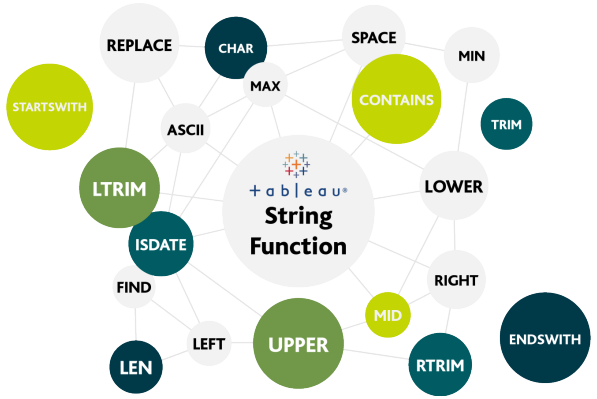
Data Visualization and Analytics with Tableau

PART 3: Calculated Fields

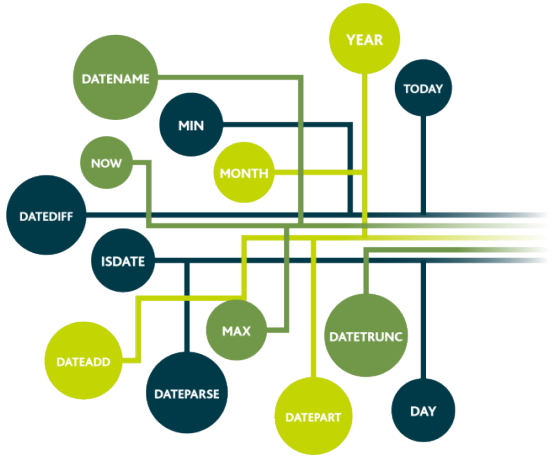
- Calculated fields allow the user to create new data from data that already exists in the data source.
- Creating a calculated field is essentially creating a new field (or column) in the data source, which cane used throughout the workbook (all worksheets).
- This will not affect the original data.

Arithmetic Functions

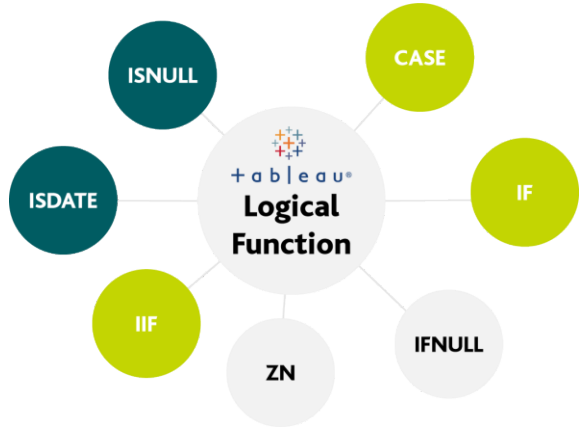
String Functions



Date Functions

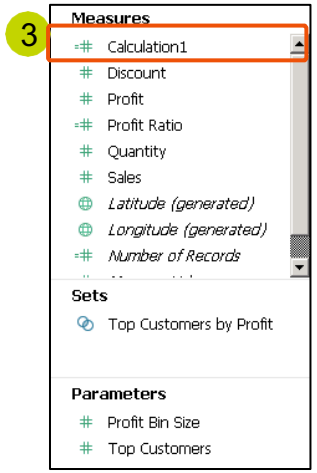


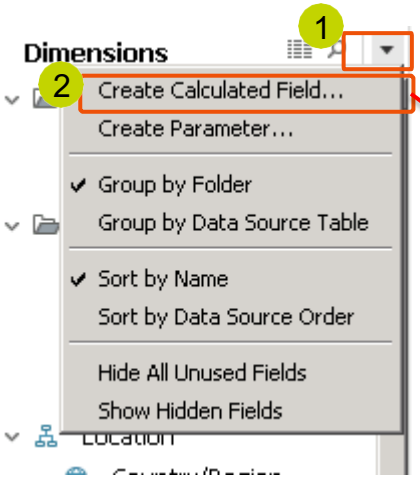
Logical Functions

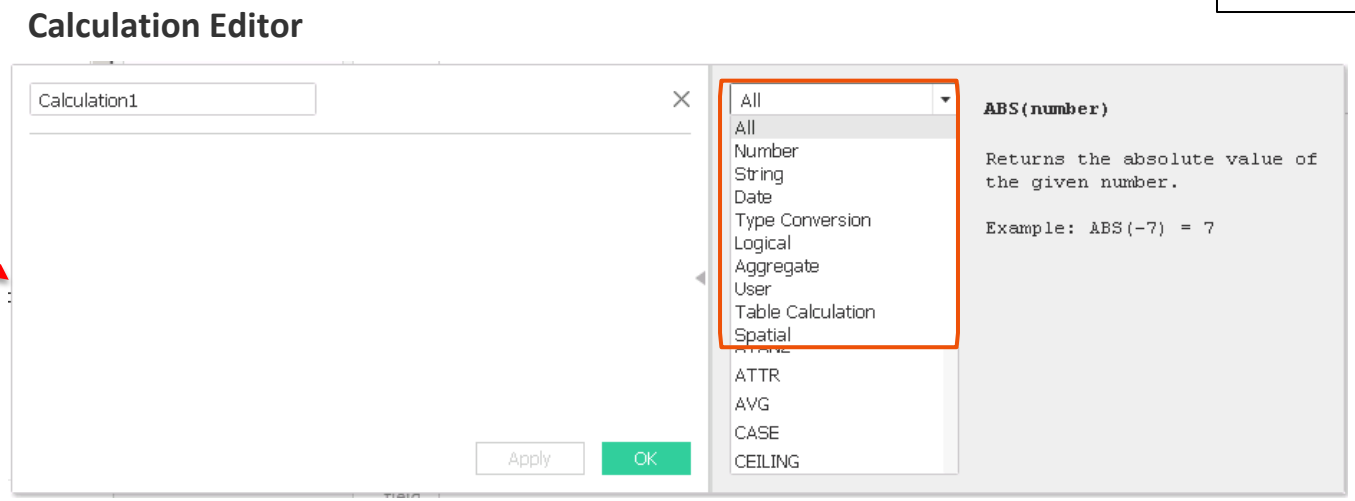


PART 3: Calculated Fields

1. Click on the downward facing triangle ▼ in the top right corner of the dimensions card.
2. Select "**Create Calculated Field**"
3. Once done, the calculated field will show in Measure/Dimension section.







To see a list of available functions, click the triangle icon on the right-side of the Calculation Editor.
Each function includes syntax, a description, and an example for your reference.
Double-click a function in the list to add it to the formula.

PART 3: Number Function

- Number functions can only be used with fields that contain numerical values.

Function	Syntax	Definition	Example
ABS	ABS(number)	Returns the absolute value of the given number.	**ABS(-7) = 7**
ACOS	ACOS(number)	Returns the arc cosine of the given number. The result is in radians.	**ACOS(-1) = 3.14159**
ASIN	ASIN(number)	Returns the arc sine of a given number. The result is in radians.	**ASIN(1) = 1.570796**
ATAN	ATAN(number)	Returns the arc tangent of a given number. The result is in radians.	**ATAN(180) = 1.56524**
ATAN2	ATAN2(y number, x number)	Returns the arc tangent of two given numbers (x and y). The result is in radians.	**ATAN2(2, 1) = 1.1071487**
CEILING	CEILING(number)	Rounds a number to the nearest integer of equal or greater value.	**CEILING(3.1415) = 4**
COS	COS(number)	Returns the cosine of an angle. Specify the angle in radians.	**COS(PI( ) /4) = 0.707106781186548**
COT	COT(number)	Returns the cotangent of an angle. Specify the angle in radians.	**COT(PI( ) /4) = 1**
DEGREES	DEGREES(number)	Converts a given number in radians to degrees.	**DEGREES(PI( )/4) = 45.0**
DIV	DIV(integer1, integer2)	Returns the integer part of a division operation, in which integer1 is divided by integer2.	**DIV(11,2) = 5**
EXP	EXP(number)	Returns e raised to the power of the given number.	**EXP(-[Growth Rate]*[Time])**
FLOOR	FLOOR(number)	Rounds a number to the nearest integer of equal or lesser value.	**FLOOR(3.1415) = 3**
HEXBINX	HEXBINX(number, number)	Maps an x, y coordinate to the x-coordinate of the nearest hexagonal bin.	**HEXBINX([Longitude], [Latitude])**
HEXBINY	HEXBINY(number, number)	Maps an x, y coordinate to the y-coordinate of the nearest hexagonal bin.	**HEXBINY([Longitude], [Latitude])**
LN	LN(number)	Returns the natural logarithm of a number. Returns Null if number is less than or equal to 0.	**LN(100) = 2**
LOG	LOG(number [, base])	Returns the logarithm of a number for the given base. If the base value is omitted, base 10 is used.	**LOG(100,2) = 6.643856**
MAX	MAX(number, number)	Returns the maximum of the two arguments, which must be of the same type.	**MAX(Sales,Profit)**
MIN	MIN(number, number)	Returns the minimum of the two arguments, which must be of the same type.	**MIN(Sales,Profit)**
PI	PI( )	Returns the numeric constant pi: 3.14159.	
POWER	POWER(number, power)	Raises the number to the specified power.	**POWER(5,2) = 25**
RADIANS	RADIANS(number)	Converts the given number from degrees to radians.	**RADIANS(180) = 3.14159**
ROUND	ROUND(number, [decimals])	Rounds numbers to a specified number of digits.	**ROUND(Sales,2)**
SIGN	SIGN(number)	The possible return values are -1 if the number is negative, 0 if the number is zero, or 1 if the number is positive.	**SIGN(AVG(Profit)) = -1**
SIN	SIN(number)	Returns the sine of an angle. Specify the angle in radians.	**SIN(PI( )/4) = 0.707106781186548**
SQRT	SQRT(number)	Returns the square root of a number.	**SQRT(25) = 5**
SQUARE	SQUARE(number)	Returns the square of a number.	**SQUARE(5) = 25**
TAN	TAN(number)	Returns the tangent of an angle. Specify the angle in radians..	**TAN(PI ( )/4) = 1.0**
ZN	ZN(expression)	Returns the expression if it is not null, otherwise returns zero. Use this function to use zero values instead of null values.	**ZN([Profit]) = [Profit]**

PART 3: String Function

- String functions allow you to manipulate string data (i.e. data made of text).

Function	Syntax	Definition	Example
ASCII	ASCII(string)	Returns the ASCII code for the first character of string.	**ASCII('A') = 65**
CHAR	CHAR(number)	Returns the character encoded by the ASCII code number.	**CHAR(65) = 'A'**
CONTAINS	CONTAINS(string, substring)	Returns true if the given string contains the specified substring.	**CONTAINS("Calculation", "alcu") = true**
ENDSWITH	ENDSWITH(string, substring)	Returns true if the given string ends with the specified substring. Trailing white spaces are ignored.	**ENDSWITH("Tableau", "leau") = true**
FIND	FIND(string, substring, [start])	Returns the index position of substring in string, or 0 if the substring isn't found.	**FIND("Calculation", "alcu") = 2**
FINDNTH	FINDNTH(string, substring, occurrence)	Returns the position of the nth occurrence of substring within the specified string.	**FINDNTH("Calculation", "a", 2) = 7**
LEFT	LEFT(string, number)	Returns the left-most number of characters in the string.	**LEFT("Matador", 4) = "Mata"**
LEN	LEN(string)	Returns the length of the string.	**LEN("Matador") = 7**
LOWER	LOWER(string)	Returns string, with all characters lowercase.	**LOWER("ProductVersion") = "productversion"**
LTRIM	LTRIM(string)	Returns the string with any leading spaces removed.	**LTRIM(" Matador ") = "Matador "**
MAX	MAX(a, b)	Returns the maximum of a and b (which must be of the same type).	**MAX ("Apple","Banana") = "Banana"**
MID	(MID(string, start, [length])	Returns the string starting at index position start. The first character in the string is position 1.	**MID("Calculation", 2, 5) ="alcul"**
MIN	MIN(a, b)	Returns the minimum of a and b (which must be of the same type).	**MIN ("Apple","Banana") = "Apple"**
REPLACE	REPLACE(string, substring, replacement)	Searches string for substring and replaces it with replacement. If substring is not found, the string is not changed.	**REPLACE("Version8.5", "8.5", "9.0") = "Version9.0"**
RIGHT	RIGHT(string, number)	Returns the right-most number of characters in string.	**RIGHT("Calculation", 4) = "tion"**
RTRIM	RTRIM(string)	Returns string with any trailing spaces removed.	**RTRIM(" Calculation ") = " Calculation"**
SPACE	SPACE(number)	Returns a string that is composed of the specified number of repeated spaces.	**SPACE(1) = " "**
SPLIT	SPLIT(string, delimiter, token number)	Returns a substring from a string, using a delimiter character to divide the string into a sequence of tokens.	**SPLIT ('a-b-c-d', '-', 2) = 'b'**
STARTSWITH	STARTSWITH(string, substring)	Returns true if string starts with substring. Leading white spaces are ignored.	**STARTSWITH("Joker", "Jo") = true**
TRIM	TRIM(string)	Returns the string with leading and trailing spaces removed.	**TRIM(" Calculation ") = "Calculation"**
UPPER	UPPER(string)	Returns string, with all characters uppercase.	**UPPER("Calculation") = "CALCULATION"**

PART 3: Date Function

- Date functions allow you to manipulate dates in data source.

