IntelliPaat

# Tableau

Module 3

## Working with Calculations & Expressions

tableau®

# Agenda

## Working with Calculations & Expressions

- Calculation Syntax and Functions in Tableau
- Types of Calculations
- LOD Expressions (concept and syntax)
- Aggregation and Replication with LOD Expressions
- Nested LOD Expressions
- Level of Details
- Fixed Level of Details
- Lower Level of Details
- Higher Level of Details
- Quick Table Calculations
- How to create Calculated Fields?
- Predefined Calculations and their Validation

# Working with Calculations & Expressions

# Calculation Syntax & Functions in Tableau


**Tableau has multiple calculation types:**

- Basic Calculations
- Table Calculations
- Level of Detail Calculations

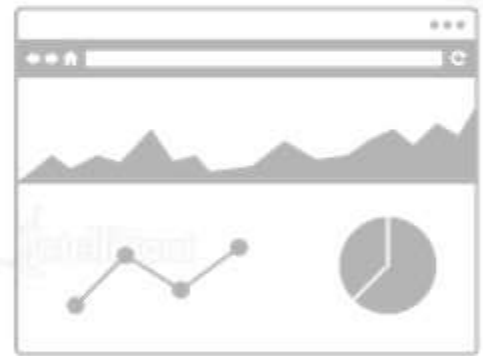
You will learn how to select the best calculation approach for different problem types.

# Types of Calculation

## Basic Expressions and LOD Expressions

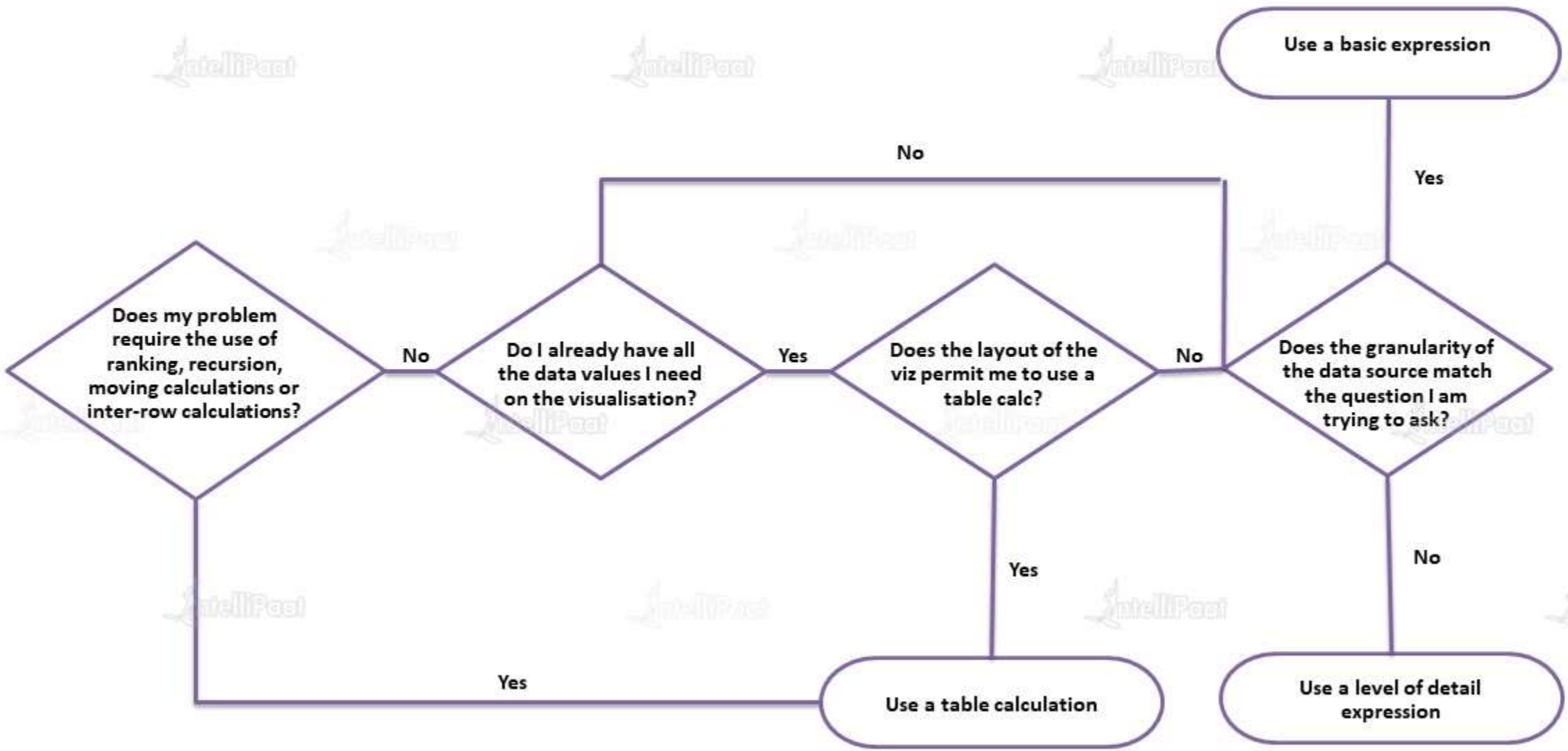
Calculated as part of the query on the underlying data

