TABLEAU NOTES

By Jay Charole

Table of Contents

Introduction to Business Intelligence

What is BI?

Business and organization have questions and goals. To answer the questions and track performance against these goals, they gather the necessary data, analyze it and determine which actions to take to reach their goals.

Organization collect data from internal IT systems and external sources, prepare it for analysis, run queries against the data and create data visualizations.

It combines variety of tools, technologies and methods with intent of driving positive changes and boosting company performance.

Example:

1) What was the sales last year?
2) How to increase sales this year by x%?

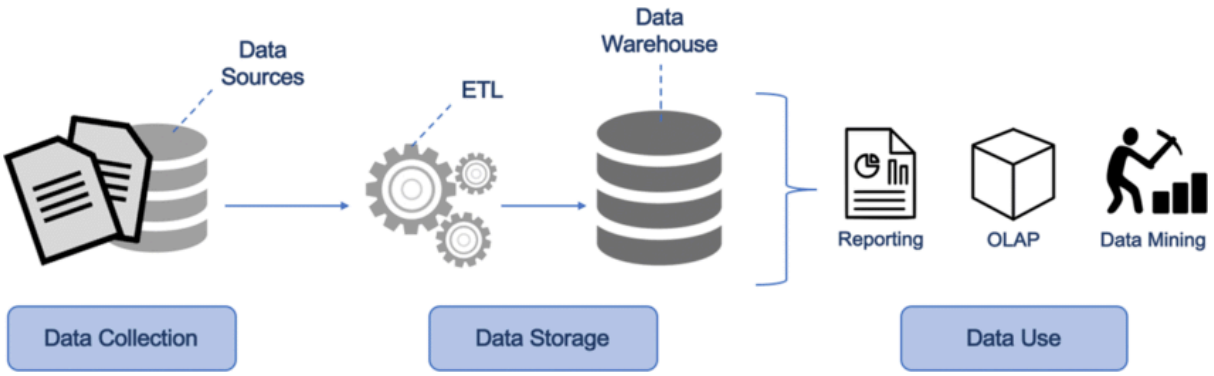
Uses of Business Intelligence

1. BI involves collecting and analysing data from multiple sources, such as internal databases, external data sources, and third-party tools.
2. The data is transformed into meaningful and actionable information through data analytics, data mining, and other advanced technologies.
3. BI provides insights that can be used to support decision-making, improve performance, and gain a competitive advantage.
4. BI tools include data warehouses, data mining software, business analytics tools, and dashboards.
5. BI dashboards and reports are designed to be user-friendly, enabling decision-makers to quickly and easily understand the data.
6. BI can help identify patterns, trends, and relationships in the data that may not be immediately apparent.
7. BI can be used in a variety of industries and business functions, including finance, marketing, operations, and human resources.
8. BI is an ongoing process of data collection, analysis, and reporting, and requires a strong data governance framework to ensure data accuracy and consistency.
9. BI is increasingly being integrated with other technologies, such as artificial intelligence and machine learning, to provide even more advanced insights and predictive analytics.
10. BI is a critical tool for businesses looking to make data-driven decisions and stay competitive in today's fast-paced and data-driven world.

Practical benefits of BI

1. Identify ways to increase profit
2. Analyze customer behaviour
3. Compare data with competitors
4. Track performance
5. Optimize operations
6. Predict success
7. Spot Market trends
8. Discover issues or trends

BI Process



The process of Business Intelligence (BI) typically involves the following steps:

Data collection: The first step in the BI process is to collect relevant data from various sources, including internal databases, external data sources, and third-party tools.

Data cleaning and preparation: Once the data has been collected, it must be cleaned, organized, and prepared for analysis. This involves removing duplicate data, filling in missing values, and ensuring consistency and accuracy of the data.

Data analysis: The next step is to analyze the data using various analytical techniques, such as data mining, statistical analysis, and machine learning algorithms. The goal is to identify patterns, trends, and relationships in the data that can provide insights and inform decision-making.

Data visualization: The insights derived from the data analysis are then presented in user-friendly dashboards, reports, and visualizations. This step is important for making the data easily understandable and actionable for decision-makers.

Interpretation and decision-making: The final step in the BI process is tc
the insights presented in the dashboards and reports, and use them to infor
decision-making. This may involve making changes to business strategies, processes, or operations based on the insights gained from the data.

Introduction to Tableau

What is Tableau?

Tableau Desktop is part of the Tableau Creator business intelligence role, which includes desktop data connection and visualization capabilities as well as online sharing.

Why Tableau?

- **Connect, join, and analyze billions of records -** Connect to any data source with 60+ built-in connectors (flat files, databases, cloud sources, APIs, etc.)
- **Understand your data by sorting, grouping, and filtering -** Use Tableau's intuitive drag-and-drop no-code interface to easily manipulate data for analysis.

- **Define robust, custom calculations using intuitive "Excel-like" fu** Enhance and refine data sources with powerful calculated fields.
- **Visualize insights with powerful dashboards and stories -** Transform flat data into actionable insights with app-like business intelligence products.
- **Engage with experts through an industry-leading user community -** Join Tableau's rich, engaging, and results-driven user community that supports its members like no other.

Tableau Products

Tableau is a leading provider of business intelligence and data visualization software. Here's a brief overview of some of their most popular products:

1. **Tableau Desktop:** This is the flagship product of Tableau and is used to create interactive data visualizations, reports, and dashboards.
2. **Tableau Prep:** Tableau Prep is a data preparation tool that allows users to clean and transform data before visualizing it in Tableau Desktop.
3. **Tableau Server:** Tableau Server is an enterprise-level platform that allows organizations to share and collaborate on Tableau content securely.
4. **Tableau Online:** Tableau Online is a cloud-based version of Tableau Server that allows users to access their Tableau content from anywhere with an internet connection.
5. **Tableau Mobile:** Tableau Mobile is a mobile app that allows users to access and interact with Tableau content on their mobile devices.
6. **Tableau Public:** Tableau Public is a free version of Tableau that allows users to create and share public visualizations online.

Tour the Tableau Environment

There are 3 options on starting page

1.Connect: - Connect to your data

2.Open: - Open your most recently used workbooks.

3.Discover: - Discover and explore content produced by the Tableau community.

Connect: - Connect to data and open saved data sources.

Open: - On the Open pane, you can do the following

1.Open recently opened workbooks: When you open Tableau Desktop for the first time, this pane is empty. As you create and save new workbooks, the most recently opened workbooks appear here. Click the workbook thumbnail to open a workbook, or

if you don't see a workbook thumbnail, click the Open a Workbook link to fi workbooks that are saved to your computer.

2.Pin workbooks: You can pin workbooks to the start page by clicking the pin icon that appears in the top-left corner of the workbook thumbnail. Pinned workbooks always appear on the start page, even if they weren't opened recently. To remove a recently opened or pinned workbook, hover over the workbook thumbnail, and then click the "x" that appears. The workbook thumbnail is removed immediately but will show again with your most recently used workbooks the next time you open Tableau Desktop.

3. Explore accelerators: Open and explore accelerator workbooks to see what you can do with Tableau. Prior to 2022.2, these were called sample workbooks.

Discover:- See popular views in Tableau Public, read blog posts and news about Tableau, and find training videos and tutorials to help you get started.

Connect Data to Tableau Desktop

To connect data with tableau desktop, follow the following steps:

1.Open Tableau Desktop and click on the "Connect" button on the start page.

2.Select the type of data you want to connect to, such as a file or a database.

3.Choose the specific data source you want to use, such as a CSV file or a SQL Server database.

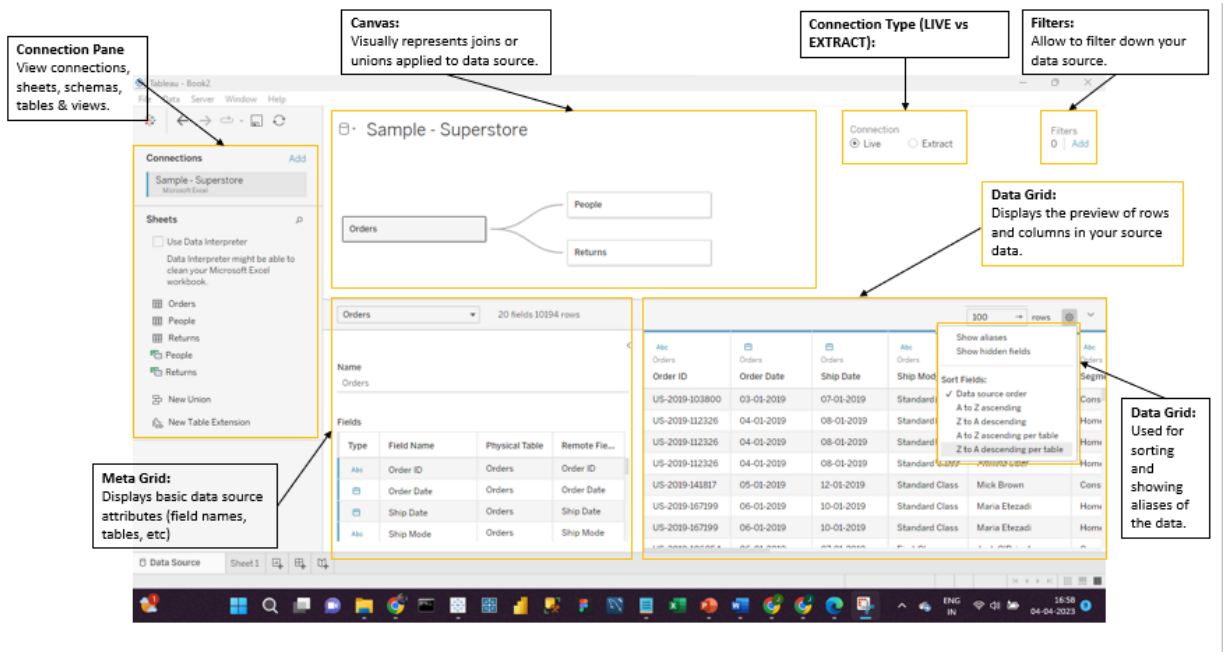
4.Provide the necessary connection details, such as the file path or server name, and login credentials if required.

5.Click on "Connect" to establish a connection with the data source.

6.Once connected, you can preview the data and select the specific tables or views you want to use in your analysis.

7.Click on the "Sheet" tab to start building your visualization. Drag and drop the fields from the data pane onto the worksheet to create your chart or graph.

Tableau Data Source Page



Connection pane - Displays the connected data source and other details about your data.

Canvas [Logical layer] - The canvas opens with the logical layer, where you can create relationships between logical tables.

Canvas [Physical layer] - Double-click a table in the logical layer to go to the physical layer of the canvas, where you can add joins and unions between tables.

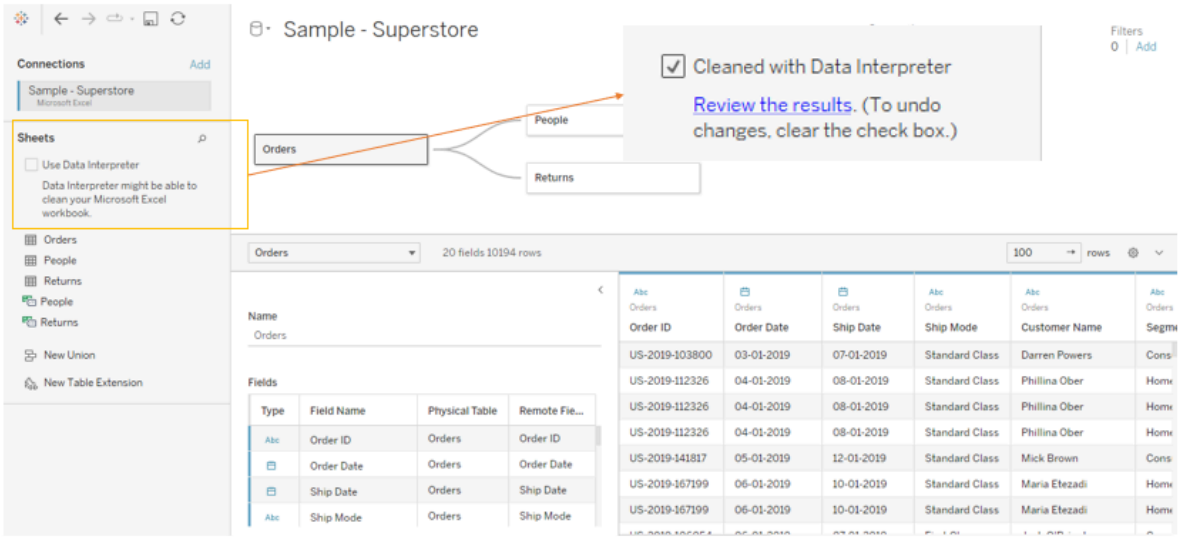
Data grid - Displays first 1,000 rows of the data contained in the Tableau data source.

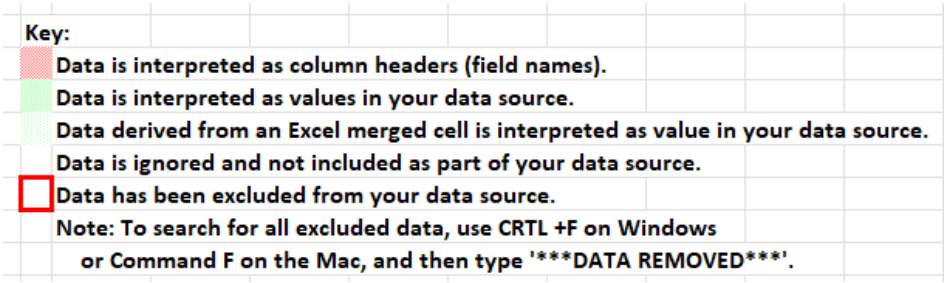
Metadata grid - Displays the fields in your data source as rows.

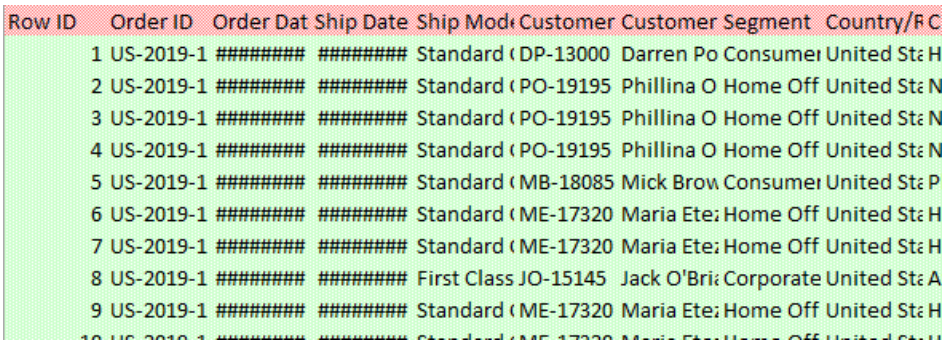
Data Interpreter in Tableau

The Data Interpreter dynamically cleans poorly formatted Excel/CSV files (extra rows, merged cells, etc.) with a single click.

NOTE: Data Interpreter is only available for Excel workbooks & CSVs.

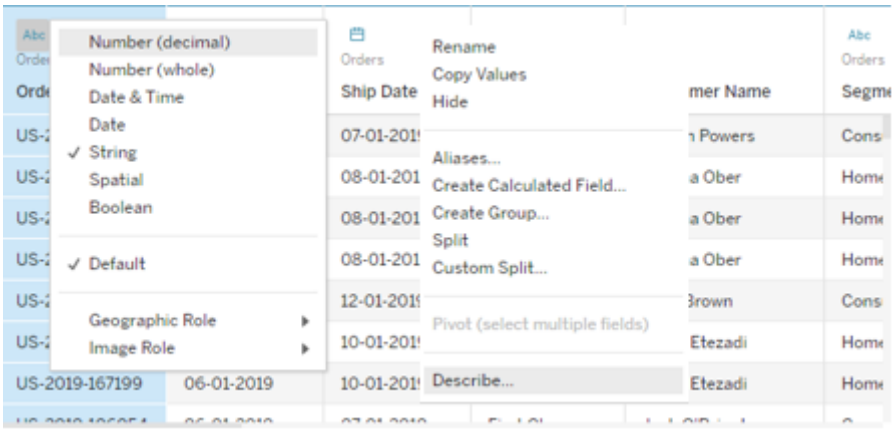






Transforming Data

The Data Interpreter dynamically cleans poorly formatted Excel/CSV files (extra rows, merged cells, etc.) with a single click.

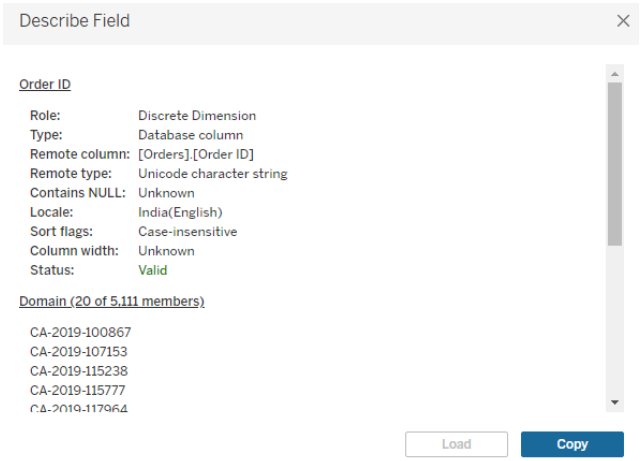


Changing Data Type:

Change Field Types or Geographic Roles, Number (decimal, whole) Date/Time, String, etc.

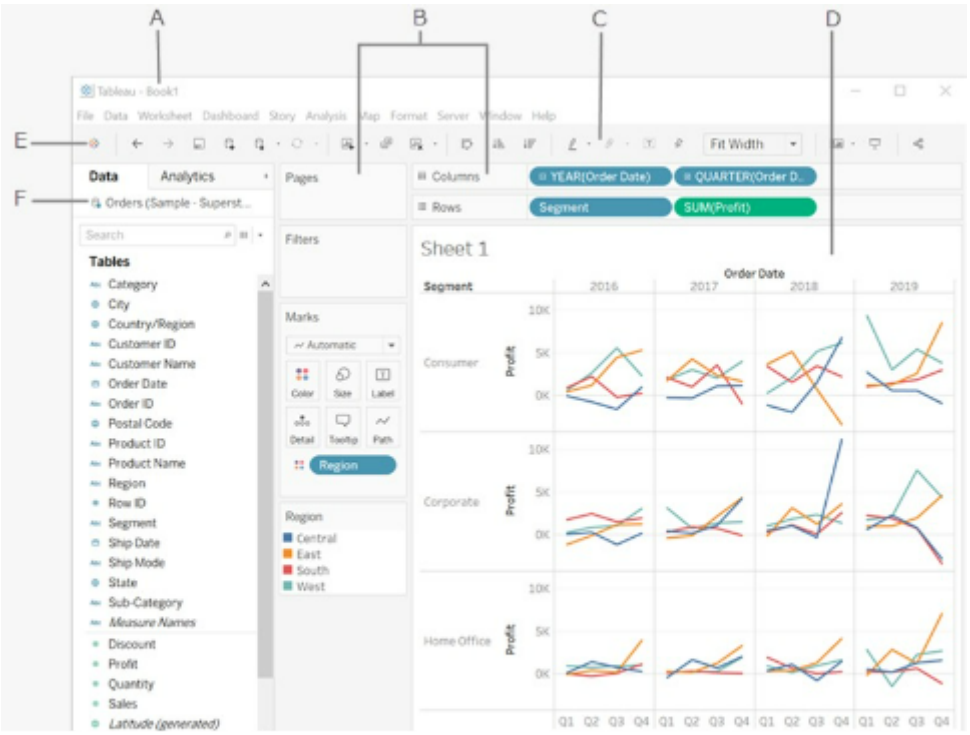
Some basic transforming features

1. **Rename fields (can also double-click the header) -** Fields are only renamed in the workbook, not the raw source.
2. **Hide unneeded columns from end-users -** Fields are only hidden in Tableau, not the raw source.
3. **Group entries in fields together -** A new field will be created, and original values will not change.
4. **Pivot multiple** fields from columns to rows.
5. **Split fields using automatic or custom delimiters -** This works like Excel's text-to-column Functionality.
6. **Describe –** Describe data is used to get the information about the particular data attribute.



The Tableau Workspace

The Tableau workspace consists of menus, a toolbar, the Data pane, cards and shelves, and one or more sheets. Sheets can be worksheets, dashboards, or stories.



A. Workbook name: - A workbook contains sheets. A sheet can be a worksheet, a dashboard, or a story. For more information, see Workbooks and Sheets.

B. Cards and shelves: - Drag fields to the cards and shelves in the workspace to add data to your view.

C. Toolbar: - Use the toolbar to access commands and analysis and navigation tools.

D. View:- This is the canvas in the workspace where you create a visualizati
referred to as a "viz").

E. Start Icon :- Click this icon to go to the Start page, where you can connect to data. For more information, see Start Page.

F. Side Bar :- In a worksheet, the side bar area contains the Data pane and the Analytics pane.

G. Data Source Tab: - Click this tab to go to the Data Source page and view your data. For more information, see Data Source Page.

H. Status bar:- Displays information about the current view.

I. Sheet tabs - Tabs represent each sheet in your workbook. This can include worksheets, dashboards, and stories. For more information, see Workbooks and Sheets.

Combining Data

Tableau Desktop provides many options for combining and modelling your data source connections, including relationships, joins, unions and blending.

Relationships: Leverage related fields between table to create contextually appropriate joins.

Joins: Merges multiple tables based on a join clause to create a new fixed table.

Unions: Appends two or more tables to form a new combined table.

Blending: Visualizes data from separate sources within the same view via blended fields.

Tableau Data Model

Every data source that you create in Tableau has a data model. You can think of a data model as a diagram that tells Tableau how it should query data in connected database tables.

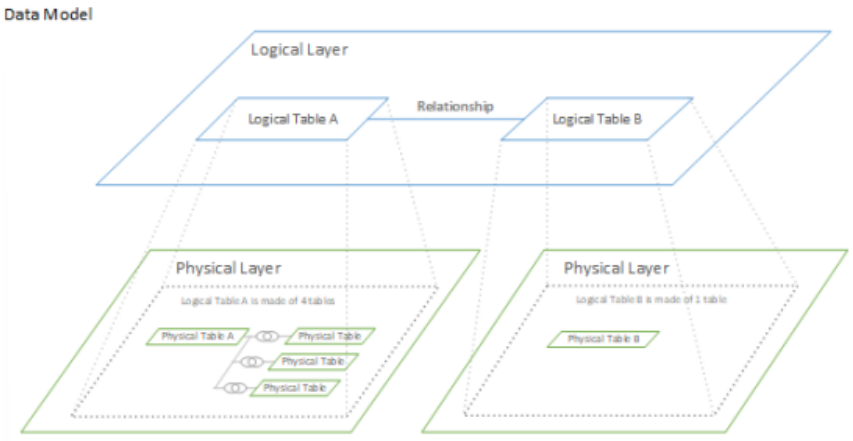
The tables that you add to the canvas in the Data Source page creates the structure of the data model. A data model can be simple, such as a single table. Or it can be more complex, with multiple tables that use different combinations of relationships, joins and unions.

Tableau Desktop's Data Model is split in two distinct layers:

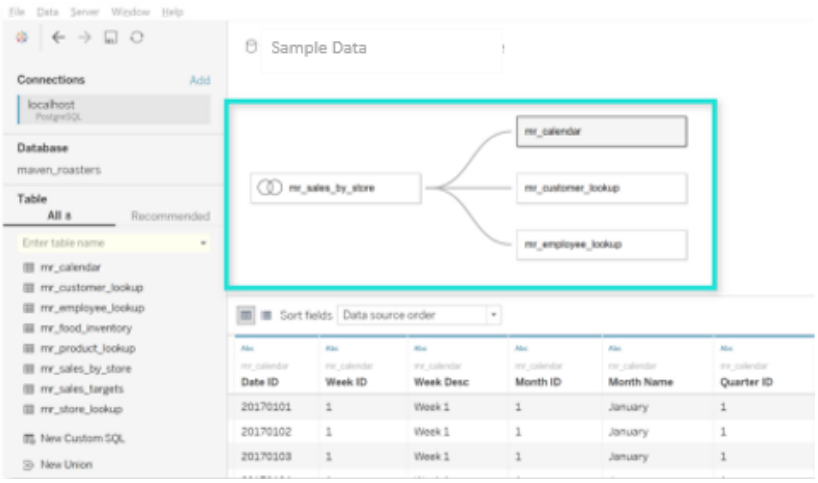
1) **Logical Layer**
2) **Physical layer.**

Both layers can be used in conjunction with each other.

Source data should be clearly understood prior to enabling either data model layer, however the logical layer is generally more forgiving than the physical layer.



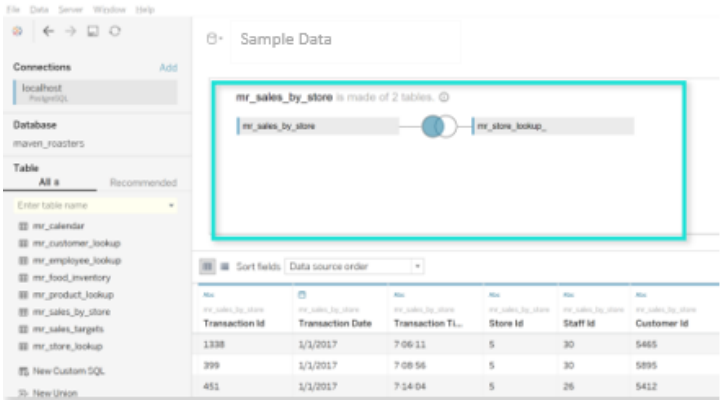
Logical Layer



The default view that we first see in the data source page canvas is logical layer of the data source. You combine data in the logical layer using relationships (or noodles). Think of this layer as the Relationships canvas in the Data Source page.

Logical layer represents the canvas for creating relationships between tables.

Physical Layer



The next layer is the physical layer. You combine data between tables at the Physical layer using Joins and Unions. Each logical table contains at least one physical table in this layer. Think of the physical layer as the Join/Union canvas in the Data Source Page. Double click a logical table to view or add joins and unions.

Physical layer represents the canvas for creating unions & joins between tables.

Note: The Physical layer joins tables together, while the Logical layer keeps tables separate, but defines relationships between them.

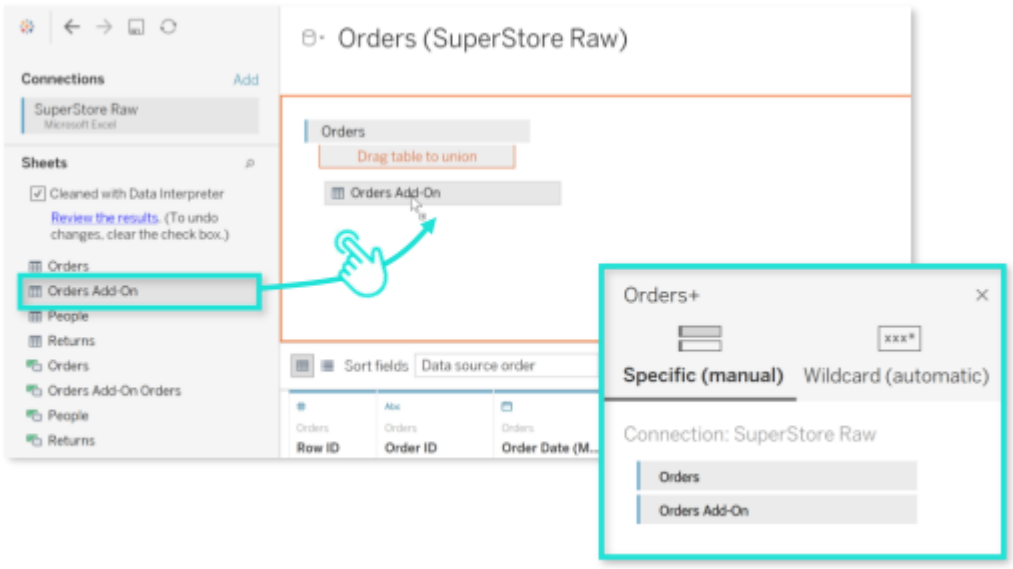
Difference between Logical Layer and Physical Layer

Logical Layer	Physical Layer
1) Relationships canvas in the data source page. 2) Tables that you drag here are called logical tables. 3) Logical tables can be related to other logical tables. 4) Logical tables are like containers for physical tables. 5) Level of details is at the row level of the logical table. 6) Logical tables remain distinct (normalized), not merged in the data source.	1) Joins/Union canvas in the data source page. 2) Tables that you drag here are called physical tables. 3) Physical tables can be joined or unioned to other physical tables. 4) Double click a logical table to see its physical tables. 5) Level of details is at the row level of merged physical tables. 6) Physical tables are merged into a single, flat table that defines the logical table.

Different Data Operations

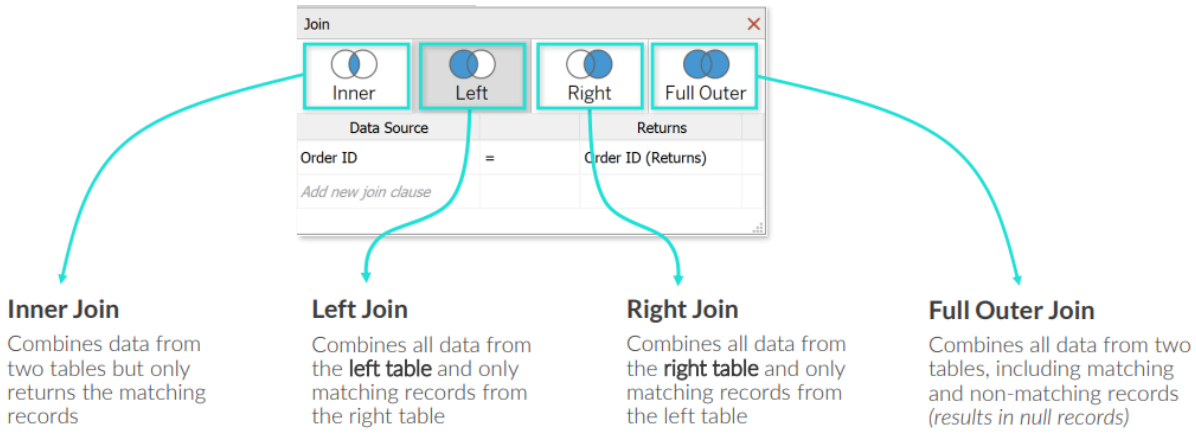
Unions

Unions append (or “stack”) rows of data from one table into another.



Joins

Joins combine data sources using keys depending on the source, you can perform inner, left, right, or full outer joins.



Note: Make sure that your join key(s) are unique to avoid duplicating records this can distort the data and skew your analysis.

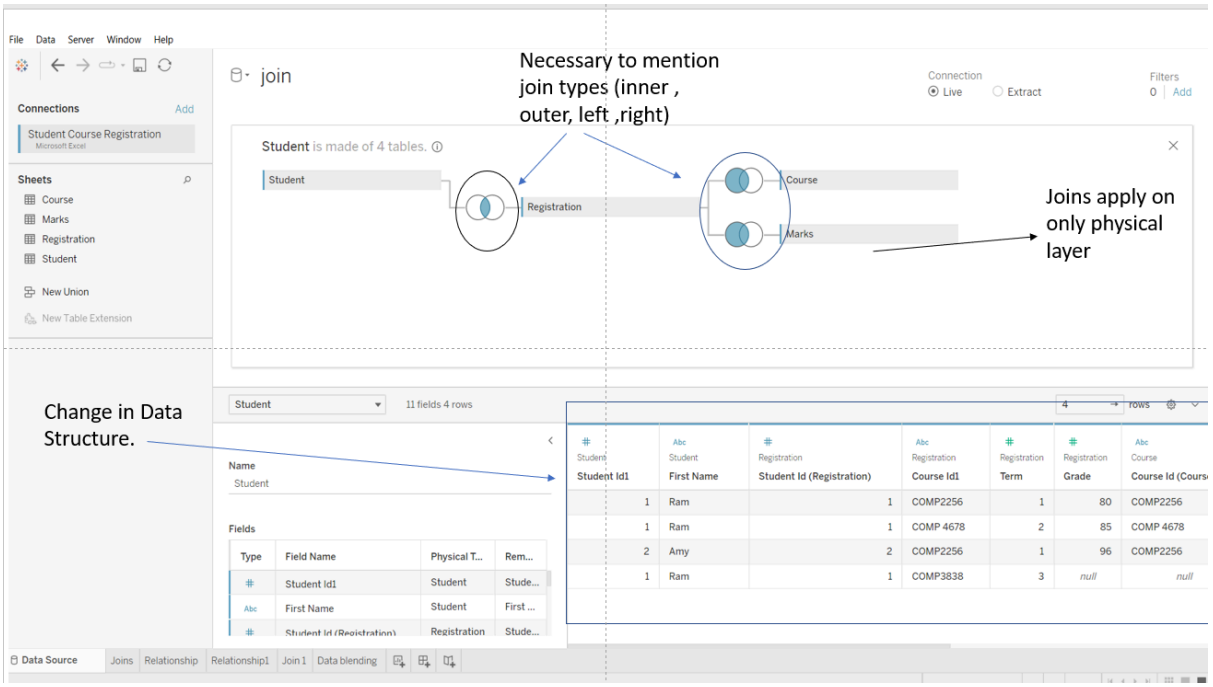
How to apply joins

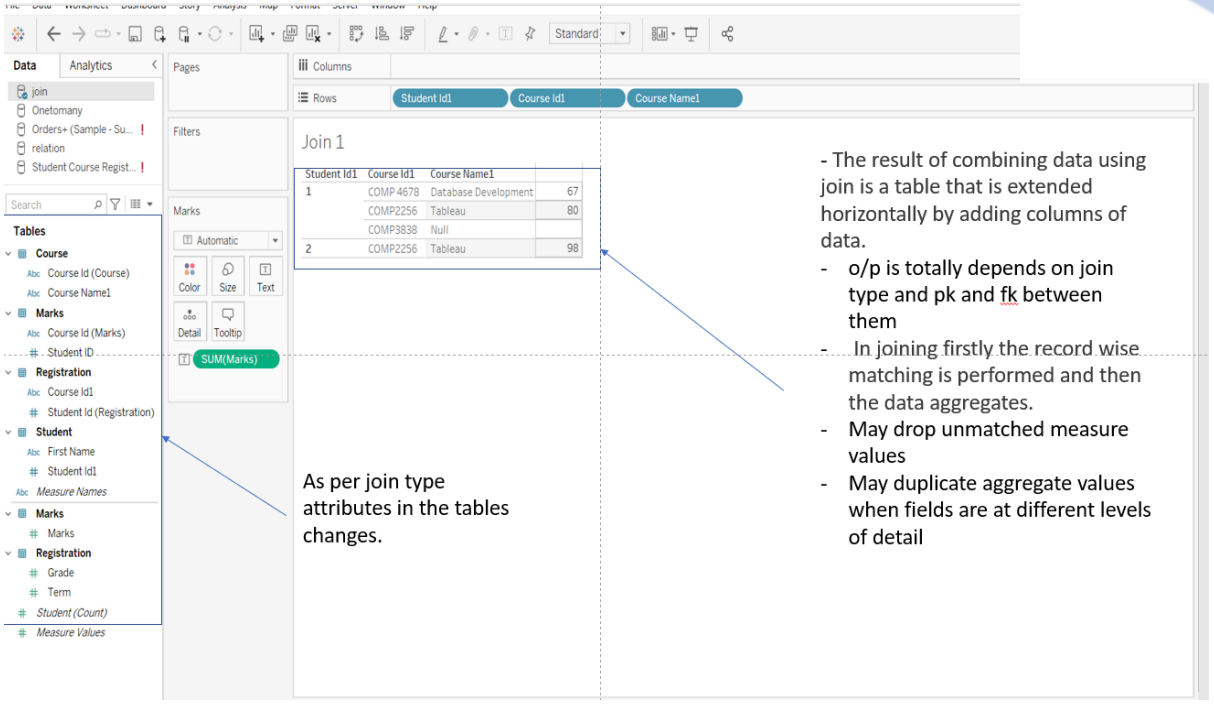
In Tableau, a "join" is used to combine data from multiple tables into a single view. The most common type of join in Tableau is an "inner join," which combines only the rows that have matching values in both tables.