Function	Syntax	Definition	Example
DATEADD	DATEADD(date_part, interval, date)	Returns the specified date with the specified number interval added to the specified date_part of that date.	**DATEADD('month', 3, #2004-04-15#) = 2004-07-15 12:00:00 AM**
DATEDIFF	DATEDIFF(date_part, date1, date2, [start_of_week])	Returns the difference between date1 and date2 expressed in units of date_part.	**DATEDIFF('week', #2013-09-22#, #2013-09-24#, 'monday')= 1**
DATENAME	DATENAME(date_part, date, [start_of_week])	Returns date_part of date as a string.	**DATENAME('year', #2004-04-15#) = "2004"**
DATEPART	DATEPART(date_part, date, [start_of_week])	Returns date_part of date as an integer.	**DATEPART('year', #2004-04-15#) = 2004**
DATETRUNC	DATETRUNC(date_part, date, [start_of_week])	Truncates the specified date to the accuracy specified by the date_part.	**DATETRUNC('quarter', #2004-08-15#) = 2004-07-01 12:00:00 AM**
DAY	DAY(date)	Returns the day of the given date as an integer.	**DAY(#2004-04-12#) = 12**
ISDATE	ISDATE(string)	Returns true if a given string is a valid date.	**ISDATE("April 15, 2004") = true**
MAKEDATE	MAKEDATE(year, month, day)	Returns a date value constructed from the specified year, month, and date.	**MAKEDATE(2004, 4, 15) = #April 15, 2004#**
MAKEDATETIME	MAKEDATETIME(date, time)	Combines a date and a time. The date can be a date, datetime, or a string type. The time must be a datetime.	**MAKEDATETIME("1899-12-30", #07:59:00#) = #12/30/1899 7:59:00 AM#**
MAKETIME	MAKETIME(hour, minute, second)	Returns a date value constructed from the specified hour, minute, and second.	**MAKETIME(14, 52, 40) = #14:52:40#**
MAX	MAX(expression) or MAX(expr1, expr2)	Returns the maximum of a and b (a and b must be of the same type).	**MAX([ShipDate1], [ShipDate2])**
MIN	MIN(expression) or MIN(expr1, expr2)	Returns the minimum of a and b (a and b must be of the same type).	**MIN([ShipDate1], [ShipDate2])**
MONTH	MONTH(date)	Returns the month of the given date as an integer.	**MONTH(#2004-04-15#) = 4**
NOW	NOW( )	Returns the current date and time.	
QUARTER	QUARTER ( )	Returns the quarter of the given date as an integer.	**QUARTER(#2004-04-15#) = 2**
TODAY	TODAY( )	Returns the current date.	**TODAY()**
WEEK	WEEK( )	Returns the week of the given date as an integer.	**WEEK (#2004-04-15#) = 16**
YEAR	YEAR (date)	Returns the year of the given date as an integer.	**YEAR(#2004-04-15#) = 2004**

PART 3: Logical Function

- Logical calculations allow you to determine if a certain condition is true or false (Boolean logic).

Function	Syntax	Definition	Example
IN	<expr1> IN <expr2>	Returns TRUE if any value in <expr1> matches any value in <expr2>.	**SUM([Cost]) IN (1000, 15, 200)**
AND	IF <expr1> AND <expr2> THEN <then> END	Performs a logical conjunction on two expressions.	**IF (ATTR([Market]) = "Africa" AND SUM([Sales]) > [Emerging Threshold])THEN "Well Performing"**
CASE	CASE <expression> WHEN <value1> THEN <return1> WHEN <value2> THEN <return2> ... ELSE <default return> END	Performs logical tests and returns appropriate values.	**CASE [Region] WHEN 'West' THEN 1 WHEN 'East' THEN 2 ELSE 3 END**
ELSE	IF <expr> THEN <then> ELSE <else> END	Tests a series of expressions returning the <then> value for the first true <expr>.	**If [Profit] > 0 THEN 'Profitable' ELSE 'Loss' END**
ELSEIF	IF <expr> THEN <then> [ELSEIF <expr2> THEN <then2>...] [ELSE <else>] END	Tests a series of expressions returning the <then> value for the first true <expr>.	**IF [Profit] > 0 THEN 'Profitable' ELSEIF [Profit] = 0 THEN 'Breakeven' ELSE 'Loss' END**
END	IF <expr> THEN <then> [ELSEIF <expr2> THEN <then2>...] [ELSE <else>] END	Tests a series of expressions returning the <then> value for the first true <expr>.	**IF [Profit] > 0 THEN 'Profitable' ELSEIF [Profit] = 0 THEN 'Breakeven' ELSE 'Loss' END**
IF	IF <expr> THEN <then> [ELSEIF <expr2> THEN <then2>...] [ELSE <else>] END	Tests a series of expressions returning the <then> value for the first true <expr>.	**IF [Profit] > 0 THEN 'Profitable' ELSEIF [Profit] = 0 THEN 'Breakeven' ELSE 'Loss' END**
IFNULL	IFNULL(expr1, expr2)	Returns <expr1> if it is not null, otherwise returns <expr2>.	**IFNULL([Profit], 0)**
IIF	IIF(test, then, else, [unknown])	Checks whether a condition is met, and returns one value if TRUE, another value if FALSE, and an optional third value or NULL if unknown.	**IIF([Profit] > 0, 'Profit', 'Loss')**
ISDATE	ISDATE(string)	Returns true if a given string is a valid date.	**ISDATE("2004-04-15") = True**
ISNULL	ISNULL(expression)	Returns true if the expression does not contain valid data (Null).	**ISNULL([Profit])**
MAX	MAX(expression) or Max(expr1, expr2)	Returns the maximum of a single expression across all records or the maximum of two expressions for each record.	**MAX([Sales])**
MIN	MIN(expression) or MIN(expr1, expr2)	Returns the minimum of an expression across all records or the minimum of two expressions for each record.	**MIN([Profit])**
NOT	IF NOT <expr> THEN <then> END	Performs logical negation on an expression.	**IF NOT [Profit] > 0 THEN "Unprofitable" END**
OR	IF <expr1> OR <expr2> THEN <then> END	Performs a logical disjunction on two expressions.	**IF [Profit] < 0 OR [Profit] = 0 THEN "Needs Improvement" END**
THEN	IF <expre> THEN <then> [ELSEIF ,expr2> THEN <then2>...] [ELSE <else>] END	Tests a series of expressions returning the <then> value for the first true <expr>.	**IF [Profit] > 0 THEN 'Profitable' ELSEIF [Profit] = 0 THEN 'Break even' ELSE 'unprofitable' END**
WHEN	CASE <expr> WHEN <Value1> THEN <return1> ... [ELSE <else>] END	Finds the first <value> that matches <expr> and returns the corresponding <return>.	**CASE [RomanNumberal] WHEN 'I' THEN 1 WHEN 'II' THEN 2 ELSE 3 END**
ZN	ZN(expression)	Returns <expression> if it is not null, otherwise returns zero.	**ZN([Profit])**
IN	<expr1> IN <expr2>	Returns TRUE if any value in <expr1> matches any value in <expr2>.	**SUM([Cost]) IN (1000, 15, 200)**

PART 3: Master Tableau Calculations

1 Basic Calculations

Row-level calculations/Aggregate calculation:
- Arithmetic calculation
- Logical operators
- String calculations
- Date calculations

2 Table Calculations

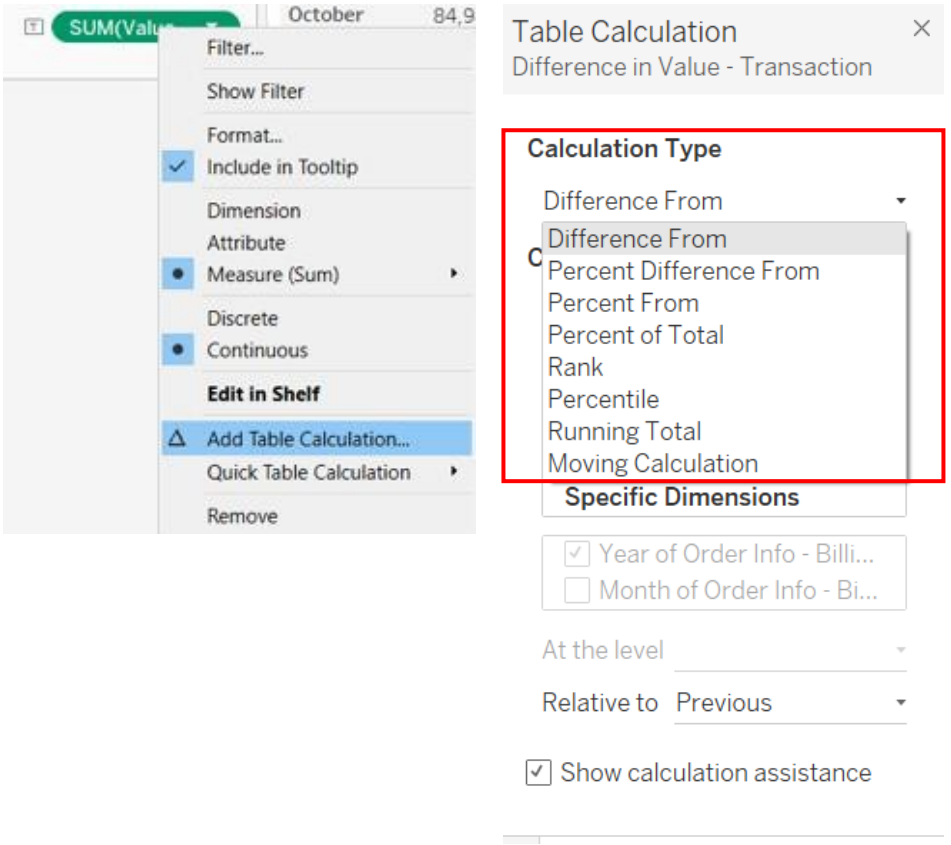
Table calculations:
- Table calculations in calc editor
- Via menu, as quick table calculation

3 Level of Detail (LOD)

Level of Details Expressions:
- More control on the level of granularity
- More granular level (INCLUDE), less granular level (EXCLUDE), entirely independent level (FIXED)

PART 3: Table Calculation

Table Calculation



Calculation Type

Difference From, Percent From:

Computes the difference between the current value and another value in the table for each mark in the visualization.

Percent of Total:

Computes a value as a percentage of all values in the current partition.

Rank:

Computes a ranking for each value in a partition.

Percentile:

Computes a percentile rank for each value in a partition.

Running Total:

Aggregates values cumulatively in a partition by summing, averaging or replacing all values with either the lowest or highest actual value.

Moving Calculation:

Determines the value for a mark in the view by performing an aggregation (sum average, minimum or maximum) across a specified number of values before and/or after the current value.

PART 3: Table Calculations

Table calculations rely on two types of fields: addressing fields and partitioning fields.

- **Partitioning fields define the scope:** They break up the view into multiple partitions or sub-views. The table calculation is then applied to the marks within each partition.
- **Addressing fields define the direction:** They define the "direction" that the calculation moves (for example, in calculating a running sum, or computing the difference between values).

Table (Across)

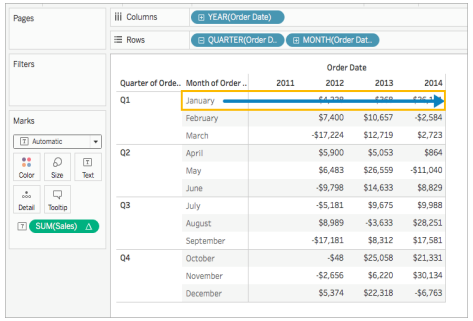
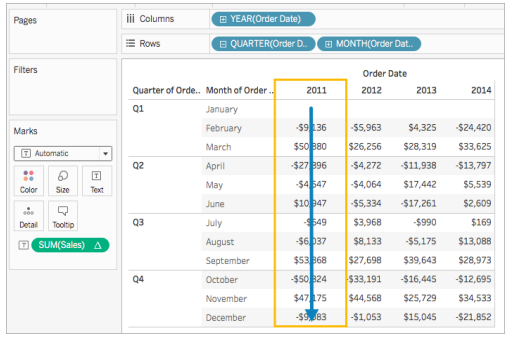


Table (Down) – Whole Table



Pane (Down)

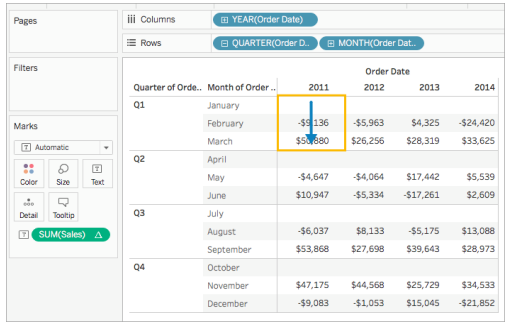
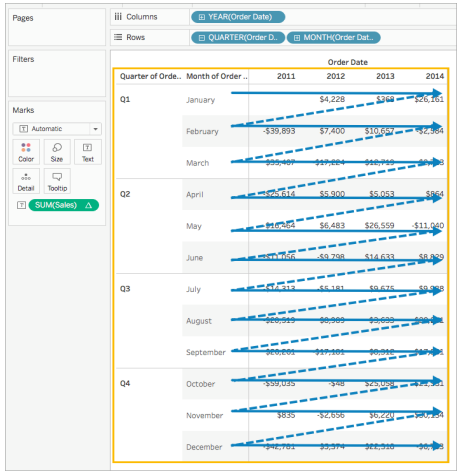


