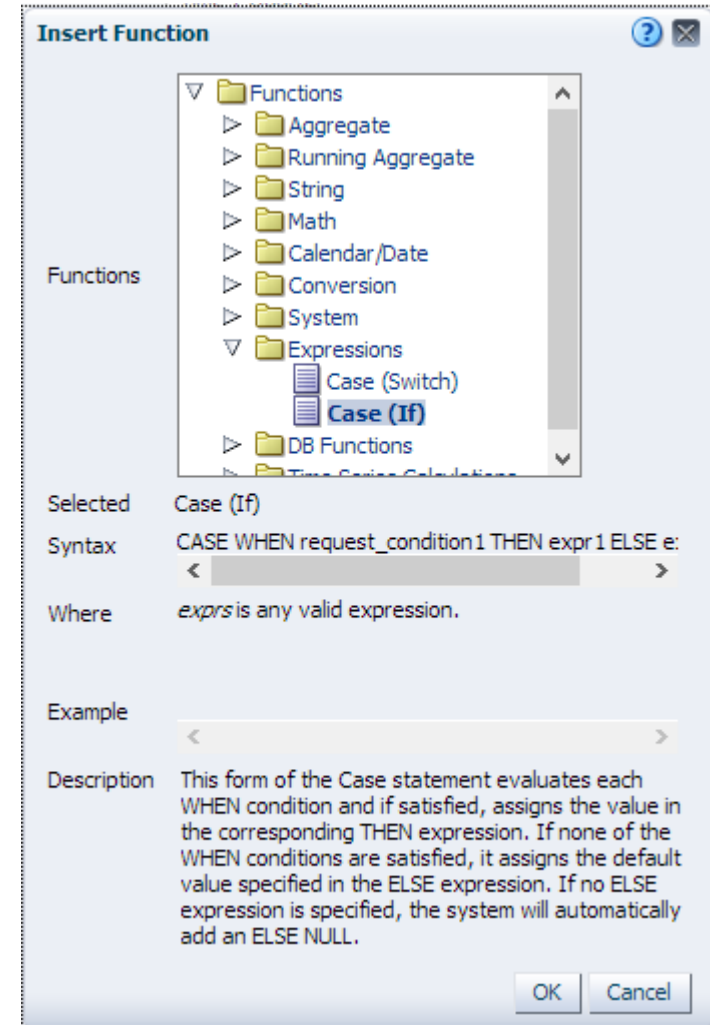
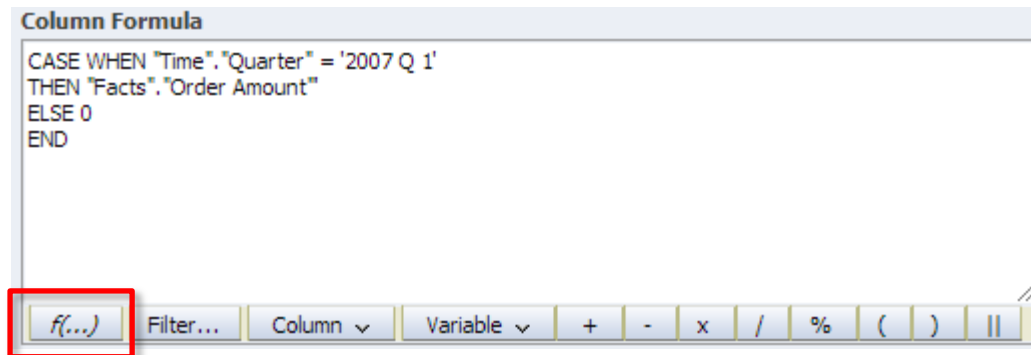
Aggregation Rule (Totals Row) Server Complex Aggregate ▼



# Overview

## Function Button

- Use the “**Function**”  button utility to help you build a variety of functions and expressions
  - There is an additional “Help” facility that provides even further details about each function available