To perform a join in Tableau:

1) Start by connecting to the data sources you want to join.
2) Drag the first table onto the canvas and then drag the second table onto the canvas as well.

3) Click on one of the tables and then click the "Join" button at the bott
"Data" pane.
4) In the "Join" dialog box, choose the type of join you want to perform (e.g., inner join, left join, right join, or full outer join).
5) Select the columns that you want to use to join the tables (these should be columns that have matching values in both tables).
6) Click the "OK" button to perform the join.
7) After the join is complete, you can create a new view that combines the data from both tables. You can also customize the join by adding filters, creating calculated fields, or changing the join type or columns used for the join.





Characteristics of joins

Joins are a more static way to combine data. Joins must be defined between physical tables up front, before analysis, and can't be changed without impacting all sheets using that data source. Joined tables are always merged into a single table. As a result, sometimes joined data is missing unmatched values, or duplicates aggregated values.

1) Are displayed with Venn diagram icons between physical tables
2) Require you to select join types and join clauses
3) Joined physical tables are merged into a single logical table with a fixed combination of data
4) May drop unmatched measure values
5) May duplicate aggregate values when fields are at different levels of detail
6) Support scenarios that require a single table of data, such as extract filters and aggregation

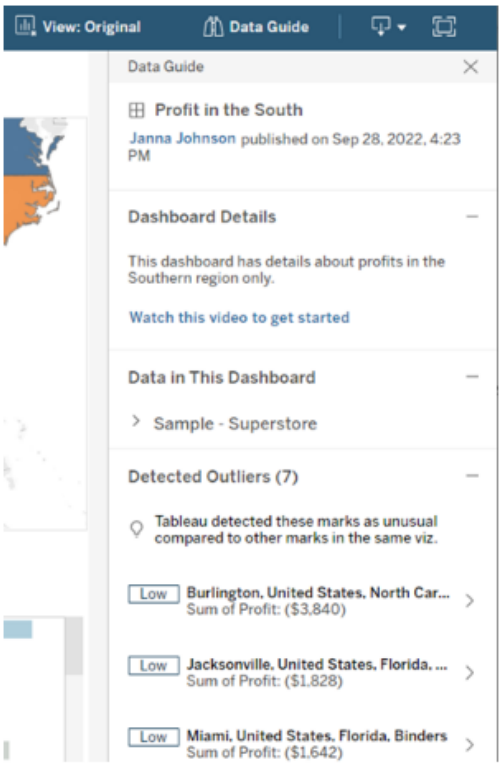
Data Guide in Tableau

Data Guide provides helpful information about a dashboard and insights about the data behind it. Data Guide allows dashboard creators to provide more explanatory context for end users – like descriptions and links to resources – directly in the dashboard. And Data Guide automatically surfaces insights powered by Explain Data and Data Change Radar to help users find outliers, identify notable data changes and learn about explanations for a mark.

Explain the Viz (powered by Explain Data) identifies outlier measures and potential key drivers behind them. Data Change Radar (Tableau Cloud only) tracks values in your

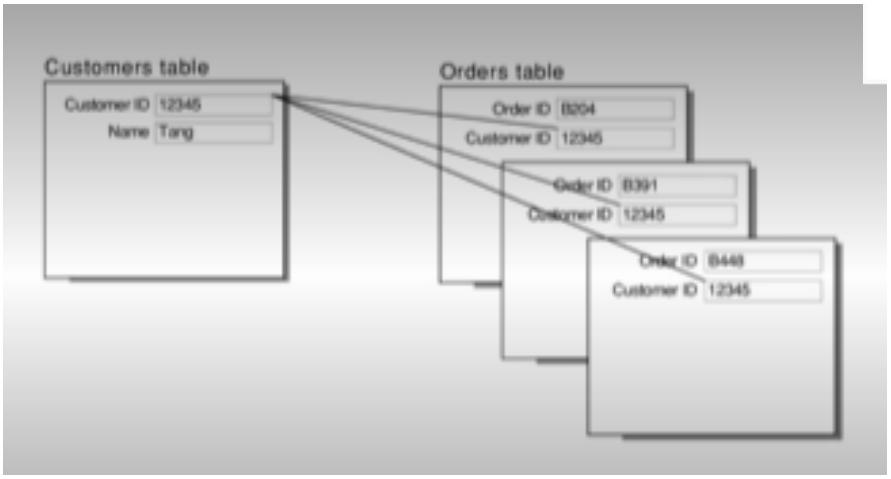
dashboard over time and automatically flags values outside of the normal p
data refreshes.

These contextually relevant details can help dashboard users navigate and use new dashboards more easily, allowing users to find insights faster, trust that they're looking at the right data, provide context for data and establish confidence in their understanding of the viz.



Relationships

Relationships are a dynamic, flexible way to combine data from multiple tables for analysis. A relationship describes how two tables relate to each other, based on common fields, but does not merge the — tables together. When a relationship is created between tables, the tables remain separate, maintaining their individual level of detail and domains.



Note: Relationships are not supported for data source connections including Cube Sources, Stored Procedures, Splunk, JSON, Salesforce (single source only), and SAP HANA (with OLAP chosen)

Performance Options in Relationships

Performance options are settings that define cardinality (uniqueness) and referential integrity (matching records) between two tables within a relationship.

Cardinality

Cardinality refers to the nature of the relationship between two or more fields in different data sources. Cardinality is important in Tableau when creating data joins or blending data from multiple sources to establish how the data should be combined or related to each other.

Tableau recognizes three types of cardinality in relationships:

- **One-to-One (1:1) Cardinality:** In a one-to-one relationship, each value in one field corresponds to exactly one value in another field. For example, if you have a "Customer ID" field in one data source and a "Customer ID" field in another data source, and each customer ID is unique in both sources, then the cardinality between these two fields would be one-to-one. In Tableau, one-to-one cardinality relationships are typically used when you want to combine data from two sources that have a direct, one-to-one relationship, such as merging customer details with order details.

- **One-to-Many (1:N) Cardinality:** In a one-to-many relationship, each value in one field can correspond to multiple values in another field. For example, if you have a "Customer ID" field in one data source and an "Order ID" field in another data source, where each customer can have multiple orders, then the cardinality

between these two fields would be one-to-many. In Tableau, one-to-
cardinality relationships are commonly used when you want to com
from a primary source with multiple related records in a secondary source, such as joining customer data with order data.

- **Many-to-Many (N:M) Cardinality:** In a many-to-many relationship, multiple values in one field can correspond to multiple values in another field. Many-to-many relationships are more complex and typically require additional data modeling techniques such as bridging tables or data blending to establish the relationship. Tableau does not directly support many-to-many relationships, but they can be modeled using advanced techniques such as custom SQL, data blending, or data prep tools.

Referential Integrity

Determines the type of join used to get dimension values for a measure during analysis

Some Records Match – if some values in the selected fields don't match or you aren't sure. Tableau uses Outer Joins, and all measures will show (even unmatched measures)

All Records Match – if all values in selected fields have a match in both tables. Joins are optimized with Inner Joins, but records can be removed/missing due to unmatched values.

NOTE: "All records match" treats records as if there are no null records in matching fields.

Advantages of Relationships

Relationships provide several advantages over using joins for multi-table data:

• You don't need to configure joins types between tables. You only need to select the fields to define the relationship.

• Related tables remain separate and distinct; they are not merged into a single table.

• Relationships use joins, but they are automatic. Tableau automatically selects join types based on the fields being used in the visualization. During analysis, Tableau adjusts join types intelligently and preserves the native level of detail in your data.

• Tableau uses relationships to generate correct aggregations and appropriate joins during analysis, based on the current context of the fields in use in a worksheet.

• Multiple tables at different levels of detail are supported in a single data source.

• You can build data models that contain more tables, and reduce the numb sources needed to build a viz

• Unmatched measure values are not dropped (no accidental loss of data).

• Avoids data duplication and filtering issues that can sometimes result from joins.

• Tableau will generate queries only for the data that is relevant to the current view.

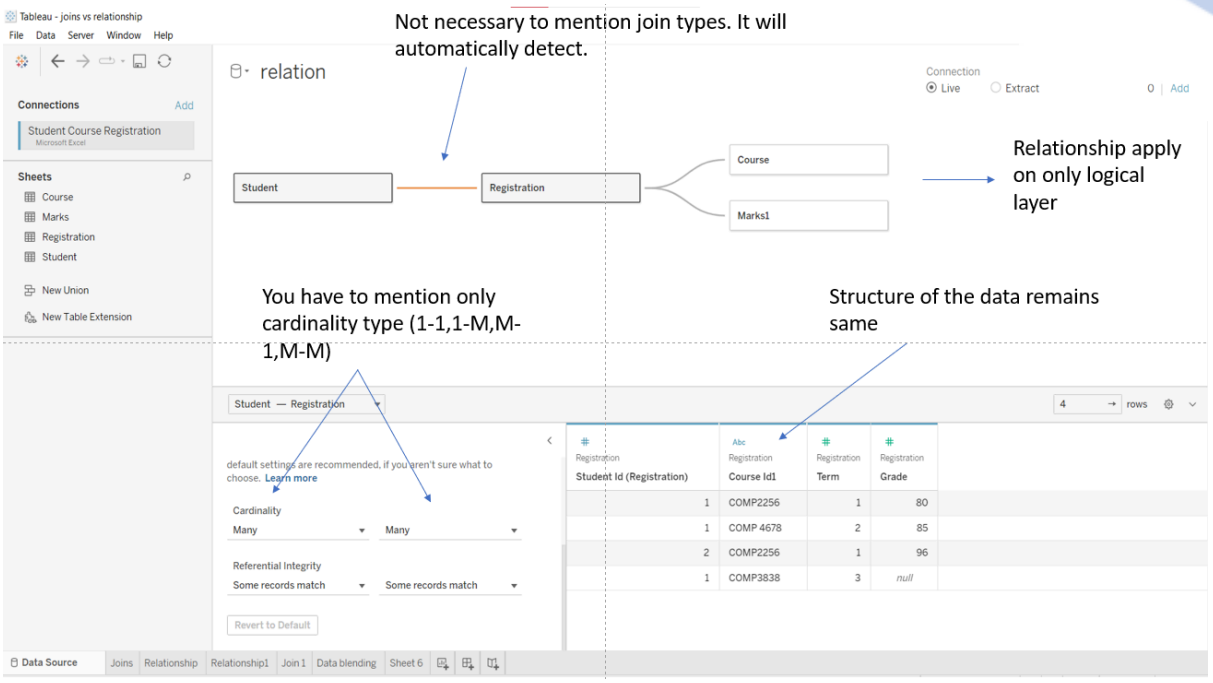
How to apply Relationships?

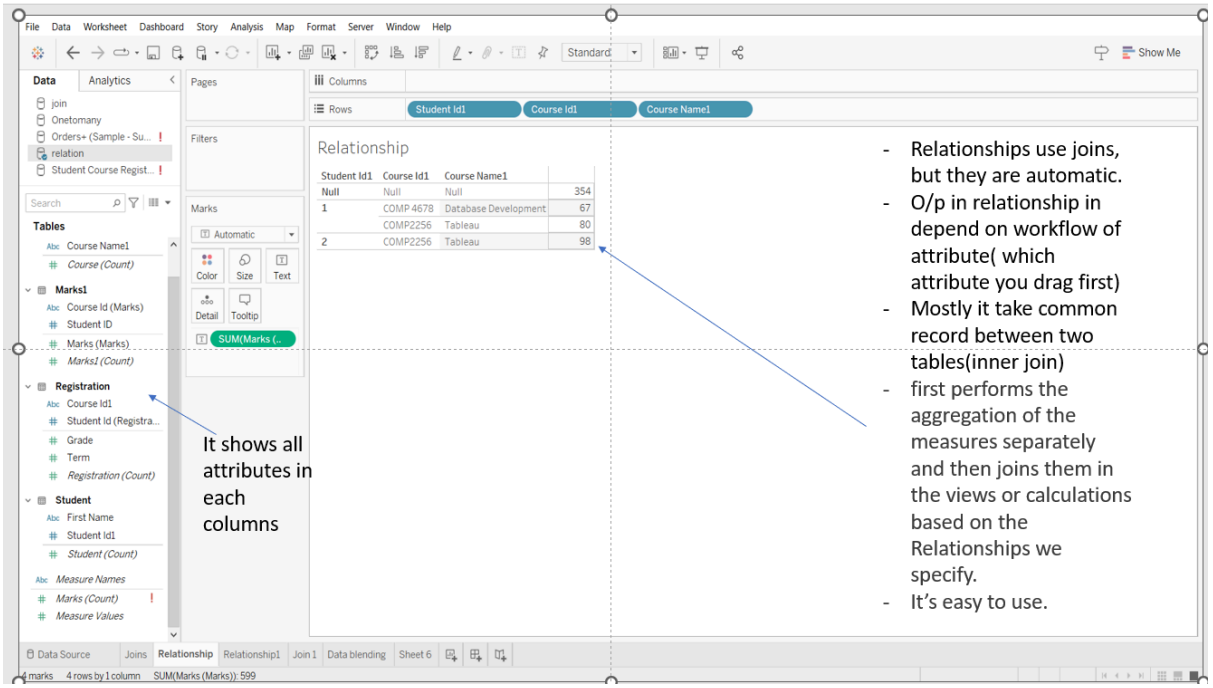
In Tableau, a "relationship" is a way to connect two or more tables in a data source without actually merging or combining them into a single table.
A relationship allows you to use fields from multiple tables in a single view, while still maintaining the integrity of the original data. This is particularly useful when working with large or complex datasets that cannot be easily combined into a single table.

To create a relationship in Tableau:

1) Start by connecting to the data sources you want to use.
2) Drag the first table onto the canvas and then drag the second table onto the canvas as well.
3) Click on one of the tables and then click the "Add" button at the bottom of the "Data" pane.
4) In the "Join Clauses" dialog box, select the columns that you want to use to create the relationship between the tables. These should be columns that have matching values in both tables.
5) Click the "OK" button to create the relationship.

After creating the relationship, you can use fields from both tables in a single view. Tableau will automatically create the appropriate join based on the relationship between the tables, allowing you to analyse and visualize your data in new ways. You can also customize the relationship by adding filters, changing the join type, or modifying the columns used for the relationship.





Some extra Points

1. Relationships are used to connect tables or data sources within a specific worksheet, while joins are used to combine data from different data sources at the workbook level, making the combined data available for use in multiple worksheets within the same workbook.
2. Relationships may cut down on duplication when tables are related and even further performance improvement.

3. Relationships often offer performance improvements based on selec
   relational criteria and not including unnecessary columns in the dat
4. Relationships may make relating tables of data easier and more flexible.
5. Relationships may not be the best tool for exploring or relating unclean data.
6. Relationships work best against tables which don't need to be re-shaped.

Difference b/w Relationships and Joins

RELATIONSHIPS	JOINS
Defined between logical tables in the Relationship canvas (logical layer)	Defined between physical tables in the Join/Union canvas (physical layer)
Don't require you to define a join type	Require join planning and join type
Act like containers for tables that are joined or unioned	Are merged into their logical table
Only data relevant to the viz is queried. Cardinality and referential integrity settings can be adjusted to optimize queries.	Run as part of every query
Level of detail is at the aggregate for the viz	Level of detail is at the row level for the single table
Join types are automatically formed by Tableau based on the context of analysis. Tableau determines the necessary joins based on the measures and dimensions in the viz.	Join types are static and fixed in the data source, regardless of analytical context. Joins and unions are established prior to analysis and don't change.
Rows are not duplicated	Merged table data can result in duplication
Unmatched records are included in aggregates, unless explicitly excluded	Unmatched records are omitted from the merged data
Create independent domains at multiple levels of detail.	Support scenarios that require a single table of data, such as extract filters, row level security, aggregation

Live vs Extract

Live Connection:

This refers to a data source which contains direct connections to the real-time data.

Live connections can be used at a place where the data is real-time data which when get updated, so our visualization also gets updated automatically.

EXAMPLE– real time transactions.

Properties:

1. Live connections are slower for complex visualizations.
2. Live connections are used especially in less complex visualizations w
   data sets, filters, calculations etc.
3. Live connections get refreshed when there is a change in the original data source.
4. Live connections always rely on the database for all queries.
5. A Live connection in Tableau basically means that Tableau is querying and reading directly from your database.

NOTE: Live connections are NOT supported in the Tableau Public

Extracts

Extracts pull data from the underlying source and into Tableau's Hyper Extract file format, allowing you to refresh data on regular basis.

Extract files are the local copy of the data source that you can use to make the view. Extracts can be used at a place where the view can be created by a subset of the data source.

EXAMPLE– Keeping the record of the monthly trends of a hospital.

Properties:

1. Extracts are much faster for visualization.
2. Extracts are used especially in more complex visualizations with large datasets, filters calculations etc.
3. Extract is a snapshot of the data, so they need to be refreshed to receive updates from the original data source.
4. However extracts databases should not always be optimized.

NOTE: Automatic refresh requires Tableau Server/Online

Why Use Extracts?

• Can improve performance (better performance than most database sources)

• Additional functionality (exposes additional features that may be limited with live connections)

• Offline access (work without a connection to the network or database)

• Supports very large datasets (into the billions of rows)

• Fast to create (quick extract times)

Types of Refresh

The extracts in tableau are refreshed in two ways:-

1. *Incremental Refresh*
2. *Full Refresh*

1. INCREMENTAL REFRESH

1. A refresh in which only the rows and columns are added which are new since the previous time you extracted the data.
2. Most data sources support an instrumental refresh as it takes less time.
3. Although it is faster but once in a period of time the user needs to use the full refresh to ensure that the data is most up to date.

2. FULL REFRESH

1. It refers to the refreshment of the entire extract
2. In this refresh, all of the rows are replaced with the data in the original data source.
3. This kind of refresh ensures that the user has an exact copy of the original data.
4. A full refresh can sometimes take a long time & be expensive on the database.

Difference between LIVE and EXTRACT

LIVE AND EXTRACT CONNECTION

LIVE	EXTRACT
A data source that contains direct connection to underlying data, which provides real-time or near real time data.	Saved subsets of data that you can use to improve performance or to take advantage of tableau functionality that might not be available in original data.
With live connection,Tableau makes queries directly against the data source.When you refresh the tableau view it will update based on underlying data.	When refreshing the data in an extract, you have the option of either doing a full refresh or an incrementel refresh which adds only new rows.
Your Tableau workbook will not work if you are unable to connect to the data source.	You would be able to use the extract even if you are not connected to your data source.
You need to enter your credentials while using a live connection.	You do not need to enter credentials while using extract.

Data Blending

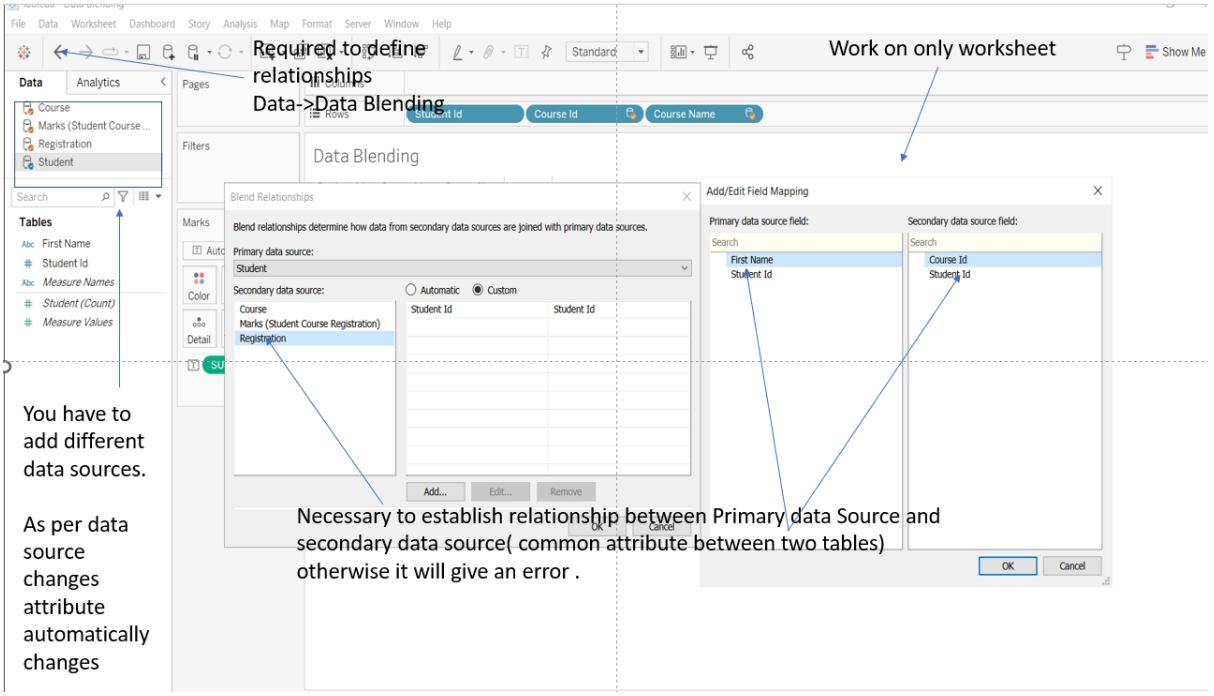
TABLEAU NOTES

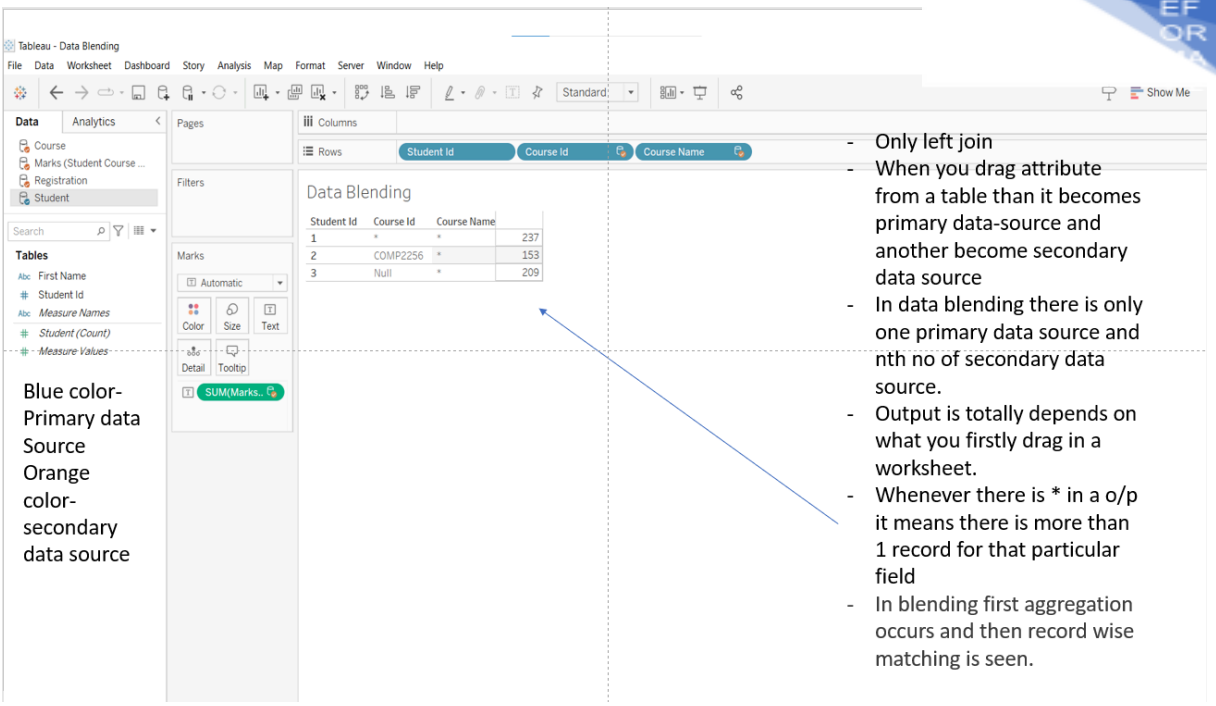
In Tableau, data blending refers to the process of combining data from mul sources within a single visualization or workbook.

Tableau supports data blending through a feature called "Data Connections," which allows users to connect to and blend data from multiple sources. This enables users to analyze and visualize data from disparate sources without the need for complex data modeling or consolidation.

Data blending in Tableau involves creating relationships between different data sources based on common fields or dimensions. Tableau then uses these relationships to combine data from the different sources into a single view or dashboard.

of the advantages of data blending in Tableau is that it allows users to work with large and complex data sets that may be difficult to consolidate into a single database or spreadsheet. Additionally, it enables users to create more sophisticated analyses by incorporating data from multiple sources into a single visualization.





Relationship vs Joins vs Blending

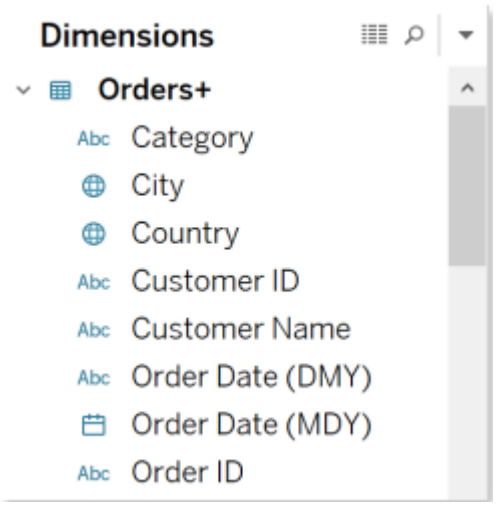
Feature	Relationship	Joins	Blending
Definition	A feature in Tableau that allows you to define and manage relationships between multiple data sources within a workbook.	A method used to combine data from different tables in a single data source using common fields.	A way to combine data from different data sources by aligning fields with similar data, without explicitly defining relationships.
Usage	Ideal for combining data from different data sources with a defined relationship, also included cardinality.	Ideal for combining data from different tables within the same data source based on shared fields.	Ideal for combining data from different data sources without a defined relationship or when the relationship between data sources is complex.
Relationship Creation	Requires manual creation of relationships using the Data source page in Tableau Desktop	Requires specifying the join type and joining fields.	Provided limited flexibility for all relationship creation. Each thing you have to do manually. And for each you need to create new relationships.
Layer	Relationship in Tableau works in logical layer	Joins in Tableau works in physical layer.	Blending in Tableau works in sheet level.

Data Integration	This allows for seamless integration of data from different data sources, enabling cross-data source filtering, parameter actions, and visualizations that combine data from multiple sources.	Tableau combines the data from the joined tables into a single data source, which can then be used to create visualizations. Joins may require additional calculations or filters to integrate the data and achieve desired visualizations.	Data inte relationships and joins, as it requires manual alignment of fields and calculations to integrate the data from different sources.
Data Duplication	No Data Duplication	Data Duplication (Additional rows)	No data duplication
Cardinality	Supports different types of data cardinality including 1-1, 1-M, M-M relationships between data sources.	Supports 1-1, 1-M relationships between tables within the same data source.	Supports 1-1, 1-M relationships between the data sources.
Performance	Typically provided better performance compared to joins and data blending, as it leverages data source relationships.	Performance depends on the type of join used and the size of data being combined.	Performance depends on the size of complexity of the data sources being blended, and the field alignment process may impact

MEASURES & DIMENSIONS

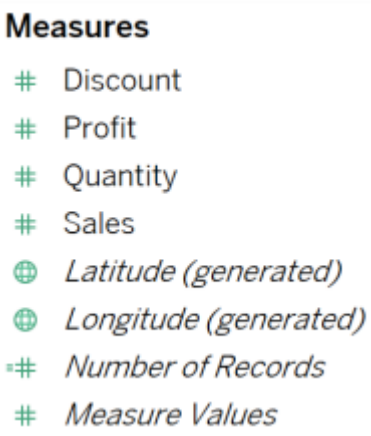
Dimensions are qualitative fields that contain information used to categorize, segment or filter your data. Common examples include: Name/ID's, Geographic fields, Dates/Times.

Dimensions are mostly Blue in colour.



Measures are quantitative, numerical fields that can be counted or aggregated (sum, average, median, max, etc.). Common examples include: Sales/Profit, Ratios/Percentages, etc.

Measures are mostly Green in colour.



By default, Tableau will divide the columns into Dimensions and Measures when you connect to any data source. In general, all the numerical columns are placed under the Measures pane. In contrast, the text columns or fields with categorical information are placed under the Dimensions pane.

Remember, you can drag the required field from Measure to Dimension or vice versa. Or right-click and select the Convert To Measure or Convert to Dimension option.

A simple example can be an Order ID or Employee ID. Tableau initially considers them as Measures, but you must drag them back to the Dimension pane.

DISCRETE VS CONTINIOUS

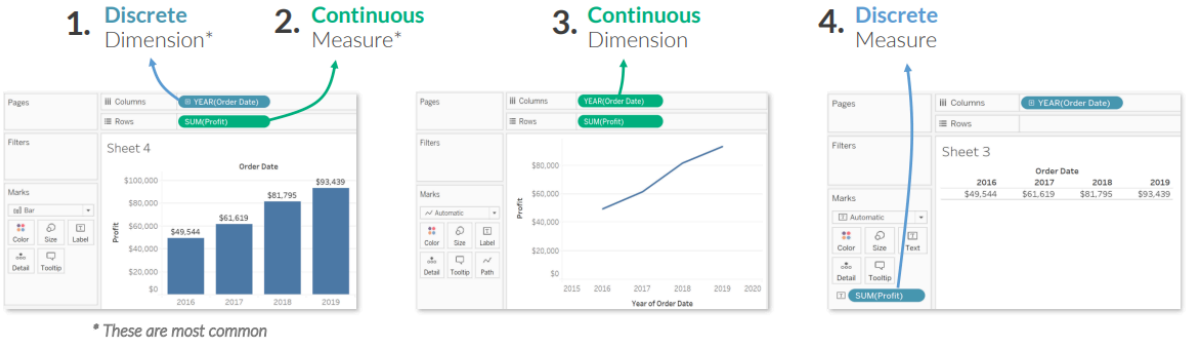
In mathematical terms, data fields can be described as either discrete or continuous.

1. Discrete fields contain a finite set of distinct values (Year, Category, Country, etc.)

2. Continuous fields can contain an infinite range of values (Age, Temp
   Profit, etc.)

Discrete are blue and continuous are green.

In Tableau, discrete fields add a new header to a view; continuous field add a new axis to a view.



DATA TYPES IN TABLEAU

There are mainly Seven Basic types in Tableau name as Text, Numeric, Boolean, Date, Date & Time, Geographic, and Mixed Data Types.

Data Type	Icon
Text (string) values	Abc
Date Values	
Date & Time Values	
Numerical Values	#
Boolean Values	T|F
Geographic Values	
Cluster Group	

1. Text (String) Data Types

The arrangement of zero or more characters enclosed in a single inverted comma or double inverted comma is known as a string. The inverted comma can be incorporated in the string by placing it twice. The string can be of a single word or group of words.

Examples of the string are 'Hello', "Hello World", etc.

String are divided into two types:

a. Char Data Type

In this type of Data Type, the length of the string is predetermined, and fixe memory allocation. If the user enters the string more than the fixed length, then an error is generated. It can store the alphanumeric data in this type of data type.

b. Varchar2 Data Type

In this type of Data Type, the length of the string is not predefined or fixed. Any number of characters can be entered in this type of string without worrying about the memory space allocation. This data type adjusts itself according to the string length entered. Alphanumeric data can be entered in this Data Type.

2. Date and Date & Time Data Types

Tableau understands almost all the date formats. The default date arrangement is "dd-mon-yy" or "dd-mon-yyyy." I want to enter a date in string data type and want the string to understand that it is a date then place # symbol before typing the date — example #3 Oct 1992. Time Data Type is an extension of Date Data Type in Tableau. It is stored in the form of Century, Year, Month, Day, Hour, Minute, and Second.

3. Number Data Types

Number Data Type can either integers or floating-point. When the float is used, it might happen that we may not be able to accumulate the decimal points after a limit. Thus, integers are mostly used. Even if floats are used, then they are often rounded up to integer by using Round function.

4. Boolean Data Types

The boolean data type has only two logical values, either True or False.

5. Mixed Values

Tableau is capable of identifying the data type by itself. If the values added in the data field does not match the data type, Tableau handles it with one of the other ways as shown in the table below;

Mapped Data Type	Treatment of other Data Types in the field
String	Dates and numbers are considered as text. A null value is not created.
Dates	Numbers are considered as dates in order of 1/1/1800. The text is null.
Number	The date is considered as the number of days. The text is null.
Boolean	Text, numbers, date, all are null.

Mixed value often creates a problem when you analyze data. To overcome tl
create a column that does not contain mixed amounts or format the empty
it with the underlying data source.

GEOGRAPHIC ROLES IN TABLEAU

You can create a group to combine related members in a field. For Example: if you are working with a view that shows average test scores by major, you might want to group certain majors together to create major categories.

Geographic Role	Assign this role to a field if it contains:
Airport	International Air Transport Association (IATA) or International Civil Aviation Organization (ICAO) airport codes.
Area Code (US)	US telephone area codes; numbers only.
CBSA/MSA (US)	US Core Based Statistical Areas (CBSA), which includes Metropolitan Statistical Areas (MSA), as defined by the US Office of Management and Budget. CBSA/MSA Codes and Names are recognized.
City	Worldwide cities with population of 15,000 or more. Names are in English (UK or US), French, German, Spanish, Brazilian-Portuguese, Japanese, Korean, and Chinese (Simplified and Traditional).
Congressional District (US)	US congressional districts.
Country/Region	Worldwide countries, regions, and territories. Names are in English (UK or US), French (Canada and France), German, Spanish, Brazilian-Portuguese, Japanese, Korean, and Chinese (Simplified and Traditional), Swedish, and Thai. Tableau also recognizes, FIPS 10, ISO 3166-1 alpha 2, and ISO 3166-1 alpha 3. Names are included in various forms, including long, short, and various abbreviations.
County	Second-level administrative divisions for select countries. For example, US counties, French departments, German kriese, etc. **Note:** Second-level administrative division definitions vary by country. In Tableau, all second-level administrative divisions are geocoded with the County geographic role.
NUTS Europe	NUTS (Nomenclature of Territorial Units for Statistics) levels 1–3 codes. Codes and names, including synonyms, are supported.
Latitude	Latitude in decimal degrees. Only available for numeric fields.
Longitude	Longitude in decimal degrees. Only available for numeric fields.
State/Province	Worldwide state, province, and other first-level administrative divisions. Names are in English (UK or US), French, German, Spanish, Brazilian-Portuguese, Japanese, Korean, and Chinese (Simplified and Traditional). **Note:** Some names are available only in their local form.
ZIP Code/Postcode	ZIP codes and postcodes for select countries. For example, US five-digit zip codes, Australian four-digit postcodes, German five-digit postcodes.