Table (across then down)



Pane (across then down)

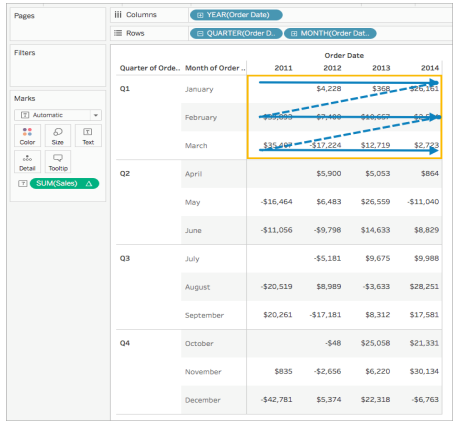
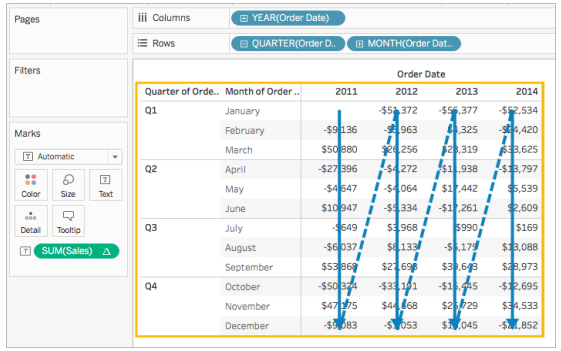
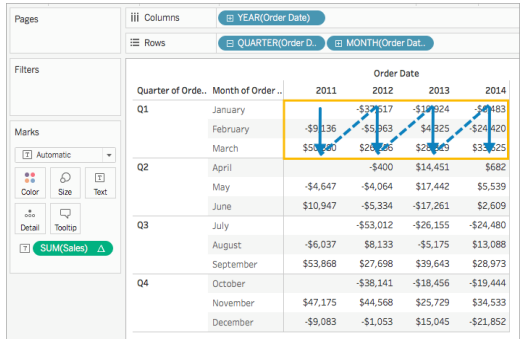


Table (down then across)



Pane (down then across)

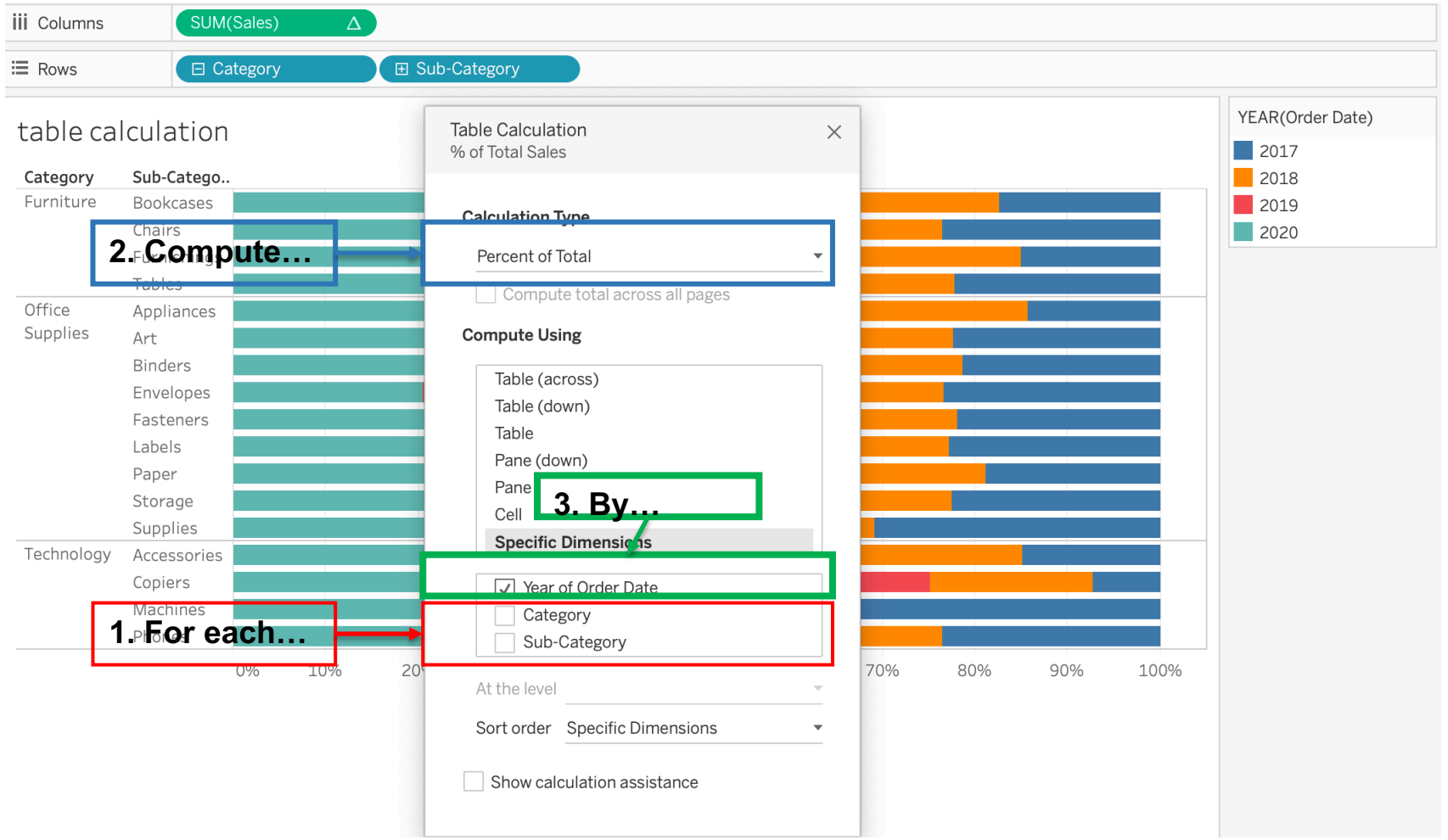


PART 3: Table Calc in English Sentences

For each Category and sub-category,

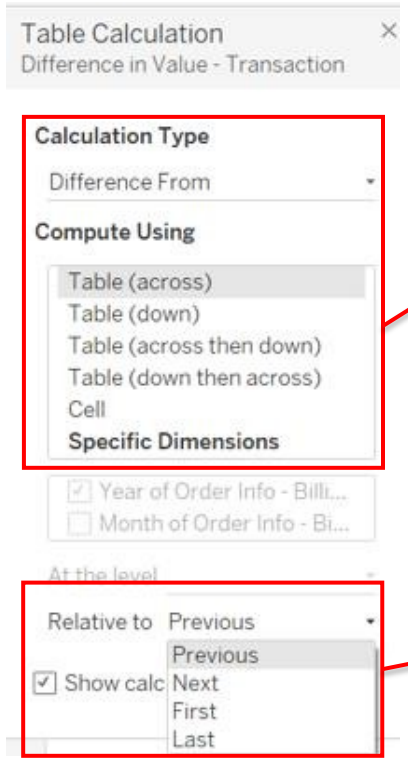
Compute the percent of total

By Year of Order Date



PART 3: Difference From

- For all difference from (number or percent), there are always two values to consider: **current value** and the value from which the difference should be calculated.
- There are two options to specify this: **“Compute Using” & “Relative to”**

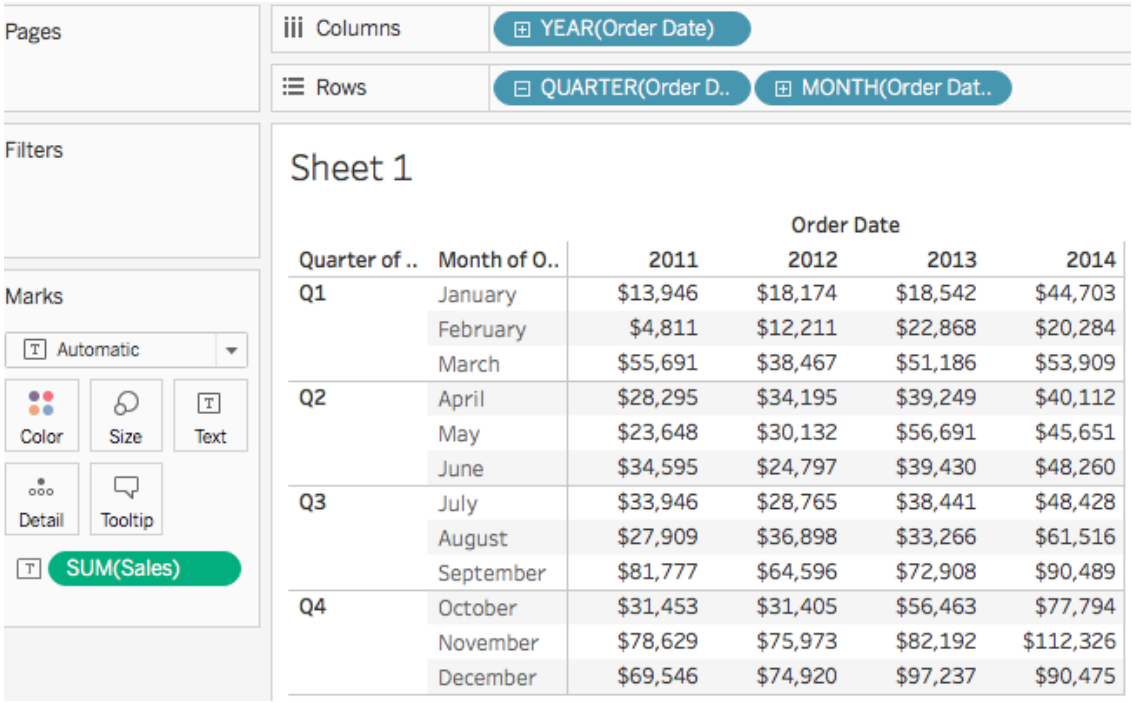


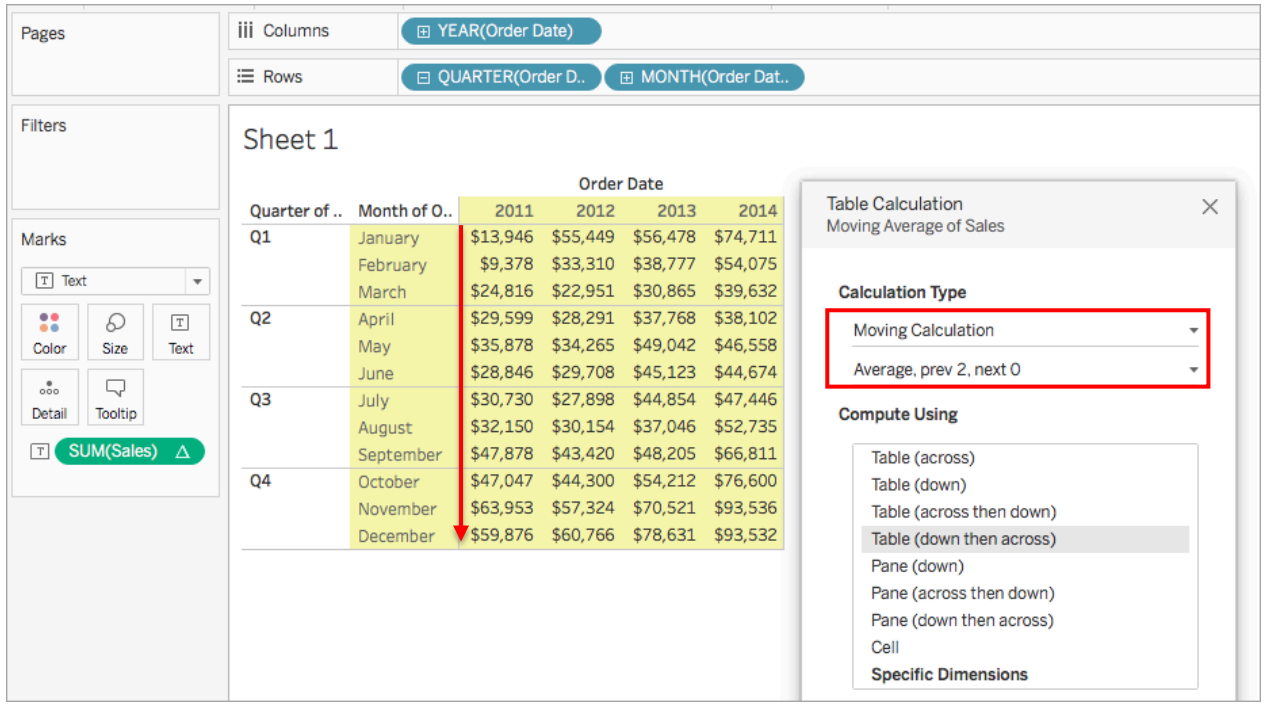
Compute Using	
Table (across)	Computes across the length of the table and restarts after every partition.
Table (down)	Computes down the length of the table and restarts after every partition.
Table (across then down)	Computes across the length of the table, then down the length of the table
Table (down then across)	Computes down the length of the table, and then across the length of the table.
Cell	Computes within a single cell
Specific Dimensions	Computes only within the dimensions the user specifics

Relative to	
Previous	Calculates the difference between current value and the previous value.
Next	Calculates the difference between current value and the next value.
First	Calculates the difference between current value and the first value.
Last	Calculates the difference between current value and the last value.