# Overview

## *Further Notes*

- It is not possible to save your column formulas so that they can be reused across multiple reports:
  - This has the drawback of code duplication when the same formula needs to be used in multiple places
  - It also increases the maintenance overhead whenever the definition changes
- If you find that you are repeatedly building in the same calculations in to lots of Analyses then you should contact your development team to request that a new Subject Area column is provided that performs the same calculation
  - The calculation can therefore be reusable across all your Analyses without duplicating any code
- Finally, be aware that extra functions and formulas can impact the performance of database queries! Always consult your development team if you notice that your own custom formulas are impacting performance



## Column Formulas Examples

# Examples

## *CASE Expressions*

- **CASE** expressions are similar to IF-THEN-ELSE expressions and can be used to return a result if a condition is satisfied, or if it is not satisfied
- In the example below, we have built a CASE expression that implements the following logic:
  - IF "Year" = '2007'
  - THEN return 'Current Year'
  - ELSE return 'Previous Years'
  - END

### Column Formula

```
CASE WHEN "Time"."Year" = '2007'  
THEN 'Current Year'  
ELSE 'Previous Year'  
END
```



Year	Total Revenue per Year
Current Year	850.78
Previous Year	438.02
	610.54
	956.96
	908.50
	957.73
	865.02
<b>Grand Total</b>	<b>5587.57</b>

- **NOTE:** For CASE expressions to work effectively, the column on which the CASE expression is based (e.g. "Time". "Year") must be included in the Analysis

# Examples

## *IFNULL Expressions*

- The **IFNULL** function is useful if you need to convert NULL values on your Analysis to something else
- In the example below, we are converting “Customer Class Code” to ‘Unknown’ if the value returned is NULL

### Column Formula

```
IFNULL("Customer"."Customer Class Code", 'Unknown')
```



Customer Class Code
Utilities
Unknown
Telecom
RETAIL
PROCESS
Other
INTERNAL
High Technology
GOVERNMENT
Financial Services
FEDERAL
COMMERCIAL

# Examples

## Aggregates

- There are several “**aggregate**” functions available that will summarise the results within the column
- The example below shows a SUM() Function being used to return the total “Total Revenue (Millions)” value in every record

### Column Formula

SUM("Facts"."Total Revenue (Millions)")



Quarter	Total Revenue (Millions)	SUM(Total Revenue (Millions))
2007 Q 1	212.16	850.78
2007 Q 2	206.01	850.78
2007 Q 3	211.37	850.78
2007 Q 4	221.24	850.78
<b>Grand Total</b>	<b>850.78</b>	<b>850.78</b>

In this case, the “**Server Complex Aggregate**” aggregation rule was required to make sure the grand total was calculated correctly!



# Examples

## *Running Aggregates*

- **Running Aggregates** are useful for showing “running” counts or totals in your request
- The example below shows how the RSUM() function can be used to simulate a “Year-to-Date” type of calculation:

### Column Formula

RSUM("Facts"."Total Revenue (Millions)")



Quarter	Total Revenue (Millions)	YTD Total Revenue
2007 Q 1	212.16	212.16
2007 Q 2	206.01	418.17
2007 Q 3	211.37	629.54
2007 Q 4	221.24	850.78
<b>Grand Total</b>	<b>850.78</b>	<b>850.78</b>

# Examples

## *Aggregate By Dimension*

- Sometimes you need perform an aggregation “**by**” a specific dimension value
  - SUM(“Measure” BY “Dimension”)
- Consider the example below, we have multiple Years on the request, so in order to calculate the “Yearly Total”, we have to sum the measure “By Year” rather than calculate the sum for all years:

### Column Formula

SUM("Facts"."Total Revenue (Millions)" by "Time"."Year")



Year	Quarter	Total Revenue (Millions)	YTD Total Revenue
2007	2007 Q 1	212.16	850.78
	2007 Q 2	206.01	850.78
	2007 Q 3	211.37	850.78
	2007 Q 4	221.24	850.78
2007 Total		850.78	850.78
2006	2006 Q 1	190.26	865.02
	2006 Q 2	226.05	865.02
	2006 Q 3	207.48	865.02
	2006 Q 4	241.23	865.02
2006 Total		865.02	865.02