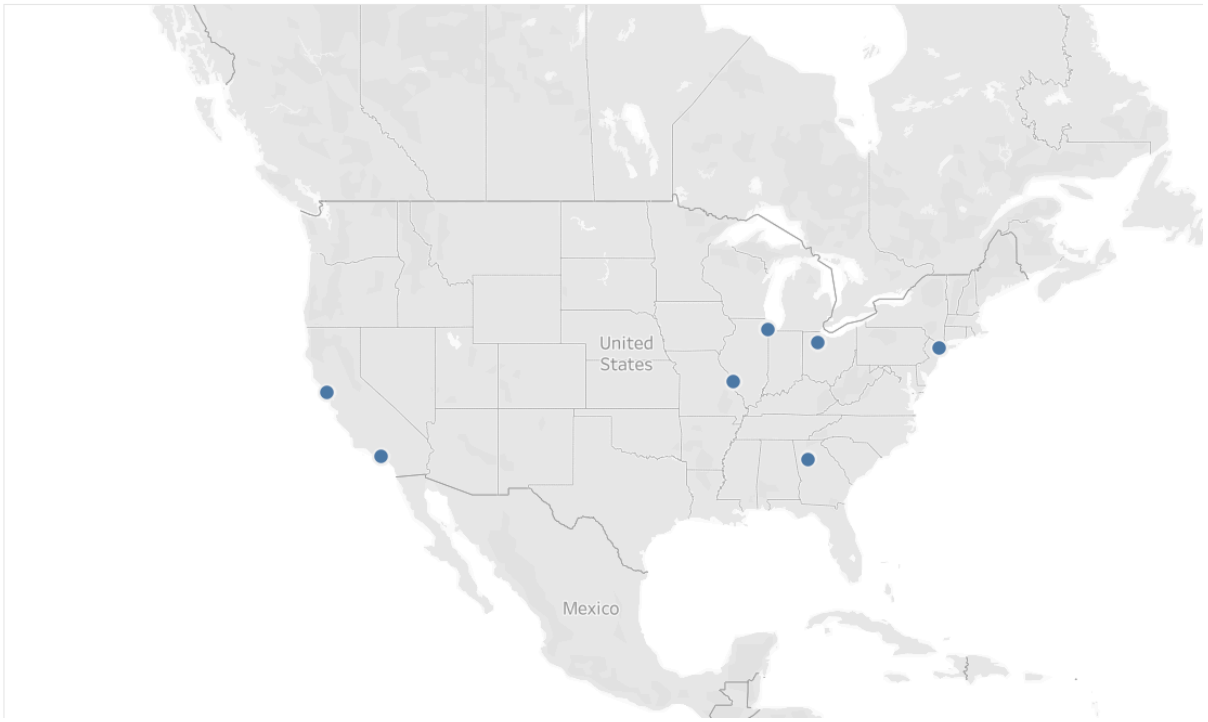
Examples:

Data:

Country	State	Postcode	Airport(IATA)	Area Code (U.S.A)	City	Latitude	Longitude	state
India	Maharashatra	412301	PNQ	212	Pune	18° 31' N	73° 55' E	Maharashatra
Canada	Alberta	T1A 0A0	YYZ	310	coldwater	44°42'N	79°40'W	Gujrat
Shri Lanka	Colombo	31000	CMB	312	Colombo	06°05'N	80°10'E	Delhi
Japan	Kanto	3620001	NRT	404	Kanto	36°15'N	139°30'E	Lakshdeep
China	Gansu	73100	PEK	415	Ganyu	34°50'N	119°08'E	Karnataka
Russia	Altai Territory	649007	VKO	314	Aim	59°00'N	133°55'E	West Bengol
United State of america	Florida	32118	MIA	420	California	37°30'N	119°30'W	Rajasthan

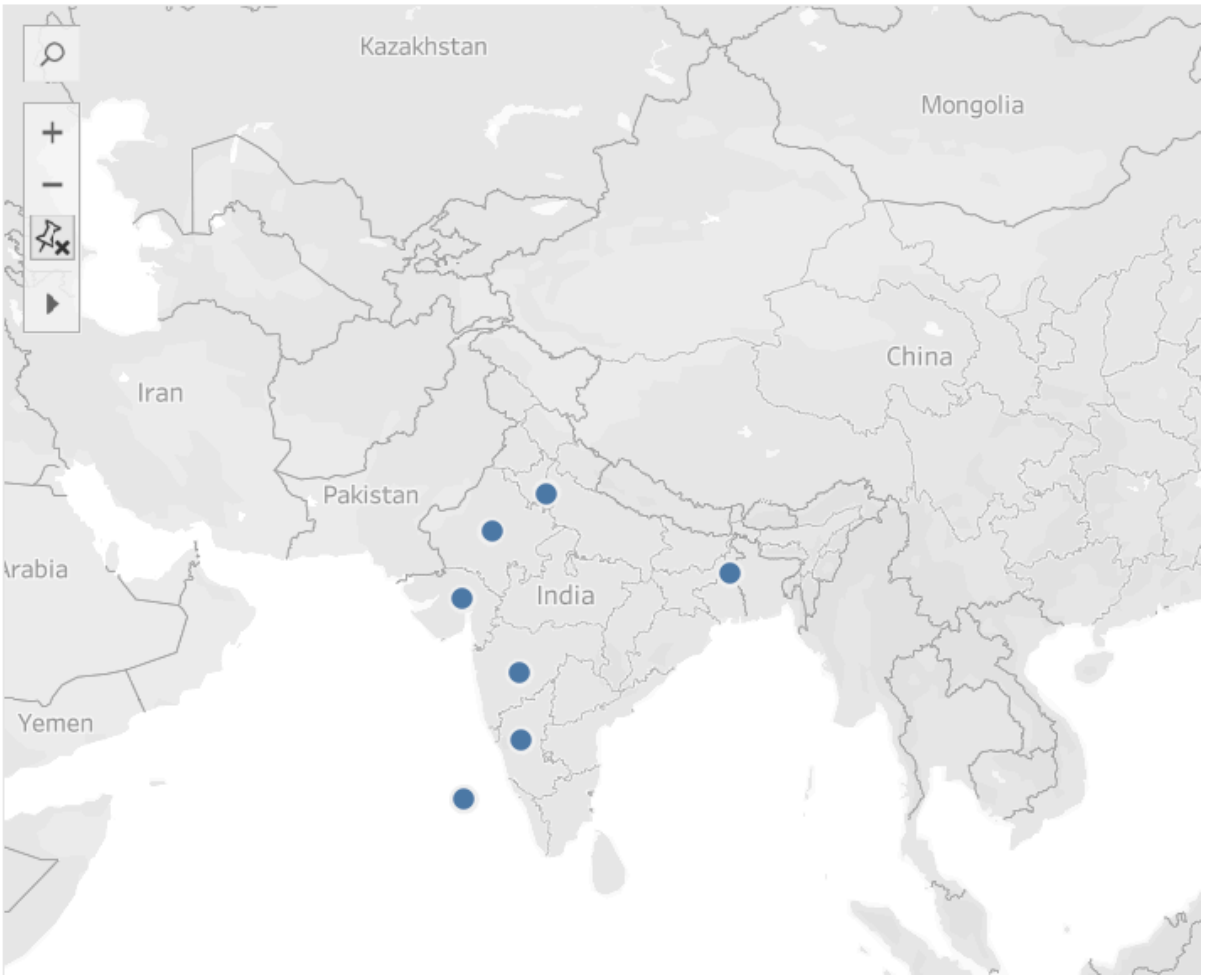
Airport

Area Code

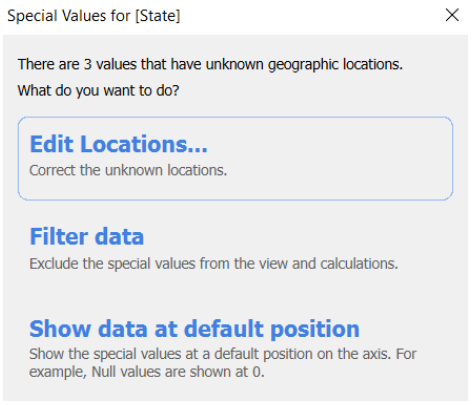


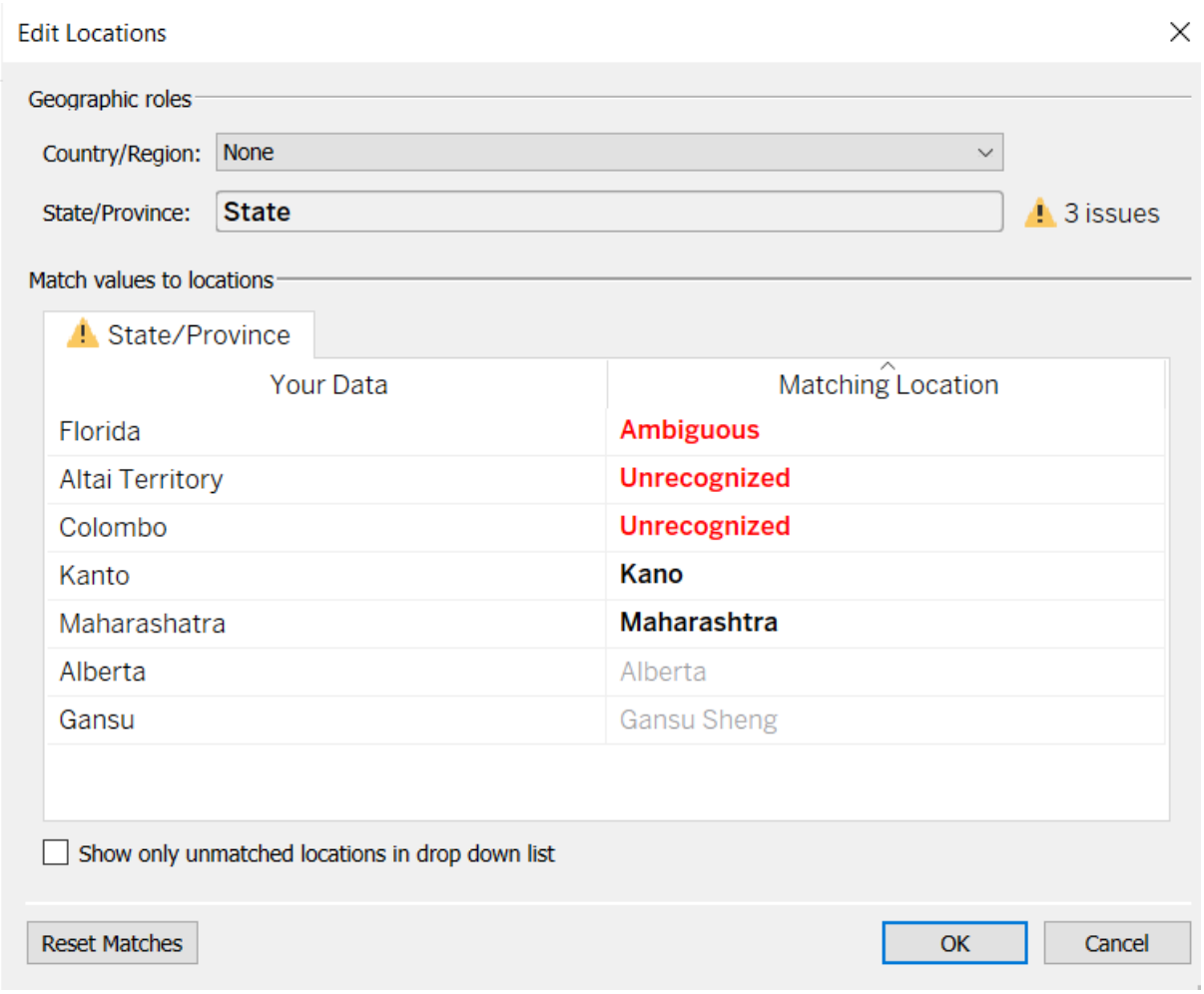
Country

State



If there is certain location that are not detected by tableau for that you are s
fill that one.
On the bottom right you see unknown records click on that

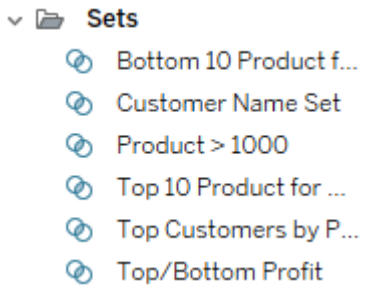




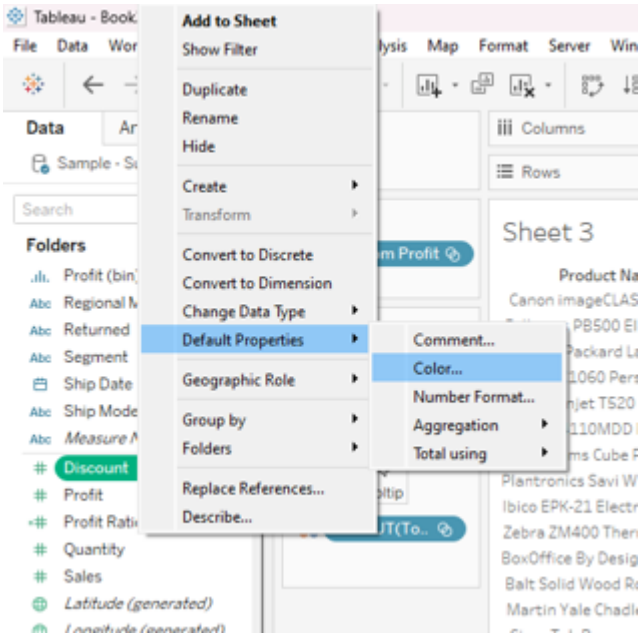
Fill that Unrecognized location.

SOME ADDITIONAL POINTS TO REMEMBER

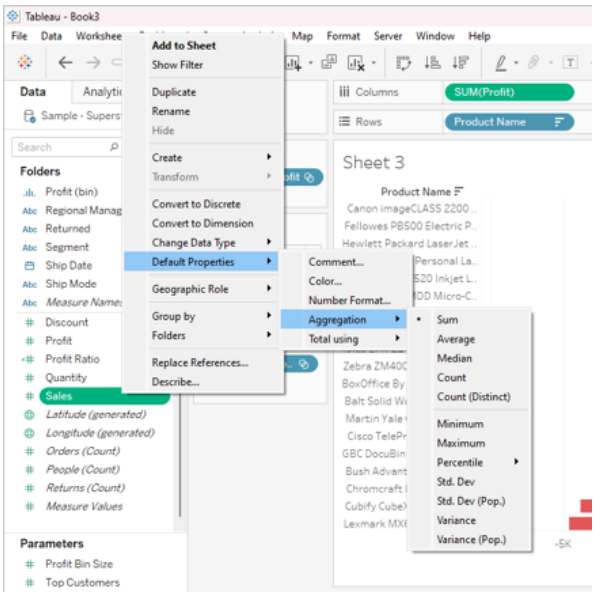
1) We can easily create folders for measures or any other thing which we want to keep separate. It's a good practice to create a folder to keep things synchronized.



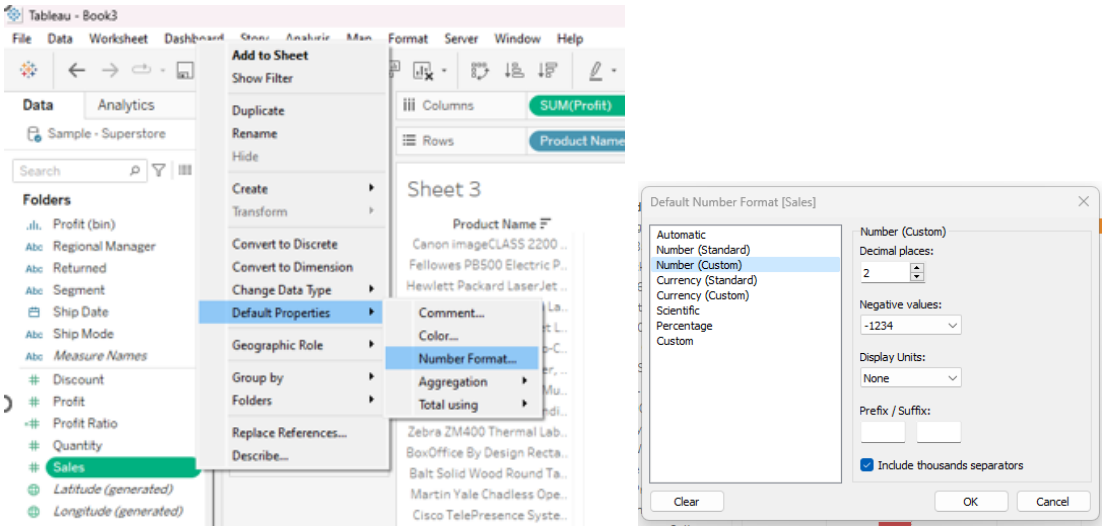
2) When you are choosing some measures or dimensions you always have to give them some colour. But there is some advantage from tableau side where you can keep some default colours throughout the dashboard.



3) We can also have some default aggregation type. For example, we are using Sales many a times in our sheets and most of the time the aggregation we are using is the SUM so we can also keep a default aggregation for our measures.



4) You can also change the default number format by changing the deci keeping some suffix or prefix to the measure, etc.



GROUPS IN TABLEAU

You can create a group to combine related members in a field. For Example: if you are working with a view that shows average test scores by major, you might want to group certain majors together to create major categories.

What are groups in Tableau?

1) A tableau group is a set of multiple members combined in a single dimension to **create a higher category of the dimension.**
2) Groups are created to combine similar members in a field.
3) You can create groups on **both Dimensions and Measures.**
4) Groups are always static.
5) Groups are useful for **both correcting data errors (e.g. combining CA, Calf and California into one data point)**

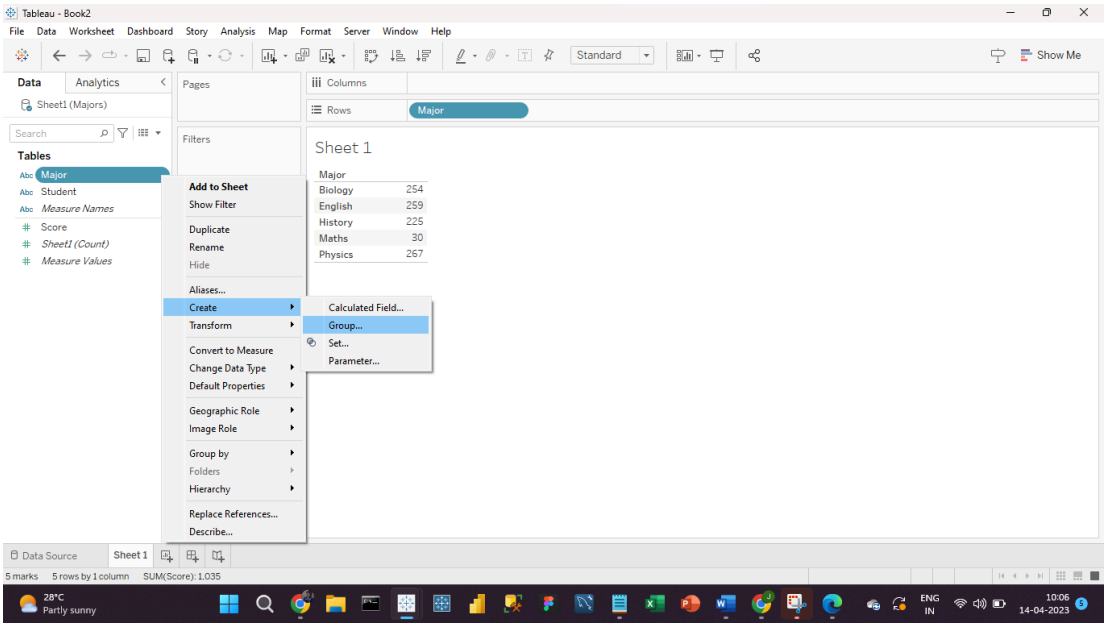
Methods to create Groups

Method 1: Traditional method

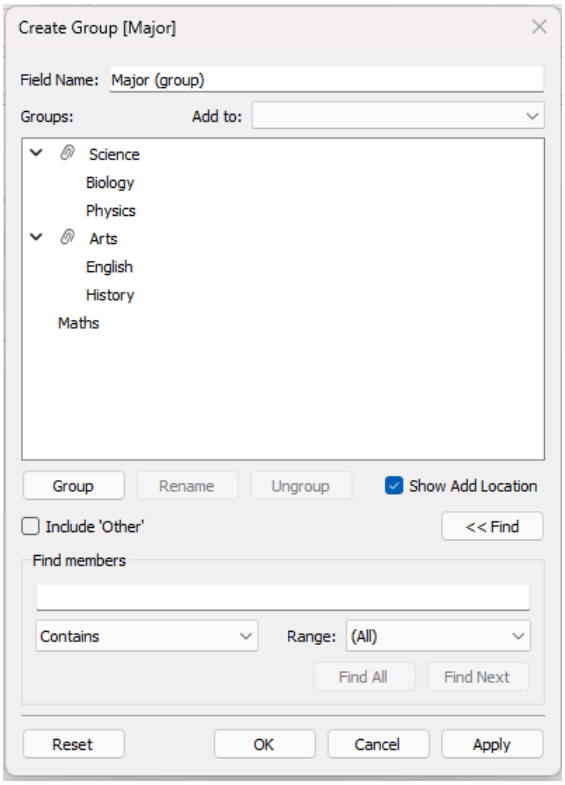
Suppose we have Majors and we want to group them according to Science and Arts with Marks.

Major	
Biology	254
English	259
History	225
Maths	30
Physics	267

We can click on the major Dimension and go on create and select group.



A dialog box will appear and you can group the majors accordingly.



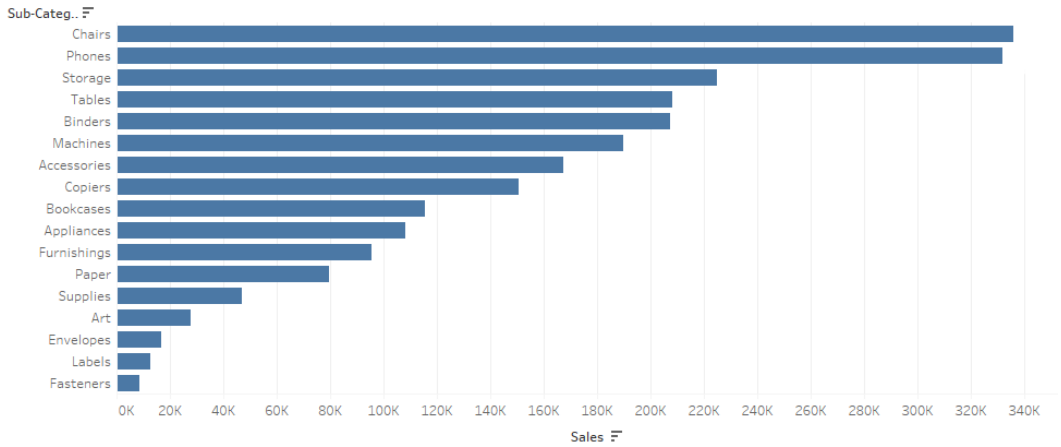
After that your group is created.

Major (Grouped)	Major	
Arts	English	259
	History	225
Maths	Maths	30
Science	Biology	254
	Physics	267

Note: You can also use find for finding the keywords and then create the groups.

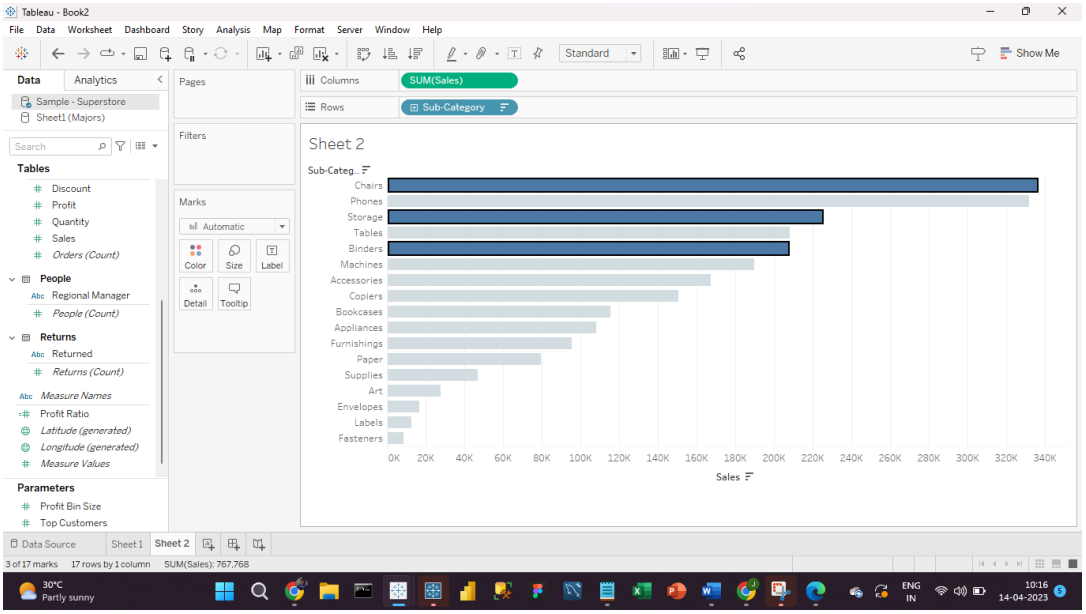
Method 2: Grouping using Visuals

Now suppose we have visuals as shown below we can group the sub-categories on basis of marks and labels.

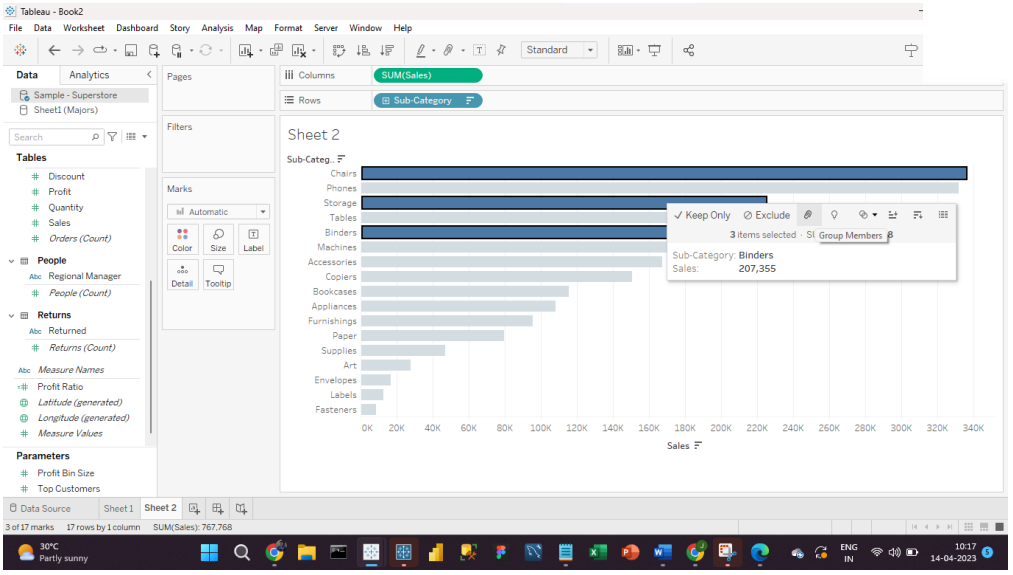


A) Grouping on basis of Marks

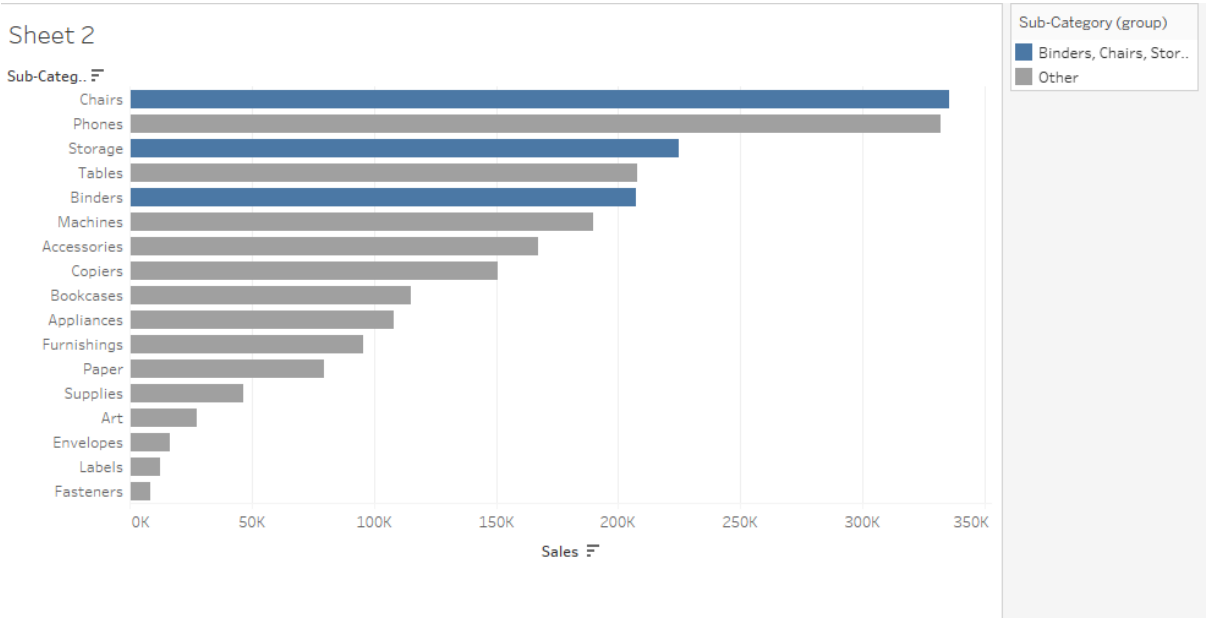
1) We can select the marks like shown in the figure.



2) Click on the group symbol

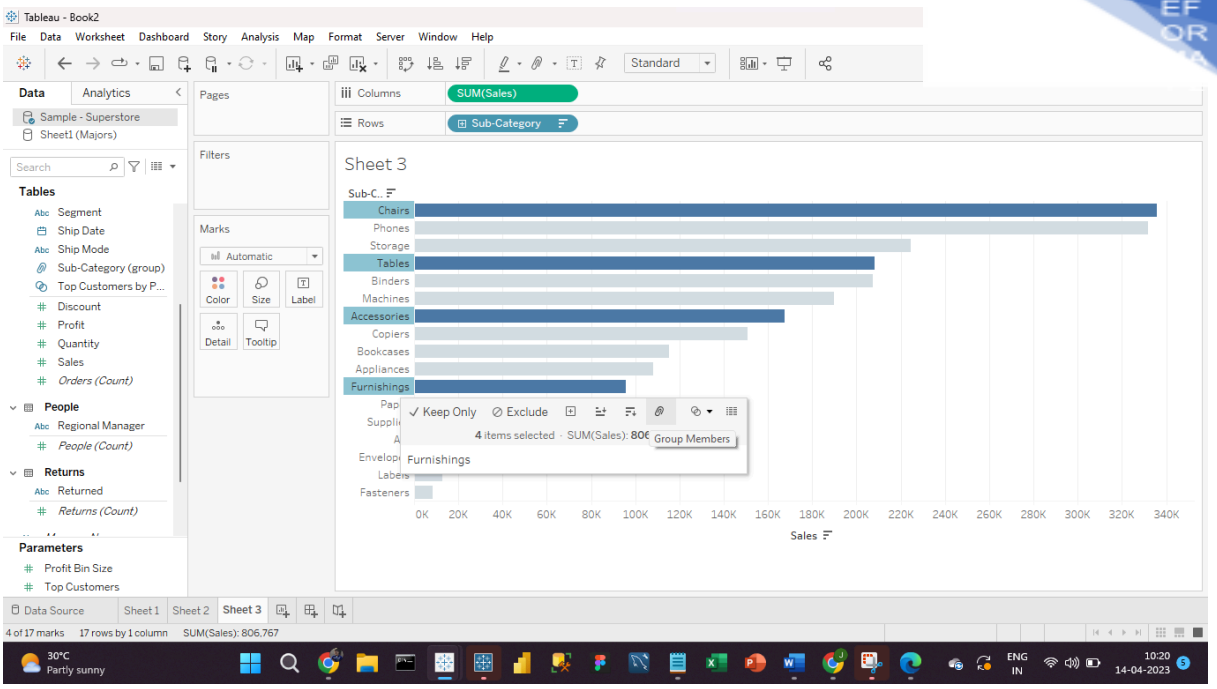


3) After grouping your marks will be grouped on basis color as grouped categories and others.

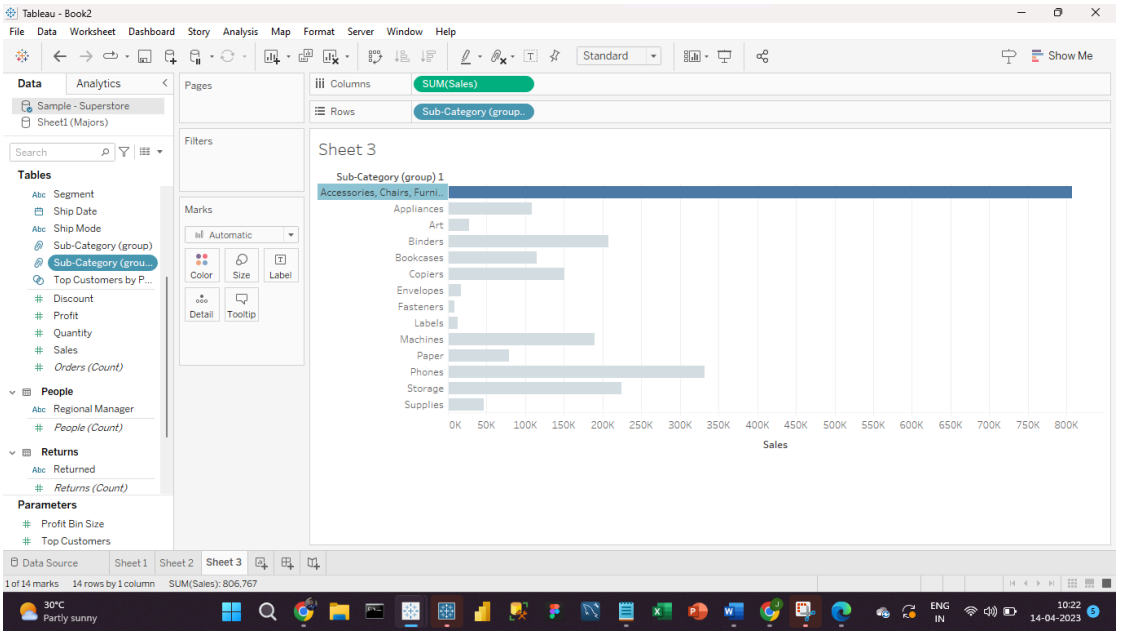


B) Grouping on basis of sub-category headers

1) Similarly select all the sub-categories and click on the group symbol to group data based on the selection of the headers.



2) The group will be as followed.



Limitations:

Tableau also has some limitations when it comes to working with groups. Some of the limitations of groups in Tableau are:

1. **Limited flexibility in modifying groups:** Once a group is created in Tableau, it cannot be easily modified. For example, you cannot add or remove members from a group dynamically based on changing data conditions or calculations. If

you need to modify a group, you typically have to create a new group from scratch.