## Table Calculations



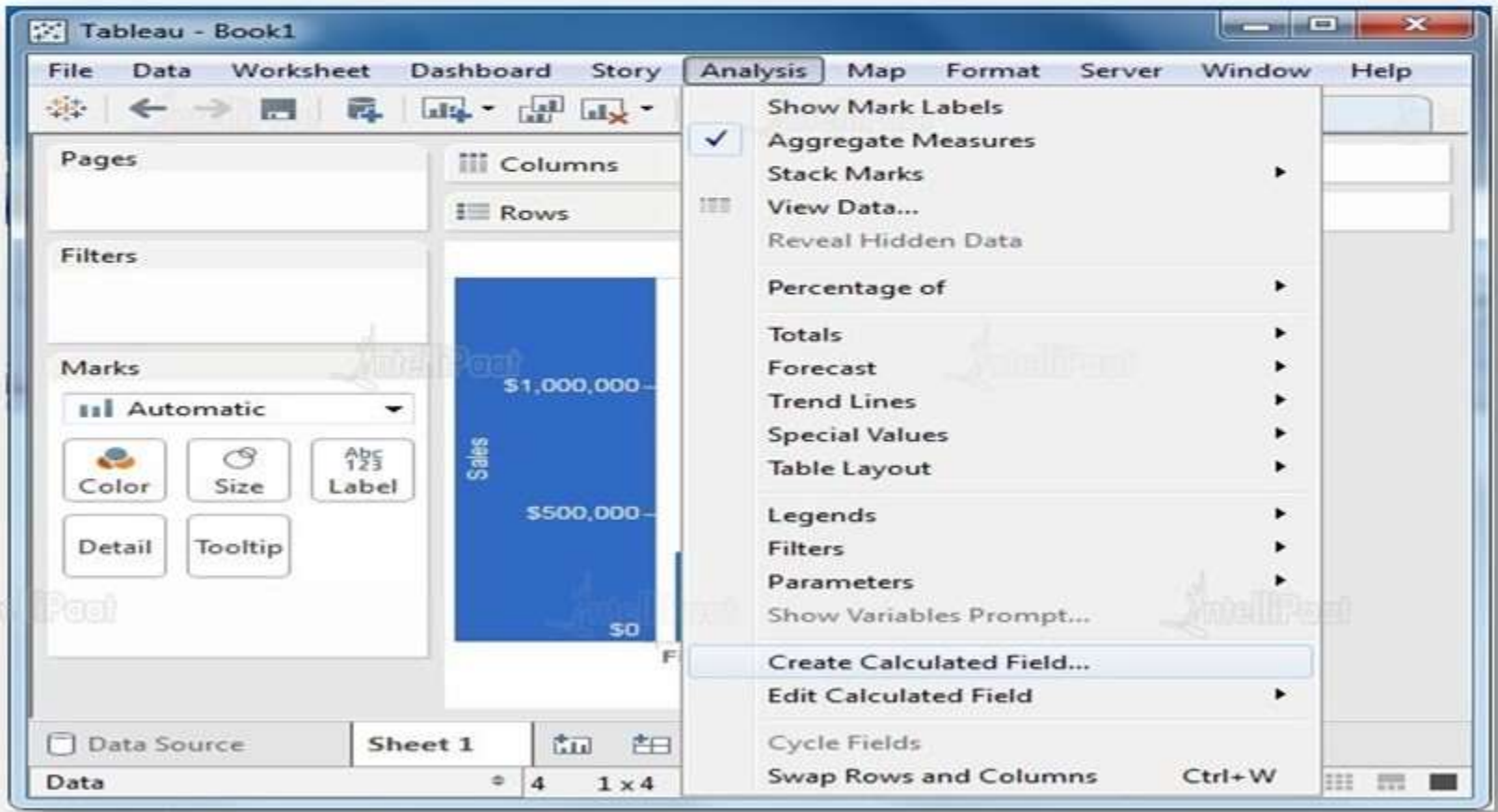
Calculated using the results from the query

# Workflow



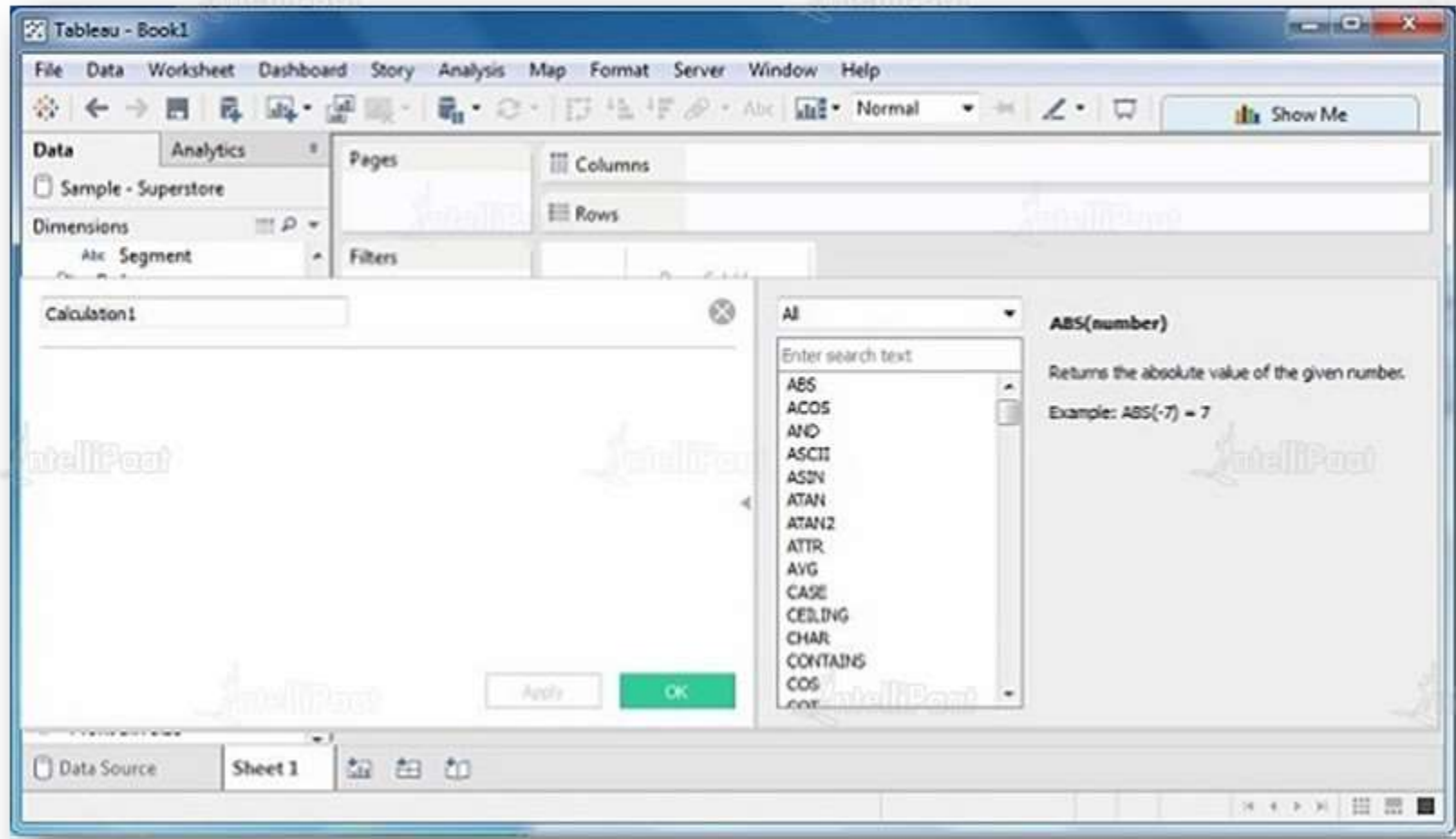
# Calculated Field in Tableau

- While connected to Sample - Superstore, go to the “Analysis” menu and click on the “Create Calculated Field”, as shown in the following screenshot.