# Examples

## Percentage

- The example below shows a column formula returning a “percentage” calculation
- In this case, the percentage calculation is showing how the “Total Revenue (Millions)” in each month compares to the entire year
  - Where the “entire year” is being calculated using a SUM() aggregate function)

### Column Formula

```
"Facts"."Total Revenue (Millions)"  
/  
SUM("Facts"."Total Revenue (Millions)")  
*  
100
```



Year	Quarter	Total Revenue (Millions)	YTD Total Revenue
2007	2007 Q 1	212.16	24.9%
	2007 Q 2	206.01	24.2%
	2007 Q 3	211.37	24.8%
	2007 Q 4	221.24	26.0%
2007 Total		850.78	100.0%

# Examples

## Filter

- The “Filter” utility provides an easy mechanism to define Facts / Measures that are filtered on specific Dimension attributes
- The example below shows how to calculate “West Europe Revenue”, which is calculated by filtering “Total Revenue (Millions)” so that only revenue for the Business Group “Vision West Europe” is included:

**Column Formula**

```
FILTER("Facts"."Total Revenue (Millions)"  
USING ("Organization"."Business Group" = 'Vision West Europe'))
```

*f(...)* **Filter...** Column ▾ Variable ▾ + - x / % ( ) ||

Year	Quarter	Total Revenue (Millions)	West Europe Revenue
2007	2007 Q 1	212.16	16.11
	2007 Q 2	206.01	10.70
	2007 Q 3	211.37	11.20
	2007 Q 4	221.24	11.47
Grand Total		850.78	49.48

**“Filters”** are especially useful when the column you wish to filter on is not already included on the Analysis

# Examples

## Bins

- The “**Bins**” tab provides a simple utility for categorising dimension values
- In the example below, we are categorising sales into “East” and “West” Europe, based upon the “Business Group” column:

Column Formula	Bins
1. Business Group is equal to / is in Vision Benelux; Vision East Europe; Vision UK and Ireland	West
2. Business Group is equal to / is in Vision East Europe; Vision Nordics	East
3. All other values	Other

Business Group	Business Group	Total Revenue (Millions)
East	Vision East Europe	92.97
	Vision Nordics	274.76
West	Vision Benelux	417.15
	Vision UK and Ireland	16.42
	Vision West Europe	49.48
Grand Total		850.78

- Note that the use of Bins can significant increase maintenance if you use them in multiple places (since you cannot save/reuse their definitions) and it is harder to implement “drill downs” and “navigation”
- If you are using Bins in lots of places then you should contact your development team for advice, as it is possible to provide Subject Area columns that implement the same logic

# Examples

## CAST

- The CAST function is used to convert values to a different data type
- For example:
  - CAST( expression AS CHAR)
  - CAST( expression AS INTEGER)
  - CAST( expression AS DOUBLE)
  - CAST( expression AS DATE)
  - CAST( expression AS TIMESTAMP)

### Column Formula

```
CAST("Facts"."Order Amount"/1000 AS DOUBLE)
```

Because 1000 is an integer, we have to convert the final result into a “**DOUBLE**” precision number otherwise the result will also be an integer (i.e. no decimal places)

# Examples

## *TimeStampDiff*

- The “TimeStampDiff” function is used to display the difference between two timestamps. The format of the function is:
  - TimeStampDiff(SQL\_TSI\_[interval], ts1, ts2)
- Where SQL\_TSI\_[interval] states the time unit for displaying the result, it can be either:
  - SQL\_TSI\_SECOND SQL\_TSI\_MINUTE SQL\_TSI\_HOUR SQL\_TSI\_DAY
  - SQL\_TSI\_WEEK SQL\_TSI\_MONTH SQL\_TSI\_QUARTER SQL\_TSI\_YEAR
- In the example below, the “TimeStampDiff” function is being used to calculate how many days are in between an “Incident Date” and the “Current Date”:

### Column Formula

`TIMESTAMPDIFF(SQL_TSI_DAY, "Time"."Date", CURRENT_DATE)`



Incident Number	Date	Days Ago
1088	29/08/2001	4256
1209	15/08/2001	4270
1579	28/08/2001	4257
1628	29/08/2001	4256
2018	08/08/2001	4277
2047	14/08/2001	4271

# Examples

## *TimeStampAdd*

- The “TimeStampAdd” function will add a specified number of intervals to a timestamp. The format of the function is:
  - TimeStampAdd(SQL\_TSI\_[interval], intervals, ts1)
- Where SQL\_TSI\_[interval] states the time unit to be added, it can be either:
  - SQL\_TSI\_SECOND    SQL\_TSI\_MINUTE    SQL\_TSI\_HOUR    SQL\_TSI\_DAY
  - SQL\_TSI\_WEEK    SQL\_TSI\_MONTH    SQL\_TSI\_QUARTER    SQL\_TSI\_YEAR
- In the example below, the “TimeStampAdd” function is being used to add 3 months on to a date column:

### Column Formula

`TIMESTAMPADD(SQL_TSI_MONTH, 3, "Time"."Date")`



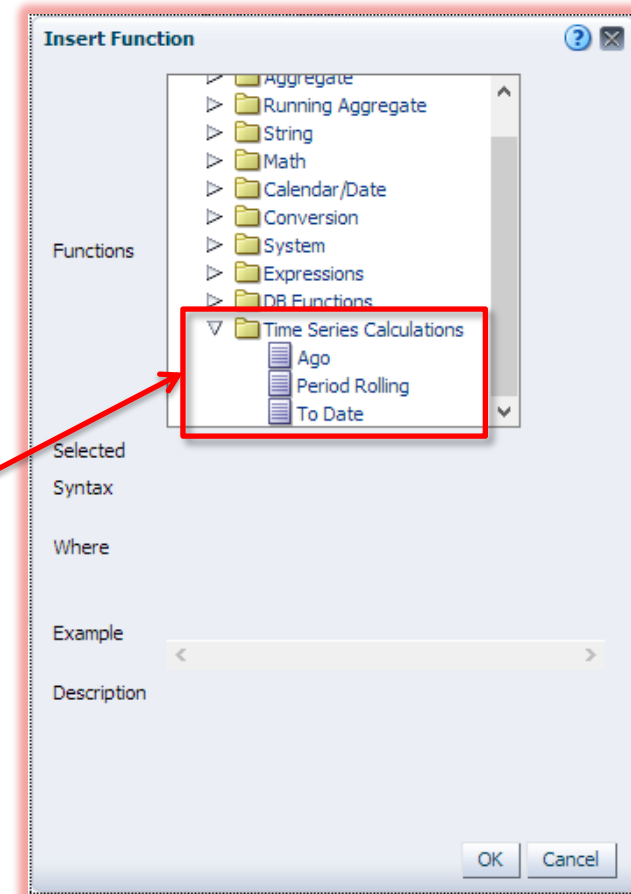
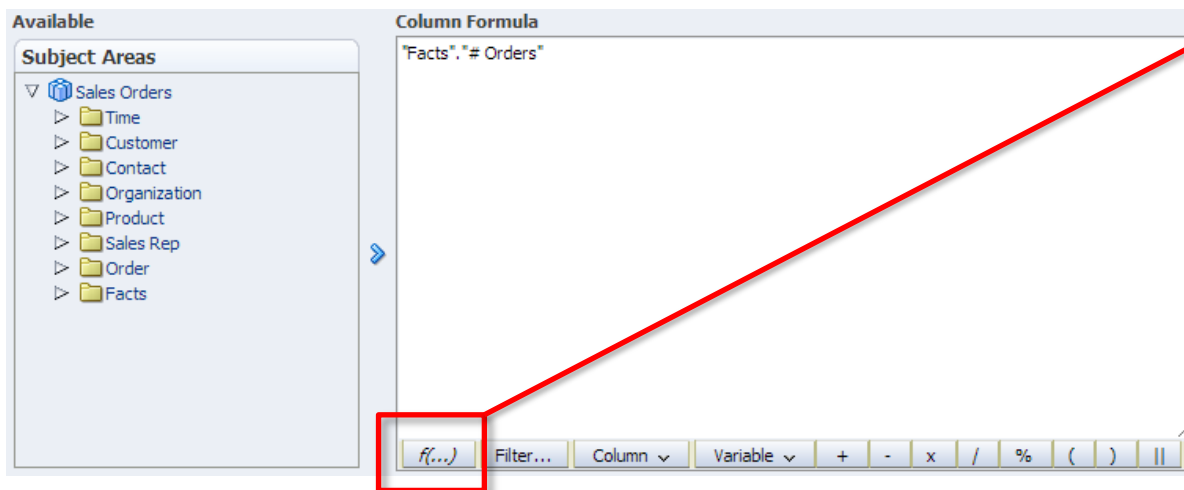
Incident Number	Date	Days + 3
1088	29/08/2000	29/11/2001
1209	15/08/2000	15/11/2001
1579	28/08/2000	28/11/2001
1628	29/08/2000	29/11/2001
2018	08/08/2000	08/11/2001