PART 3: Moving Calculation

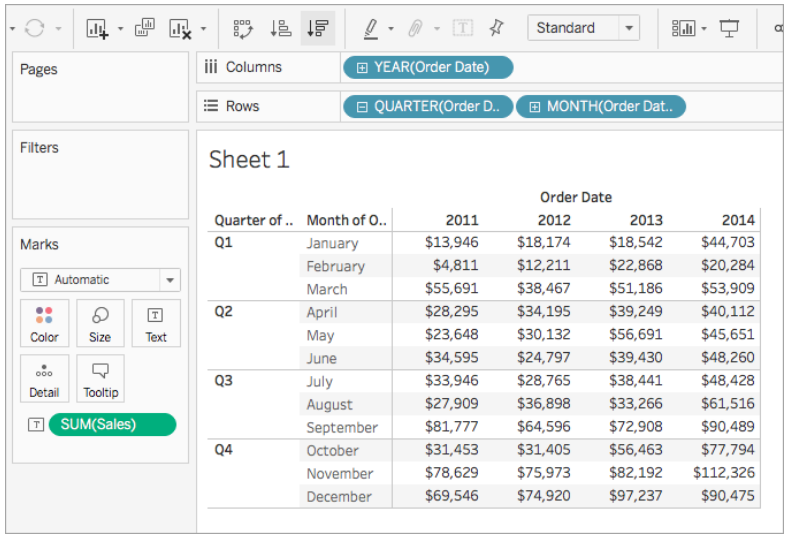
- A moving calculation is typically used to **smooth short-term fluctuations** in your data so that you can see **long-term trends**.
- Moving Calculation table calculation determines the value for a mark in the view by performing an aggregation (sum, average, minimum, or maximum) across a specified number of values before and/or after the current value.
- Consider table below, Moving Average, Previous 2 and Next 0 averages the monthly total for it and the two previous months over time. 9378 = (13946 + 4811)/2; 24816 = (55691 + 4811 + 13946)/3; 29599 = (4811+55691+28295)/3





PART 3: Percent of Total

- Percent of Total table calculation computes a value as a percentage of all values in the current partition..
- January 2011 makes up 18.73% of sales made in Q1. (Pane down, percent of total)
- January 2011 makes up 2.88% of sales made in 2011. (Table down, percent of total)



Pane (Down)

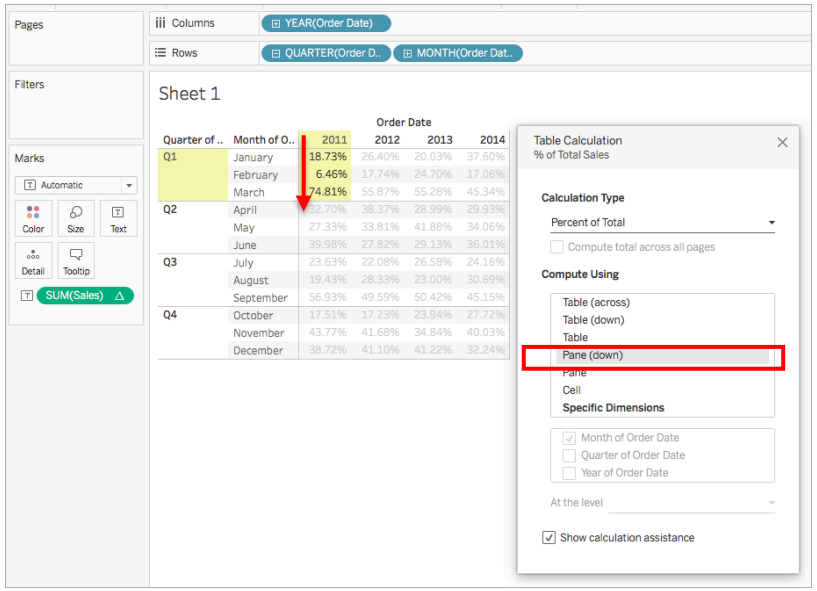
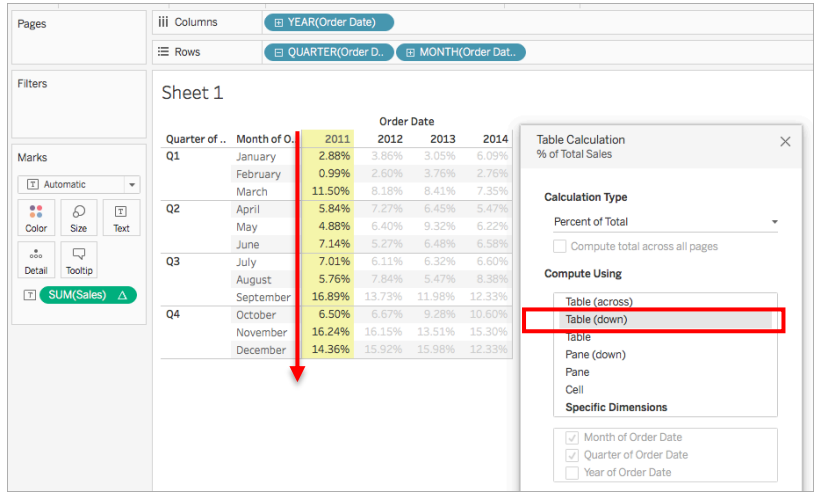
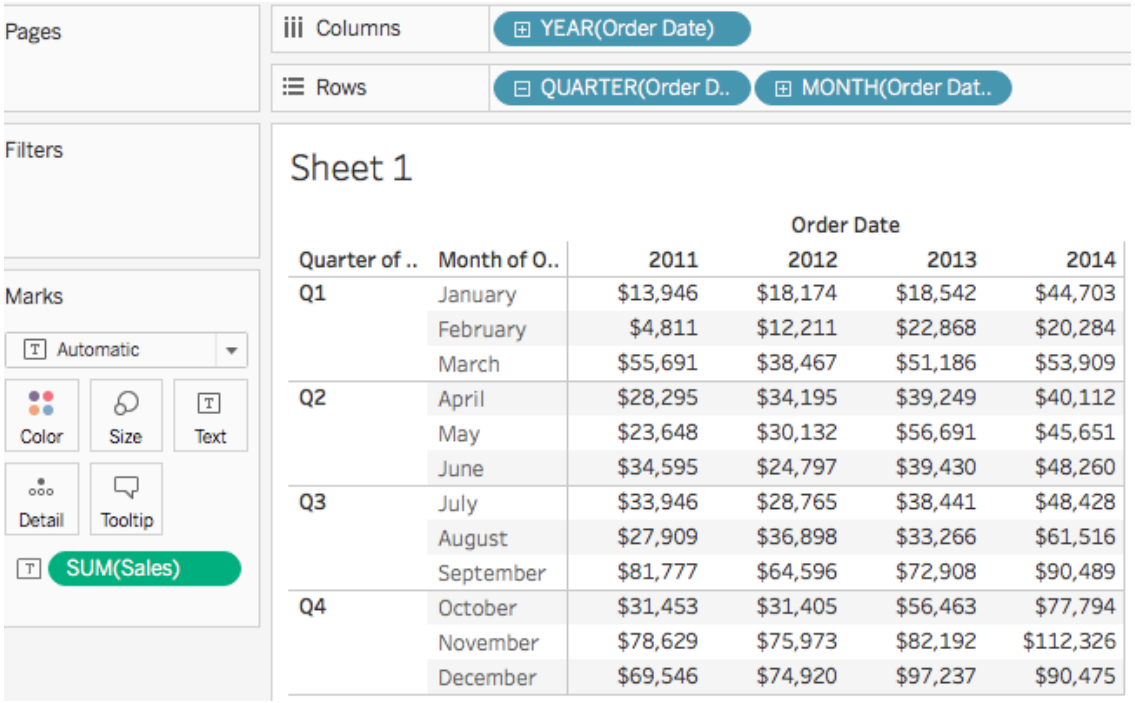


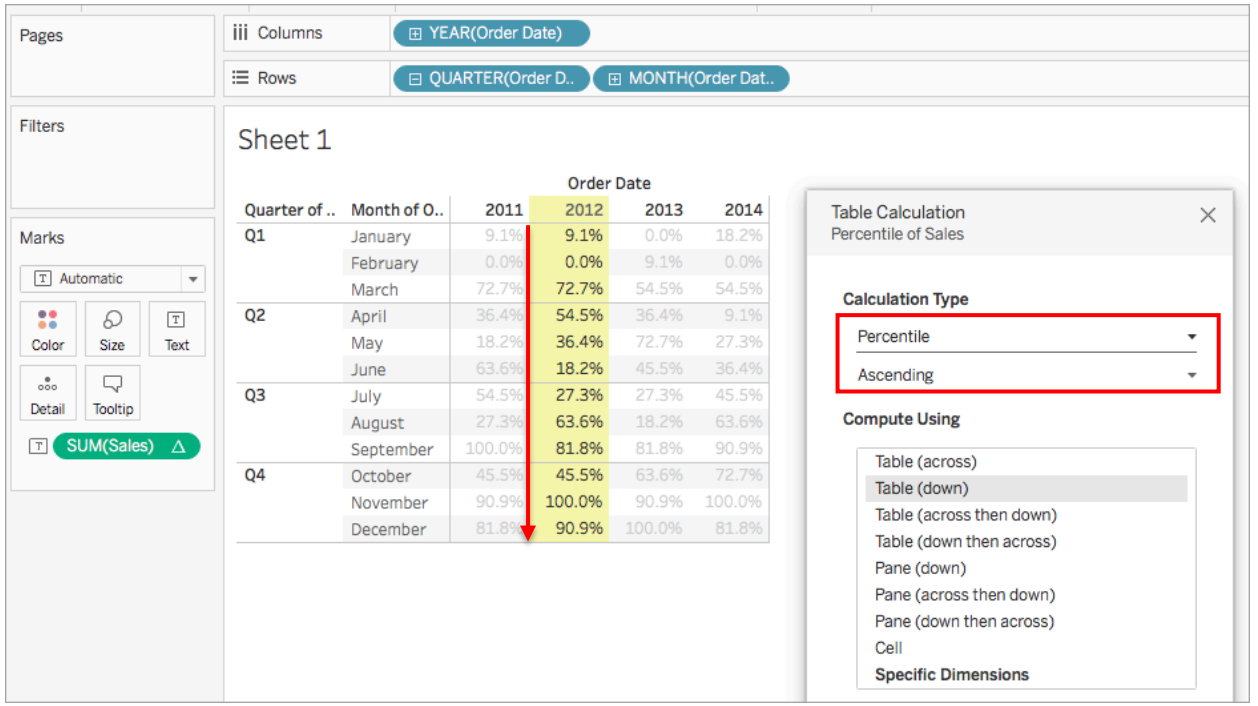
Table (Down)



PART 3: Percentile

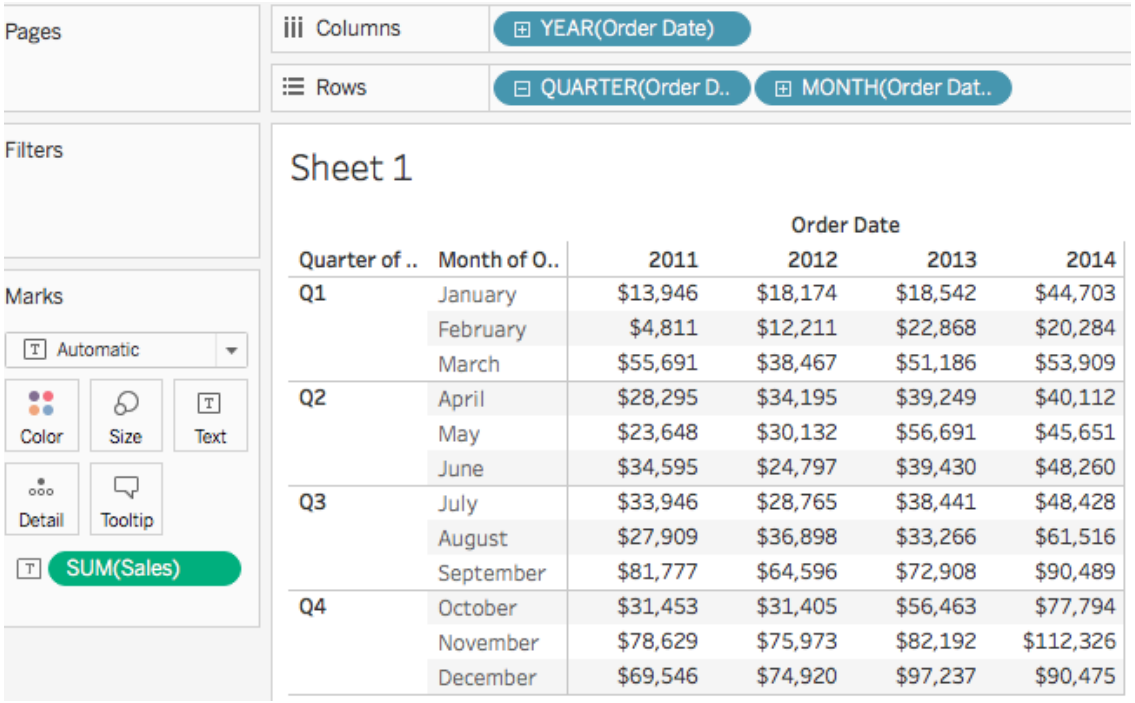
- **Percentile** table calculation computes a percentile rank for each value in a partition.
- The **percentile rank** of a score is the percentage of scores in its frequency distribution that are equal to or lower than it.
- Ascending order ranks values from least to most. Descending order ranks values from most to least.
- Consider table below, January 2012 sales is 18174, second lowest sales in 2012, 1/11 = 9.1%; February 2012 sales is 1211, the lowest sales in 2012, 0/11 = 0%; March 2012 sales is 38467, ranked 9th sales in 2012, 8/11 = 72.7%.