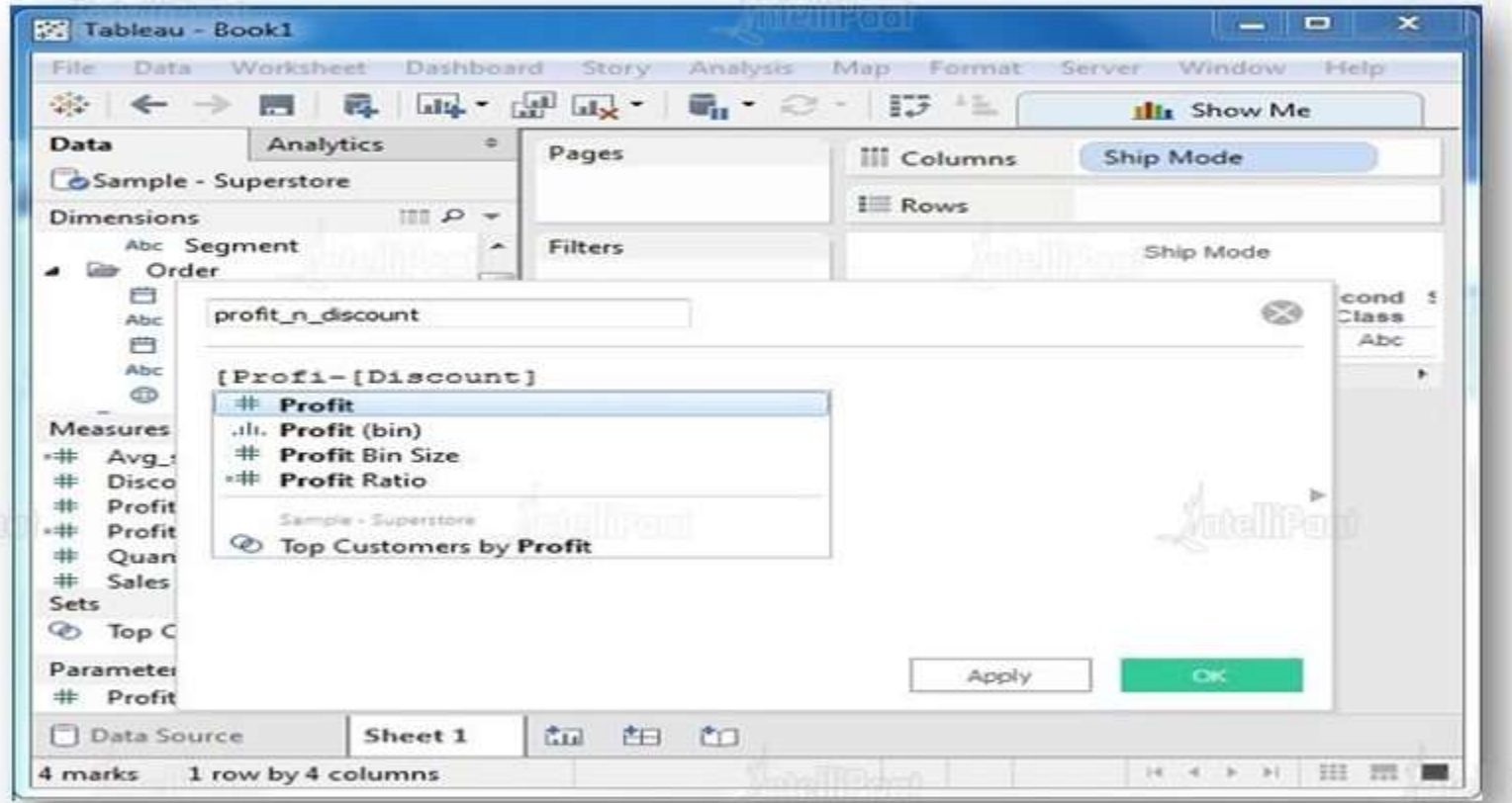
# Calculation Editor

- The previous step opens a calculation editor which lists all the functions that is available in Tableau. You can change the drop-down value and see only the functions related to numbers.



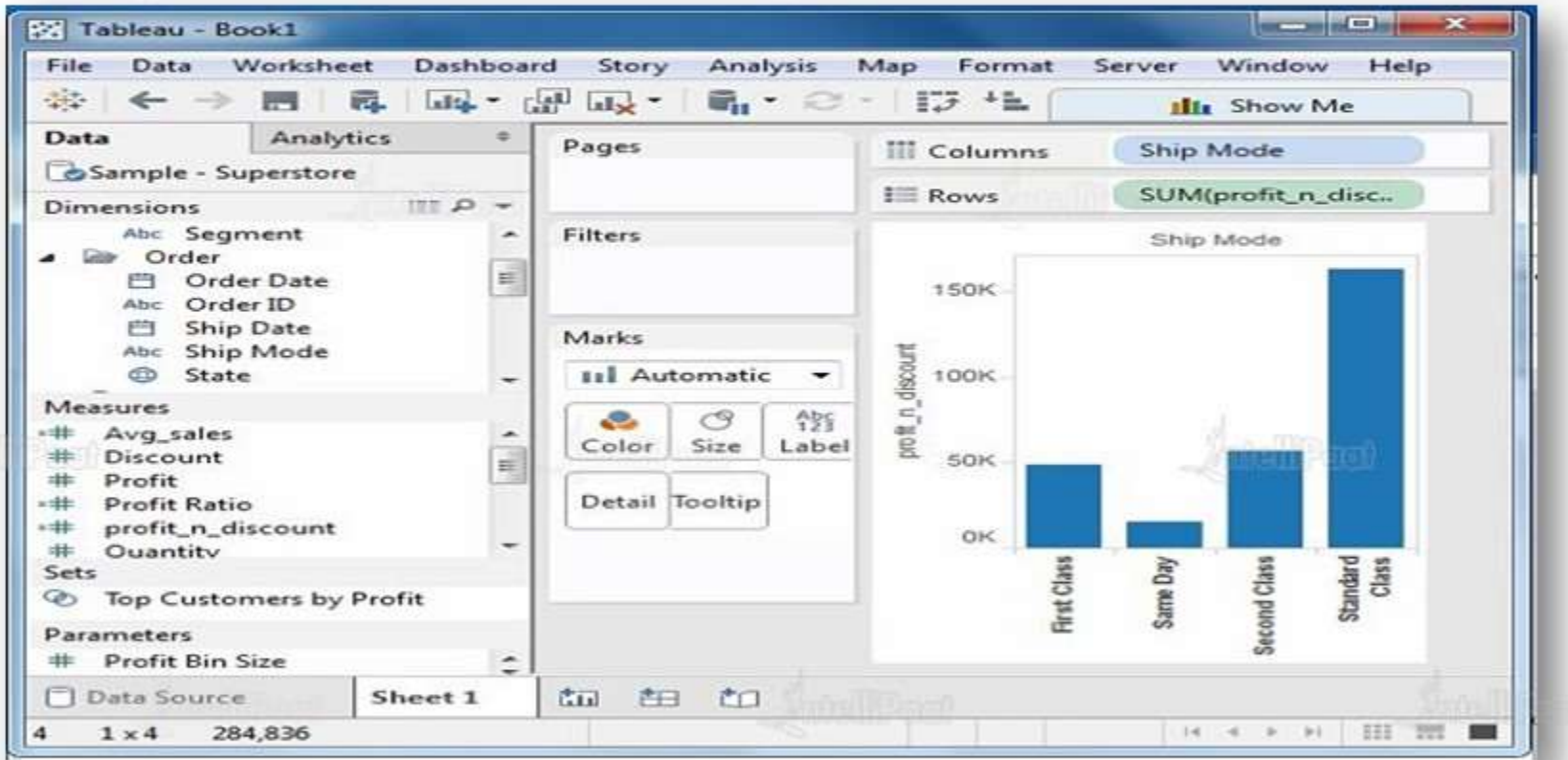
# Creating a Formula

- To study the difference between profit and discount for different shipping modes of products, create a formula subtracting the discount from the profit as shown in the screenshot. Also, name this field as **profit_n_discount**.