# Examples

## *Time Series*

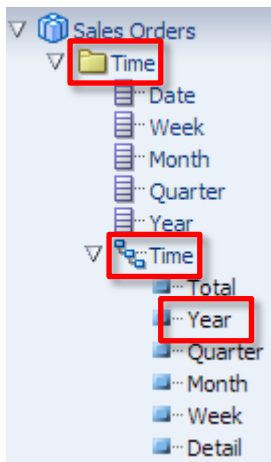
- There are also powerful “Time Series” functions available for report developers to use in their Analyses:
  - To Date
  - Ago
  - Period Rolling



# Examples

## ToDate

- The “**ToDate**” function provides Month/Week/Year-to-Date capability and has the following syntax:
  - `TODATE(expr, time_level)`
- Where the *time\_level* parameter should reference a level within a “Time” hierarchy which you can find in the Subject Area:
  - For example: Time.Time.Year



### Column Formula

`TODATE("Facts"."# Orders", Time.Time.Year)`

Year-to-Date  
for # Orders

Year	Month	# Orders	YTD # Orders
2007	2007 / 01	572	572
	2007 / 02	288	860
	2007 / 03	438	1298
	2007 / 04	437	1735
	2007 / 05	432	2167
	2007 / 06	477	2644
	2007 / 07	408	3052
	2007 / 08	461	3513
	2007 / 09	411	3924
	2007 / 10	461	4385
	2007 / 11	433	4818
	2007 / 12	448	5266

# Examples

## Ago

- The “**Ago**” function is used to deliver calculations such as “1 Year Ago # Orders”. It has the following syntax:
  - *AGO(expr, time\_level, number of periods)*
- Where the *time\_level* parameter should reference a level within a “Time” hierarchy which you can find in the Subject Area:
  - For example: Time.Time.Year



### Column Formula

AGO("Facts"."Total Revenue (Millions)", Time.Time.Year, 1)

1 Year-Ago  
Total Revenue

Year	Total Revenue (Millions)	Year Ago
2001	438.02	0.00
2002	610.54	438.02
2003	956.96	610.54
2004	908.50	0.00
2005	957.73	956.96
2006	865.02	957.73
2007	850.78	865.02
2008	0.00	850.78
<b>Grand Total</b>	<b>5587.57</b>	<b>4679.06</b>

# Examples

## PeriodRolling

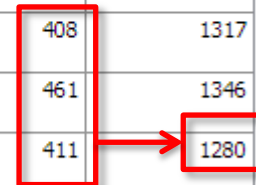
- The “**PeriodRolling**” function is used to deliver calculations such as “3 Months Rolling # Orders”. It has the following syntax:
  - PERIODROLLING(*expr*, *from\_period*, *to\_period*)
- In this example, a 3 month rolling sum is performed on “# Orders” (from -2 periods to the current period):