2. **Inability to use groups in all calculations:** Groups in Tableau are not supported in all types of calculations. For example, you cannot use groups in table calculations or in calculations that involve level of detail (LOD) expressions. This can limit the flexibility and complexity of calculations that you can perform using groups.
3. **Group membership not reflected in data source:** Groups in Tableau are created at the visualization level and do not affect the underlying data source. This means that group membership is not reflected in the original data source, and you cannot use groups as a basis for filtering or aggregating data in other parts of your Tableau workbook or in other visualizations that use the same data source.
4. **Inability to create nested or hierarchical groups:** Tableau does not support creating nested or hierarchical groups, where you can create groups within groups. This can limit the ability to create complex groupings based on multiple levels of criteria.
5. **Group limitations in data blending:** If you are using data blending in Tableau, groups created in one data source cannot be used in the blending process with another data source. This can limit the ability to create groups that span across multiple data sources.
6. **Limited group management options:** Tableau does not provide robust group management options, such as the ability to centrally manage groups across multiple visualizations or workbooks. This can make it challenging to maintain consistency and manage groups in a large-scale Tableau deployment.
7. **Performance considerations:** Groups in Tableau can impact performance, especially when dealing with large datasets or complex calculations. Grouping can add overhead in terms of processing time and may impact the performance of your visualizations, especially if you have multiple groups or complex groups in your workbook.

Despite these limitations, Tableau offers a wide range of powerful data visualization and analysis capabilities, and groups can still be very useful for simplifying data analysis and creating custom groupings for visualization purposes. It's important to be aware of these limitations and consider them when designing your Tableau visualizations to ensure that your results are accurate and meet your requirements.

SETS IN TABLEAU

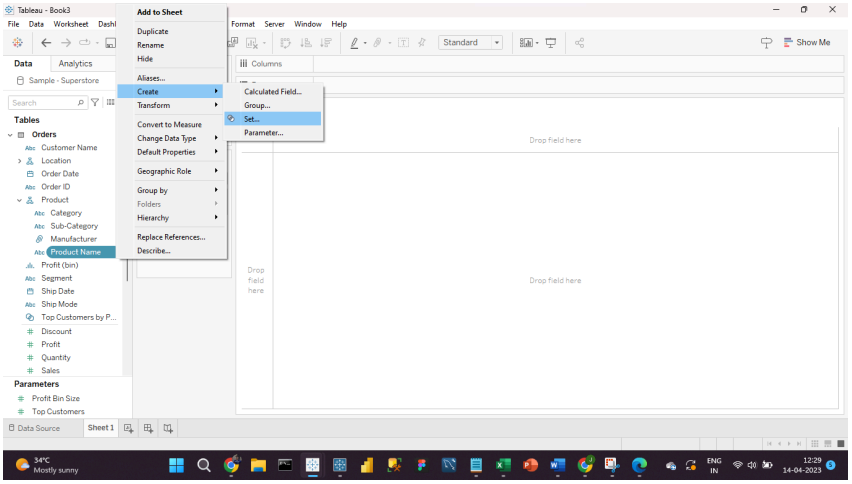
Sets are custom fields that define a subset of data based on some conditions. Sets can be Constant or Computed, and can be made more dynamic using set actions

Characteristics of Sets in Tableau

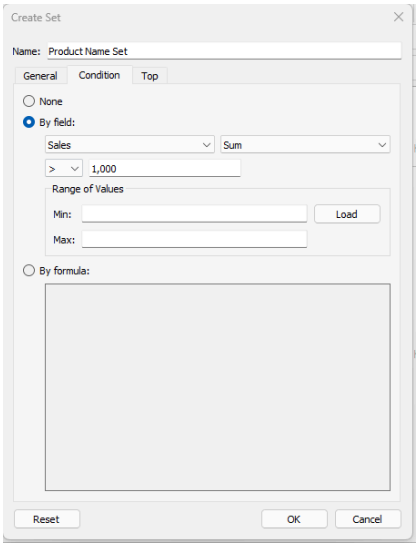
1. Tableau sets are custom fields used to hold the subset of data based on a given condition.
2. In real-time, you can create a set by selecting members from the list or a visualization.
3. Sets are more complex as compared to groups.
4. Sets can be static as well as dynamic in nature.
5. You can only create sets on dimensions.
6. The members of a static set do not change, they are fixed even when the data changes.
7. A static set can be based on single or multiple dimensions.
8. The user can remove the elements but cannot add in a static set.

Example of Set Creation

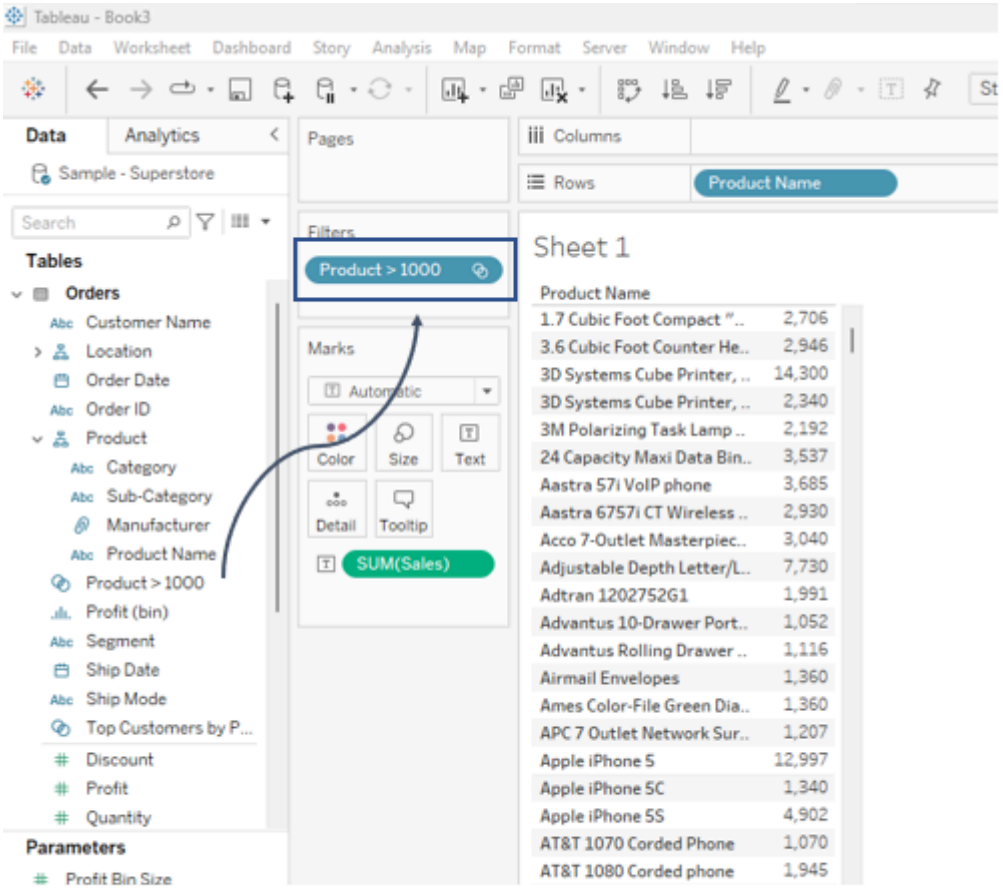
You can easily create set by clicking on the dimension and select create in select you have to choose set.



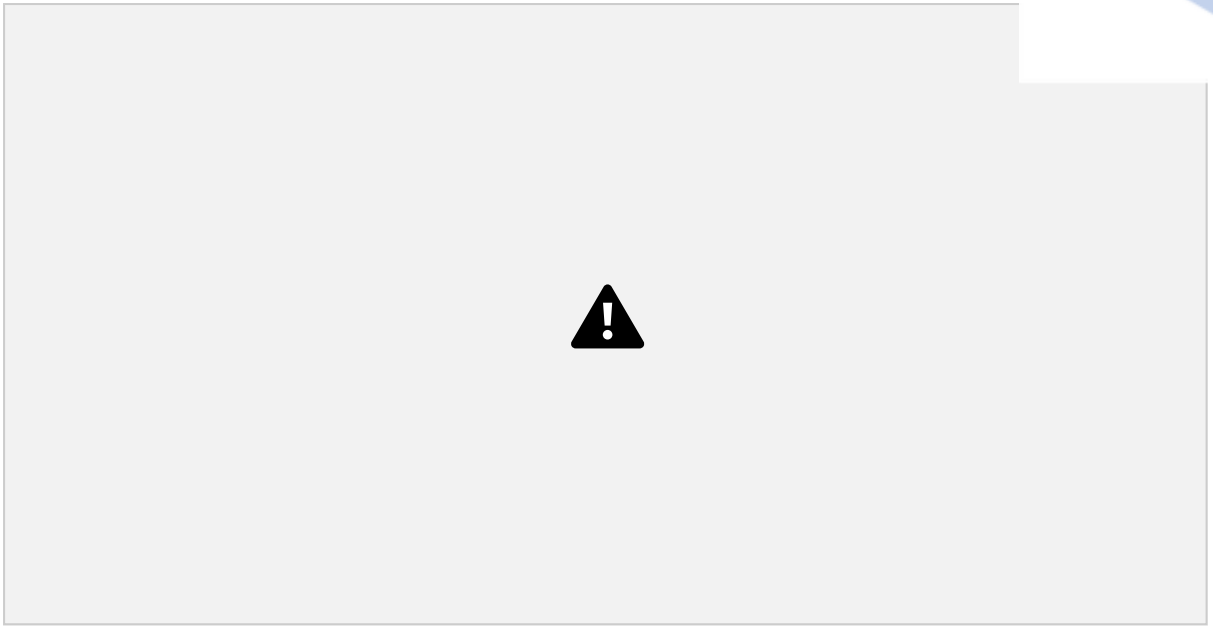
After clicking on set a dialog box will appear and you can create a set based on some condition.



And a set is created and you can add it in the filter.

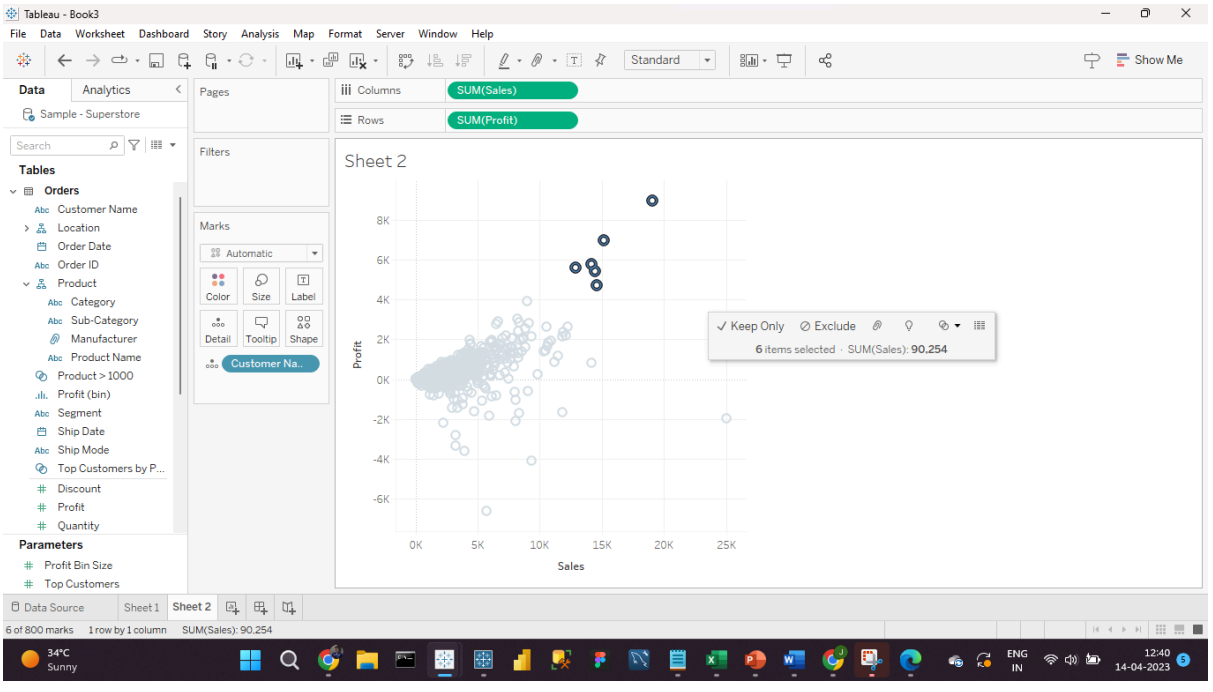


You can also observe that everything in SET is in form of IN/OUT. The rows which are considered in IN are the rows which has satisfied your condition of SET and OUT are the rows which have not satisfied the condition of your SET.

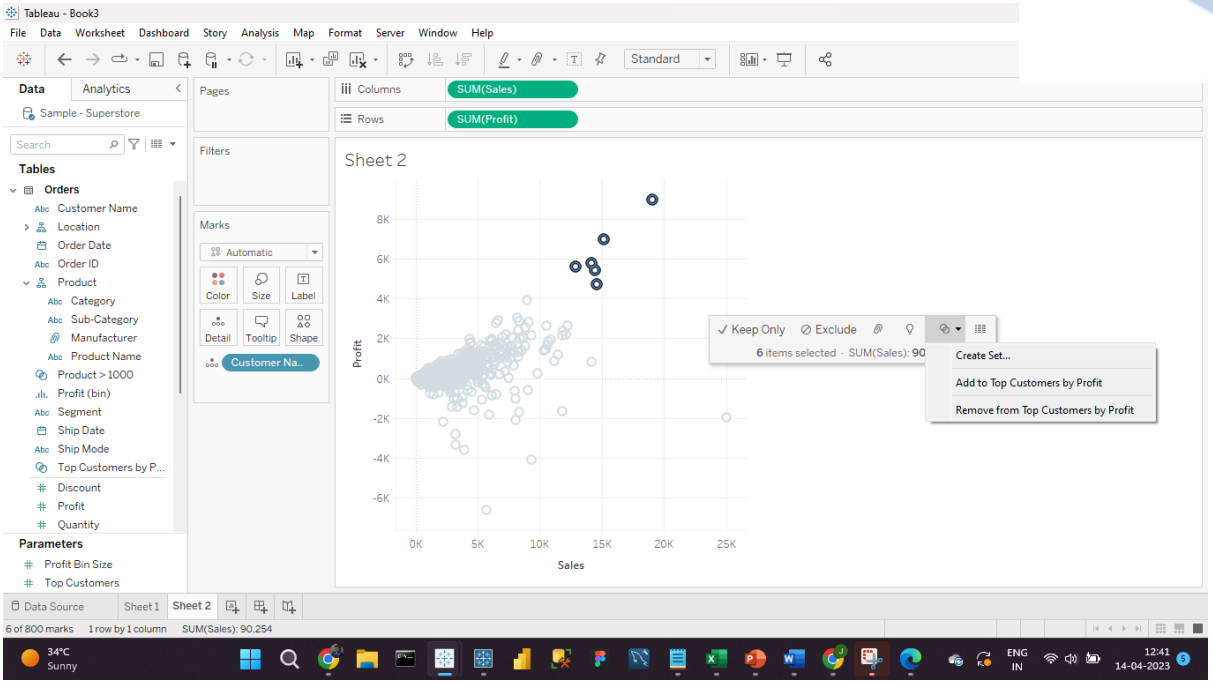


Example of Static Set

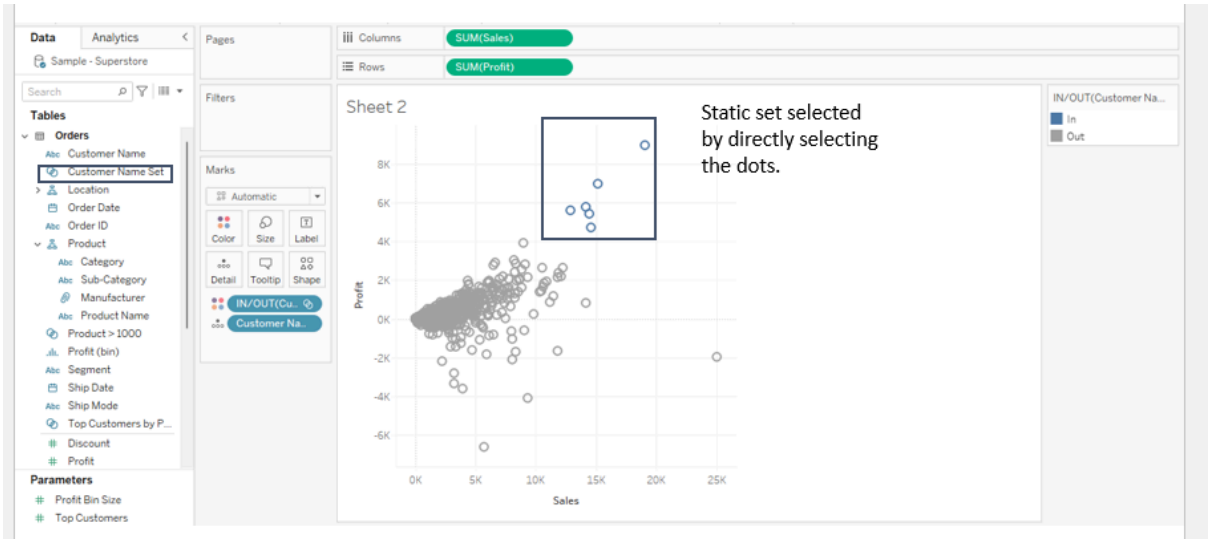
Now if you want to create a static set you can select the points and click on the set icon from the dialog box.



After clicking on that button just select Create Set and your set will be created.

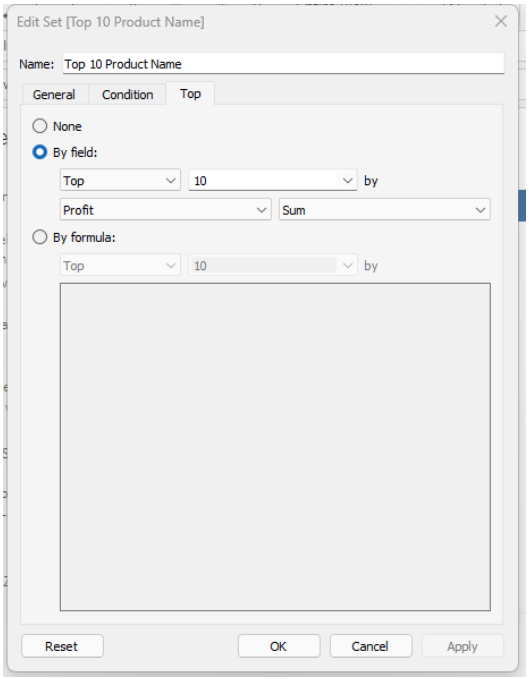


But the set which will be created will be a static set and you can change the values in the static set.

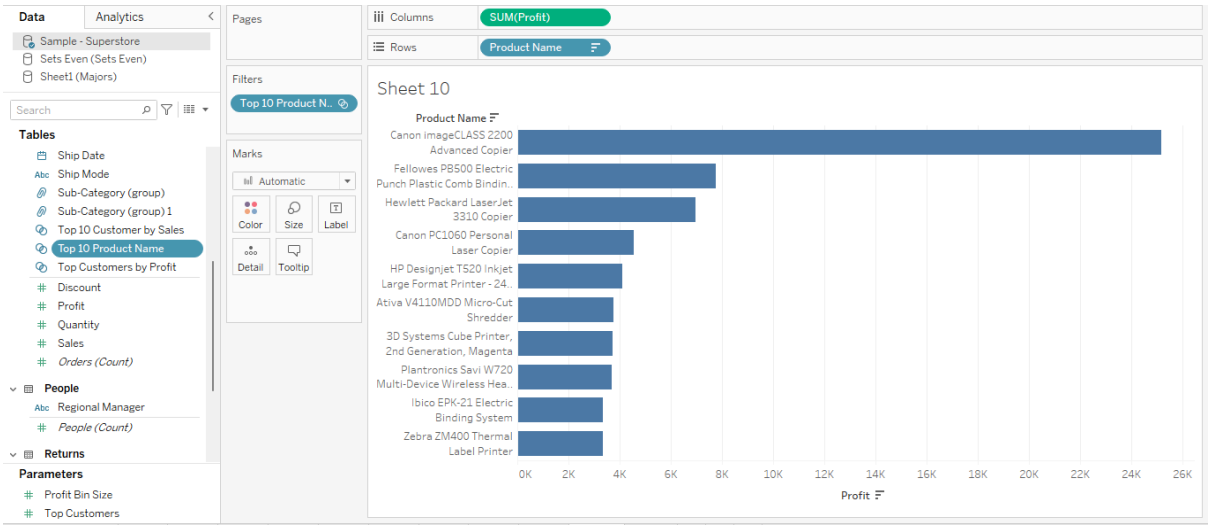


Example of Top using Sets

You can create Top N Values set by simply going on create set and choosing Top and put down your values.



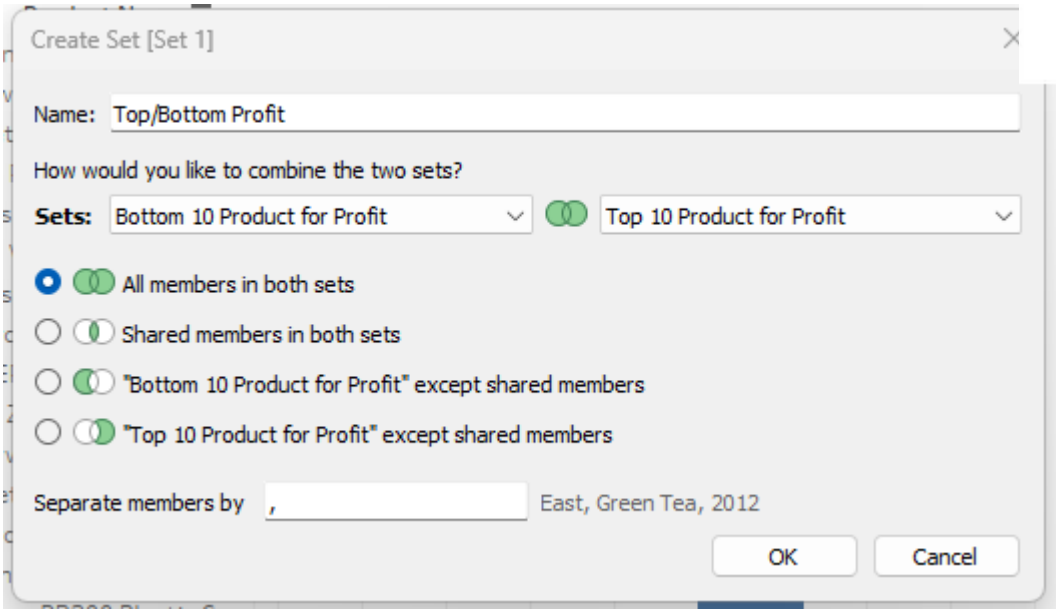
Once it is created you can simply drag down that into filters and your top 10 products according to the sales are created.



Combined sets in Tableau

The Combined Sets in Tableau are handy for comparing two existing sets for further analysis.

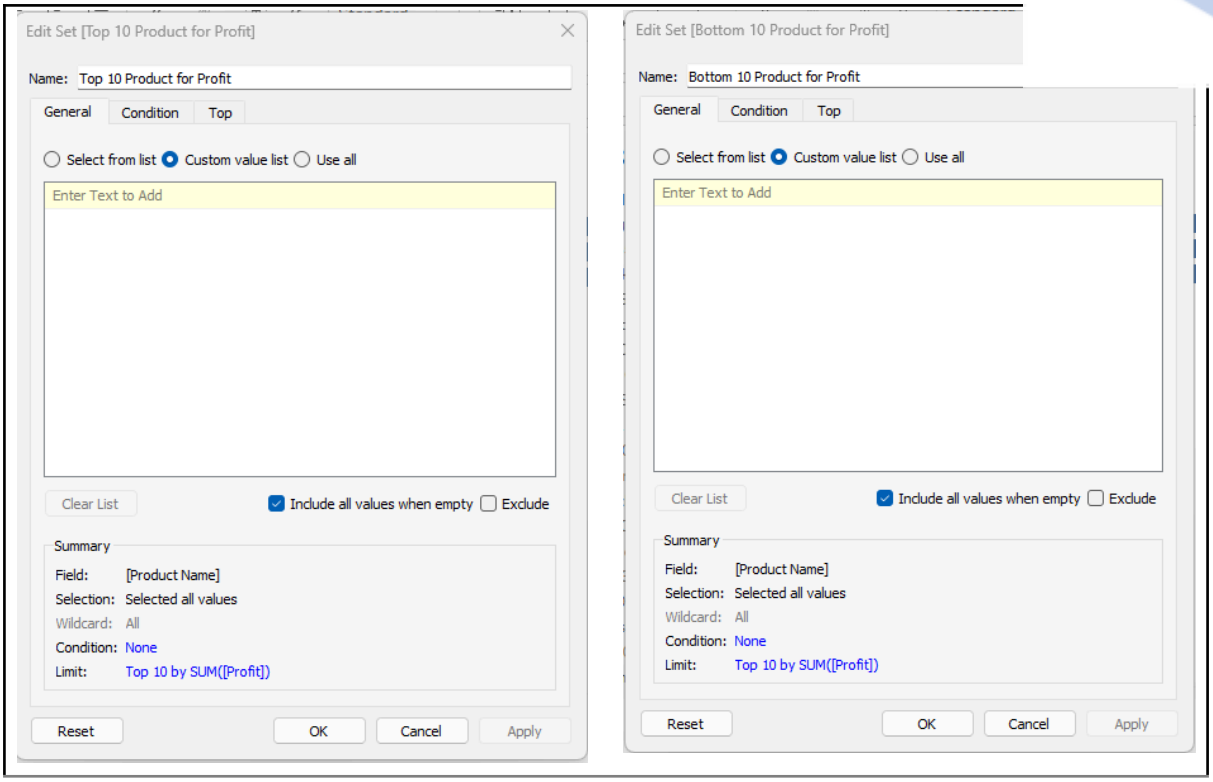
The following screenshot will show you the window used to create a Combined Set in Tableau, and their properties are:



- **Name:** Specify a unique name for the set. The title should reflect the set functionality.
- **Sets:** Please select or change the existing Set from Drop down menu. The first set act as Left Set and the second set as a Right set
- **All members in Both Sets:** This option is similar to SQL Full Join. If you select this option, Tableau Combined Set holds all the members from the Left and Right Sets.
- **Shared Members in Both Sets:** This option is similar to SQL Inner Join. This Combined Set option holds matching members from Left Set and Right Set. It means Every record should match the condition present in the Left and Right sets.
- **Left Set Except Shared Members:** If you select this option, Combined Set will hold all the members from Left Set except the matching members from the Right one.
- **Right Set Except for Shared Members:** This option holds all the members from Right Set except matching members from Left Set.

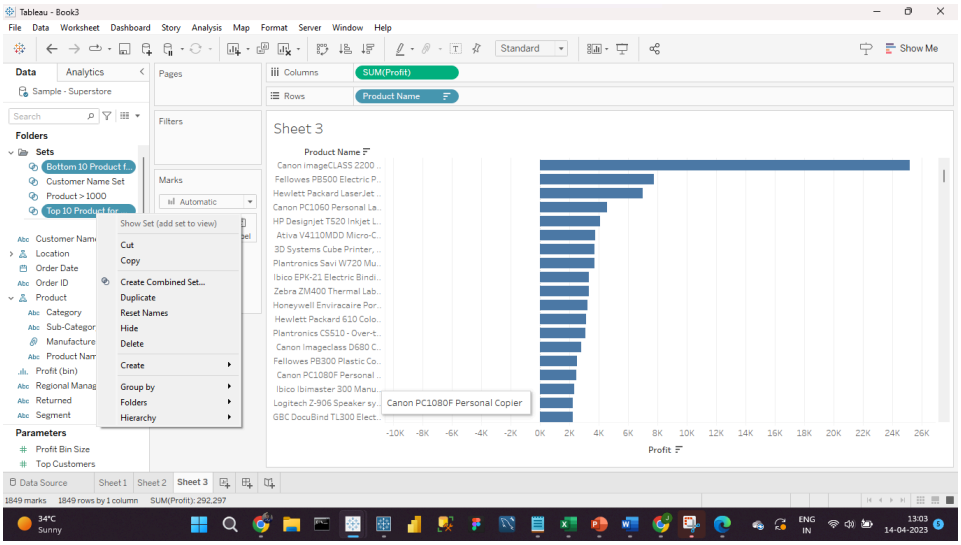
Example: Now we want to know the Top 10 and Bottom 10 product according to profit into one single chart. We can simply do this using combined sets.

Lets first make a set for Top 10 Products according to Profit and Bottom 10 Products according to Profit.

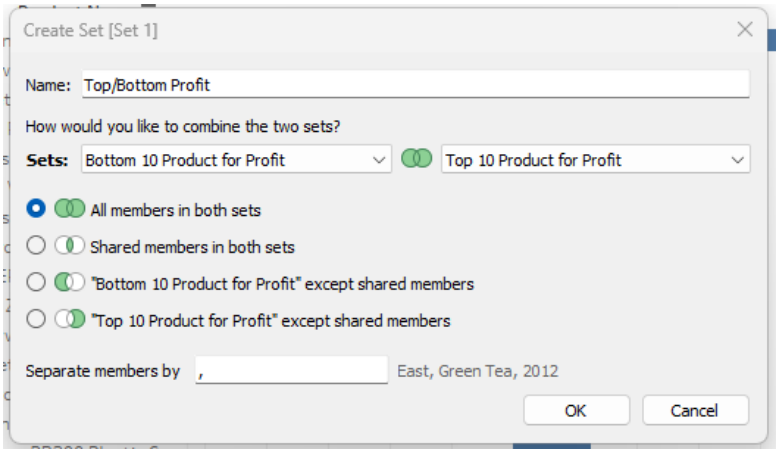


Now once we have created this two set next we are going to use is the combined sets.

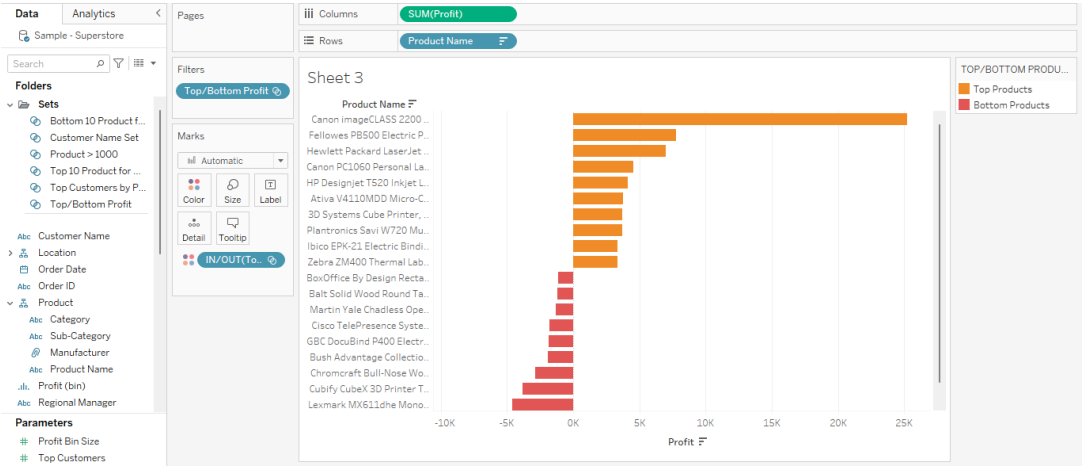
For making a combined set you have to select both the sets than right click and select combined sets.



Give it a name and select all members from both the set.



Click on OK and then drag and drop Top/Bottom Profit in your filter section you will see the clear segregation between the Top and Bottom Products. You can also change the colour by adding the sets and make it look attractive.



Limitations of Sets

Tableau provides sets as a way to create custom groups or subsets of data for analysis and visualization. However, sets also have some limitations in Tableau. Some of the limitations of sets in Tableau are:

1. **Fixed membership:** Once a set is created in Tableau, its membership is fixed based on the criteria used to define the set. The set does not update dynamically as data changes or as filters are applied. This means that if your data changes, the set membership will not automatically adjust, and you may need to manually update the set.
2. **Limited flexibility in modifying sets:** Sets in Tableau cannot be easily modified once they are created. You cannot add or remove members from a set dynamically based on changing data conditions or calculations. If you need to modify a set, you typically have to create a new set from scratch.

3. **Inability to use sets in all calculations:** Sets in Tableau are not sup types of calculations. For example, you cannot use sets in table calcu calculations that involve level of detail (LOD) expressions. This can limit the flexibility and complexity of calculations that you can perform using sets.
4. **Set membership not reflected in data source:** Sets in Tableau are created at the visualization level and do not affect the underlying data source. This means that set membership is not reflected in the original data source, and you cannot use sets as a basis for filtering or aggregating data in other parts of your Tableau workbook or in other visualizations that use the same data source.
5. **Performance considerations:** Sets in Tableau can impact performance, especially when dealing with large datasets or complex calculations. The creation and use of sets can add overhead in terms of processing time and may impact the performance of your visualizations, especially if you have multiple sets or complex sets in your workbook.
6. **Limited set management options:** Tableau does not provide robust set management options, such as the ability to centrally manage sets across multiple visualizations or workbooks. This can make it challenging to maintain consistency and manage sets in a large-scale Tableau deployment.
7. **Set limitations in data blending:** If you are using data blending in Tableau, sets created in one data source cannot be used in the blending process with another data source. This can limit the ability to create sets that span across multiple data sources.

Despite these limitations, sets in Tableau can be useful for creating custom groups or subsets of data for analysis and visualization. It's important to be aware of these limitations and consider them when designing your Tableau visualizations to ensure that your results are accurate and meet your requirements.

Difference between Groups and Sets

Features	Groups	Sets
Definition	Combining data elements from a field or multiple fields into a single group.	Creating a subset of data based on specific conditions or criteria.
Usage	Aggregate data and create a new category or level of detail.	Create a separate group of data points that meet certain conditions or thresholds.
Dimension vs Measure	Created within a single dimension.	Can be created based on measures, dimensions, or a combination of both.

Data Elements	Combines data elements from one or more fields into a single group.	Creates a subset of data specific conditions or cr
Applications	Summarize or filter data in visualizations.	Perform calculations, create custom filters, or compare subsets of data in visualizations.
Result	Creates a new aggregated category or level of detail in the field.	Creates a separate subset of data points, which can be used for further analysis or visualization.
Flexibility	More flexible in combining data elements from multiple fields.	Less flexible as it is based on specific conditions or criteria.
Dynamic vs static	Dynamic, as the group can change as the underlying data changes.	Static, as the set is fixed based on the conditions or criteria at the time of creation.
Position	Tableau generates a new entry in the data source as well as dimensions selection when a new group is created.	Sets get positioned only in the data pane under sets section.
Forms of result	Group aggregate the data	Sets differentiate the data in the viz, through IN/OUT notations without aggregating the data.

Calculations in Tableau

In Tableau, a calculated field is a user-defined field that is created by applying a formula or expression to existing fields in a dataset. It allows you to manipulate and transform data in ways that are not possible with the original dataset.

Calculated fields can be created using a variety of mathematical, logical, and statistical functions. For example, you can use calculated fields to perform calculations like addition, subtraction, multiplication, and division, or to create more complex formulas that involve functions like IF statements, logical operators, and aggregate functions like SUM, COUNT, AVG, and MAX.

Calculated fields are useful for performing analysis that goes beyond simple data aggregation or filtering. They allow you to create custom metrics, identify trends or patterns in your data, and gain insights that may not be readily apparent from the original dataset.