# Using the Calculated Field

- Aforementioned calculated field can be used in the view by dragging it to the Rows shelf as shown in the screenshot. It produces a bar chart showing the difference between profit and discount for different shipping modes.



# Level of Detail

- Tableau level of detail (LOD) expressions allow you to change the most granular place where an analysis takes place. When can we use LOD expressions?

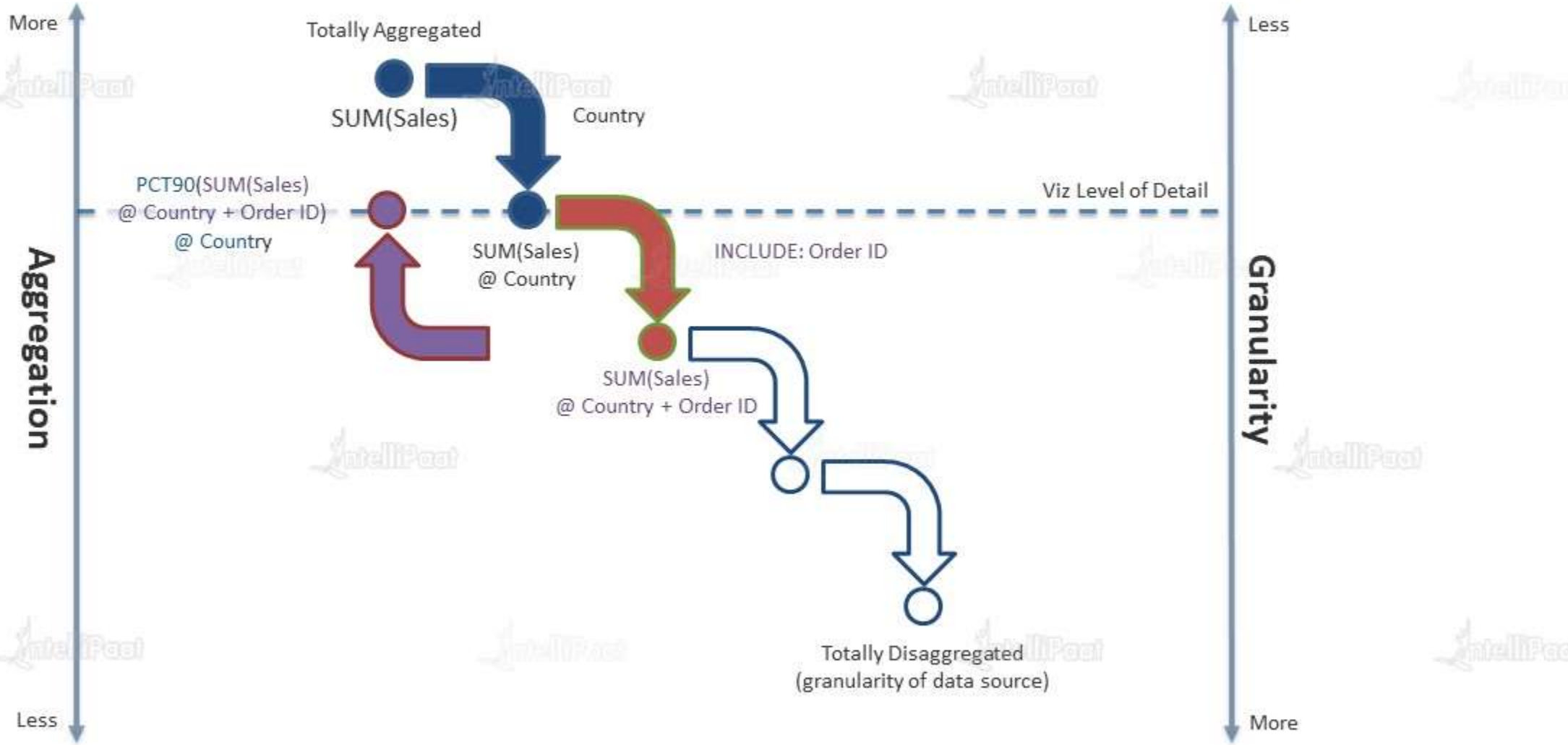
Consider using LOD expressions when:

- There is a requirement to show the data at a level different from the dimensions/level present in the view. There is a need to obtain some static calculated value that is not affected by any filters that are applied to the view.
- You need to compute things such as AVG(Sales) by State minus AVG(Sales) for the entire dataset to see how the sales per state compare to the overall average, with level of detail expressions in Tableau.