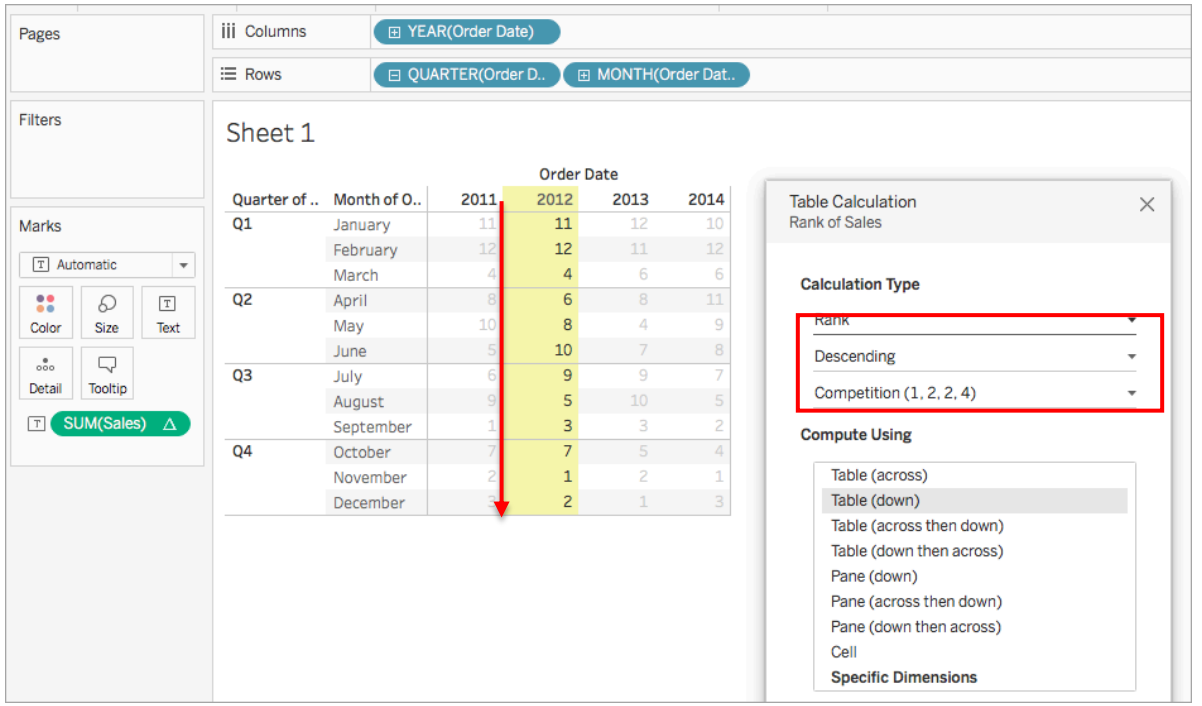


PART 3: Rank

- Rank table calculation computes a ranking for each value in a partition. For Rank table calculation, the default value is Descending.
- Ascending order ranks values from least to most. Descending order ranks values from most to least.
- There are 4 Rank Type: Competition (1, 2, 2, 4); Modified Competition (1, 3, 3, 4); Dense (1, 2, 2, 3); Unique (1, 2, 3, 4).
- Consider table below, 75973 is the largest sales in 2012 It ranks first. 12211 is the smallest sales in 2012. It ranks last (12th).

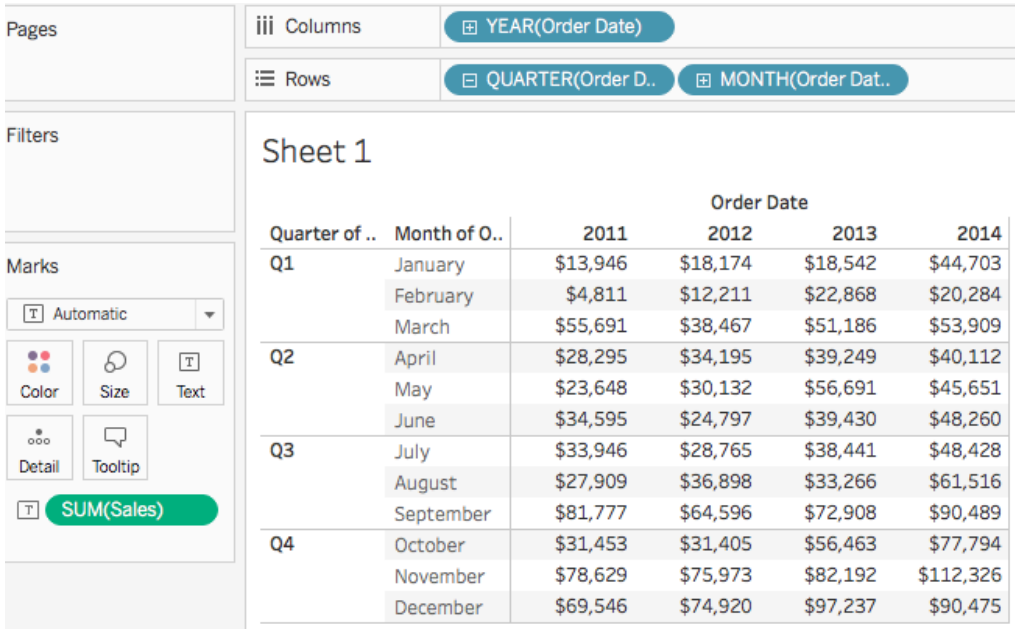


		Order Date			
Quarter of ..	Month of O..	2011	2012	2013	2014
Q1	January	$13,946	$18,174	$18,542	$44,703
	February	$4,811	$12,211	$22,868	$20,284
	March	$55,691	$38,467	$51,186	$53,909
Q2	April	$28,295	$34,195	$39,249	$40,112
	May	$23,648	$30,132	$56,691	$45,651
	June	$34,595	$24,797	$39,430	$48,260
Q3	July	$33,946	$28,765	$38,441	$48,428
	August	$27,909	$36,898	$33,266	$61,516
	September	$81,777	$64,596	$72,908	$90,489
Q4	October	$31,453	$31,405	$56,463	$77,794
	November	$78,629	$75,973	$82,192	$112,326
	December	$69,546	$74,920	$97,237	$90,475



PART 3: Running Total

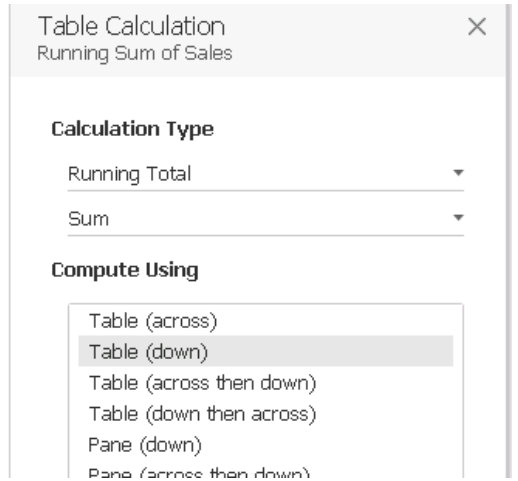
- **Running Total** table calculation aggregates values cumulatively in a partition. It can do this by summing values, averaging values, or replacing all values with either the lowest or highest actual value.
- The Running Total doesn't have to be a sum, it can be **Average, Minimum, Maximum** as well.



Sheet 1

		Order Date			
Quarter of ..	Month of O..	2011	2012	2013	2014
Q1	January	$13,946	$18,174	$18,542	$44,703
	February	$4,811	$12,211	$22,868	$20,284
	March	$55,691	$38,467	$51,186	$53,909
Q2	April	$28,295	$34,195	$39,249	$40,112
	May	$23,648	$30,132	$56,691	$45,651
	June	$34,595	$24,797	$39,430	$48,260
Q3	July	$33,946	$28,765	$38,441	$48,428
	August	$27,909	$36,898	$33,266	$61,516
	September	$81,777	$64,596	$72,908	$90,489
Q4	October	$31,453	$31,405	$56,463	$77,794
	November	$78,629	$75,973	$82,192	$112,326
	December	$69,546	$74,920	$97,237	$90,475

		Order Date			
Quarter of ..	Month of O..	2016	2017	2018	2019
Q1	January	$14,237	$18,174	$18,542	$43,971
	February	$18,757	$30,125	$41,521	$64,273
	March	$74,448	$68,852	$93,237	$123,145
Q2	April	$102,743	$103,047	$131,987	$159,666
	May	$126,391	$133,179	$188,975	$203,928
	June	$160,987	$157,976	$229,319	$256,909
Q3	July	$194,933	$186,741	$268,581	$302,174
	August	$222,842	$223,640	$299,697	$365,295
	September	$304,620	$288,236	$373,107	$453,161
Q4	October	$336,073	$319,640	$432,795	$530,938
	November	$414,702	$395,613	$512,207	$649,386
	December	$484,247	$470,533	$609,206	$733,215



PART 3: Master Tableau Calculations

1 Basic Calculations

Row-level calculations/Aggregate calculation:
- Arithmetic calculation
- Logical operators
- String calculations
- Date calculations

2 Table Calculations

Table calculations:
- Table calculations in calc editor
- Via menu, as quick table calculation

3 Level of Detail (LOD)

Level of Details Expressions:
- More control on the level of granularity
- More granular level (INCLUDE), less granular level (EXCLUDE), entirely independent level (FIXED)

PART 3: Level of Details (LOD)

LOD allows you to present (or simply calculate) data at a higher or lower level of detail than what is currently in your visualization.

1. FIXED expression

- **{FIXED [whatever field you want things to be fixed by separated by commas]: aggregate of the measure you want to be fixed}**
- Level of details defined by […]= Irrespective of level of detail in view
- { FIXED [College]: COUNTD([U_ID])} would provide you with the distinct count of students (or employees) for a given college, even if the user filters by a particular department

01

2. INCLUDE expression

- **{INCLUDE [name of field(s) you want included in the calculation]: aggregate of the measure you want calculated}**
- Include […] to level of detail from view = more granular than level of detail in view

02

3. EXCLUDE expression

- **{EXCLUDE [name of field(s) you want excluded in the calculation]: aggregate of the measure you want calculated}**
- Excludes […] from level of detail in view = less granular than level of detail in view

03

PART 3: Level of Details (LOD)

FIXED vs INCLUDE/EXCLUDE

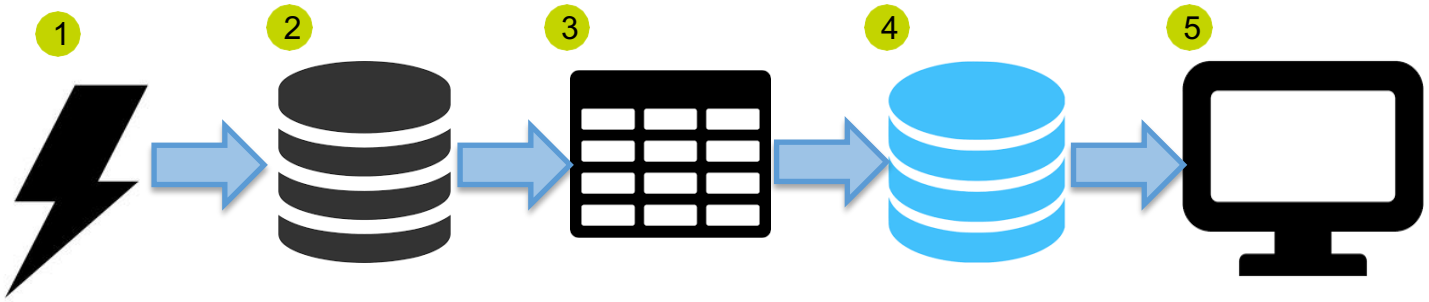
- Depending on what's in the view, an INCLUDE/EXCLUDE function can often be used to create a field that is equivalent to what FIXED will provide
- However, INCLUDE/EXCLUDE fields are **always measures**, while FIXED creates a **dimension or measure.**
- Generally, if you're creating a field that will categorize groups, use FIXED; otherwise, use either FIXED or INCLUDE/EXCLUDE