Steps to Create Calculated fields

To create a calculated field in Tableau, you can use the "Create Calculated Field" option in the "Analysis" menu or right-click on a blank area in the "Fields" pane and select "Create Calculated Field." You can then enter your formula or expression and give the calculated field a name. Once created, the calculated field can be used like any other field in Tableau, including in visualizations, filters, and calculations.

Why use calculated fields?

- Calculated fields allow you to create new data from data that already exists in your data source.
- When you create a calculated field, you are essentially creating a new field (or column) i your data source, the values or members of which are determined by a calculation that you control.
- This new calculated field is saved to your data source in ‘Tableau, and can be used to create visualizations.
- But don't worry: your original data remains untouched.

You can use calculated fields for many, many reasons. Some examples might include:

- To segment data

- To convert the data type of a field, such as converting a string to a da
- To aggregate data
- To filter results
- To calculate ratios

Types of Calculations

You create calculated fields using calculations. There are three main types of calculations you can use to create calculated fields in Tableau:

- **Basic calculations** Basic calculations allow you to transform values or members at the data source level of detail (a row-level calculation) or at the visualization level of detail (an aggregate calculation).
- **Level of Detail (LOD) expressions -** Just like basic calculations, LOD calculations allow you to compute values at the data source level and the visualization level. However, LOD calculations give you even more control on the level of granularity you want to compute. They can be performed at a more granular level (INCLUDE), a less granular level (EXCLUDE), or an entirely independent level (FIXED) with respect to the granularity of the visualization.
- **Table calculations -** Table calculations allow you to transform values at the level of detail of the visualization only.

How to aggregate?

The rules that apply to aggregate calculations are as follows:

For any aggregate calculation, you cannot combine an aggregated value and a disaggregated value. For example, SUM(Price) [Items) is not a valid expression because SUM(Price) is aggregated and items is not. However, SUM(Price Items) and SUM(Price) SUM(Items) are both valid.

Constant terms in an expression act as aggregated or disaggregated values as appropriate. For example: SUM(Price 7) and SUM(Price)*7 are both valid expressions. • All of the functions can be evaluated on aggregated values. However, the arguments to any given function must either all be aggregated or all disaggregated. For example: MAX(SUM(Sales), Profit) is not a valid expression because Sales is aggregated and Profit is not. However, MAX(SUM(Sales), SUM(Profit)) is a valid expression.

The result of an aggregate calculation is always a measure.

Functions in Tableau

Any data analysis involves a lot of calculations. In Tableau, the calculations editor is used to apply calculations to the fields being analyzed. Tableau has a number of inbuilt functions which help in creating expressions for complex calculations.

Following are the description of different categories of functions:

1. Number functions
2. String functions
3. Date functions
4. Logical functions
5. Aggregate functions
6. Type conversion
7. User functions
8. Table calculations functions
9. Spatial functions

Aggregate functions

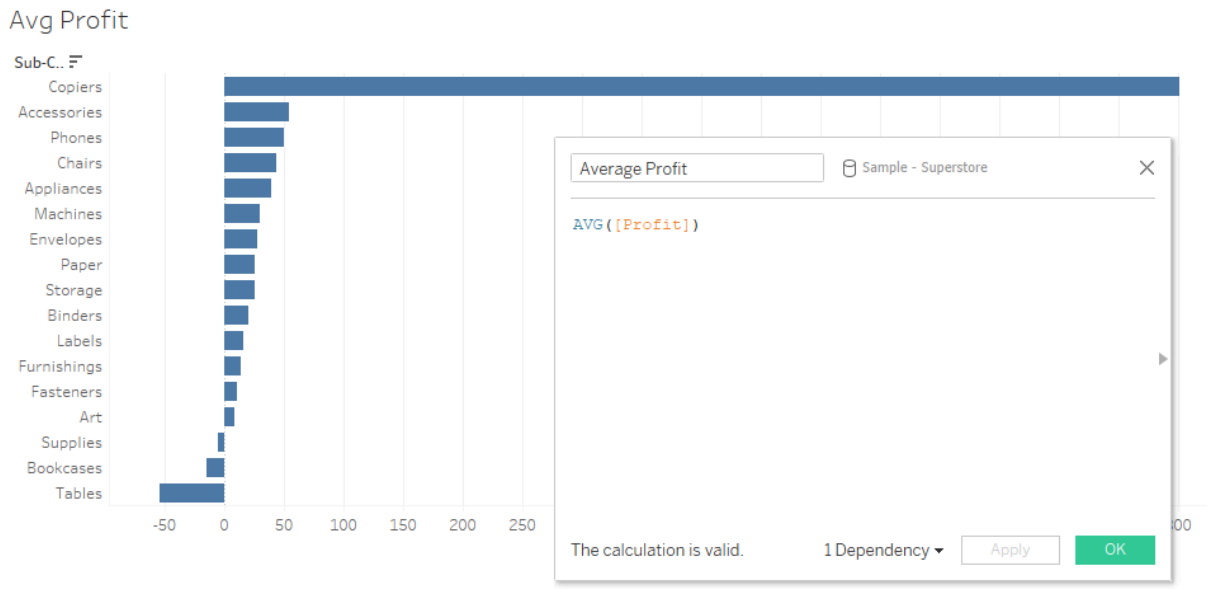
Tableau provides various Aggregate Functions, which help you to perform aggregations such as calculating the sum, avg, minimum, maximum, etc.

SUM: calculates the total sum of a field for all records in the dataset.

Syntax: SUM(expression)

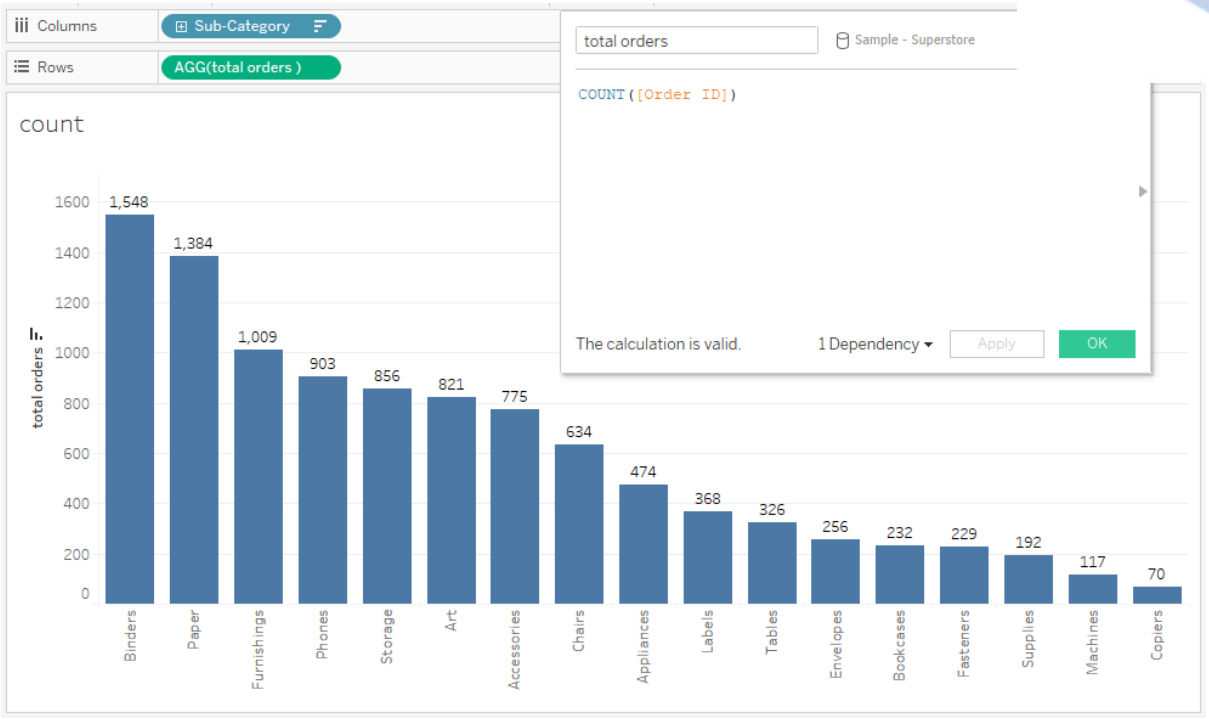
AVG: calculates the average of a field for all records in the dataset.

Syntax: AVG(expression)



COUNT: counts the number of records in the dataset.

Syntax: COUNT(expression)

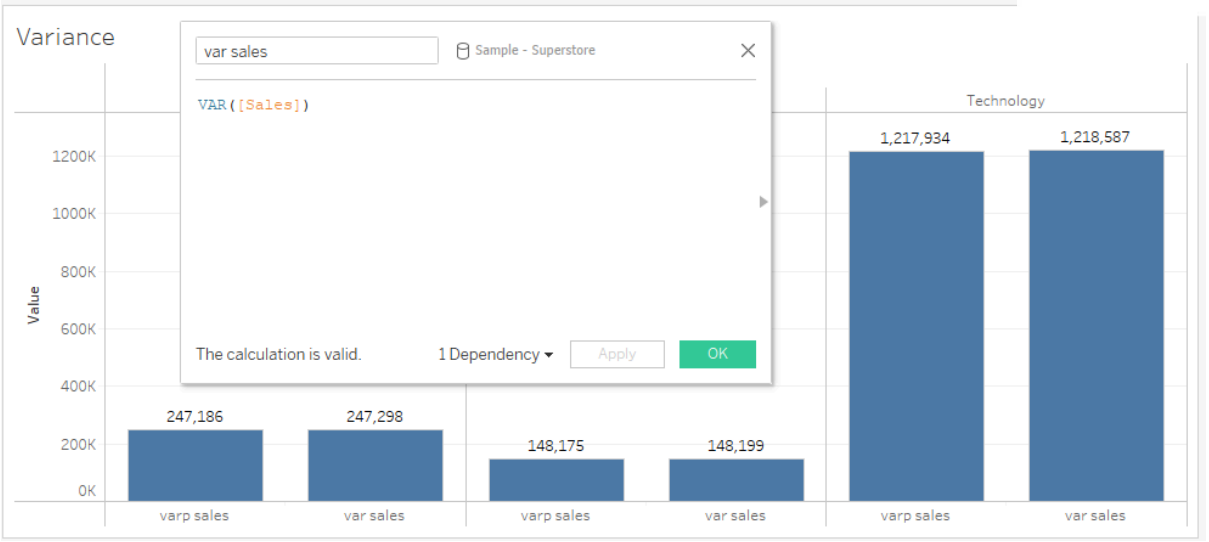


COUNTD: counts the number of unique values in a field.

Syntax: COUNTD(expression)

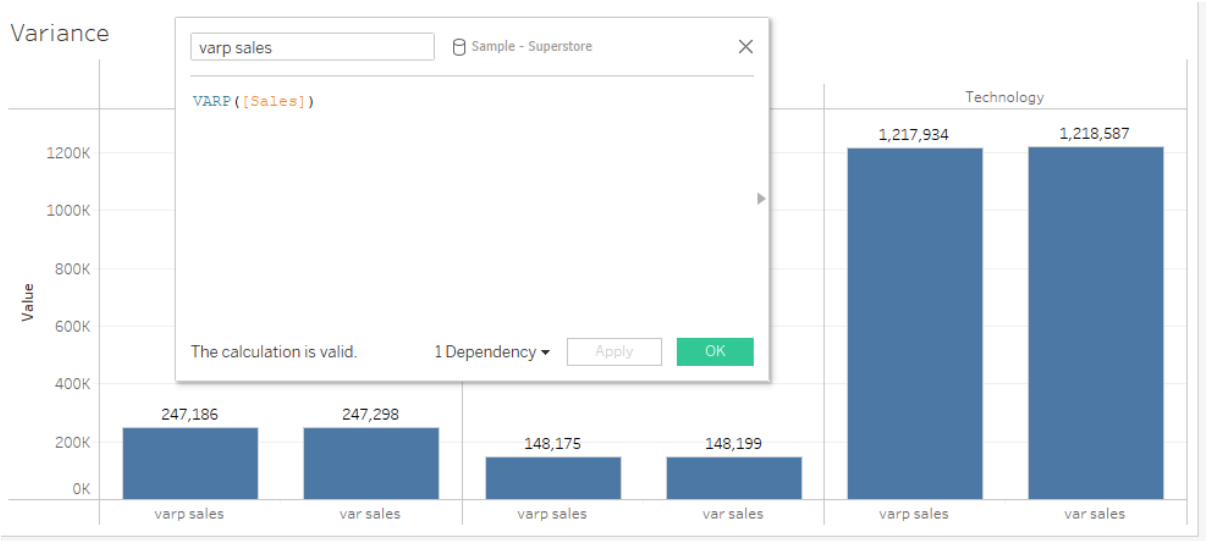
VAR: calculates the sample variance of a field for all records in the dataset.

Syntax: VAR(expression)



VARP: calculates the population variance of a field for all records in the dataset.

Syntax: VARP(expression)



MEDIAN: calculates the median value of a field for all records in the dataset. *Median is the value in the very middle of a set of measurements. A percentile (or a centile) is a measure used in statistics*

indicating the value below which a given percentage of observations in a group of observatio
example, the 20th percentile is the value (or score) below which 20% of the observations may
Median is also called 50th Percentile

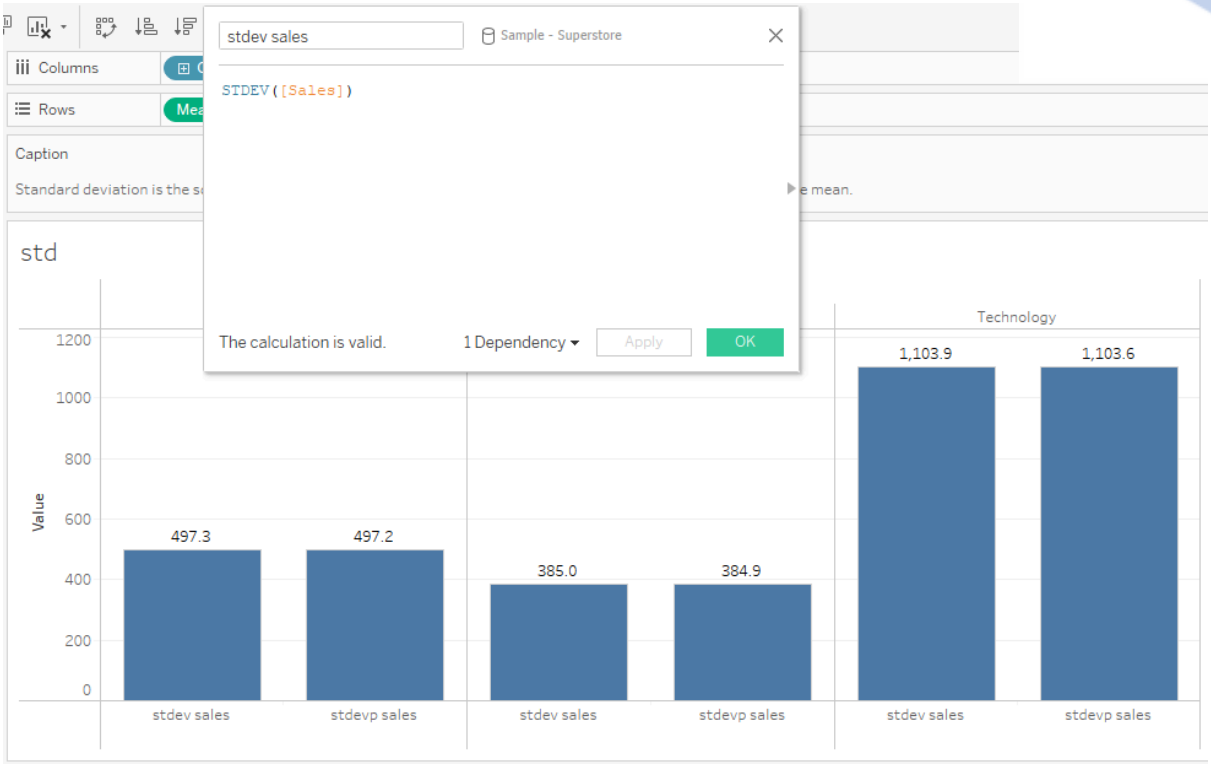
Syntax: MEDIAN(expression)

Median/Percentile

Category	median	percent..
Furniture	173.9	173.9
Office Supplies	27.3	27.3
Technology	166.2	166.2

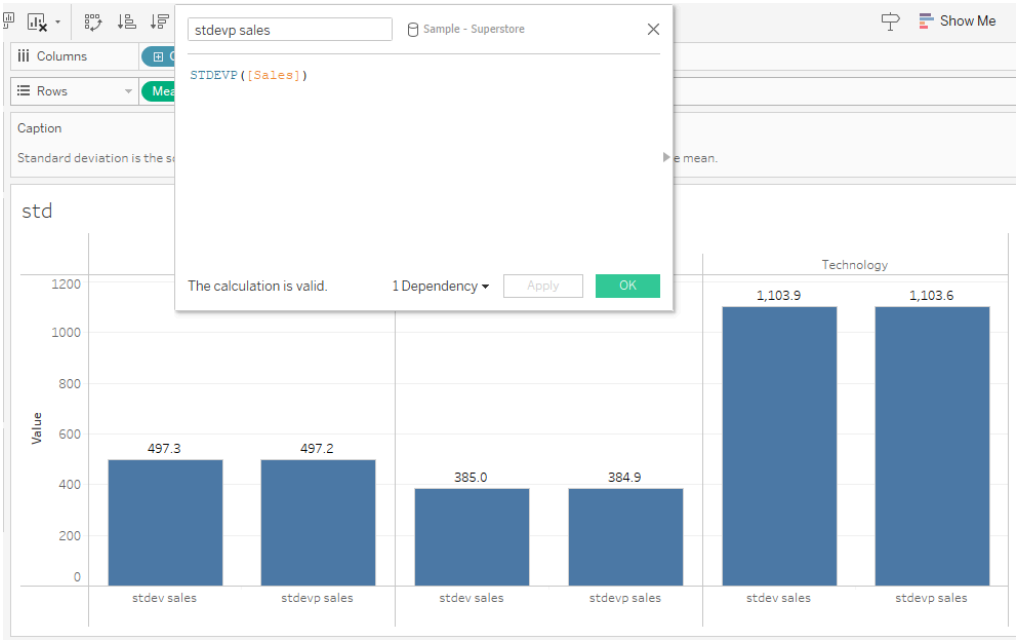
STDEV: calculates the sample standard deviation of a field for all records in the dataset. Standard deviation is the square root of the variance. It measures the average distance of each data point from the mean.

Syntax: STDEV(expression)



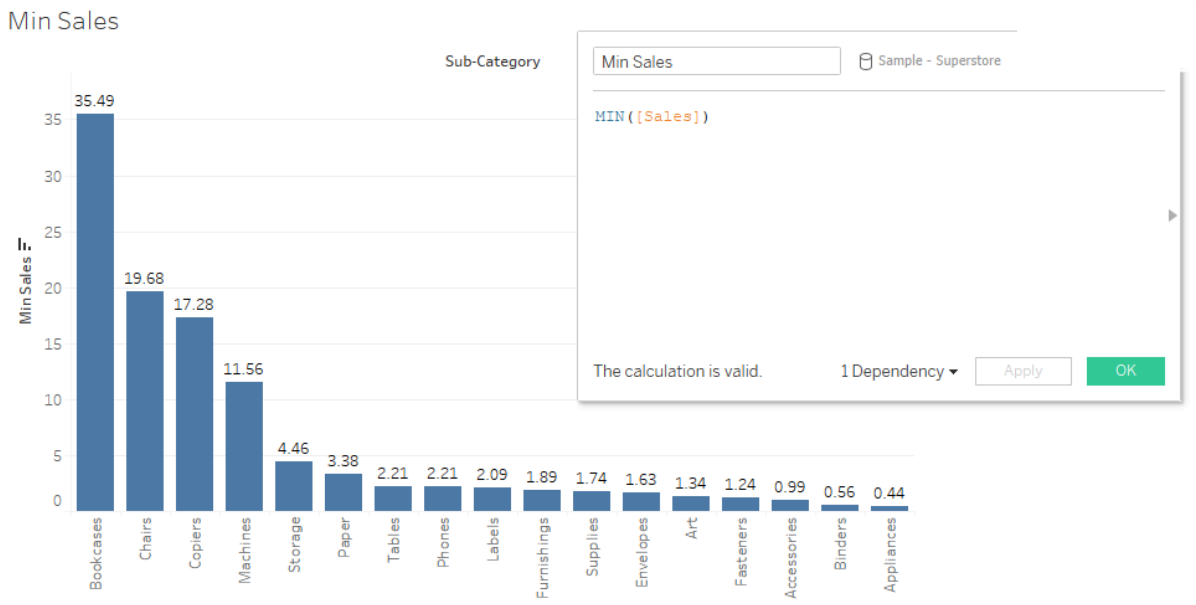
STDP: calculates the population standard deviation of a field for all records in the dataset.

Syntax: STDEVP(expression)



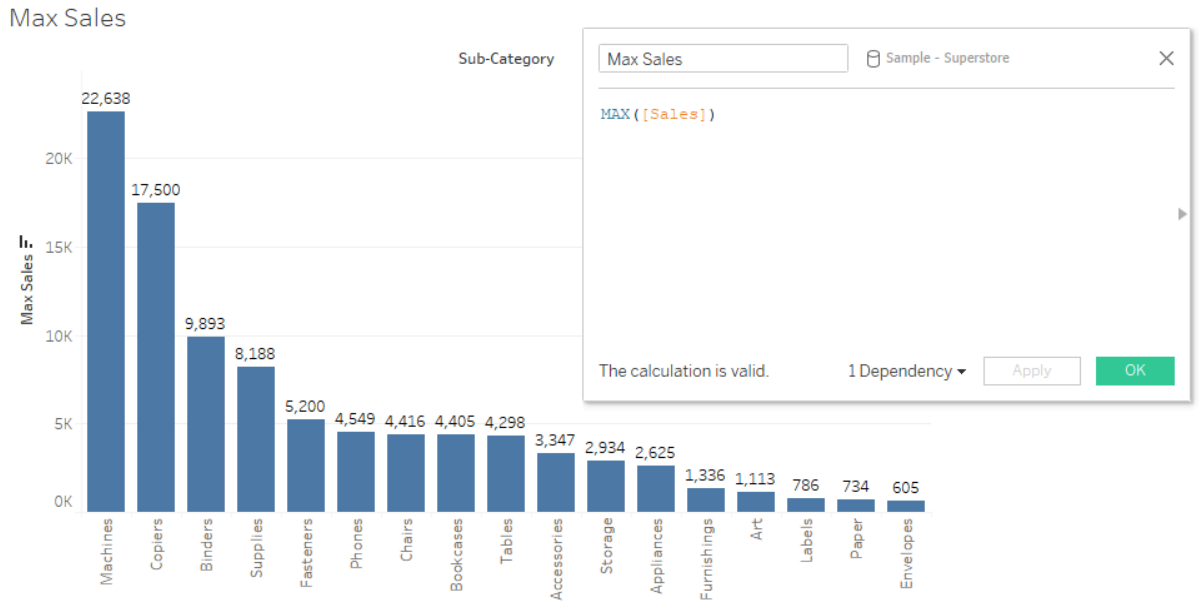
MIN: returns the minimum value of a field for all records in the dataset.

Syntax: MIN(expression) or MIN(expr1, expr2)



MAX: returns the maximum value of a field for all records in the dataset.

Syntax: MAX(expression) or Max(expr1, expr2)



COVAR: calculates the sample covariance between two fields for all records in the dataset.

Covariance and correlation are two statistical measures that describe the relationship between two variables. If the covariance is positive, it indicates that the two variables tend to increase or decrease together. If the covariance is negative, it indicates that the two variables tend to have an inverse relationship, meaning that as one variable increases, the other decreases.

Syntax: COVAR(expression 1, expression2)

covariance

covariance	69,382
covariancep	69,375
correlation	0

COVARP: calculates the population covariance between two fields for all records in the dataset.

Syntax: COVARP(expression 1, expression2)

covariance

covariance	69,382
covariancep	69,375
correlation	0

CORR: calculates the correlation between two fields for all records in the dataset. Correlation, on the other hand, measures the strength and direction of the linear relationship between two variables. It is a standardized version of covariance that ranges from -1 to 1.

Syntax: CORR(expression 1, expression2)

covariance

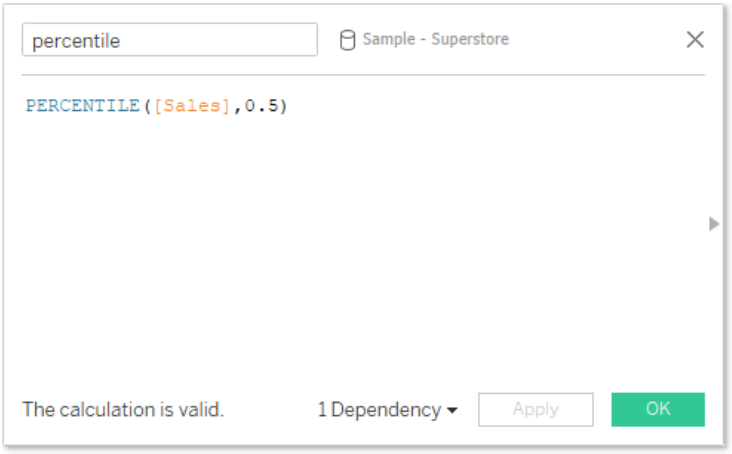
covariance	69,382
covariancep	69,375
correlation	0

PERCENTILE: returns the nth percentile of a field for all records in the data

Syntax: PERCENTILE(expression, number)

Median/Percentile

Category	median	percent..
Furniture	173.9	173.9
Office Supplies	27.3	27.3
Technology	166.2	166.2



Logical Functions

Tableau provides various Aggregate Functions, which help you to perform aggregations such as calculating the sum, avg, minimum, maximum, etc. In this article, we will show you how to use Tableau Aggregate Functions with examples.

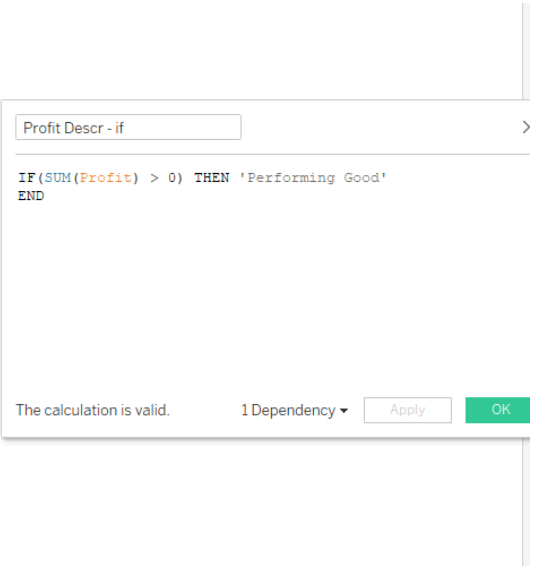
IF: The Tableau If statement returns the result only if the given condition is True; otherwise, it returns nothing.

Syntax: IF <expr> THEN <then> [ELSEIF <expr2> THEN <then2>...] [ELSE <else>] END

In this scenario you can also observe NULL values as there is no condition for negative values.

if

State/Province	Profit Descr - if	Discount	Quantity	Sales	Profit
Alabama	Performing Good	0	256	19,511	5,787
Arizona	Null	68	862	35,282	-3,428
Arkansas	Performing Good	0	240	11,678	4,009
California	Performing Good	146	7,667	457,688	76,381
Colorado	Null	58	693	32,108	-6,528
Connecticut	Performing Good	1	281	13,384	3,511
Delaware	Performing Good	1	367	27,451	9,977
District of Columbia	Performing Good	0	40	2,865	1,060
Florida	Null	115	1,379	89,474	-3,399
Georgia	Performing Good	0	705	49,096	16,250
Idaho	Performing Good	2	64	4,382	827
Illinois	Null	192	1,845	80,166	-12,608
Indiana	Performing Good	0	578	53,555	18,383
Iowa	Performing Good	0	112	4,580	1,184
Kansas	Performing Good	0	74	2,914	836
Kentucky	Performing Good	0	523	36,592	11,200
Louisiana	Performing Good	0	156	9,217	2,196
Maine	Performing Good	0	35	1,271	454
Maryland	Performing Good	1	420	23,706	7,031
Massachusetts	Performing Good	2	491	28,634	6,786
Michigan	Performing Good	2	946	76,270	24,463
Minnesota	Performing Good	0	331	29,863	10,823



IF ELSE: The If Else function will test the condition.

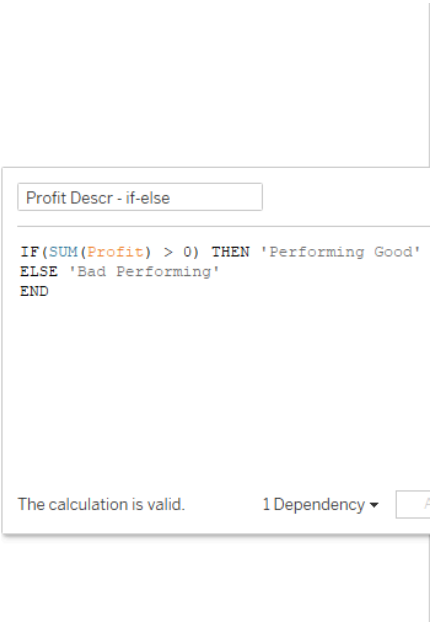
When the test condition is true, the statement after the THEN keyword will return

When it is False, the statement after the Else keyword will return.

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

if-else

State/Province	Profit Descr - if	Profit Descr - if-else	Discount	Quantity	Sales	Profit
Alabama	Performing Good	Performing Good	0	256	19,511	5,787
Arizona	Null	Bad Performing	68	862	35,282	-3,428
Arkansas	Performing Good	Performing Good	0	240	11,678	4,009
California	Performing Good	Performing Good	146	7,667	457,688	76,381
Colorado	Null	Bad Performing	58	693	32,108	-6,528
Connecticut	Performing Good	Performing Good	1	281	13,384	3,511
Delaware	Performing Good	Performing Good	1	367	27,451	9,977
District of Columbia	Performing Good	Performing Good	0	40	2,865	1,060
Florida	Null	Bad Performing	115	1,379	89,474	-3,399
Georgia	Performing Good	Performing Good	0	705	49,096	16,250
Idaho	Performing Good	Performing Good	2	64	4,382	827
Illinois	Null	Bad Performing	192	1,845	80,166	-12,608
Indiana	Performing Good	Performing Good	0	578	53,555	18,383
Iowa	Performing Good	Performing Good	0	112	4,580	1,184
Kansas	Performing Good	Performing Good	0	74	2,914	836
Kentucky	Performing Good	Performing Good	0	523	36,592	11,200
Louisiana	Performing Good	Performing Good	0	156	9,217	2,196
Maine	Performing Good	Performing Good	0	35	1,271	454
Maryland	Performing Good	Performing Good	1	420	23,706	7,031
Massachusetts	Performing Good	Performing Good	2	491	28,634	6,786
Michigan	Performing Good	Performing Good	2	946	76,270	24,463
Minnesota	Performing Good	Performing Good	0	331	29,863	10,823



ELSE IF : The ElseIf function is handy to check multiple conditions. Remember, these Tableau conditions will only execute if it's previous IF or ELSEIF statement fails. The ElseIf function sequentially executes the statement. It will check the first condition, If the condition is TRUE, it executes the statement after the THEN keyword. If it is FALSE, it checks the Next one (ElseIf condition) and so on.

Syntax: IF <expr> THEN <then> [ELSEIF <expr2> THEN <then2>...] [ELSE <else>] END

elseif

State/Province	Profit Descr - if	Profit Descr - if-else	Profit Descr - elseif	Profit	Sales
Alabama	Performing Good	Performing Good	Breakeven	5,787	19,511
Arizona	Null	Bad Performing	Loss	-3,428	35,282
Arkansas	Performing Good	Performing Good	Breakeven	4,009	11,678
California	Performing Good	Performing Good	Performing Good	76,381	457,688
Colorado	Null	Bad Performing	Loss	-6,528	32,108
Connecticut	Performing Good	Performing Good	Breakeven	3,511	13,384
Delaware	Performing Good	Performing Good	Breakeven	9,977	27,451
District of Columbia	Performing Good	Performing Good	Breakeven	1,060	2,865
Florida	Null	Bad Performing	Loss	-3,399	89,474
Georgia	Performing Good	Performing Good	Breakeven	16,250	49,096
Idaho	Performing Good	Performing Good	Breakeven	827	4,382
Illinois	Null	Bad Performing	Loss	-12,608	80,166
Indiana	Performing Good	Performing Good	Breakeven	18,383	53,555
Iowa	Performing Good	Performing Good	Breakeven	1,184	4,580
Kansas	Performing Good	Performing Good	Breakeven	836	2,914
Kentucky	Performing Good	Performing Good	Breakeven	11,200	36,592
Louisiana	Performing Good	Performing Good	Breakeven	2,196	9,217
Maine	Performing Good	Performing Good	Breakeven	454	1,271
Maryland	Performing Good	Performing Good	Breakeven	7,031	23,706
Massachusetts	Performing Good	Performing Good	Breakeven	6,786	28,634
Michigan	Performing Good	Performing Good	Breakeven	24,463	76,270
Minnesota	Performing Good	Performing Good	Breakeven	10,823	29,863

AND: The Tableau AND function are used to check multiple expressio function accepts two arguments. If both the conditions are True, it Otherwise, it returns False.

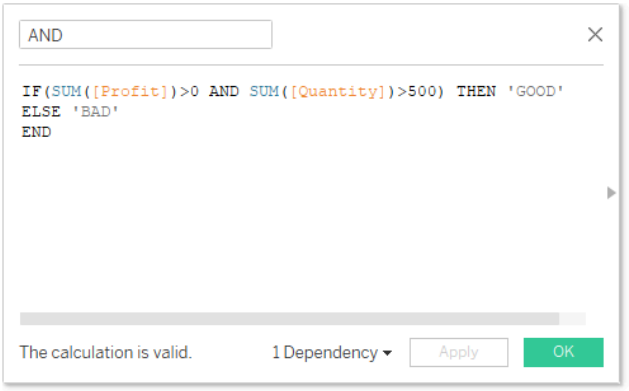
Syntax: IF <expr1> AND <expr2> THEN <then> END

Once you click on the Create Calculated Field… option, the following window will be opened. Here, we renamed the default calculation name as AND Function.

It will check whether the Profit is greater than 0 and the Quantity is greater than 500. If both these conditions are true, the Tableau logical AND function will return Good; otherwise, it will return Bad.

and

State/Province	AND	Profit	Sales	Quantity
Alabama	BAD	5,787	19,511	256
Arizona	BAD	-3,428	35,282	862
Arkansas	BAD	4,009	11,678	240
California	GOOD	76,381	457,688	7,667
Colorado	BAD	-6,528	32,108	693
Connecticut	BAD	3,511	13,384	281
Delaware	BAD	9,977	27,451	367
District of Columbia	BAD	1,060	2,865	40
Florida	BAD	-3,399	89,474	1,379
Georgia	GOOD	16,250	49,096	705
Idaho	BAD	827	4,382	64
Illinois	BAD	-12,608	80,166	1,845
Indiana	GOOD	18,383	53,555	578
Iowa	BAD	1,184	4,580	112
Kansas	BAD	836	2,914	74
Kentucky	GOOD	11,200	36,592	523
Louisiana	BAD	2,196	9,217	156
Maine	BAD	454	1,271	35
Maryland	BAD	7,031	23,706	420
Massachusetts	BAD	6,786	28,634	491
Michigan	GOOD	24,463	76,270	946
Minnesota	BAD	10,823	29,863	331

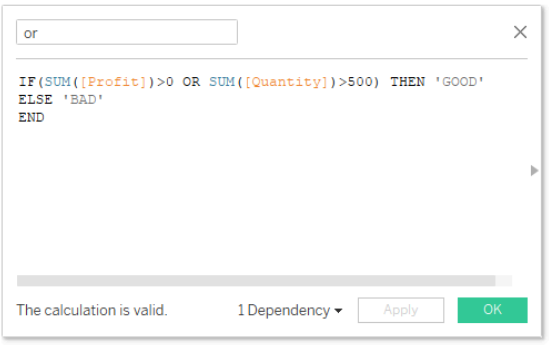


OR: The Tableau OR function is like either or statement in English. If both the conditions are False, It will return False; otherwise, it returns True.