**Syntax:**

{FIXED/INCLUDE/EXCLUDE [dim1, dim2,...] : aggregate-expression}

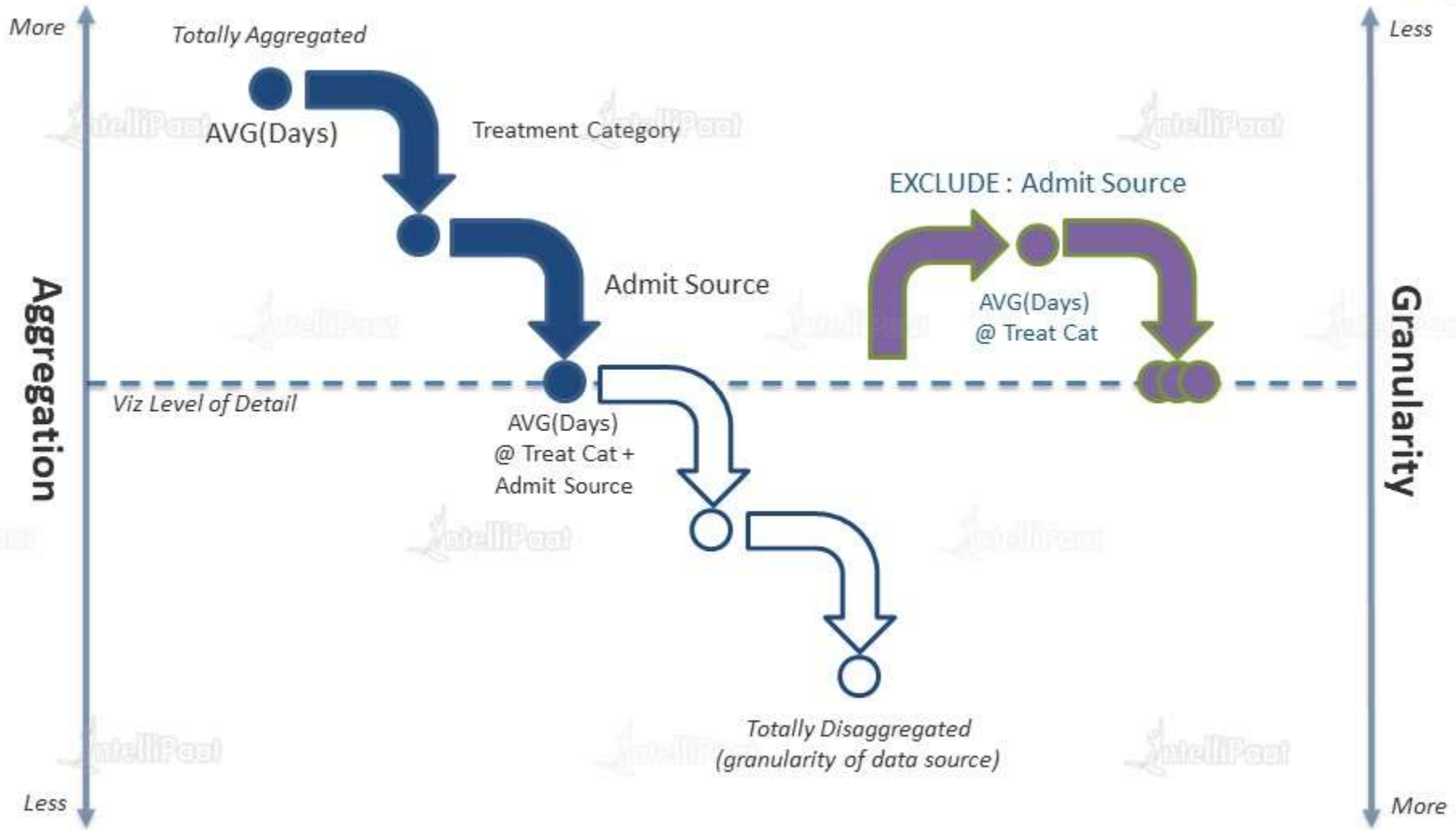
# Level of Detail



# AVG({INCLUDE [ORDER ID]: SUM(Profit)})

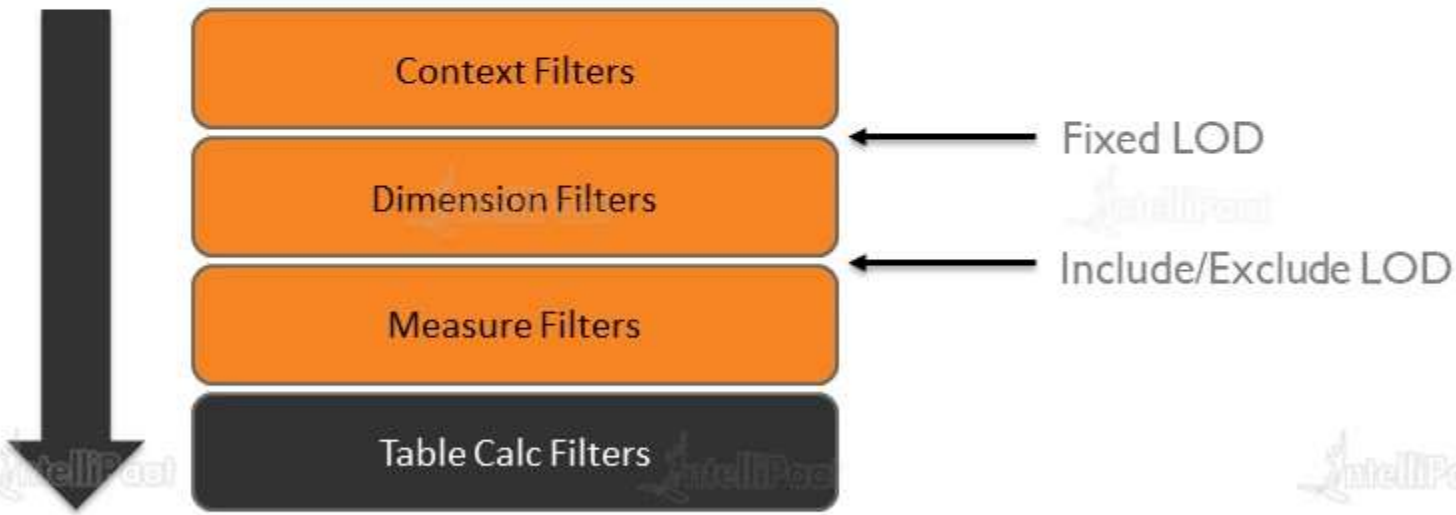
| Region | Country | Order ID | Row ID | |
|---|---|---|---|---|
| Central | Belgium | ES-2011-237.. | 130 | €309 |
| | | | 131 | €2,076 |
| | | | 132 | €525 |
| | | ES-2011-239.. | 2693 | €152 |
| | | | 2694 | €957 |
| | France | ES-2011-101.. | 6048 | €140 |
| | | | 6049 | €58 |
| | | ES-2011-108.. | 6777 | €76 |
| | | ES-2011-112.. | 4476 | €58 |
| | | ES-2011-113.. | 4456 | €599 |
| | Germany | ES-2011-100.. | 2531 | €229 |
| | | ES-2011-105.. | 9543 | €13 |
| | | | 9544 | €800 |
| | | | 9545 | €1,908 |
| | | | 9546 | €748 |
| North | Denmark | ES-2011-158.. | 4602 | €87 |
| | | ES-2011-450.. | 1068 | €82 |
| | | | 1069 | €25 |
| | | ES-2011-482.. | 626 | €26 |
| | | ES-2012-384.. | 6437 | €189 |
| | Norway | ES-2011-163.. | 5114 | €132 |
| | | | 5115 | €50 |
| | | ES-2011-181.. | 8639 | €27 |
| | | | 8640 | €38 |
| | | | 8641 | €92 |

- The SUM of profit is the result across the order IDs.
- The AVG is then the average of these order ID sum values per country.
- Multi-level aggregation!