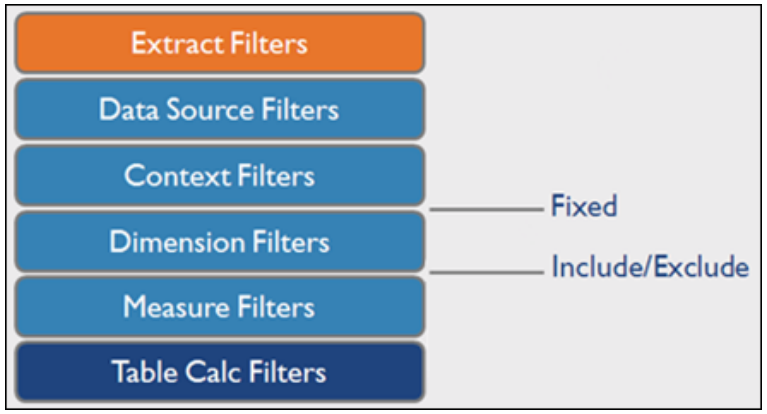
Nested LOD

- LOD functions can be nested
- Imagine you wanted to know the average number of math courses taken by each student by department at the level of the college, i.e., you don't want a particular student OR a particular department to skew the average

{FIXED [Dept]: {FIXED [U_ID]: AVG(MATH_COURSE_COUNT)}}

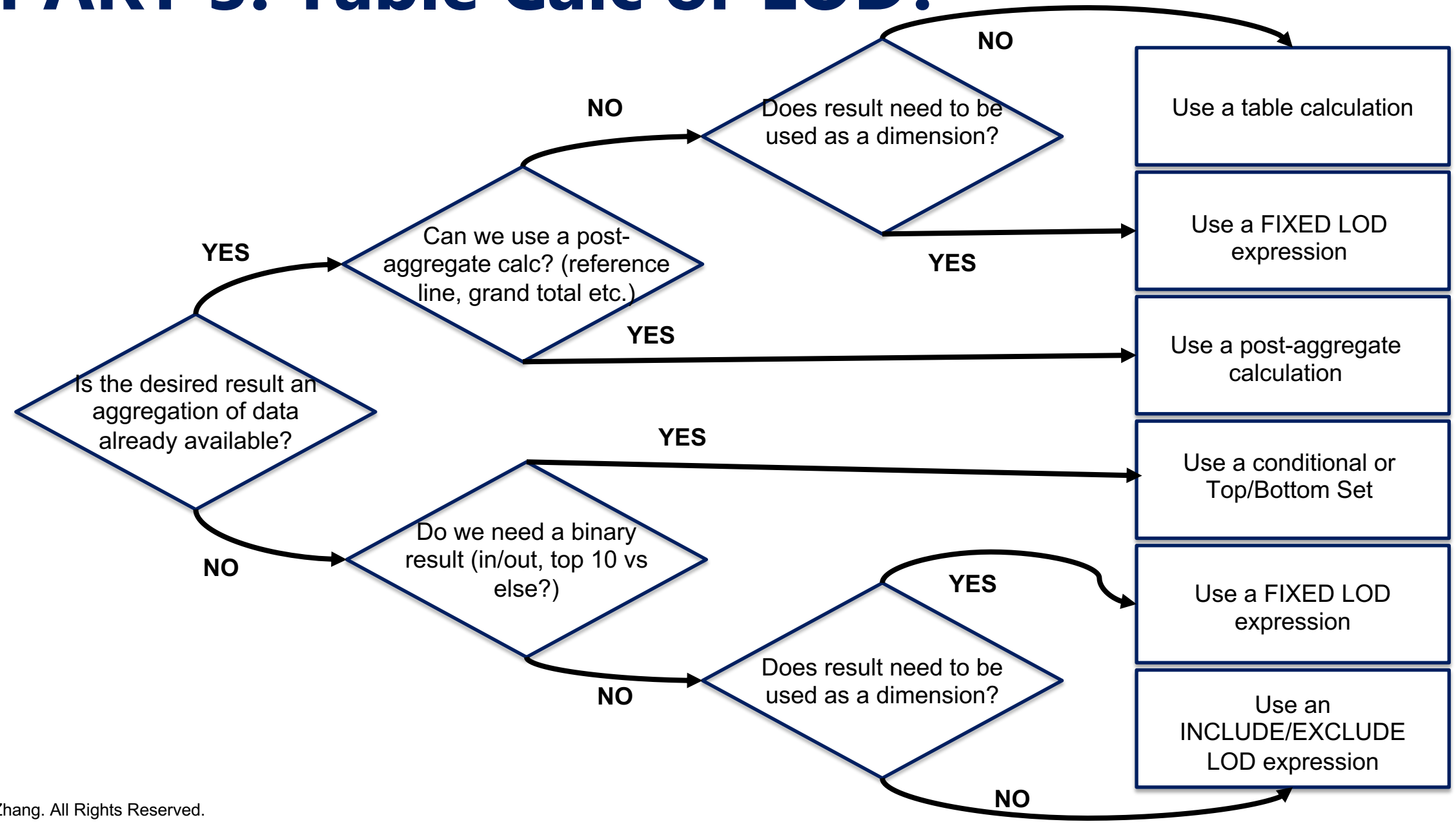
PART 3: Level of Details (LOD)





1. Tableau generates a query and sends it to the database
2. Database process the query. This is where Tableau considers **calculated fields**, including **level of detail calculations**
3. Tableau generates a temporary table that is already filtered and aggregated with any new columns produced as a result of calculated fields.
4. Only after all of that are **table calculation** applied.

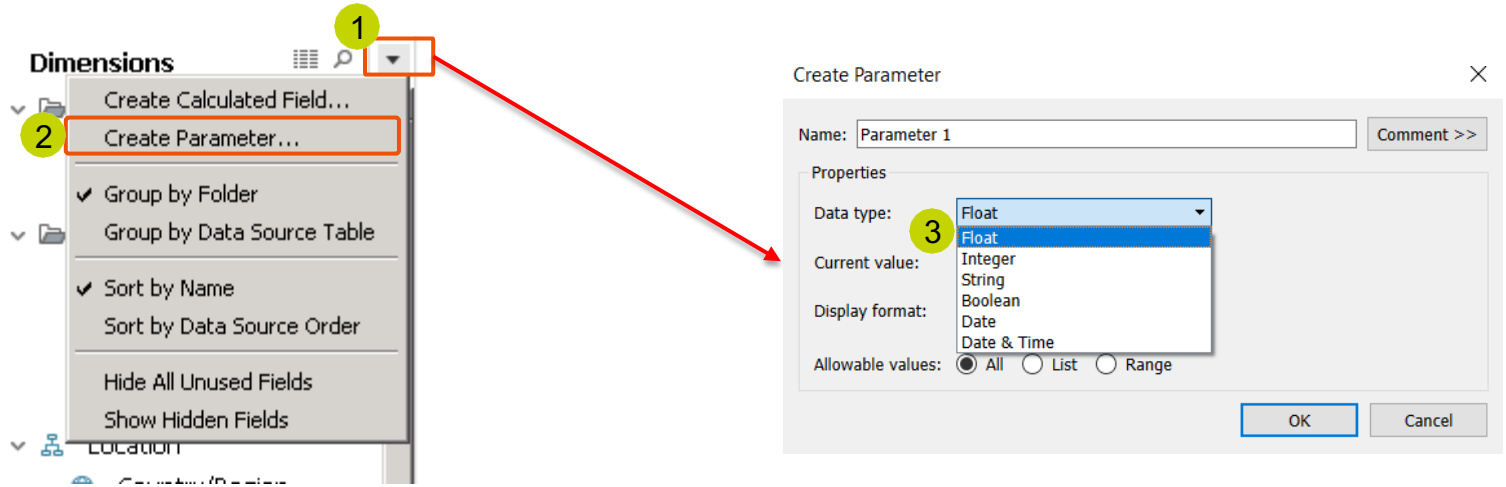
PART 3: Table Calc or LOD?



PART 3: Create Parameter

- A parameter is any value passed to a program to customize the program for a particular purpose. This could be anything:
  - A string of text
  - A range of values
  - An amount
- Common uses include: **What-If-Analysis & User Input Analysis**

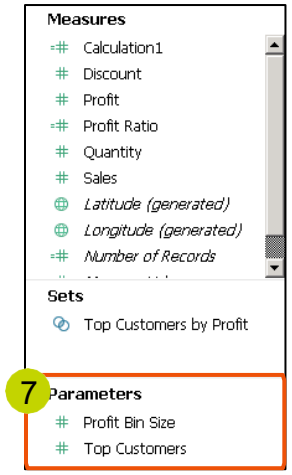
1. Click on the downward facing triangle ▼ in the top right corner of the dimensions card.
2. Select "**Create Parameter**"
3. Specify the data type (Float/Integer/String/Boolean/Date/Date & Time) for values.

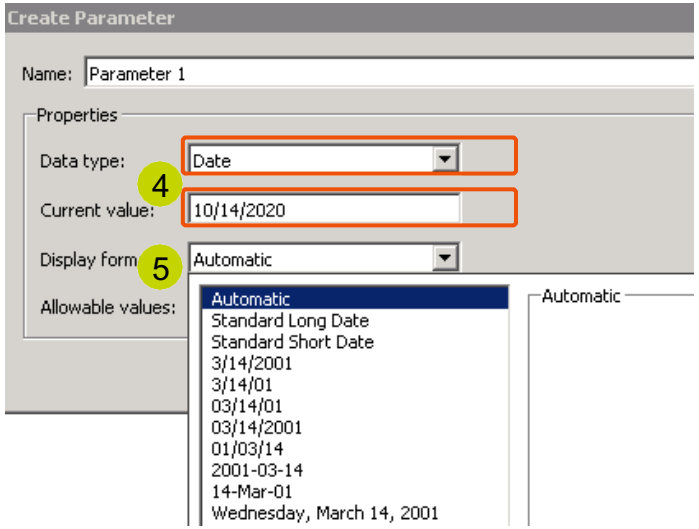


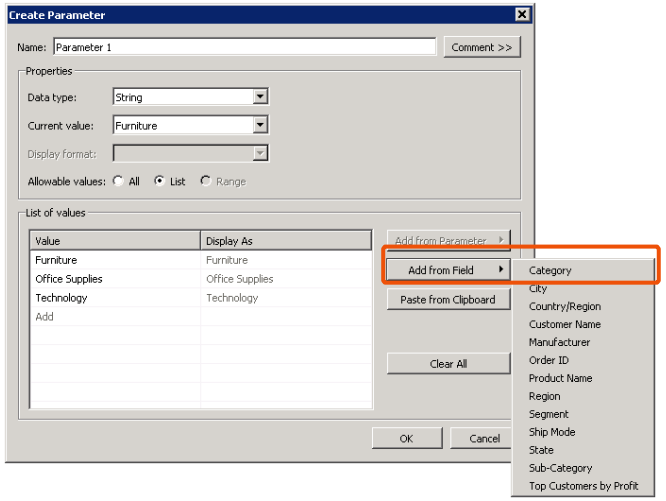
PART 3: Create Parameter

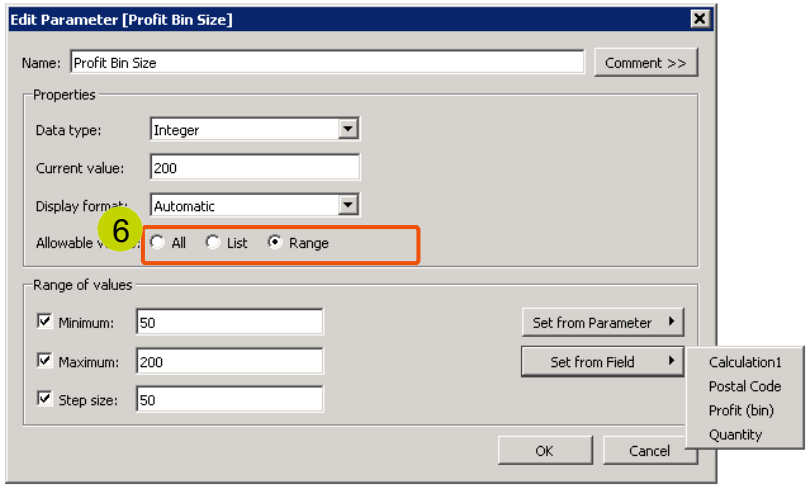
4. Specify a current value. This is the default value for the parameter.
5. Specify the display format to use in the parameter control.
6. Specify the parameter allowable values: **“All”, “List”, “Range”**
   If you select List, you can specify the list of values.
   - ➢ You can add members by **Add from Field**
   - ➢ Or you can **Paste from Clipboard**.

   If you select Range, you must specify minimum , maximum and step size.
7. Once done, the parameter will show in Parameter section.