Syntax: IF <expr1> OR <expr2> THEN <then> END

or

State/Province	or	Profit	Sales	Quantity
Alabama	GOOD	5,787	19,511	256
Arizona	GOOD	-3,428	35,282	862
Arkansas	GOOD	4,009	11,678	240
California	GOOD	76,381	457,688	7,667
Colorado	GOOD	-6,528	32,108	693
Connecticut	GOOD	3,511	13,384	281
Delaware	GOOD	9,977	27,451	367
District of Columbia	GOOD	1,060	2,865	40
Florida	GOOD	-3,399	89,474	1,379
Georgia	GOOD	16,250	49,096	705
Idaho	GOOD	827	4,382	64
Illinois	GOOD	-12,608	80,166	1,845
Indiana	GOOD	18,383	53,555	578
Iowa	GOOD	1,184	4,580	112
Kansas	GOOD	836	2,914	74
Kentucky	GOOD	11,200	36,592	523
Louisiana	GOOD	2,196	9,217	156
Maine	GOOD	454	1,271	35
Maryland	GOOD	7,031	23,706	420
Massachusetts	GOOD	6,786	28,634	491
Michigan	GOOD	24,463	76,270	946
Minnesota	GOOD	10,823	29,863	331



IFF: The IIF function is the simple version of the If Else Function. If the condition is True, then it will return First Statement otherwise, the second statement.

Syntax: IIF(test, then, else, [unknown])

IIF

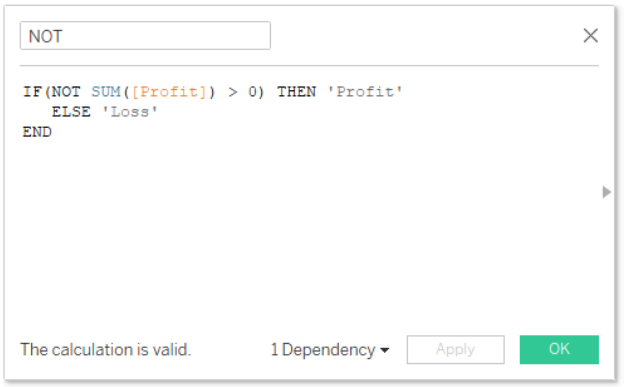
State/Province	IIF	Profit	Sales	Quantity
Alabama	Profit	5,787	19,511	256
Arizona	Loss	-3,428	35,282	862
Arkansas	Profit	4,009	11,678	240
California	Profit	76,381	457,688	7,667
Colorado	Loss	-6,528	32,108	693
Connecticut	Profit	3,511	13,384	281
Delaware	Profit	9,977	27,451	367
District of Columbia	Profit	1,060	2,865	40
Florida	Loss	-3,399	89,474	1,379
Georgia	Profit	16,250	49,096	705
Idaho	Profit	827	4,382	64
Illinois	Loss	-12,608	80,166	1,845
Indiana	Profit	18,383	53,555	578
Iowa	Profit	1,184	4,580	112
Kansas	Profit	836	2,914	74
Kentucky	Profit	11,200	36,592	523
Louisiana	Profit	2,196	9,217	156
Maine	Profit	454	1,271	35
Maryland	Profit	7,031	23,706	420
Massachusetts	Profit	6,786	28,634	491
Michigan	Profit	24,463	76,270	946
Minnesota	Profit	10,823	29,863	331

NOT: The Tableau NOT function return the exact opposite which means, True will become false, and vice versa.

Syntax: IF NOT <expr> THEN <then> END

NOT

State/Province	NOT	Profit	Sales	Quantity
Alabama	Loss	5,787	19,511	256
Arizona	Profit	-3,428	35,282	862
Arkansas	Loss	4,009	11,678	240
California	Loss	76,381	457,688	7,667
Colorado	Profit	-6,528	32,108	693
Connecticut	Loss	3,511	13,384	281
Delaware	Loss	9,977	27,451	367
District of Columbia	Loss	1,060	2,865	40
Florida	Profit	-3,399	89,474	1,379
Georgia	Loss	16,250	49,096	705
Idaho	Loss	827	4,382	64
Illinois	Profit	-12,608	80,166	1,845
Indiana	Loss	18,383	53,555	578
Iowa	Loss	1,184	4,580	112
Kansas	Loss	836	2,914	74
Kentucky	Loss	11,200	36,592	523
Louisiana	Loss	2,196	9,217	156
Maine	Loss	454	1,271	35
Maryland	Loss	7,031	23,706	420
Massachusetts	Loss	6,786	28,634	491
Michigan	Loss	24,463	76,270	946
Minnesota	Loss	10,823	29,863	331



ISNULL: The ISNULL function will check whether it is NULL or Not. If it is NULL, then it returns TRUE; otherwise, False will return.

Syntax: ISNULL(expression)

ISNULL

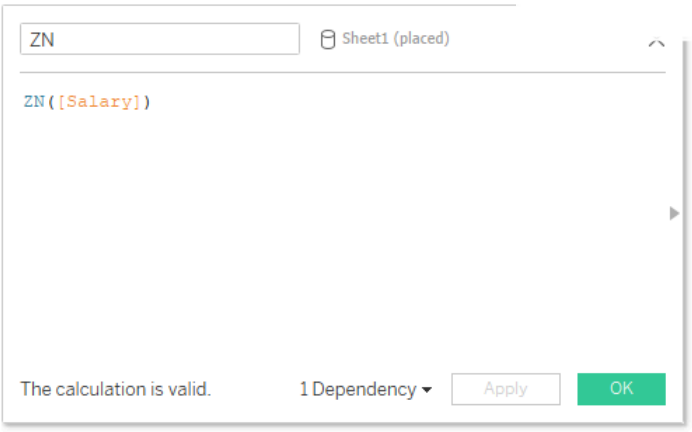
Name	Company	ISNULL	
Aarya	CRG	False	8,000
Aditya	CRG	False	8,000
Bhavesh	CRG	False	8,000
Jay	CRG	False	8,000
Malhar	Null	True	0
Tanmay	Null	True	0

ZN: The ZN will return the original values of Not Null values and 0 for Null values. In simple English, ZN in Tableau is used to replace the NULL values with 0

Syntax: ZN(expression)

ZN

Name	Company	Salary	
Aarya	CRG	8,000	8,000
Aditya	CRG	8,000	8,000
Bhavesh	CRG	8,000	8,000
Jay	CRG	8,000	8,000
Malhar	Null	Null	0
Tanmay	Null	Null	0



ISDATE: The ISDATE statement tests whether or not the string argument are often born-again to a legitimate date (TRUE) or if it cannot (FALSE).

Syntax: ISDATE(string)

IN: Returns TRUE if any value in <expr1> matches any value in <expr2>. The values in <expr2> can be a set, list of literal values, or combined field.

Eg: SUM([Cost]) IN (1000, 15, 200)

Syntax: <expr1> IN <expr2>

IFNULL: The IFNULL function is used to replace the NULL values with your own.

Syntax: IFNULL(expr1, expr2)

IFNULL

Name	Company	
Aarya	CRG	CRG
Aditya	CRG	CRG
Bhavesh	CRG	CRG
Jay	CRG	CRG
Malhar	Null	Unplaced
Tanmay	Null	Unplaced

CASE Statements

Tableau Case Function is similar to the IF ELSE or Else If function. The Case function evaluates a series of conditional expressions based on the condition result, and it will return the output.

Syntax:

CASE <Input_Expression>

WHEN <test_condition1> THEN result_1

WHEN <test_condition2> THEN result_2

.....

ELSE default_result

END

Arguments for this are:

- Input_Expression: This may be any column or expression on which you want to operate.
- test_condition: Desktop will compare this expression against the Input_Expression, and if it is correct, the TRUE result will be returned.
- result: If the test_condition is equal to Input_Expression, then this will return as an output. If they are not equal, default_result will return as output.

CASE -1

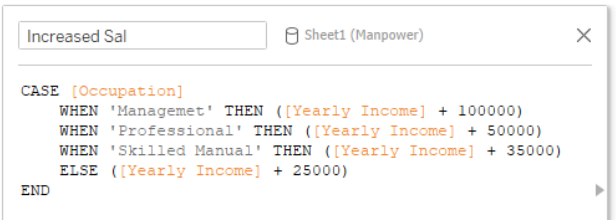
Emp ID	First Name	Last Name	Occupation	Case 1	
1	Jay	Charole	Professional	Sr. Software Developer	1,50,751
2	Malhar	Jadhav	Managemet	Administrators	1,97,302
3	Bhavesh	Jagtap	Skilled Manual	Software Developer	66,325
4	Aditya	Deodhar	Professional	Sr. Software Developer	1,54,669
5	Aarya	Charole	Clerical	Freshers	2,13,163
6	Prabhu	Sharma	Managemet	Administrators	1,36,277
7	Sachin	Baditya	Clerical	Freshers	2,01,249
8	Karan	Bakshi	Professional	Sr. Software Developer	1,77,449
9	Supriya	Kher	Professional	Sr. Software Developer	1,56,202
10	Deepika	Chopra	Clerical	Freshers	2,01,579
11	Surabhi	Thatte	Professional	Sr. Software Developer	1,96,683
12	Pratyusha	Beesa	Managemet	Administrators	1,28,158

Case 1 | Sheet1 (Manpower)

```
CASE [Occupation]
    WHEN 'Managemet' THEN 'Administrators'
    WHEN 'Professional' THEN 'Sr. Software Developer'
    WHEN 'Skilled Manual' THEN 'Software Developer'
    ELSE 'Freshers'
END
```

CASE-2

Emp ID	First Name	Last Name	Case 1	Occupation	Yearly Income	Increased Sal
1	Jay	Charole	Sr. Software Developer	Professional	150,751	200,751
2	Malhar	Jadhav	Administrators	Managemet	197,302	297,302
3	Bhavesh	Jagtap	Software Developer	Skilled Manual	66,325	101,325
4	Aditya	Deodhar	Sr. Software Developer	Professional	154,669	204,669
5	Aarya	Charole	Freshers	Clerical	213,163	238,163
6	Prabhu	Sharma	Administrators	Managemet	136,277	236,277
7	Sachin	Baditya	Freshers	Clerical	201,249	226,249
8	Karan	Bakshi	Sr. Software Developer	Professional	177,449	227,449
9	Supriya	Kher	Sr. Software Developer	Professional	156,202	206,202
10	Deepika	Chopra	Freshers	Clerical	201,579	226,579
11	Surabhi	Thatte	Sr. Software Developer	Professional	196,683	246,683
12	Pratyusha	Beesa	Administrators	Managemet	128,158	228,158



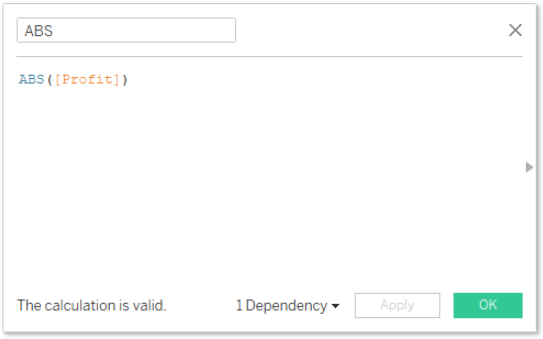
MATH Functions

ABS: The Tableau ABS math function is used to return the absolute positive value.

Syntax: ABS(number)

ABS

Sub-Catego..	Profit	ABS
Accessories	41,937	43,798
Appliances	18,329	35,589
Art	6,653	6,701
Binders	31,426	108,553
Bookcases	-3,632	21,068
Chairs	27,224	47,494
Copiers	56,094	56,094
Envelopes	6,988	6,988
Fasteners	2,429	2,495
Furnishings	13,892	27,265
Labels	5,573	5,636
Machines	3,462	63,699
Paper	34,512	34,524
Phones	45,051	60,112
Storage	21,285	34,246
Supplies	-1,171	4,860
Tables	-17,753	47,254

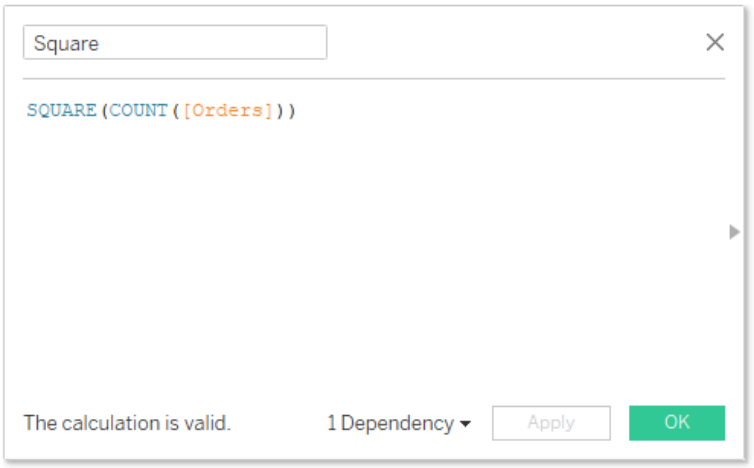


SQUARE: The Tableau Square function finds the square of a given number

Syntax: SQUARE(Number)

SQUARE

Sub-Catego..	Count of Orders	Square
Accessories	775	600,625
Appliances	474	224,676
Art	821	674,041
Binders	1,548	2,396,304
Bookcases	232	53,824
Chairs	634	401,956
Copiers	70	4,900
Envelopes	256	65,536
Fasteners	229	52,441
Furnishings	1,009	1,018,081
Labels	368	135,424
Machines	117	13,689
Paper	1,384	1,915,456
Phones	903	815,409
Storage	856	732,736
Supplies	192	36,864
Tables	326	106,276



SQRT: The Tableau Sqrt function finds the square root of a given number.

Syntax: SQRT(Number)

SQRT

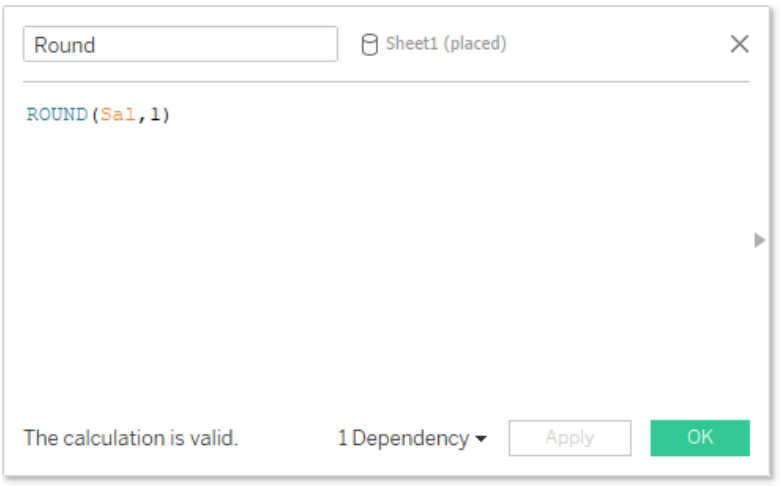
Sub-Catego..	Count of Orders	Square	SQRT
Accessories	775	600,625	775
Appliances	474	224,676	474
Art	821	674,041	821
Binders	1,548	2,396,304	1,548
Bookcases	232	53,824	232
Chairs	634	401,956	634
Copiers	70	4,900	70
Envelopes	256	65,536	256
Fasteners	229	52,441	229
Furnishings	1,009	1,018,081	1,009
Labels	368	135,424	368
Machines	117	13,689	117
Paper	1,384	1,915,456	1,384
Phones	903	815,409	903
Storage	856	732,736	856
Supplies	192	36,864	192
Tables	326	106,276	326

ROUND: The Tableau math Round function returns the nearest integer value. It accepts the second argument (optional) to specify the number of decimals.

Syntax: ROUND(Number, no_of_decimals)

ROund

Name	Round	
Aarya	66,755.9	66,755.9
Aditya	65,756.5	65,756.5
Bhavesh	79,796.9	79,796.9
Jay	7,698.8	7,698.78
Malhar	79,797.9	79,797.89
Tanmay	6,767.8	6,767.8

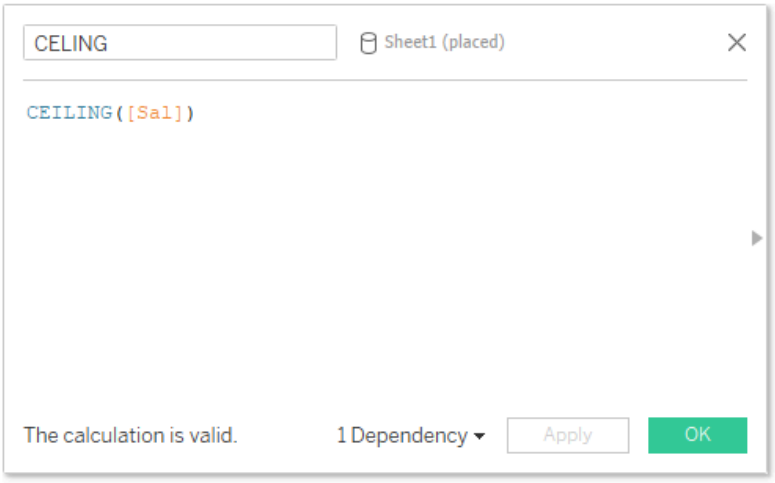


CEILING: The Tableau Ceiling function returns the closest integer value, which is greater than or equal to a given value.

Syntax: CEILING(Number)

Floor

Name	CELING	
Aarya	66,756	66,755.9
Aditya	65,757	65,756.5
Bhavesh	79,797	79,796.9
Jay	7,699	7,698.78
Malhar	79,798	79,797.89
Tanmay	6,768	6,767.8

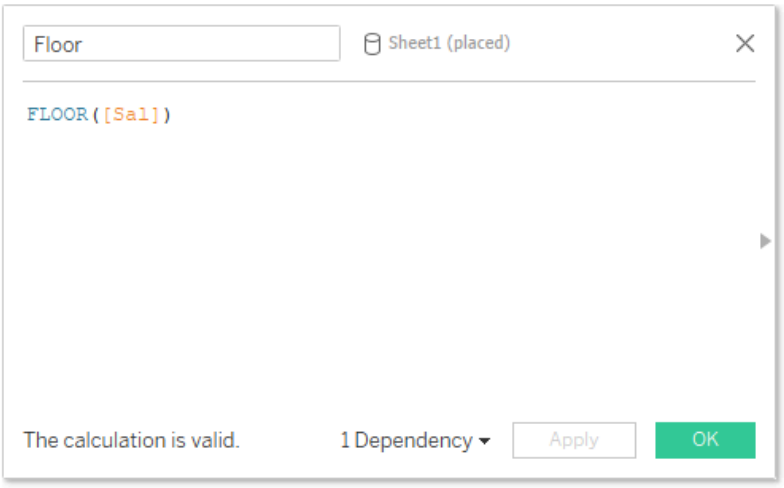


FLOOR: The Tableau math Floor function is used to return the closest integer value, which is less than or equal to a given value.

Syntax: FLOOR(Number)

Floor

Name	Floor	
Aarya	66,755	66,755.9
Aditya	65,756	65,756.5
Bhavesh	79,796	79,796.9
Jay	7,698	7,698.78
Malhar	79,797	79,797.89
Tanmay	6,767	6,767.8



POWER: This function is used to find the Power of a given number. Use the second argument to specify the Power.

Syntax: POWER(Number, power)

DIV: The Tableau DIV function divides one number by another.

Syntax: DIV(integer, integer)

String Functions

ASCII Function: The ASCII String functions returns the ASCII code for the first character in the string.

Syntax: ASCII(string)

Example: ASCII("Alabama") = 65

CHAR Function: This String functions works in the reverse of the ASCII function. It converts an integer ASCII code into a character.

Syntax: CHAR(integer)

Example: CHAR(65) = 'A'

CONTAINS Function: The CONTAINS String functions will return a TRUE value if the string contains the substring and a FALSE if it does not.

Syntax: CONTAINS(string, substring)

Example: CONTAINS("InterWorks", "Works") = TRUE

ENDSWITH Function: A similar function to the one above, the ENDSWITH string functions tests if the string ends with the selected substring, returning either TRUE or FALSE.

Syntax: ENDSWITH(string, substring)

Example: ENDSWITH("software", "ware") = TRUE

FIND Function: The FIND string functions returns the start of the substring within the string. The first character in the string is position 1. If the substring is not found, it will return a value of 0.

Syntax: FIND(string, substring, [start])

Example: FIND("Oklahoma", "la") = 3

If the start argument is also defined, any instance of the substring that appears before the start will be ignored.

Example: FIND("Mississippi", "iss", 4) = 5

ISDATE Function: This is a logical test that is also included in the list of logical functions. It tests a string to determine if it is a valid date (true/false). This String functions is also included in the Date Functions.

Syntax: ISDATE(string)

Example: ISDATE("September 29, 2014") = true

LEFT Function: This String functions returns the characters of the string using the specified number as the amount.

Syntax: LEFT(string, num_chars)

Example: LEFT("cheetah", 4) = 'chee'

If the start_of_week is omitted, then it is determined by the data source.

LEN Function: The LEN String functions (or length function) returns the character count of the given string field.

Syntax: LEN(string)

Example: LEN("Missouri") = 8

LOWER Function: This String functions converts all characters in the given string into lower case letters.

Syntax: LOWER(string)

Example: LOWER("Tableau") = "tableau"

LTRIM Function: This String functions will remove any spaces starting the string.

Syntax: LTRIM(string)

Example: LTRIM(" Harry") = "Harry"

MAX Function: The MAX function exists in several categories of functions, including String Functions. When used for strings, the MAX function returns the value that is highest in the sort sequence defined by the database for that field's column. If either field is NULL, then the function will return NULL.

Syntax: MAX(a, b)

Example: MAX("Apple", "Banana") = "Banana"

MID Function: The MID String functions returns the characters from the middle of a text string. The start argument is where the returned value will begin and the length argument is how many characters will be returned. If the length is not included, then all remaining characters after the start position will be included. The first character in the string is position 1.

Syntax: MID(string, start, [length])

Example: MID("Stillwater", 3, 2) = "il"

MIN Function: Similar to the MAX function, the MIN function returns the minimum between a and b, which must be of the same data type (i.e. string). With strings, the MIN function will return the lower value based on the sort sequence as defined in the database. If either argument is NULL, then this String functions will return NULL.

Syntax: MIN(a, b)

Example: MIN("Apple", "Banana") = "Apple"

REPLACE Function: This String functions will replace find any occurrence of the substring in the string and replace those characters with the replacement string. If the substring is not found in the string, then there is no change.

Syntax: REPLACE(string, substring, replacement)

Example: REPLACE("calculation", "ion", "ed") = "calculated"

RIGHT Function: This String functions is the opposite of the LEFT function. It returns the characters from the end of a given string, the amount determined by the number of characters argument.

Syntax: RIGHT(string, num_chars)

Example: RIGHT("Nebraska", 6) = "braska"

RTRIM Function: The partner function to LTRIM. The RTRIM String functions returns the string with any trailing spaces removed.

Syntax: RTRIM(string)

Example: RTRIM("Harry ") = "Harry"

SPACE Function: The SPACE function returns a string with the number of spaces defined by the number argument.

Syntax: SPACE(number)

Example: SPACE(1) = " "

STARTSWITH Function: This is the opposite of the ENDSWITH function, and it returns a TRUE or FALSE result if the string starts with the substring.

Syntax: STARTSWITH(string, substring)

Example: STARTSWITH("Michigan", "Mic") = TRUE

TRIM Function: The TRIM function removes any leading or trailing spaces from the string.

Syntax: TRIM(string)

Example: TRIM(“ Harry ”) = “Harry”

UPPER Function: The UPPER function works in opposite to the LOWER function. It takes all the characters in the string and converts them to uppercase characters.

Syntax: UPPER(string)

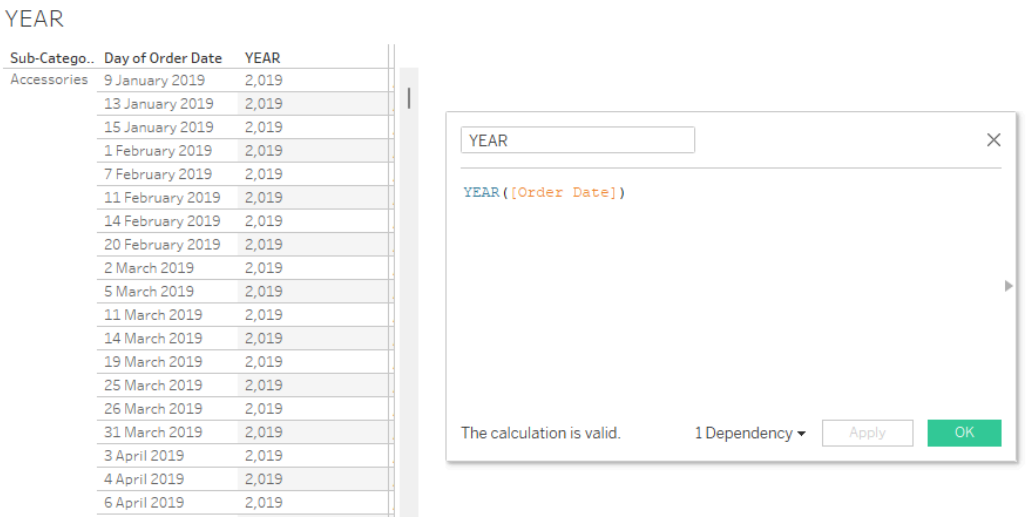
Example: UPPER(“nasa”) = “NASA”

Date Functions

Tableau provides various Date Functions such as Year, Month, Day, makedate, maketime, datediff, datepart, dateadd, datename, datetrunc, now, today, etc.

YEAR: The Tableau YEAR function is used to return Year from the given date

Syntax: YEAR(Date)

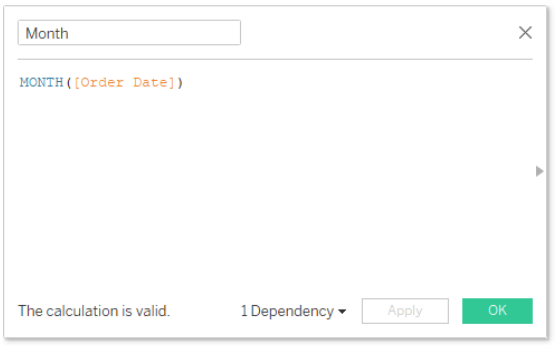


MONTH: The Tableau Month function is used to return the Month number

Syntax: MONTH(Date)

MONTH

Sub-Catego..	Day of Order Date	Month
Accessories	9 January 2019	1
	13 January 2019	1
	15 January 2019	1
	1 February 2019	2
	7 February 2019	2
	11 February 2019	2
	14 February 2019	2
	20 February 2019	2
	2 March 2019	3
	5 March 2019	3
	11 March 2019	3
	14 March 2019	3
	19 March 2019	3
	25 March 2019	3
	26 March 2019	3
	31 March 2019	3
	3 April 2019	4
	4 April 2019	4
	6 April 2019	4
	8 April 2019	4

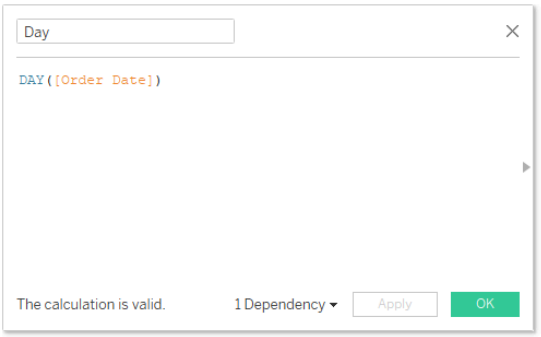


DAY: The Tableau DAY function is used to extract or return the Day number from a given date.

Syntax: DAY(Date)

YEAR()

Sub-Catego..	Day of Order Date	Day
Accessories	9 January 2019	9
	13 January 2019	13
	15 January 2019	15
	1 February 2019	1
	7 February 2019	7
	11 February 2019	11
	14 February 2019	14
	20 February 2019	20
	2 March 2019	2
	5 March 2019	5
	11 March 2019	11
	14 March 2019	14
	19 March 2019	19
	25 March 2019	25
	26 March 2019	26
	31 March 2019	31
	3 April 2019	3
	4 April 2019	4

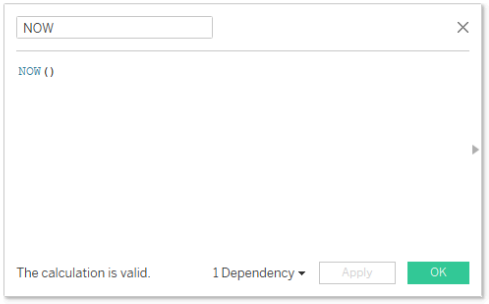


NOW: The Tableau NOW function is used to return today's date and time.

Syntax: NOW()

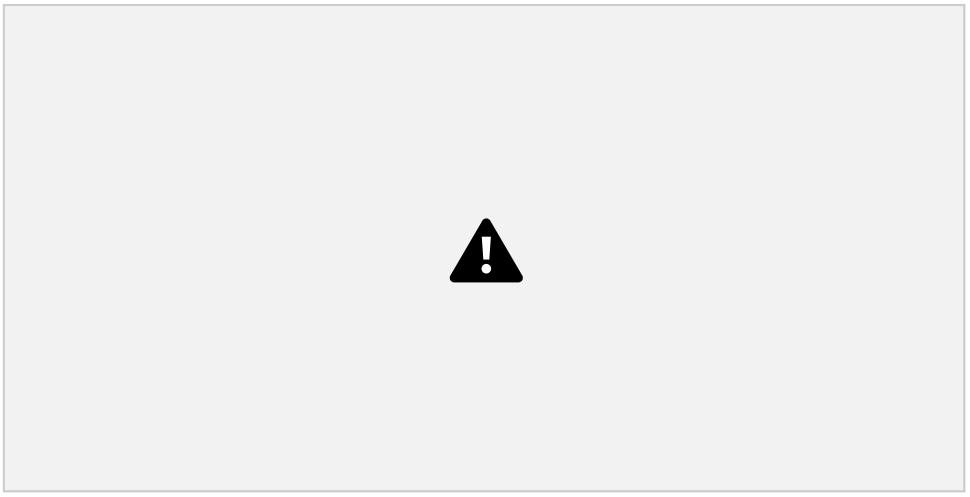
NOW

Sub-Catego..	Day of Order Date	NOW
Accessories	9 January 2019	28-04-2023 14:46:54
	13 January 2019	28-04-2023 14:46:54
	15 January 2019	28-04-2023 14:46:54
	1 February 2019	28-04-2023 14:46:54
	7 February 2019	28-04-2023 14:46:54
	11 February 2019	28-04-2023 14:46:54
	14 February 2019	28-04-2023 14:46:54
	20 February 2019	28-04-2023 14:46:54
	2 March 2019	28-04-2023 14:46:54
	5 March 2019	28-04-2023 14:46:54
	11 March 2019	28-04-2023 14:46:54
	14 March 2019	28-04-2023 14:46:54
	19 March 2019	28-04-2023 14:46:54
	25 March 2019	28-04-2023 14:46:54
	26 March 2019	28-04-2023 14:46:54
	31 March 2019	28-04-2023 14:46:54
	3 April 2019	28-04-2023 14:46:54
	4 April 2019	28-04-2023 14:46:54
	6 April 2019	28-04-2023 14:46:54
	8 April 2019	28-04-2023 14:46:54



TODAY: The Tableau Today function is used to return today's date

Syntax: TODAY()

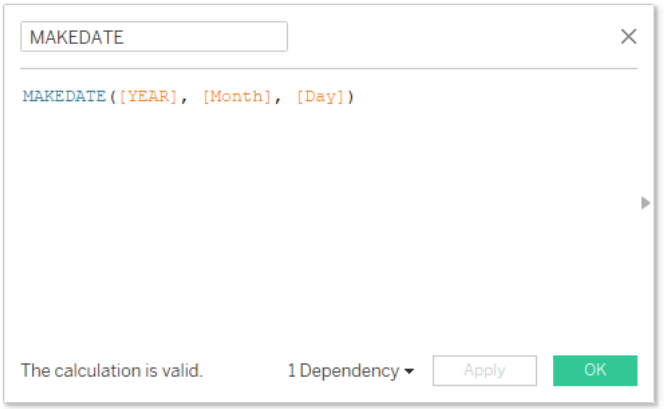


MAKEDATE: The Tableau MakeDate function is used to return date from the year, month, and day.

Syntax: MAKEDATE(year, month, day)

MAKEDATE

Sub-Catego..	Day of Order Date	MAKEDATE
Accessories	9 January 2019	09-01-2019
	13 January 2019	13-01-2019
	15 January 2019	15-01-2019
	1 February 2019	01-02-2019
	7 February 2019	07-02-2019
	11 February 2019	11-02-2019
	14 February 2019	14-02-2019
	20 February 2019	20-02-2019
	2 March 2019	02-03-2019
	5 March 2019	05-03-2019
	11 March 2019	11-03-2019
	14 March 2019	14-03-2019
	19 March 2019	19-03-2019
	25 March 2019	25-03-2019
	26 March 2019	26-03-2019
	31 March 2019	31-03-2019
	3 April 2019	03-04-2019
	4 April 2019	04-04-2019
	6 April 2019	06-04-2019

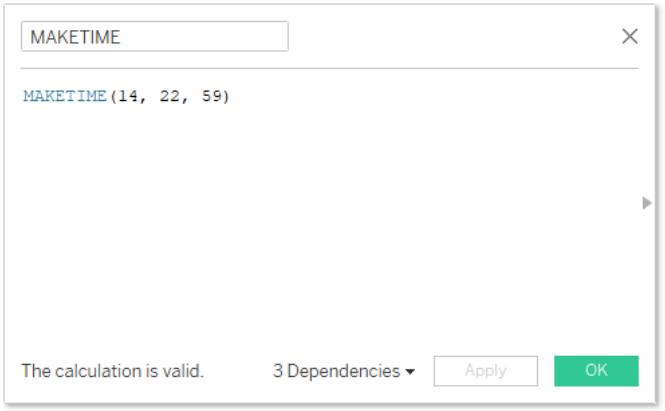


MAKETIME: The Tableau MakeTime date function is used to return time from an hour, minute, and second.