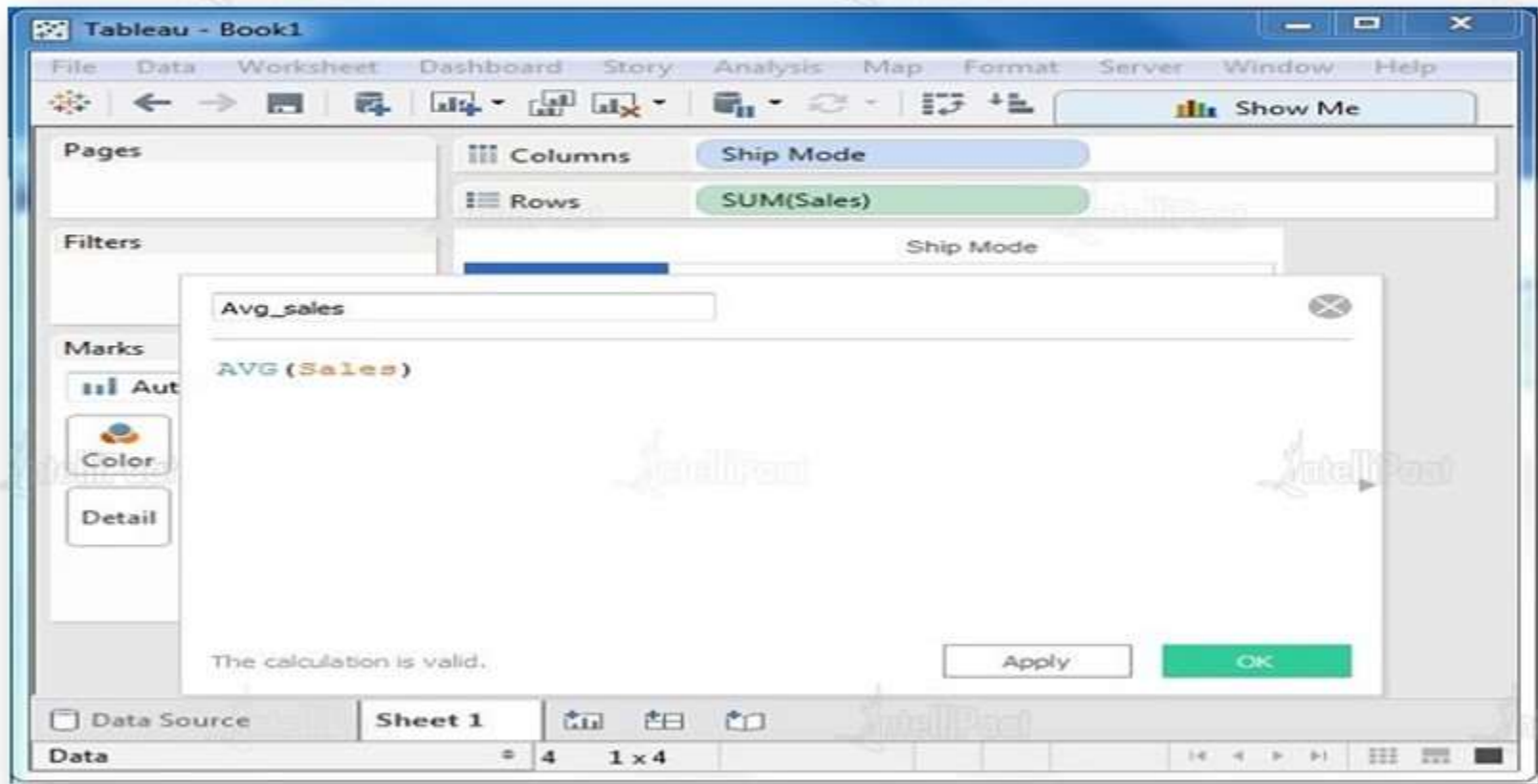
# Filtering Pipeline

- LOD expressions are part of the query, whereas table calculations are applied after the query.



# Applying Aggregate Calculations

- Create AVG(sales) values for different ship modes.
- Write the formula in the calculation editor as shown in the screenshot.



# DATEADD

DATEADD(date_part, interval, date)

Adds an increment to the date and returns a new date

## Example:

```
DATEADD('month', 3, #4/16/2014#)
```

**Returns 7/16/2014**

# DATEDIFF

DATEDIFF(date_part, date1, date2, [start_of_week])

Returns the difference between two dates, expressed in units of the date part

## Example:

```
DATEDIFF('year',#7/3/2011#,#5/27/2015#)
```

**Returns 4**

# TODAY

TODAY()

Returns the date of today

**Example:**

```
TODAY ( )
```

**Returns 06/02/2019**

# IF…THEN…END

IntelliPaat

IF <expr> THEN <then> ELSE <else> END

Tests a series of expressions returning the <then> value for the first true <expr>

## Example:

```
IF [Profit] > 0 THEN "Profitable" ELSE "Loss" END
```

# IF and Only IF

IIF(test, then, else)

Checks whether the condition is met and returns one value if TRUE and another value if FALSE

**Example:**

```
IIF([Profit] > 0, 'Profit', 'Loss')
```

# CONTAINS

CONTAINS(string, substring)

Returns True if the string contains the substring

**Example:**

```
CONTAINS("Texas", "Tex")
```

# LEFT

LEFT(string, num_chars)

Returns the specified number of characters from the start of a given string

**Example:**

```
LEFT("Visualization",3)
```

**Returns Vis**

# FIND

FIND(string, substring)

Returns the position of a substring within a string

**Example:**

```
FIND("Data Rockstar", "Rockstar")
```

**Returns 6**

# Which Aggregation?

ATTR()

MIN()

MAX()

COUNT()

COUNTD()