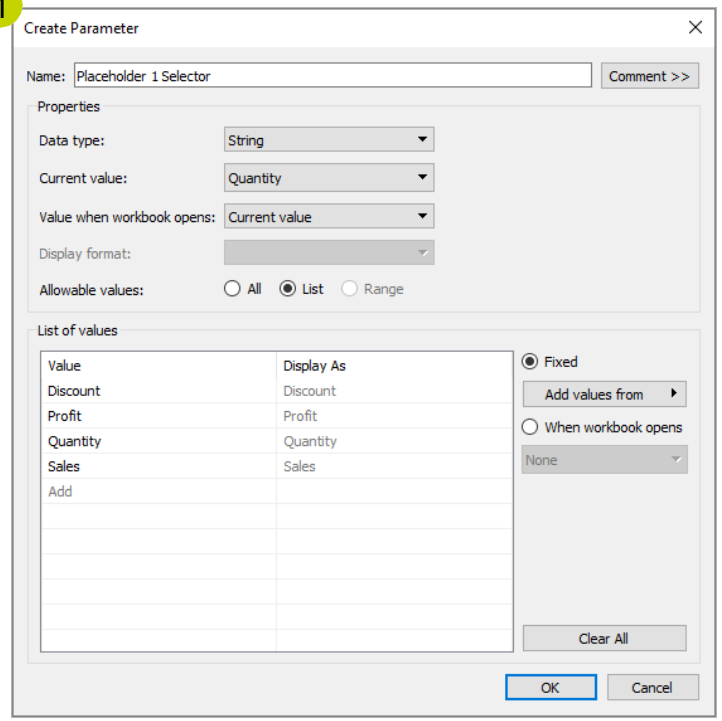


PART 3: Use Parameter to Swap Measures

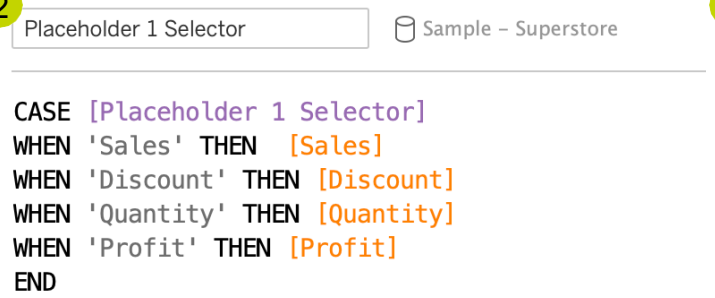
Suppose you want to add a control to the view that would let any user select the measures to be compared.
You can create such a view using parameters and calculated fields.

1. Create the Parameter with a list of measures
2. Use the parameter in a Calculation to change the measures
3. Show the Parameter Control for the user
4. Use the Calculated Field in your visuals

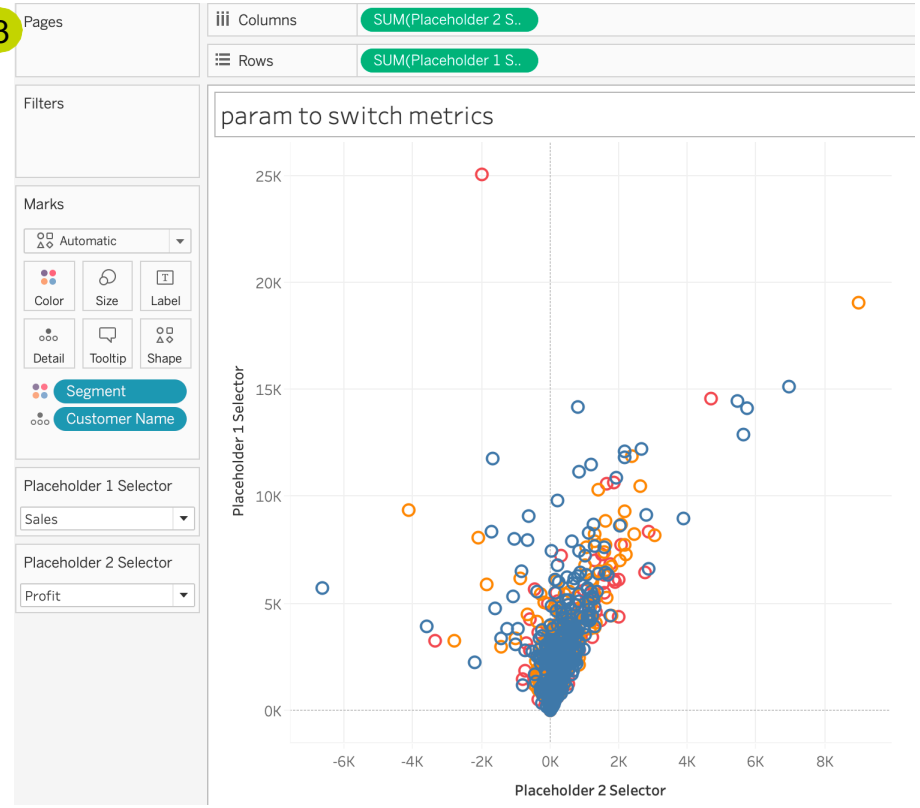
1



2



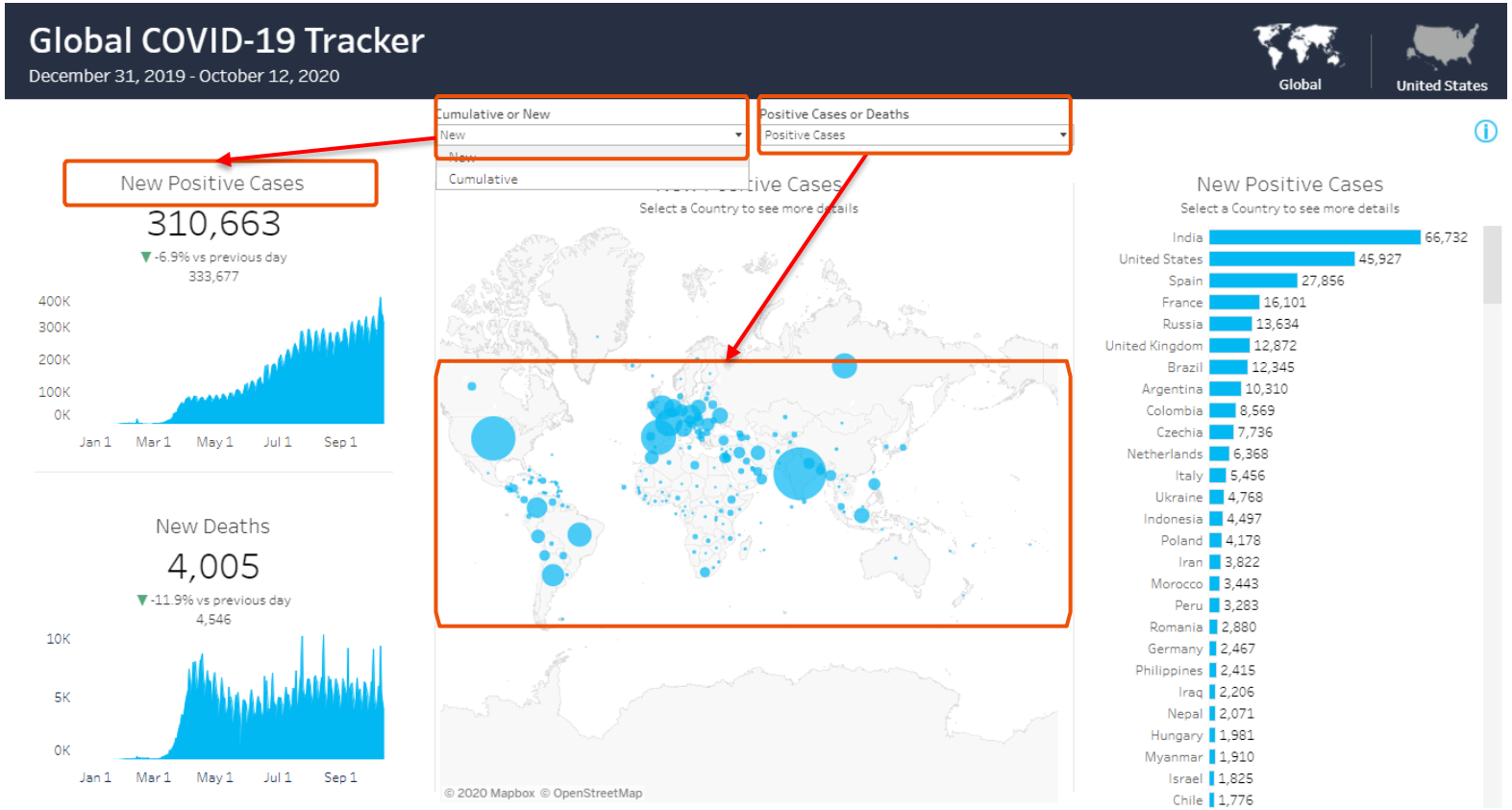
3



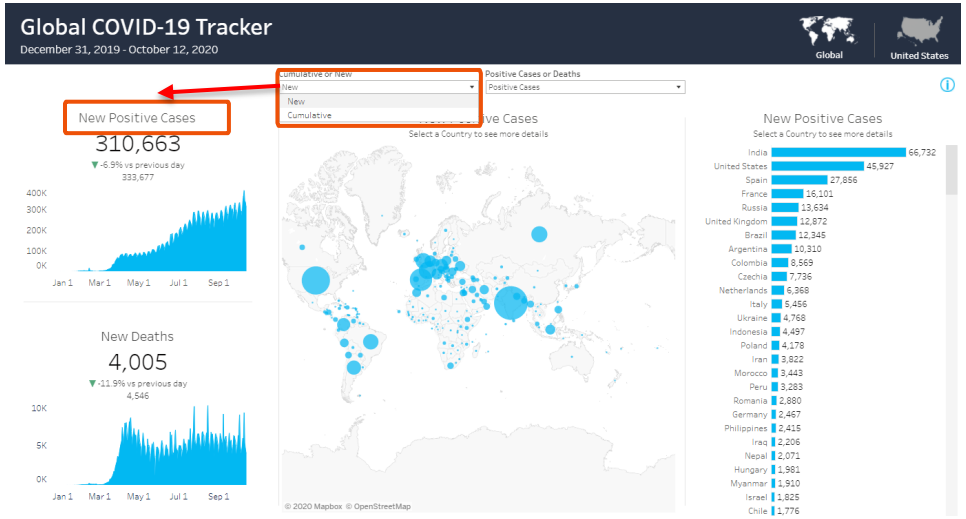
PART 3: Use Parameter to Swap Measures

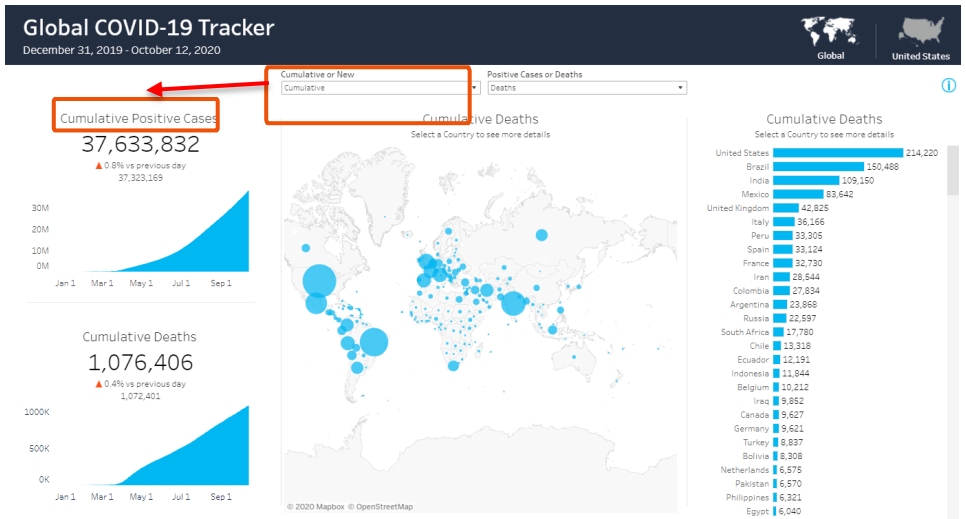
Common uses for Parameters are what-if analysis and user-input analysis.

1. Create the Parameter
2. Use the parameter in a Calculation
3. Show the Parameter Control for the user
4. Use the Calculated Field in your visuals