Syntax: MAKETIME(hour, minute, second)

MAKETIME

Sub-Catego..	Day of Order Date	MAKETIME
Accessories	9 January 2019	30-12-1899 14:22:59
	13 January 2019	30-12-1899 14:22:59
	15 January 2019	30-12-1899 14:22:59
	1 February 2019	30-12-1899 14:22:59
	7 February 2019	30-12-1899 14:22:59
	11 February 2019	30-12-1899 14:22:59
	14 February 2019	30-12-1899 14:22:59
	20 February 2019	30-12-1899 14:22:59
	2 March 2019	30-12-1899 14:22:59
	5 March 2019	30-12-1899 14:22:59
	11 March 2019	30-12-1899 14:22:59
	14 March 2019	30-12-1899 14:22:59
	19 March 2019	30-12-1899 14:22:59
	25 March 2019	30-12-1899 14:22:59
	26 March 2019	30-12-1899 14:22:59
	31 March 2019	30-12-1899 14:22:59
	3 April 2019	30-12-1899 14:22:59
	4 April 2019	30-12-1899 14:22:59
	6 April 2019	30-12-1899 14:22:59
	8 April 2019	30-12-1899 14:22:59
	11 April 2019	30-12-1899 14:22:59

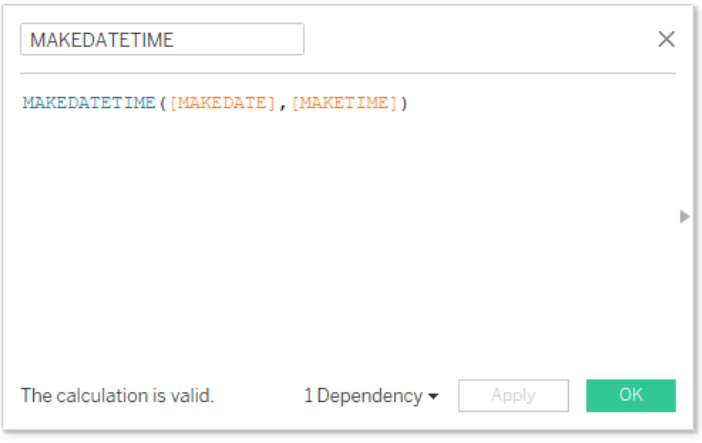


MAKEDATETIME: The Tableau MakeDateTime function is used to return date and time from Date, Time.

Syntax: MAKEDATETIME(Date, Time)

MAKEDATETIME

Order Date	MAKEDATETIME
03-01-2019	03-01-2019 14:22:59
04-01-2019	04-01-2019 14:22:59
05-01-2019	05-01-2019 14:22:59
06-01-2019	06-01-2019 14:22:59
07-01-2019	07-01-2019 14:22:59
09-01-2019	09-01-2019 14:22:59
10-01-2019	10-01-2019 14:22:59
11-01-2019	11-01-2019 14:22:59
13-01-2019	13-01-2019 14:22:59
14-01-2019	14-01-2019 14:22:59
15-01-2019	15-01-2019 14:22:59
16-01-2019	16-01-2019 14:22:59
18-01-2019	18-01-2019 14:22:59
19-01-2019	19-01-2019 14:22:59
20-01-2019	20-01-2019 14:22:59
21-01-2019	21-01-2019 14:22:59
23-01-2019	23-01-2019 14:22:59
26-01-2019	26-01-2019 14:22:59
27-01-2019	27-01-2019 14:22:59
28-01-2019	28-01-2019 14:22:59

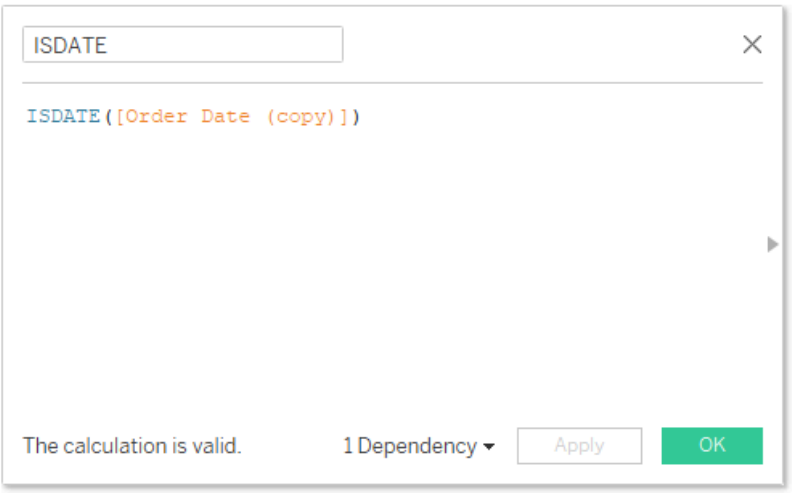


ISDATE: The Tableau IsDate function is used to check whether the given string is a date or not. It returns either True or False.

Syntax: ISDATE(string)

ISDATE

Order Date ..	ISDATE
2019-01-03	True
2019-01-04	True
2019-01-05	True
2019-01-06	True
2019-01-07	True
2019-01-09	True
2019-01-10	True
2019-01-11	True
2019-01-13	True
2019-01-14	True
2019-01-15	True
2019-01-16	True
2019-01-18	True
2019-01-19	True
2019-01-20	True
2019-01-21	True
2019-01-23	True
2019-01-26	True
2019-01-27	True

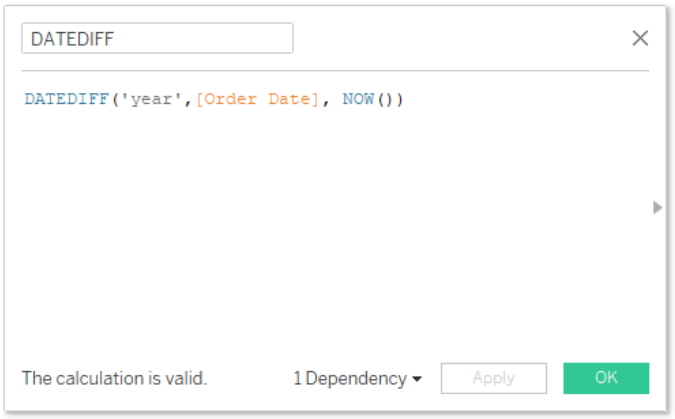


DATEDIFF: The Tableau DATEDIFF function is used to return the date difference between the start and the end date. Use the first argument to specify the difference term. This datediff function accepts YEAR, MONTH, DAY, etc.

Syntax: DATEDIFF(Difference_term, Start_Date, End_Date)

DATEDIFF

Order Date	DATEDIFF
03-01-2019	4
04-01-2019	12
05-01-2019	4
06-01-2019	36
07-01-2019	8
09-01-2019	8
10-01-2019	8
11-01-2019	4
13-01-2019	44
14-01-2019	4
15-01-2019	4
16-01-2019	16
18-01-2019	4
19-01-2019	16
20-01-2019	68
21-01-2019	24
23-01-2019	8
26-01-2019	36
27-01-2019	12



DATEADD: The Tableau DATEADD function is used to add user-specified intervals to an actual date. Use the first argument to define the date part term, and the second arg to specify the interval. This DATEADD function accepts YEAR, MONTH, DAY, etc.

Syntax: DATEADD(Date_part, interval, Date)

DATEADD

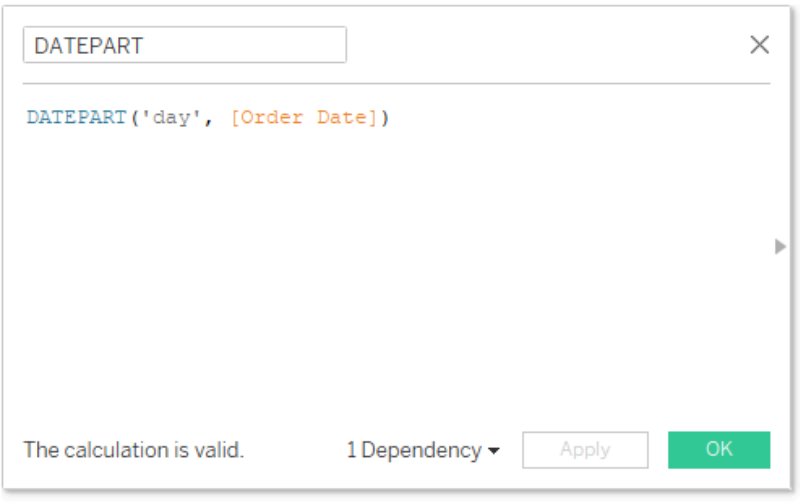
Order Date	Year of DAT..
03-01-2019	2024
04-01-2019	2024
05-01-2019	2024
06-01-2019	2024
07-01-2019	2024
09-01-2019	2024
10-01-2019	2024
11-01-2019	2024
13-01-2019	2024
14-01-2019	2024
15-01-2019	2024
16-01-2019	2024
18-01-2019	2024
19-01-2019	2024
20-01-2019	2024
21-01-2019	2024
23-01-2019	2024
26-01-2019	2024
27-01-2019	2024
28-01-2019	2024

DATEPART: The Tableau DATEPART function is used to extract or return part of a date. Use the first arg to specify the date part. This DatePart function accepts YEAR, MONTH, DAY, HOUR, MIN, etc.

Syntax: DATEPART(Date_part, Date)

DATEPART

Order Date	DATEPART
03-01-2019	3
04-01-2019	12
05-01-2019	5
06-01-2019	54
07-01-2019	14
09-01-2019	18
10-01-2019	20
11-01-2019	11
13-01-2019	143
14-01-2019	14
15-01-2019	15
16-01-2019	64
18-01-2019	18
19-01-2019	76
20-01-2019	340
21-01-2019	126
23-01-2019	46
26-01-2019	234
27-01-2019	81

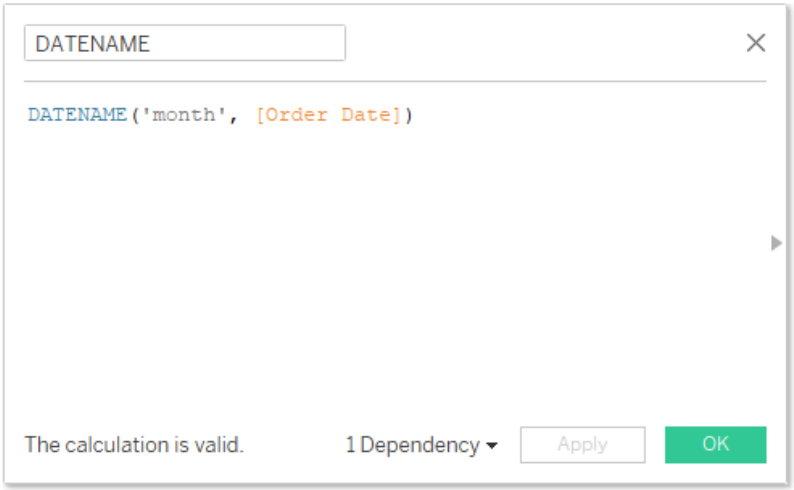


DATENAME: The Tableau DATENAME function is used to return the date part name. Use the first argument to specify the date part, and it accepts YEAR, MONTH, DAY.

Syntax: DATENAME(Date_part, Date)

DATENAME

Order Date	DATENAME
03-01-2019	January
04-01-2019	January
05-01-2019	January
06-01-2019	January
07-01-2019	January
09-01-2019	January
10-01-2019	January
11-01-2019	January
13-01-2019	January
14-01-2019	January
15-01-2019	January
16-01-2019	January
18-01-2019	January
19-01-2019	January
20-01-2019	January
21-01-2019	January
23-01-2019	January
26-01-2019	January
27-01-2019	January



DATETRUNC: The Tableau DATETRUNC is one of the date function. It is used to return the first day of the specified date part. Use the first argument to specify the date part, and this datetrunc accepts YEAR, MONTH, DAY, etc.

Syntax: DATETRUNC(Date_part, Date)

DATETRUNC

Order Date	DATETRUNC
03-01-2019	01-01-2019 00:00:00
04-01-2019	01-01-2019 00:00:00
05-01-2019	01-01-2019 00:00:00
06-01-2019	01-01-2019 00:00:00
07-01-2019	01-01-2019 00:00:00
09-01-2019	01-01-2019 00:00:00
10-01-2019	01-01-2019 00:00:00
11-01-2019	01-01-2019 00:00:00
13-01-2019	01-01-2019 00:00:00
14-01-2019	01-01-2019 00:00:00
15-01-2019	01-01-2019 00:00:00
16-01-2019	01-01-2019 00:00:00
18-01-2019	01-01-2019 00:00:00
19-01-2019	01-01-2019 00:00:00
20-01-2019	01-01-2019 00:00:00
21-01-2019	01-01-2019 00:00:00
23-01-2019	01-01-2019 00:00:00
26-01-2019	01-01-2019 00:00:00
27-01-2019	01-01-2019 00:00:00

DATEPARSE: This Tableau Date Function returns the date_string as a date convert a field to date. The date_format argument describes how the s arranged.

Syntax: DATEPARSE(date_format, [date_string])

Parameters

Parameters in Tableau enable users to add some advanced calculations and calculated fields. Parameters provide adding a non-existing variable to the entire work and simplify the needs and requirements to analyze and visualize the data.

The parameters in Tableau are the workbook variables like a number, date, or calculated field that allows users to replace a constant value in a calculation, filter, or reference line.

For example, the user can create a new calculated field that returns True if the aggregate of total marks is greater than 90% and returns False if it is less than 90%. Users can replace the constant value of "90%" in the formula with the parameters in Tableau as per the requirements. With the parameter control, users can dynamically vary the threshold values in their calculation.

What is the Use of Tableau Parameters

Following are the benefits of using parameters in Tableau software:

1. You can process the data and add a filter using tableau parameters which provides you more flexibility to work with datasets and apply convenient tricks to them.
2. Parameters are a method that transfers the control on datasets from the author of the report to the user.
3. Improves user engagement as he can apply different filters to create his own perspective from the available datasets.
4. Helps to extract better insights from the raw datasets through a data-driven parameter extension tableau.
5. Parameter actions tableau also allows the user to dynamically change the range of the values without creating the filters again and again.
6. Allows the collaborators and other stakeholders to only see the insights associated with their department and avoid the remaining ones.
7. Features like sheet swapping using parameter tableau and tableau filter measure names with parameters are mostly used to hide, display, and share only a finite number of visualizations or pages of a report with other users. It's similar to creating an abstraction layer on the top of your database.

Some Case Studies

Top N Parameters in Tableau

The following are the steps to create Top N Parameters in Tableau.

In this Parameters demo, you will find the Top N Sub-Categories giving maximum sales.

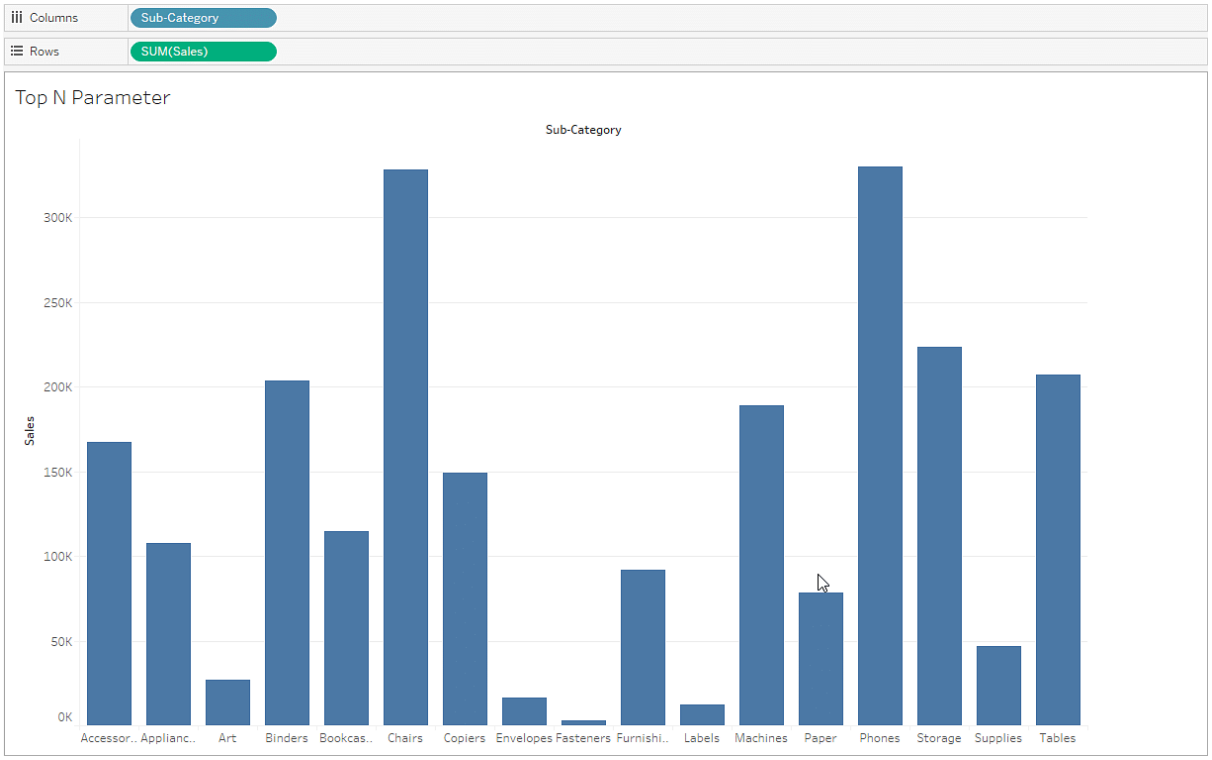
Start Tableau and import the sample superstore dataset.

Please create a new sheet and rename it as Top N Parameters, for your reference.

Now, select sales from the measures tab and drag it to rows.

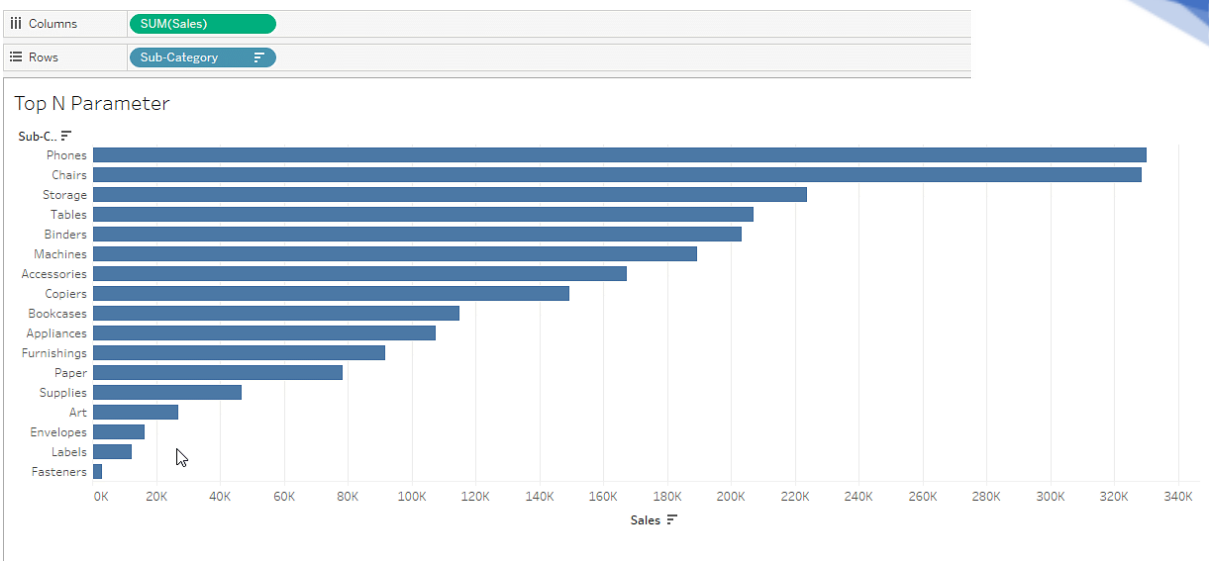
Select sub-category from dimensions and drag it to columns.

Tableau will automatically generate a bar chart as shown below.

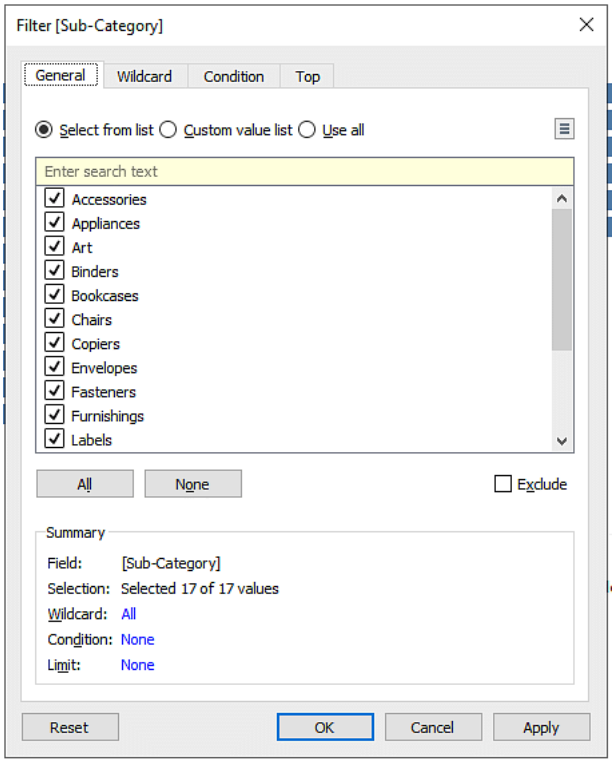


The orientation can be changed if necessary.

The next step is to organize the chart in descending order of the sum of sales.

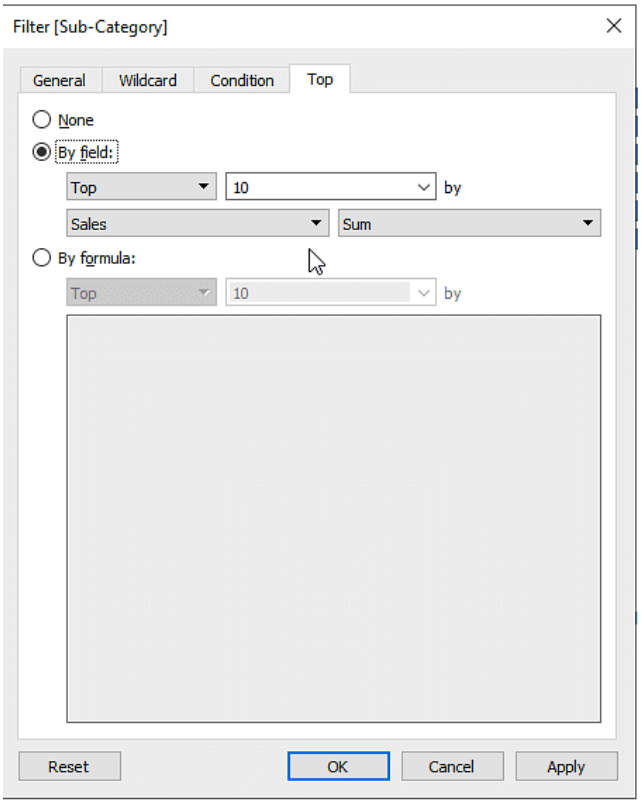


To find the Top N Sub-Categories giving the maximum sales, drag the sub-category pill from dimensions onto the filter in the marks-sheet.



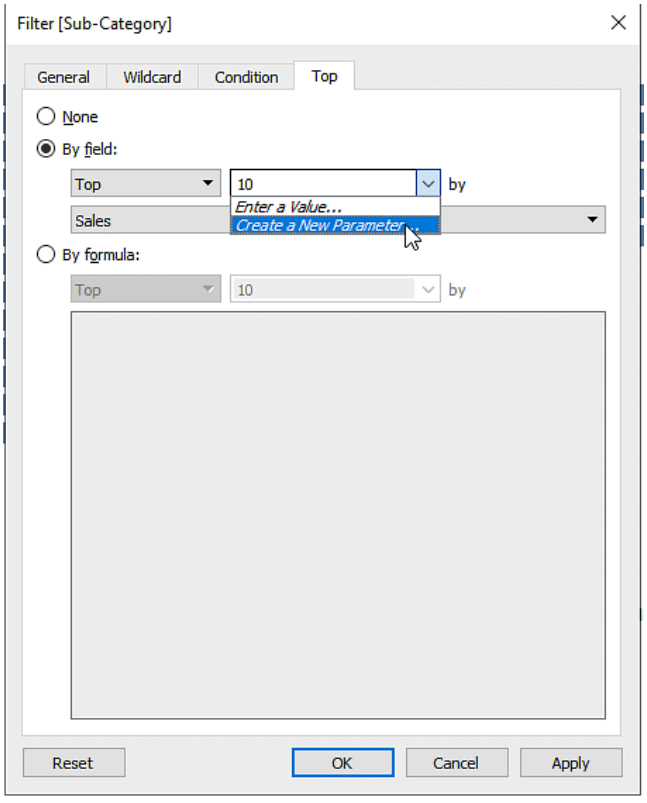
A new "filter" dialogue box will pop-up on the screen.

Select the top option in the menu bar.



Select the field option

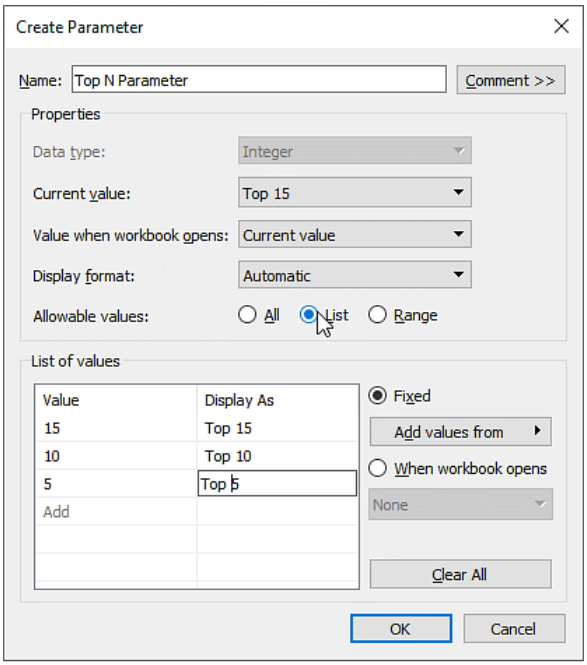
Select the new parameter option



Rename the parameter

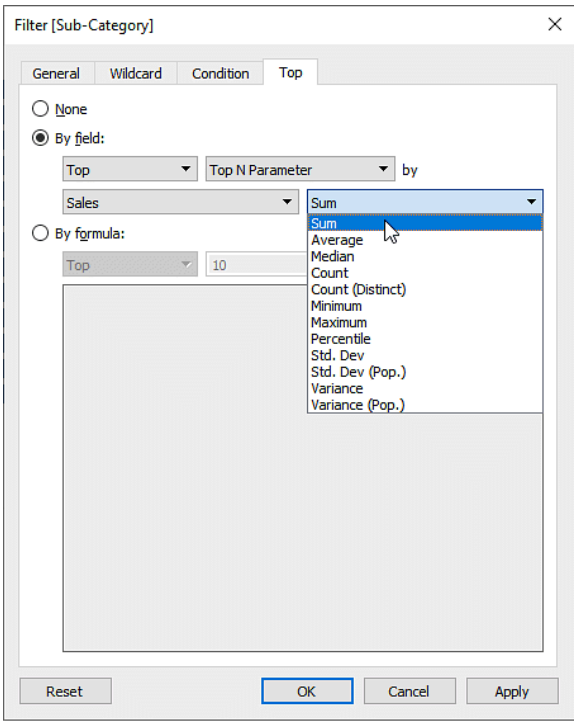
Select the list option

Write in the values and type in the text into the "display as" option.



Make sure the aggregation is SUM.

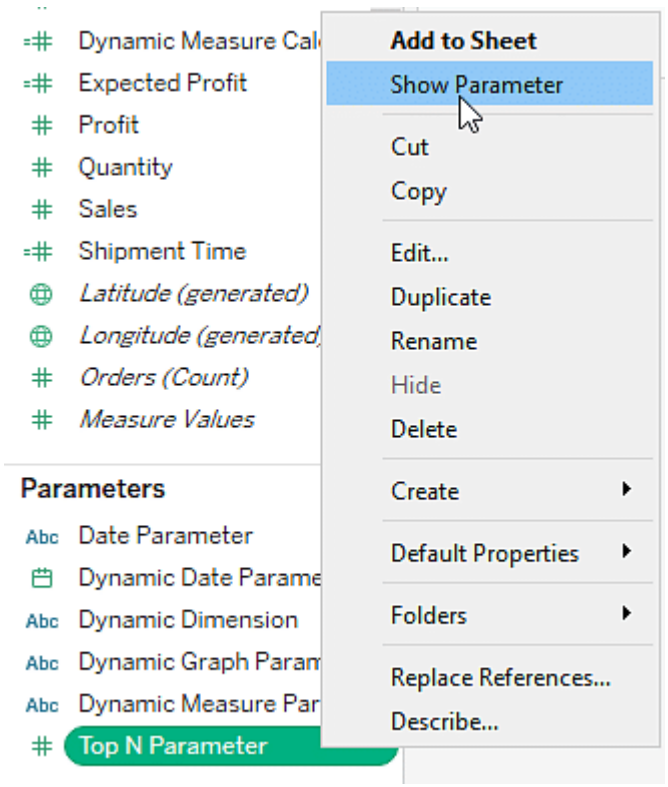
Select the OK option.



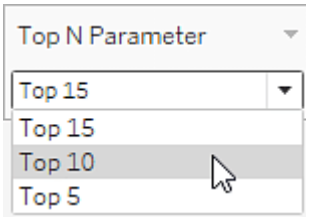
Under the measures template, you can find the newly created parameter.

Select the newly created parameter and right-click the pill.

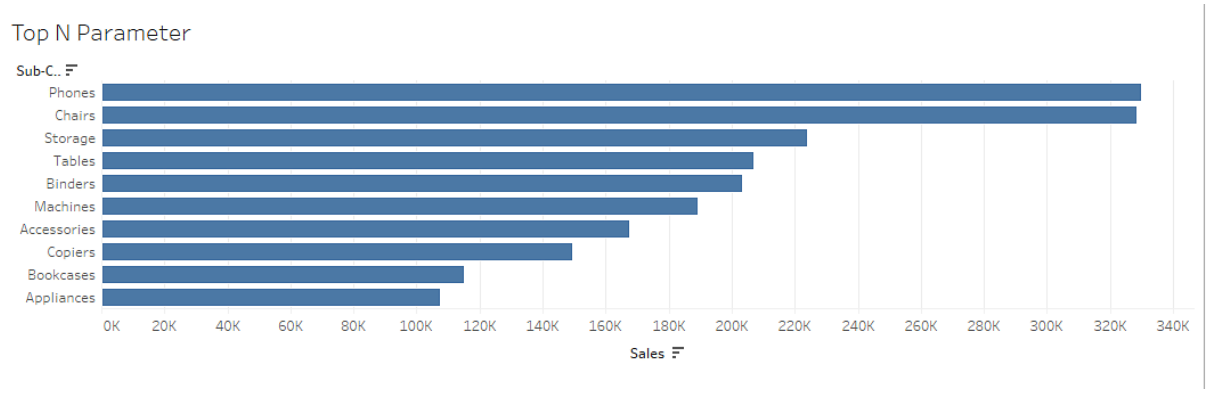
Select the show parameter option and select the top 10 option.



The Tableau will provide the Top N Parameter list on the screen.



Select Top 10 and Tableau will present the output.



With that, you have seen how to create Top N Parameters in Tableau. Next, this tutorial will look into the Date Parameters in Tableau.

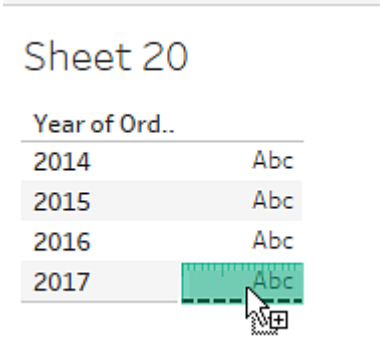
Data Field Parameters

In the date parameters, you will find out the sales according to the dates. It could be in terms of a year, quarter, month, week, or day. The "Parameter options" will be available in the parameter.

Now, the following steps are needed to be followed to create a date parameter.

Please create a new sheet and rename it as Date Parameter.

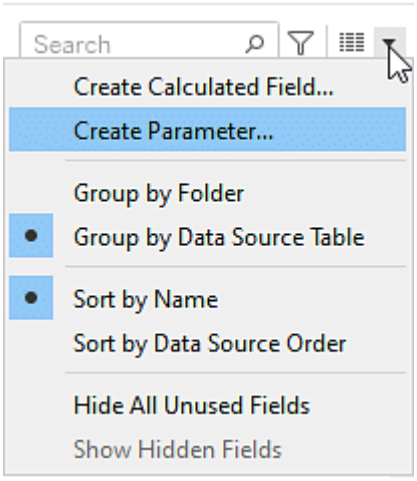
Drag order date to rows and sales to the tt table as shown below.



The resultant text table will as follows.

Year of Ord..	
2014	49,544
2015	61,619
2016	81,795
2017	93,439

To create a new parameter, select the arrow icon in the top left corner and select the create parameter option as shown below.



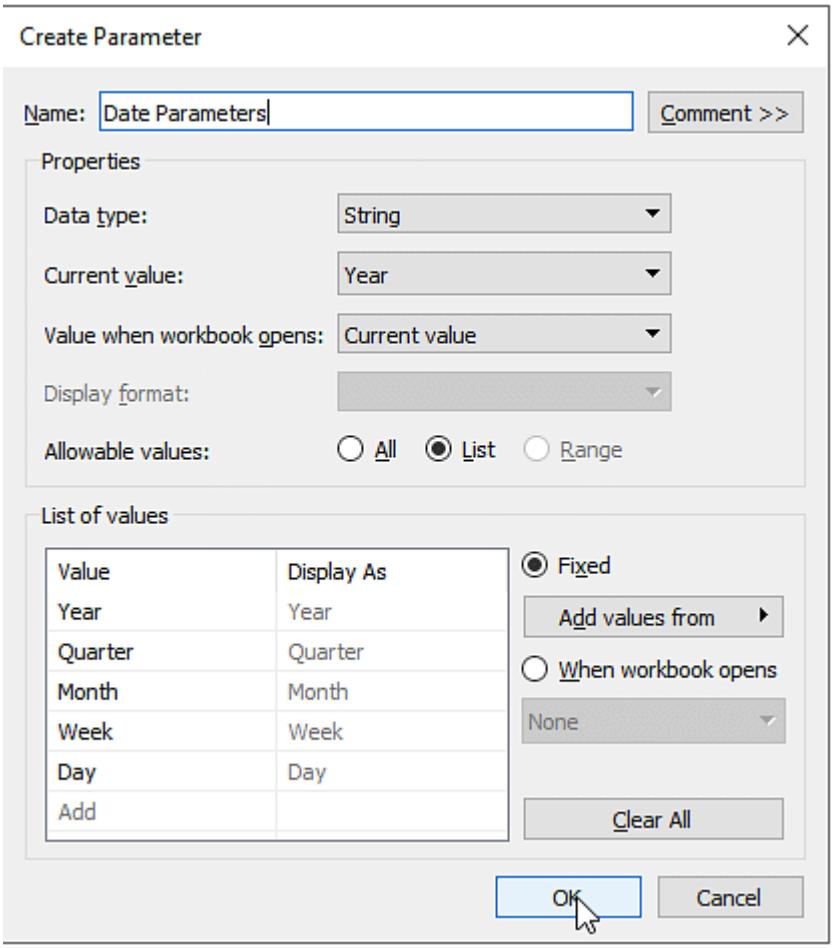
The new parameter dialogue box will appear on the screen.

Rename the parameter as date parameters

Select the datatype as string.

Select the list option

Add the values and "display as" details and select OK as shown below.