### Column Formula

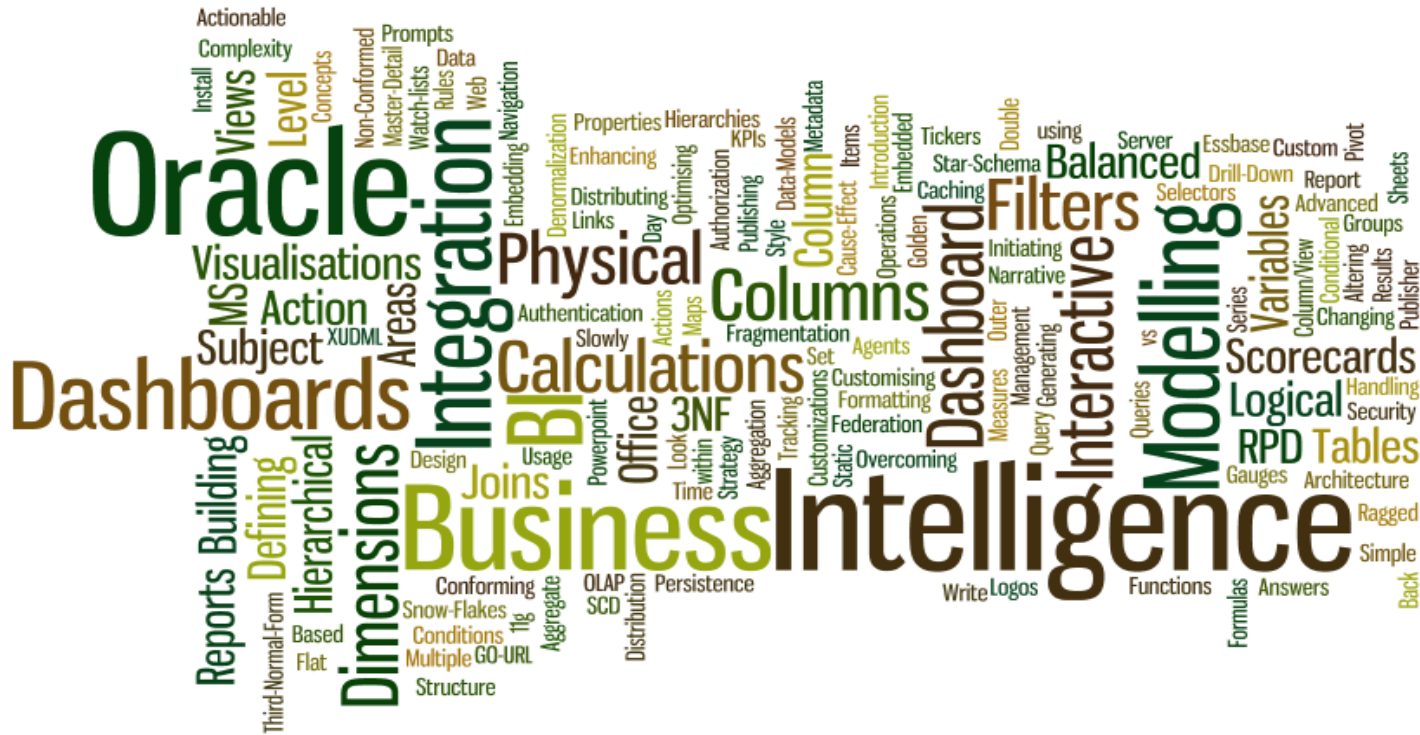
PERIODROLLING("Facts"."# Orders", -2,0)



Month	# Orders	3 Month Rolling
2007 / 01	572	1498
2007 / 02	288	1378
2007 / 03	438	1298
2007 / 04	437	1163
2007 / 05	432	1307
2007 / 06	477	1346
2007 / 07	408	1317
2007 / 08	461	1346
2007 / 09	411	1280
2007 / 10	461	1333
2007 / 11	433	1305
2007 / 12	448	1342



# Questions?



# PEAK

## indicators

**PEAK**  
indicators

Helping Your Business  
Intelligence Journey