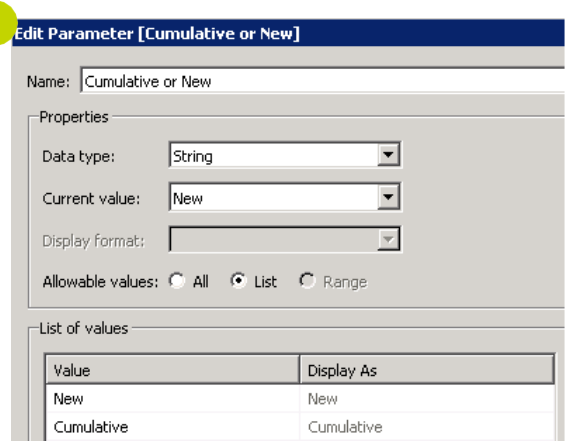


PART 3: Use Parameter to Swap Measures





1

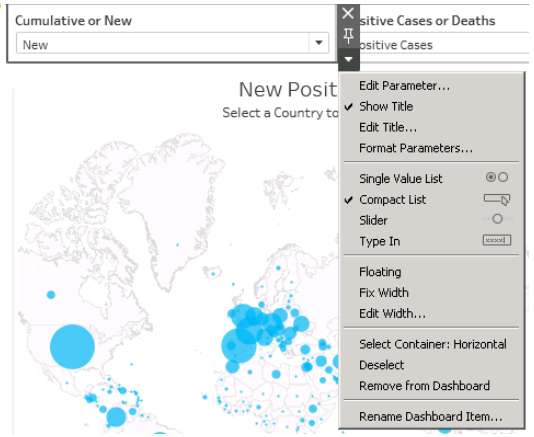


2

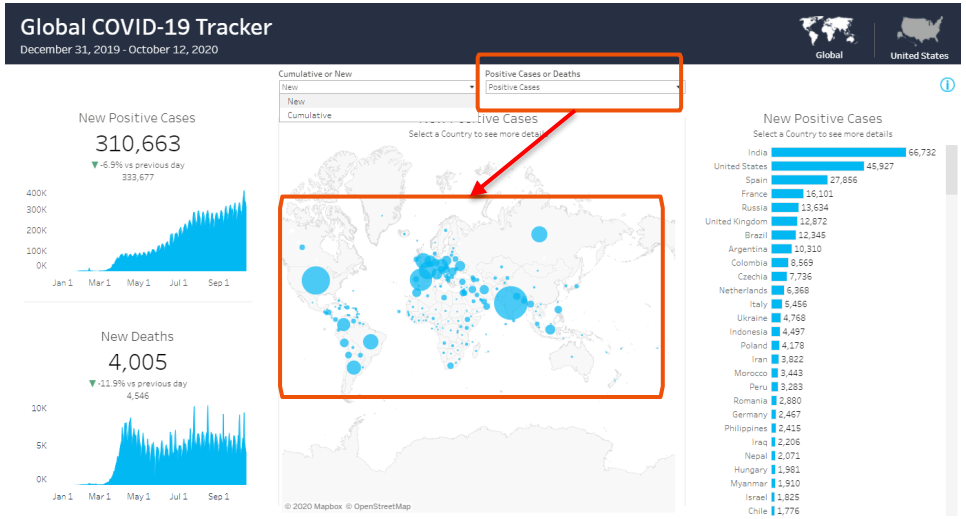
Total or New Positive Cases

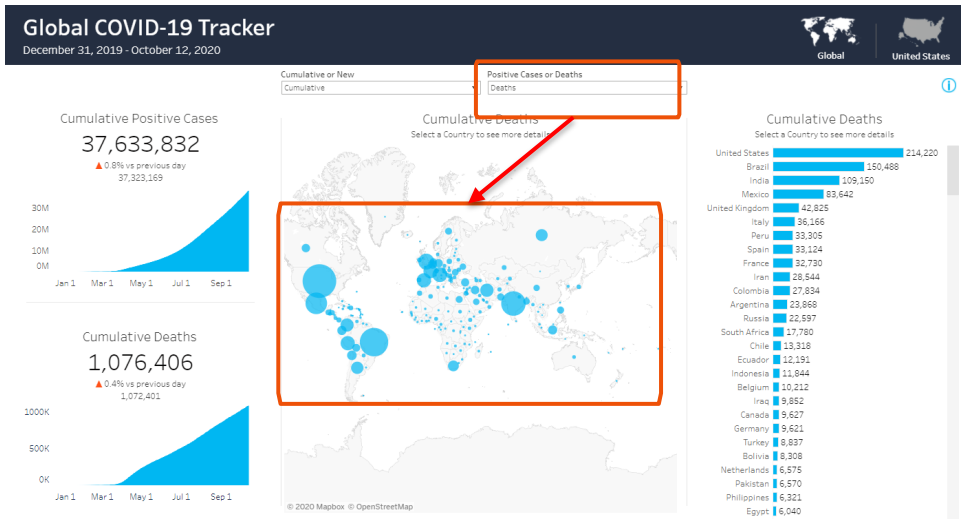
```
IF [Cumulative or New] = "New" THEN [People Positive New Cases Count]
ELSE [People Positive Cases Count] END
```

3

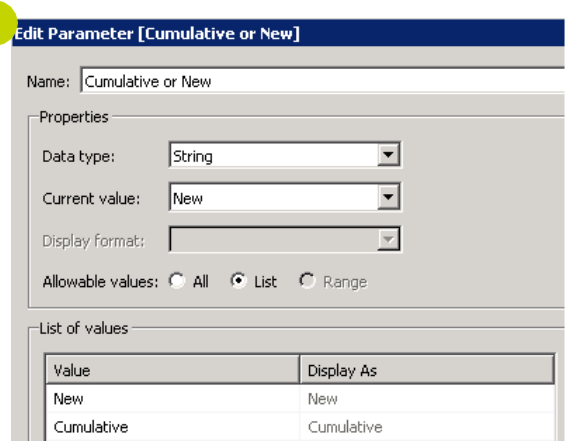


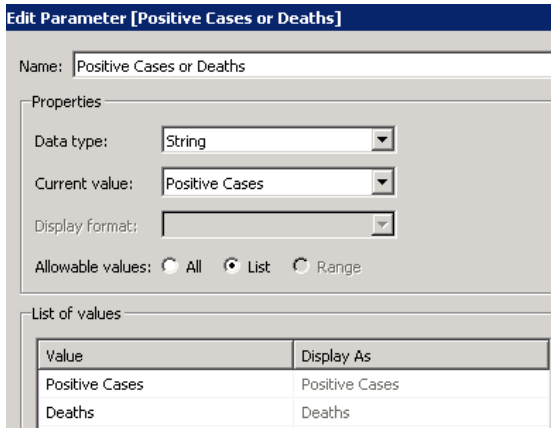
PART 3: Use Parameter to Swap Measures





1



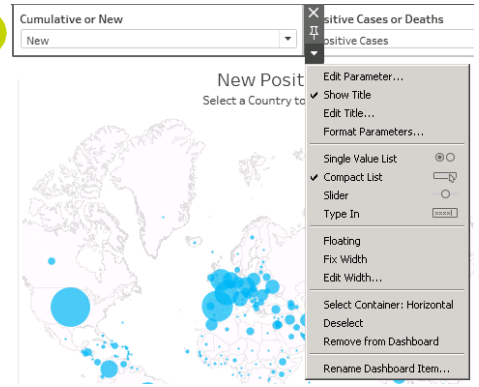


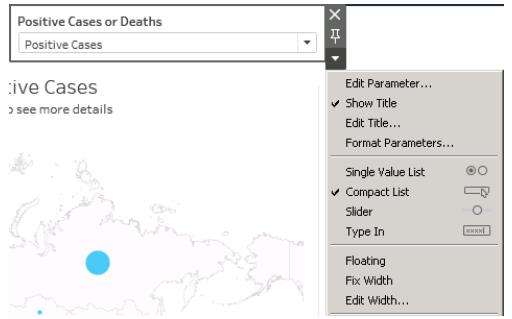
2

Selected Metric

```
IF [Cumulative or New] = "Cumulative" and [Positive Cases or Deaths] = "Positive Cases"
THEN [People Positive Cases Count]

ELSEIF [Cumulative or New] = "New" and [Positive Cases or Deaths] = "Positive Cases"
THEN [People Positive New Cases Count]

ELSEIF [Cumulative or New] = "New" and [Positive Cases or Deaths] = "Deaths"
THEN [People Death New Count]

ELSEIF [Cumulative or New] = "Cumulative" and [Positive Cases or Deaths] = "Deaths"
THEN [People Death Count]

END
```

3



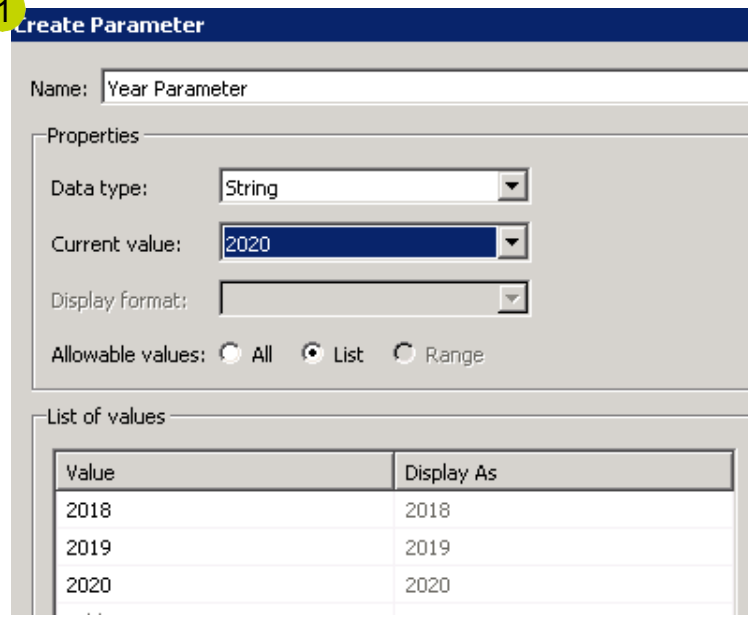


PART 3: Use Parameter as Filter

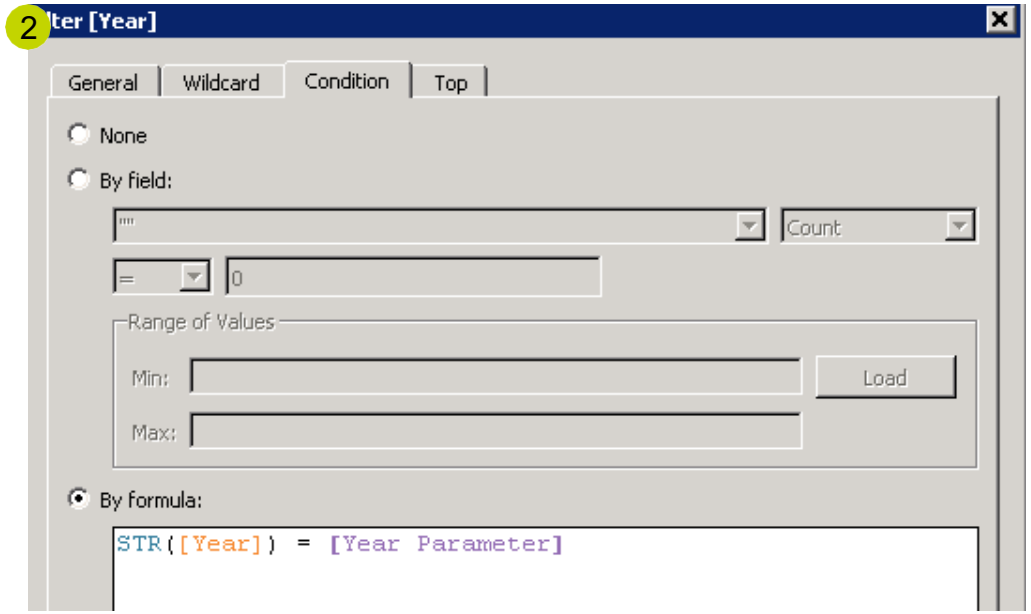
Suppose a dashboard are using multiple data sources and if you want to filter all the sheets in the dashboard by year.
Option 1: You can use Quick Filter for each individual data source.
Option 2: You can use Parameter to filter

- Write a Boolean condition to say the field and your parameter are the same.
- This will then pass the value you select in your parameter and subsequently filter the field by this.

1



2

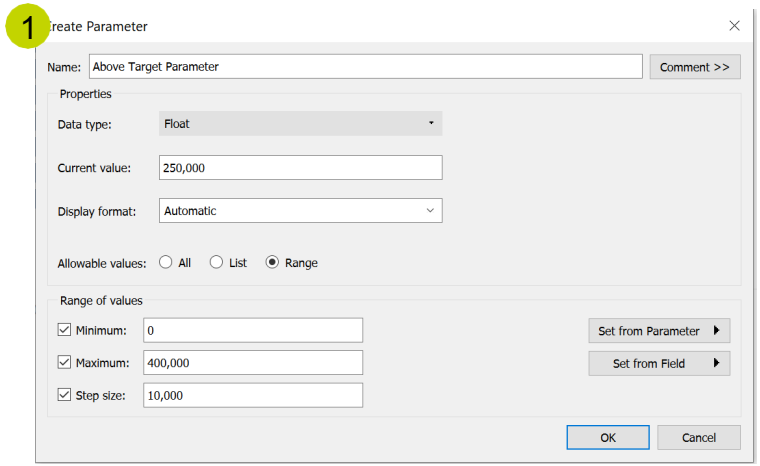


PART 3: Use Parameter to Change Color

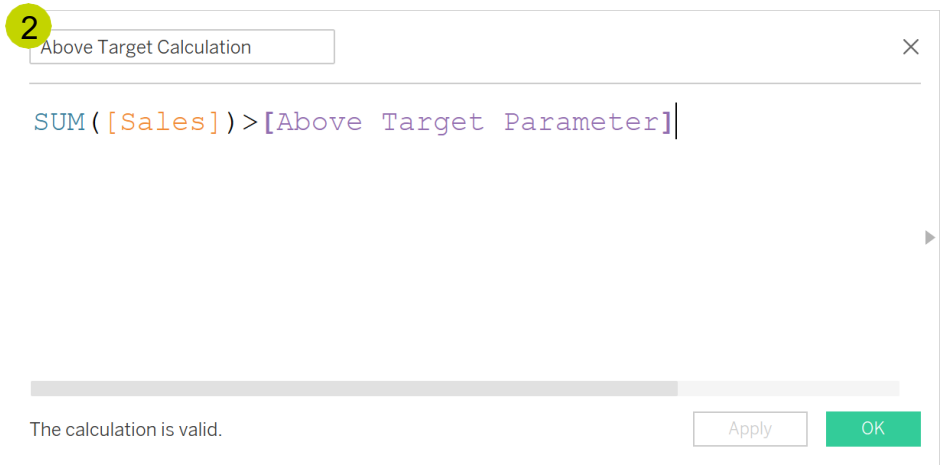
Suppose you wan to create a parameter which will automatically change the colors of a bar graph depending on whether a target has been met or not.

1. Use can define the target level.
2. The metric will benchmark against the target level. If above, the graph shows a color, if below, the graph shows another color.
    - ➢ Write a Boolean condition to benchmark against target level.
    - ➢ This will then pass the target value in your parameter and subsequently add to the color.

1



2



3