# Quick Table Calculations

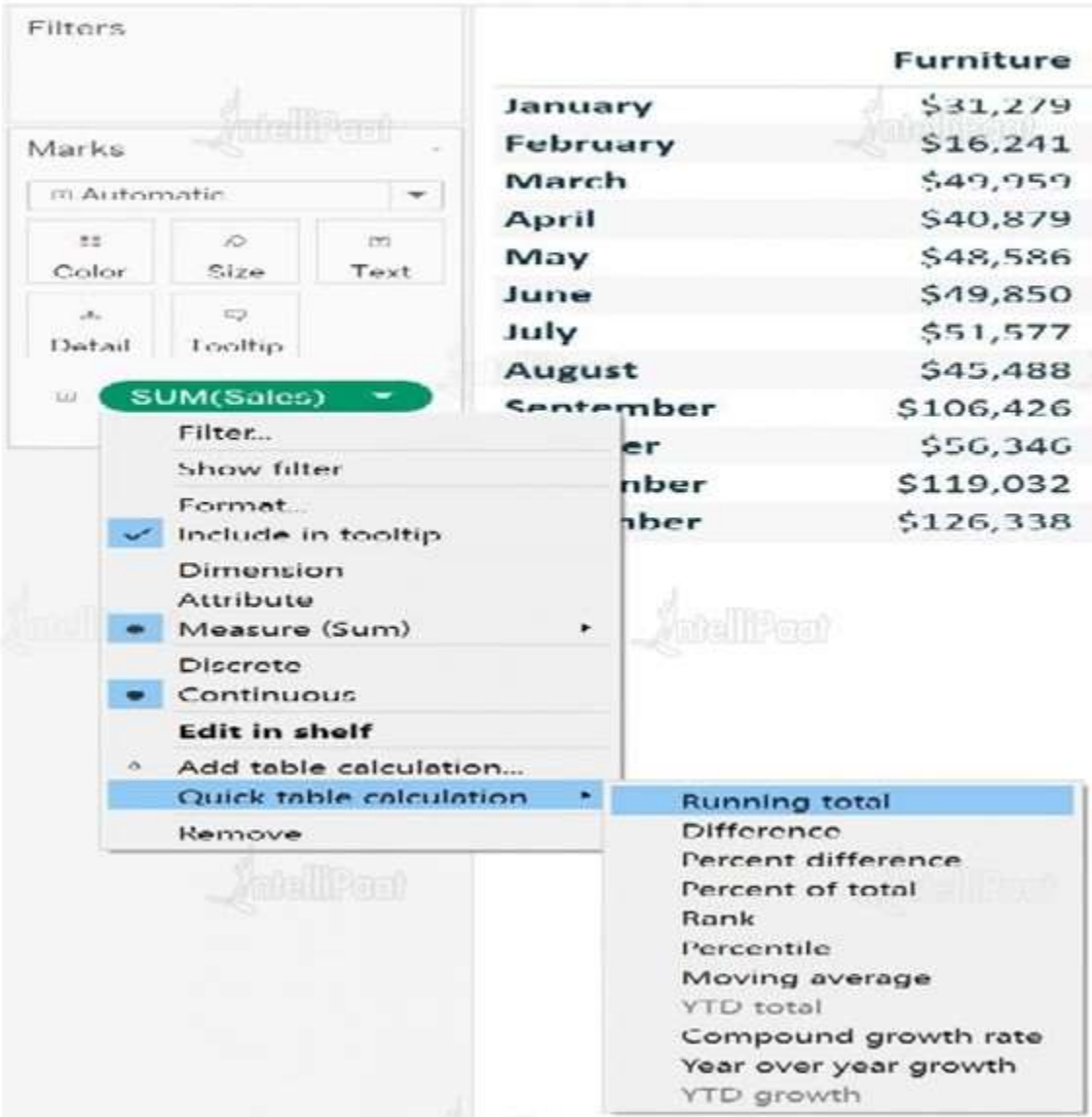
A quick table calculation is a one-step process where you choose a common table calculation type from a list. Tableau automatically applies the most typical settings for that calculation type.

**To apply a quick table calculation, do the following:**

Click on a Measure in the view and choose **Quick Table Calculation** from the Context menu.

Choose a Calculation type from the sub-menu.

# Quick Table Calculations




| | Category | | |
|---|---|---|---|
| | Furniture | Office Supplies | Technology |
| January | $31,279 | $33,663 | $30,424 |
| February | $16,241 | $20,572 | $23,359 |
| March | $49,959 | $56,205 | $93,088 |
| April | $40,879 | $49,275 | $51,697 |
| May | $48,586 | $43,394 | $64,142 |
| June | $49,850 | $49,800 | $47,433 |
| July | $51,577 | $43,217 | $54,787 |
| August | $45,488 | $62,307 | $51,795 |
| September | $106,426 | $102,774 | $100,570 |
| [illegible]er | $56,346 | $54,086 | $86,683 |
| [illegible]nber | $119,032 | $99,579 | $130,509 |
| [illegible]nber | $126,338 | $104,175 | $101,665 |

# Fixed Level of Details

LOD Expressions also provide an option to create an aggregation level completely independent of the Viz LOD.

Consider that you want to analyze YELP data to find the yearly cohorts in which a business had its first review. Does each cohort have the same review trends? With LOD expressions, we can specify the cohort at an exact level of detail:

# Predefined Calculations

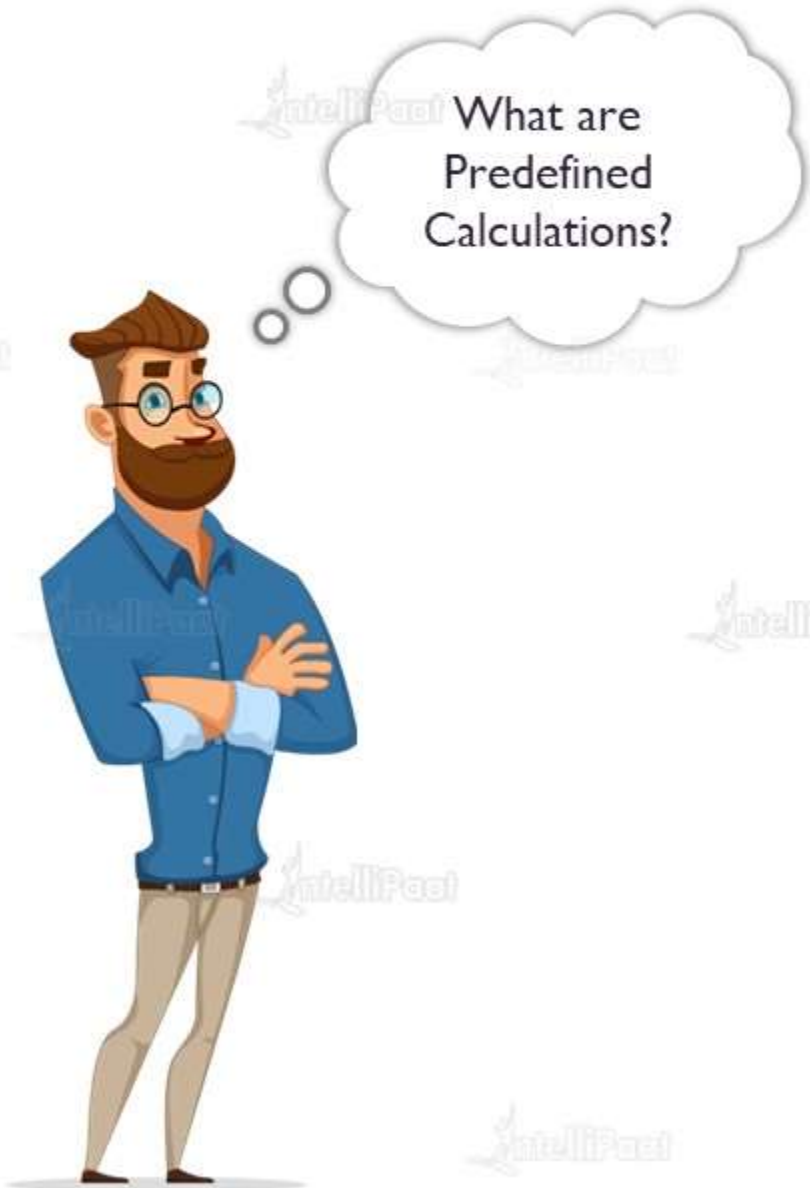



Tableau comes with several predefined calculations to compute with the numbers on a view including running total, difference, percent difference, percent of total, moving average and more. These predefined calculations are called table calculations because they compute the result based on a virtual table that includes only the numbers on the view.

# Running Total

Running total (RT) is adding each column to the total of all the preceding columns (to the left).

For example, the running sum of sales for office supplies = sum(sales) for furniture + sum(sales) for office supplies.

The RT of technology equals the total for all three categories. The values you get back for each category depend on the sort order and how the table calculation is computed.

RUNNING_SUM(SUM([Sales]))

The calculated field behind the running total uses the RUNNING_SUM function. If there are null values in your table, this is not an issue with this function as it just ignores them and continues the running total at the next non-null value.

You could edit this calculation by putting an If statement within the brackets:

e.g., RUNNING_SUM(if sum([Sales]) >100 then sum([Sales]) else 0 end)

Or, you could have a percent of total sales within the brackets (a secondary table calculation):

e.g., RUNNING_SUM(SUM([Sales])) / TOTAL(SUM([Sales]))

| | Furniture | Office Supplies | Technology |
|---|---|---|---|
| Sum (Sales) | 82,135 | 76,921 | 99,252 |
| Running Sum of Runni.. | 82,135 | 159,056 | 258,308 |

# Difference

Difference subtracts the current column from the previous column (depending on how the calculation is computed). The default quick table calculation is as follows:

ZN(SUM([Sales])) – LOOKUP(ZN(SUM([Sales])), -1)

| | Furniture | Office Supplies | Technology |
|---|---|---|---|
| Sum (Sales) | 82,135 | 76,921 | 99,252 |
| Difference (Sales) | | -5,214 | 22,330 |

Difference is bit more interesting as it contains ZN and LOOKUP in the calculation. The ZN stands for “Zero Null” and turns Null values into Zero.

If you do not have ZNs in the calculation, then the difference would be calculated as null if either the current or the previous value is null. This may be something that you need to play with inside the calculation, depending on what null means within your data.

LOOKUP is not to be confused with how the lookup functions work in Excel.

LOOKUP in Tableau looks at a relative position in the table, e.g., the previous row: LOOKUP(Sum([Sales]),-1) or two rows ahead: LOOKUP(SUM([Sales]),2).

# Percent Difference

- Percent difference takes the calculation from Difference (current row minus previous row) and divides by the absolute value of the previous row. The default quick table calculation is as follows:
- (ZN(SUM([Sales])) – LOOKUP(ZN(SUM([Sales])), -1)) / ABS(LOOKUP(ZN(SUM([Sales])), -1))

ABS() turns a number into an absolute value.

This means that the output number is positive whether the input number is positive or negative. This is required when calculating the percentage difference between negative values. Dividing a negative by a negative will give a positive which will be not show the correct percentage difference. In the chart below, I have compared the results you would get when running the calculation with or without the ABS function included:

| | Furniture | Office Supplies | Technology |
|---|---|---|---|
| Sum (Sales) | 82,135 | 76,921 | 99,252 |
| % Difference (Sales) | | -6.35% | 29.03% |

# Percent of Total

- Percent of total can be computed to calculate across rows, columns, panes, the whole table or specific dimensions.

- SUM([Sales]) / TOTAL(SUM([Sales]))

The TOTAL function totals all the values in a row, column, cell, table or specific dimension.

| | Furniture | Office Supplies | Technology |
|---|---|---|---|
| Sum (Sales) | 82,135 | 76,921 | 99,252 |
| % of Total (Sales) | 31.80% | 29.78% | 38.42% |

# Moving Average

When you add moving average as a quick calculation, the default is to take the average of the current value and the two previous values in the row (or however the table calculation is computed).

The default quick table calculation is as follows:

WINDOW_AVG(SUM([Sales]), -2, 0)

WINDOW_AVG(SUM([Sales]), [start], [end])

| | Furniture | Office Supplies | Technology |
|---|---|---|---|
| Sum (Sales) | 82,135 | 76,921 | 99,252 |
| Moving Average (Sales) | 82,135 | 79,528 | 86,103 |

The WINDOW_AVG function takes the average of all cells from the start reference to the end reference. In this case, it is taking the average of the values between column –2 (2 columns previous) and column 0 (the current column). You could edit this calculation by replacing with different start and end numbers, e.g., 6,0 to get a 7 day moving average if your table is aggregated by days. You could also replace the start and end references with other functions such as first() and last(). Last gives you the number of rows from the current row to the last row in the partition, whereas first gives you the number of rows from the current row to the first row. Using first in with window_avg or lookup means that any cell will be referencing the first cell in the partition, whereas using last will always reference the last cell in the partition.

# REGULAR EXPRESSIONS

- REGEXP_EXTRACT - This function lets one extract a particular pattern from a string variable.
- REGEXP_EXTRACT_NTH
- REGEXP_MATCH - This function is used for finding a repeated pattern in a string and returning a boolean.
- REGEXP_REPLACE - Replace a pattern with a set of characters.

Let us say we have the word "cat" that we want to extract from a string field.

- (cat) will extract the word cat and drop everything else in the string
- ^cat will extract only cat if its at the beginning of the string
- cat$ will extract cat only if its at the end of the string
- ^cat$ will only extract cat if it is alone in a string
- [cat] will only extract the specific characters c,a,t from a string
- [c-t] extracts any characters from c to t
- [^c-t] will extract any character except those between c to t
- [cat]+ will give back one or more characters that are c, a, or t
- (cat|dog) will extract the word cat OR the word dog
- (cat){2,4} extract cat when it is repeated 2 to 4 times
- c*t extracts any pattern that starts with c and ends with t
- ca?t extracts cat but will also extract ct
- ca\st will extract ca t
- cat\d{3} will extract cat012, cat111, cat356, or any pattern with cat and 3 digits afterwords.
- cat\w{3} is the same as above but with character values instead of numbers.
- cat. will match any character one time after cat. (i.e. catt, cata, cats,..)

## EXAMPLE WITH SAMPLE SUPERSTORE

Using Regexp() with the field OrderID

Order ID
CA-2017-152156

Pattern for orderid -> ([A-Z]+)-(\d+)-(\d+)

1)REGEXP_EXTRACT(OrderID,'-(\d+)-') - returns the numbers inbetween both the "-"

2)REGEXP_EXTRACT_NTH(OrderID,'([A-Z]+)-(\d+)-(\d+)',1) - return the 1st part of the pattern (CA)

3)REGEXP_MATCH(OrderID,'CA') - returns TRUE if the row contains 'CA' else FALSE

4)REGEXP_REPLACE(OrderID,'CA','RE') - replaces CA by RE

# Compute using options

TABLE ACROSS

TABLE DOWN

TABLE ACROSS THEN DOWN

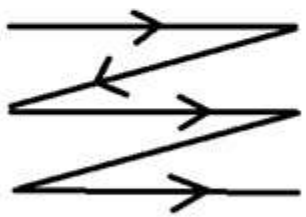
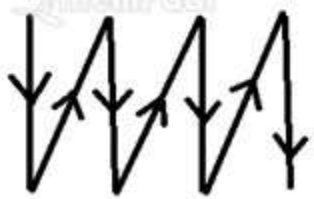


TABLE DOWN THEN ACROSS



# Thank You