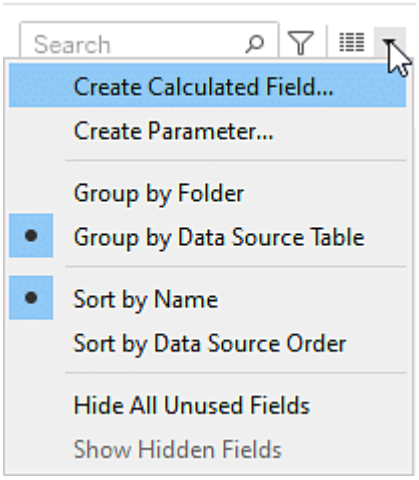


According to the year, quarter, month, week, and day, you need a new calculated field to display the sales.

Create a new calculated field named date calculation.

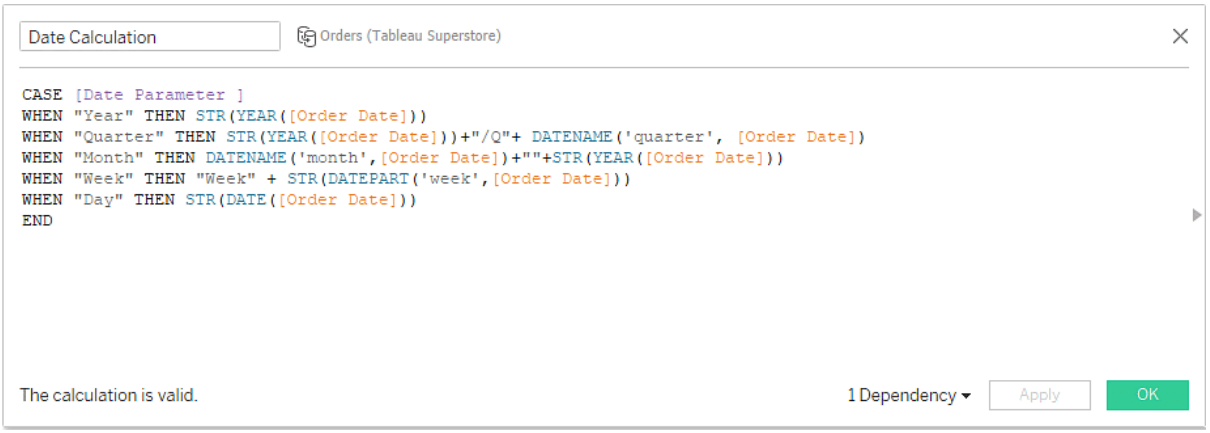
Go to the arrow icon option in the top left corner.

Select the arrow icon and click on the "create a calculated field" option, as shown below.



A new window will appear on your screen. Here you can add your queries, as shown below.

Write the following query in the date calculation window and select OK as shown below.

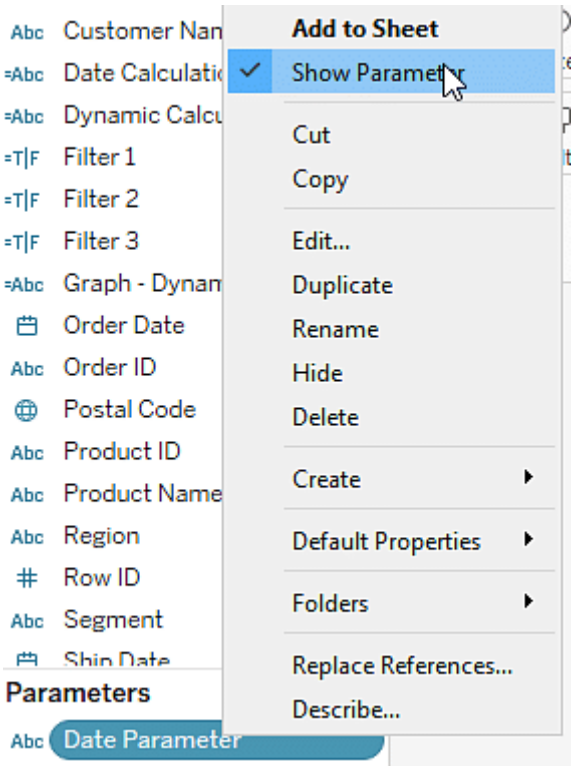


Now, you have the date parameter in the parameters section and date calculation in the dimensions section.

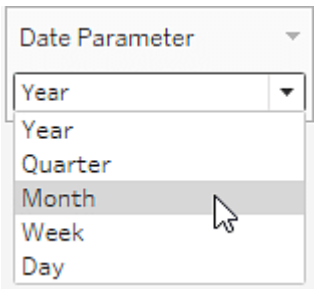
Next, change the text table to a bar graph. Select the 'Show me' tab and choose the option of a bar graph in the list.

The text table will not get converted into a bar graph.

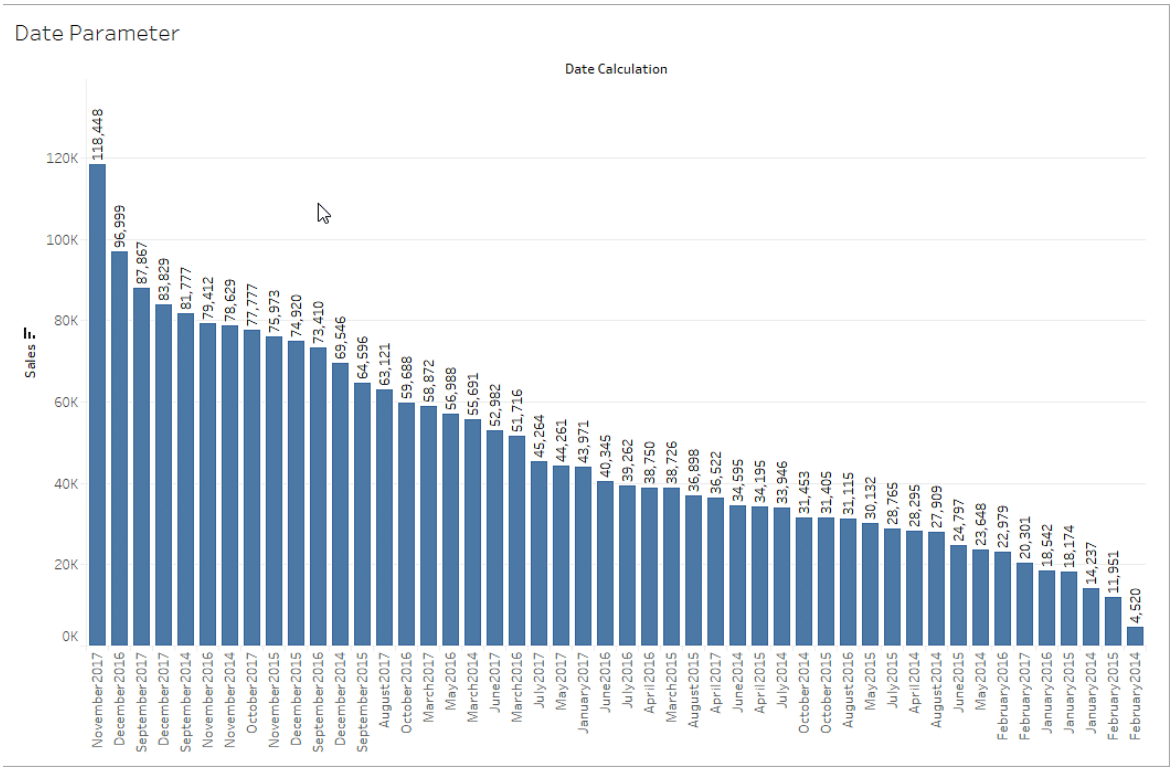
Select the marks sheet's label option to give you the details related to sales in a readable format.



The parameters option will be shown as follows.



Moving on, select the month option, and Tableau will provide you with sale per month.

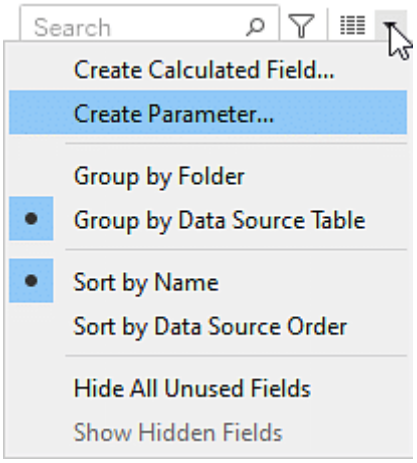


Dynamic Dimensions

To create the Dynamic Dimensions parameter, you need to execute the following steps.

Please create a new sheet and rename it as Dynamic Dimensions.

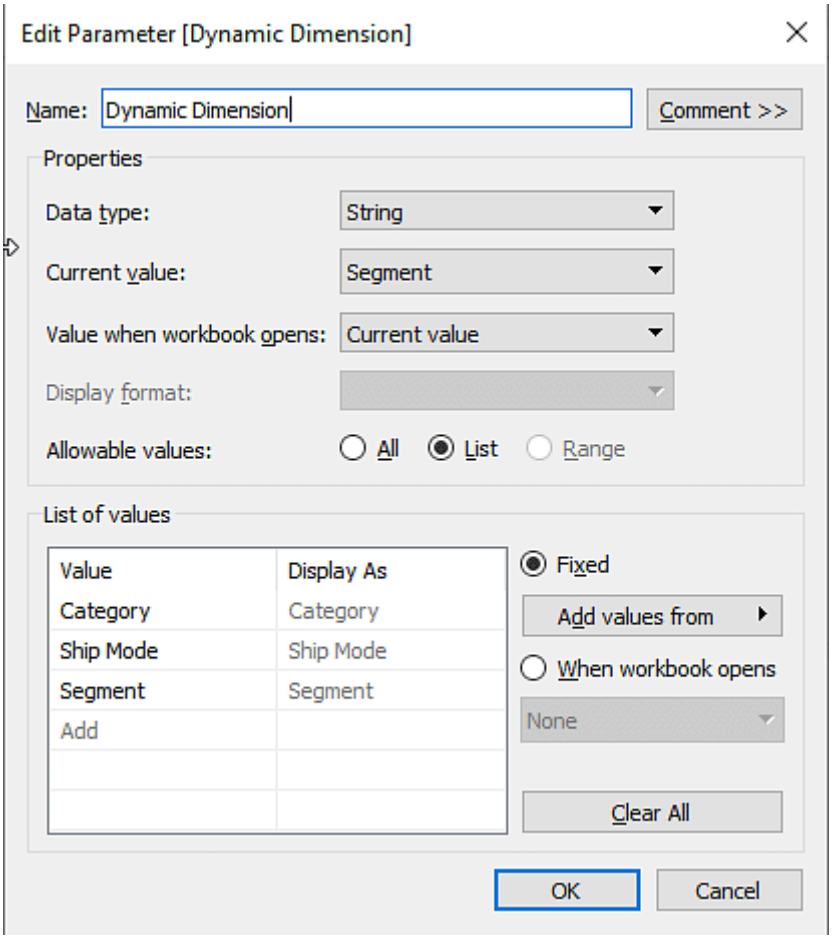
Create a new parameter. Select the arrow icon and click on the new parameter option.



Now, rename the parameter as Dynamic Dimension Parameter.

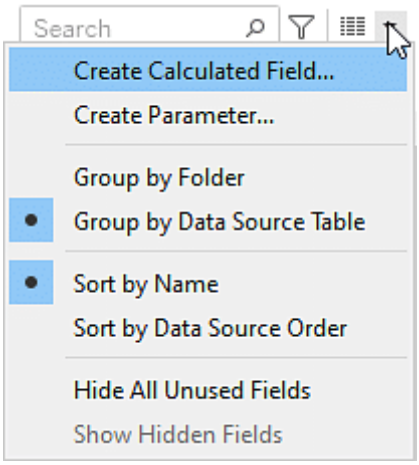
Select the datatype as a string.

Select category, ship-mode, and segment dimensions from the dimensions panel.



Select the OK option and now, create a new calculated field.

Hover over to the arrow icon and select "create a calculated field."



Write the following code in the calculated field window and select OK as below.

CASE [Dynamic Dimension]

WHEN "Category" THEN [Category]

WHEN "Ship Mode" THEN [Ship Mode]

WHEN "Segment" THEN [Segment]

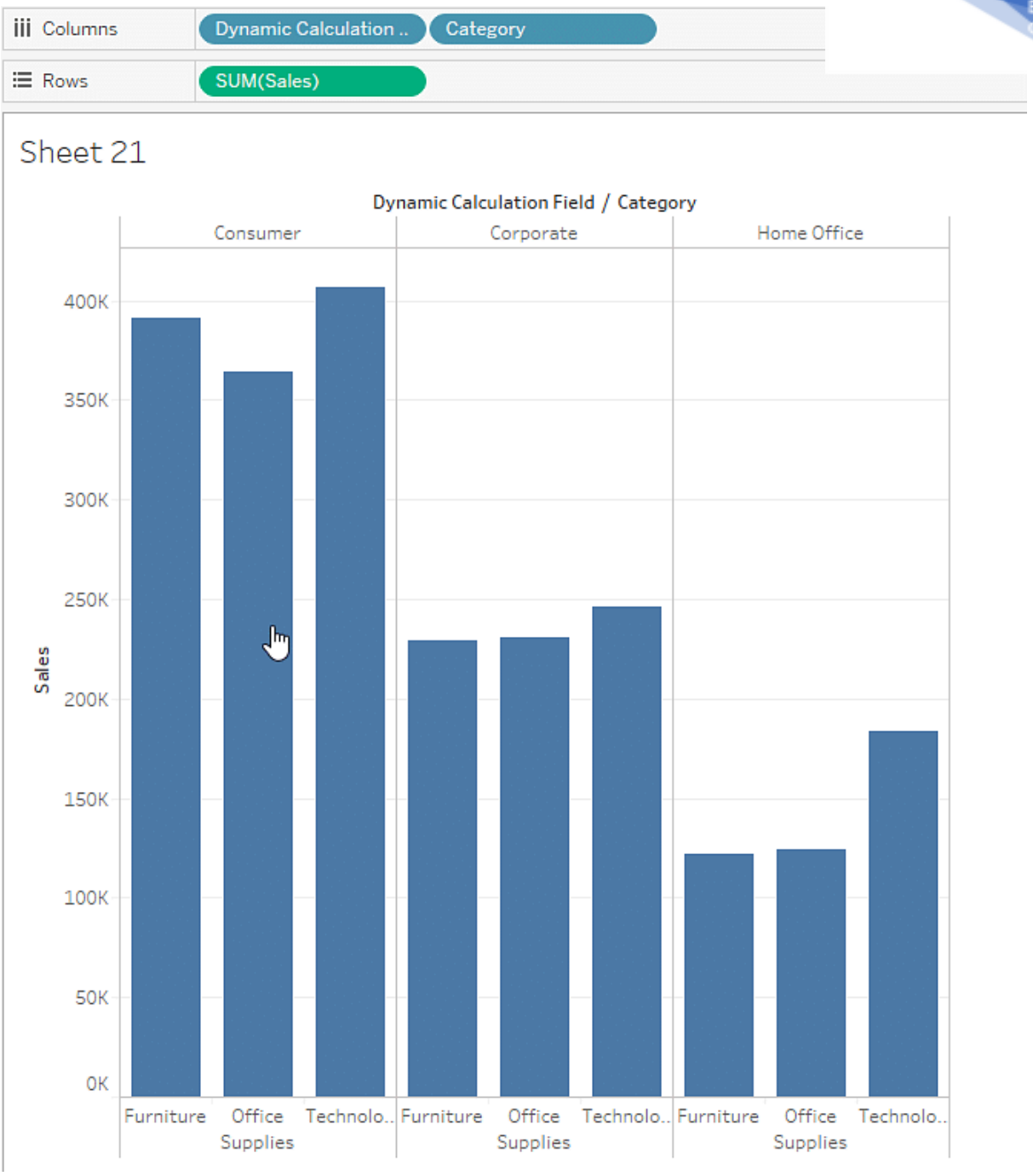
END

Now, you can find the newly created dynamic dimension parameter and dynamic dimension calculated fields in the left panel.

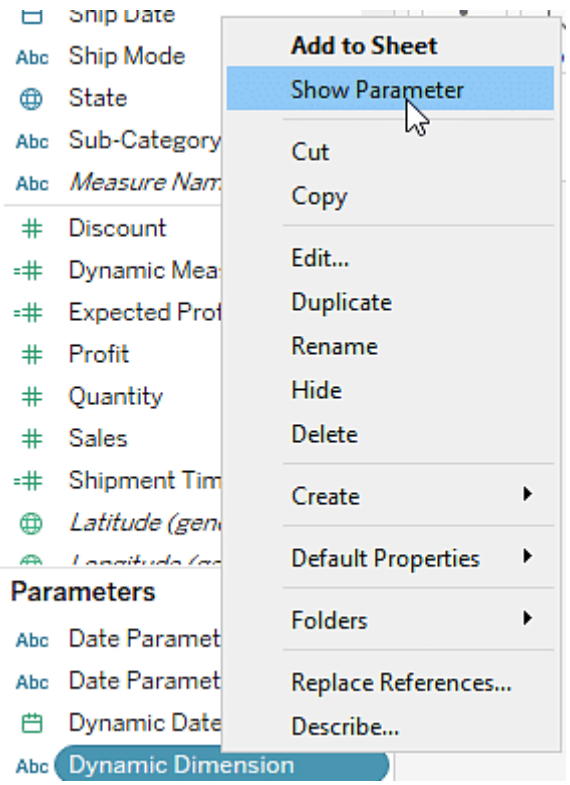
Drag the Dynamic Calculation field from dimensions panel to columns

Drag category to columns

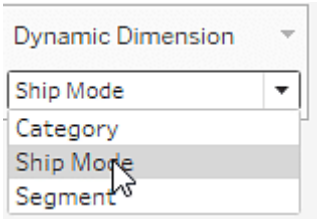
Drag sales to rows

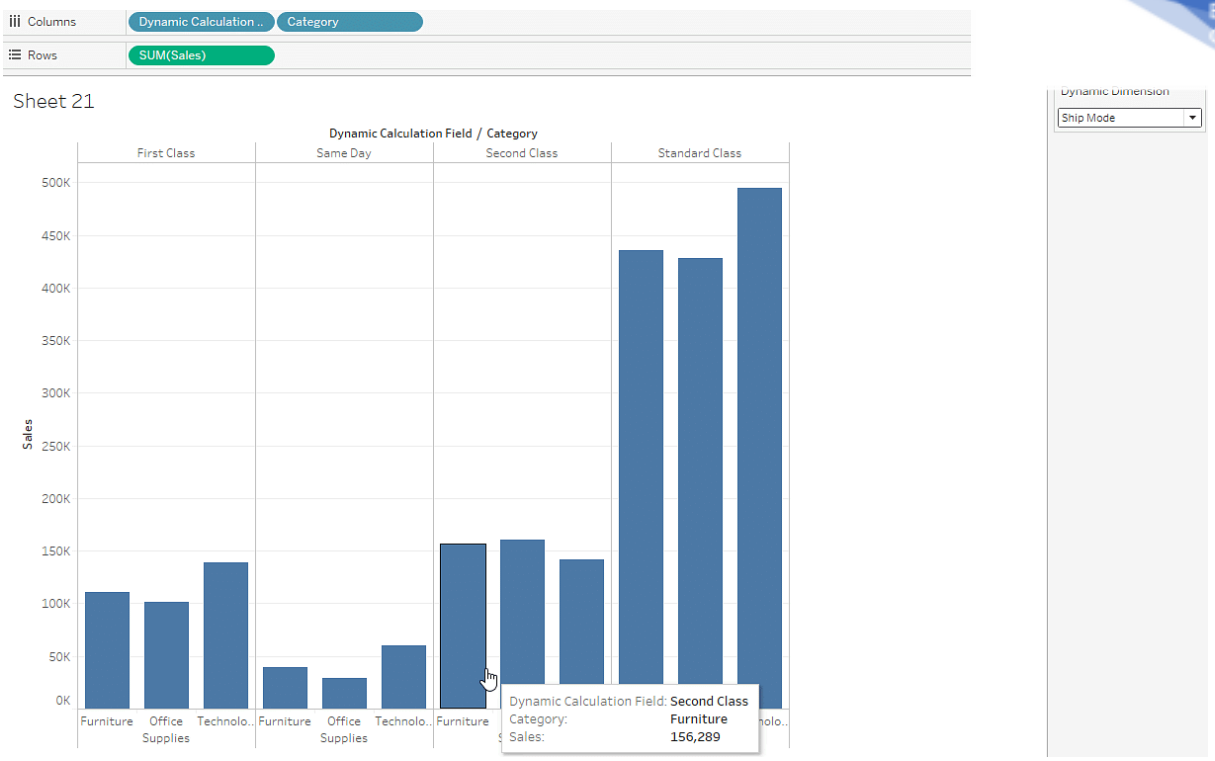


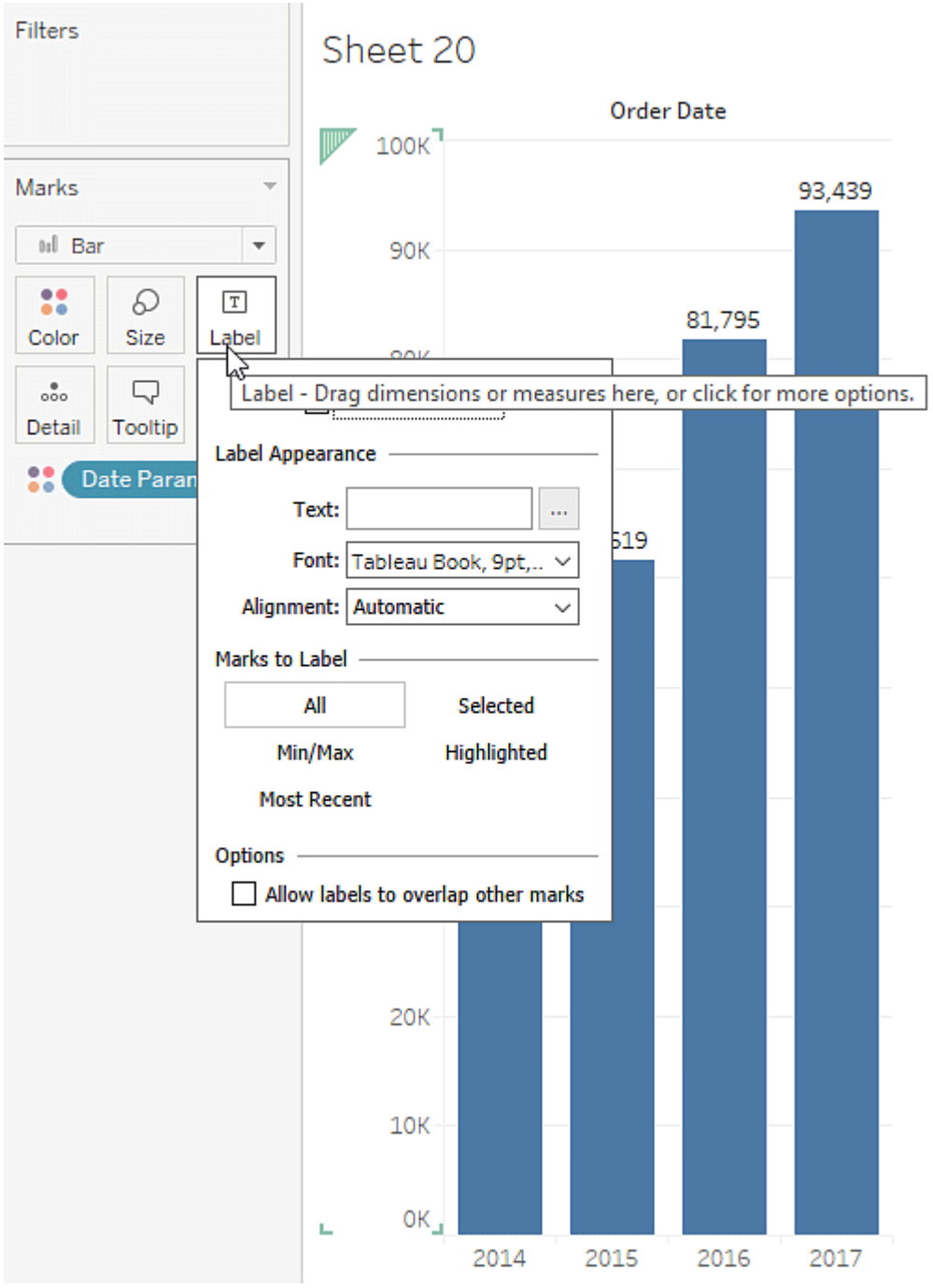
Now right click on the Dynamic Dimensions parameter pill and select the show parameter option.



Now, take a look at the Parameter menu. The output is now capable of visualizing dynamic dimension data based on dimensions like Category, Ship-mode, and Segment, as shown in the following output image.




Columns
Dynamic Calculation ..
Category
Rows
SUM(Sales)
Sheet 21
Dynamic Calculation Field / Category
First Class
Same Day
Second Class
Standard Class
Sales
500K
450K
400K
350K
300K
250K
200K
150K
100K
50K
0K
Furniture
Office Supplies
Technolo..
Dynamic Calculation Field: Second Class
Category: Furniture
Sales: 156,289
Dynamic Dimension
Ship Mode



To visualize the sales in terms of the year, quarter, month, week, or day, you need to use the Date Parameters. Select the Date Parameters option from the parameters panel and right-click the pill, and select the show parameter option as shown below.

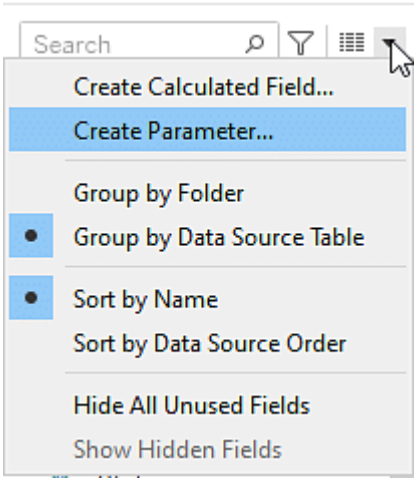
Dynamic Measures

Now, Dynamic Measures are completely similar to Dynamic Dimensions, with only one difference. Instead of Dimension values, we use Measure values.

To create a Dynamic Measures Parameter, you need to follow the steps below.

Please create a new sheet and rename it as Dynamic Measures Parameter for reference.

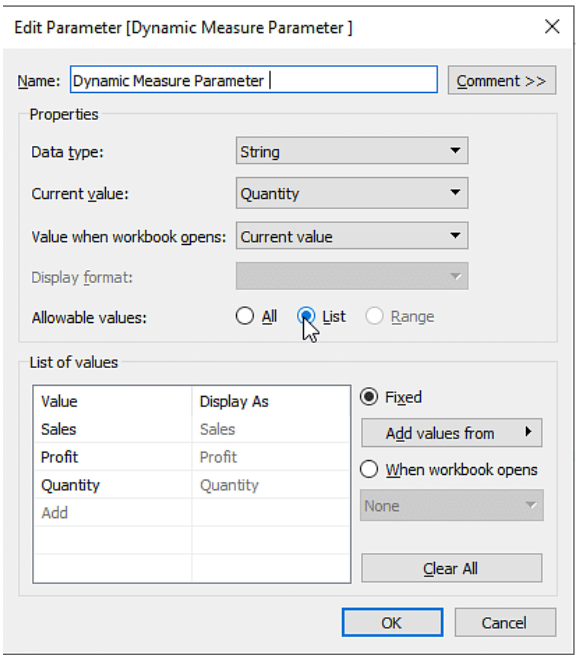
Create a new parameter. Select the arrow icon click on the new parameter option.



Next up, rename the parameter as Dynamic measures parameter.

Select the data type as a string.

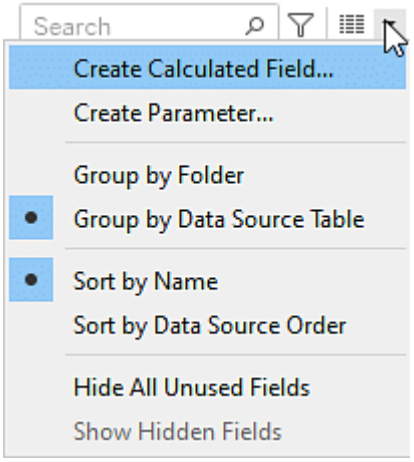
Moving on, select the list option and write the sales, profit, and quantity values as shown below.



Now you will find a newly created pill in the parameters panel.

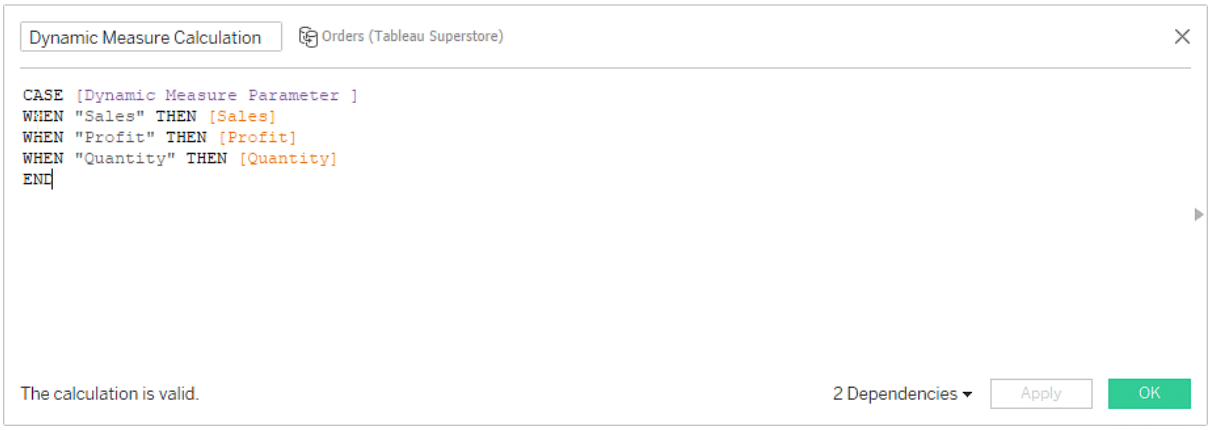
The next step is to create a new calculated field.

Select the arrow icon, click on the create new calculated field option.



A new calculated field window will appear on the screen.

Type the following formula and select OK to create a new calculated field.

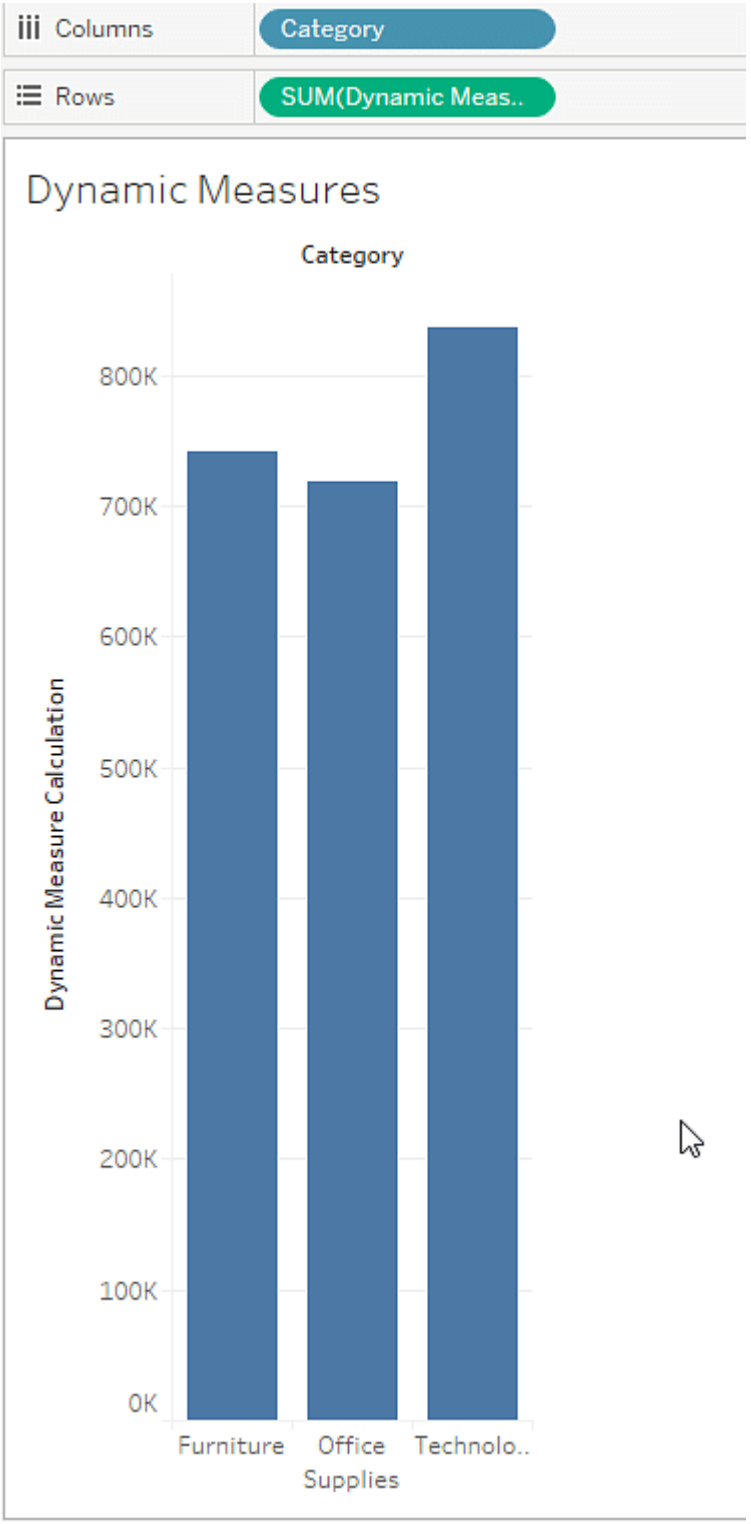


Now, you can find a newly created calculated field in the measures panel.

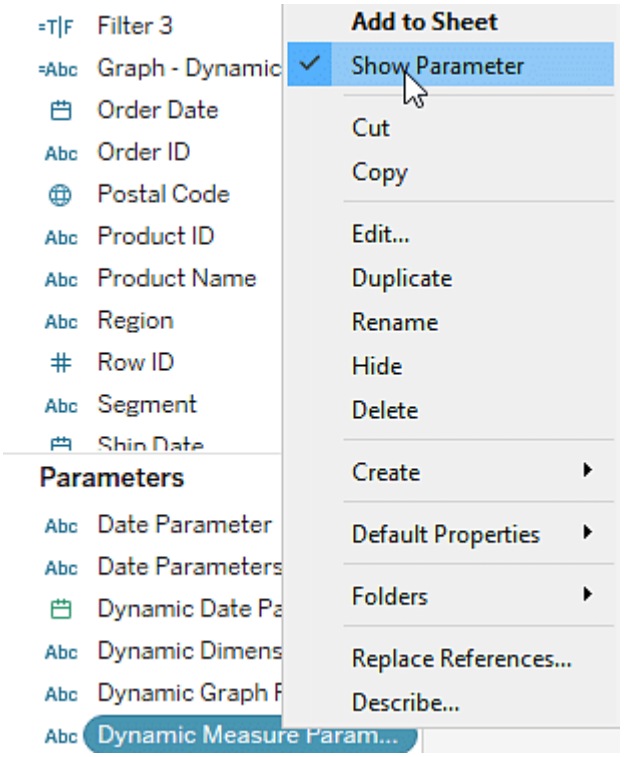
Drag the newly created calculated field in the measures panel to rows.

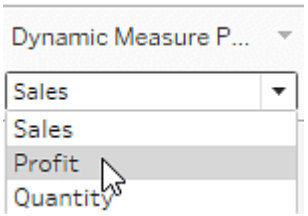
Drag category from dimensions to columns.

Tableau will automatically generate a bar chart.

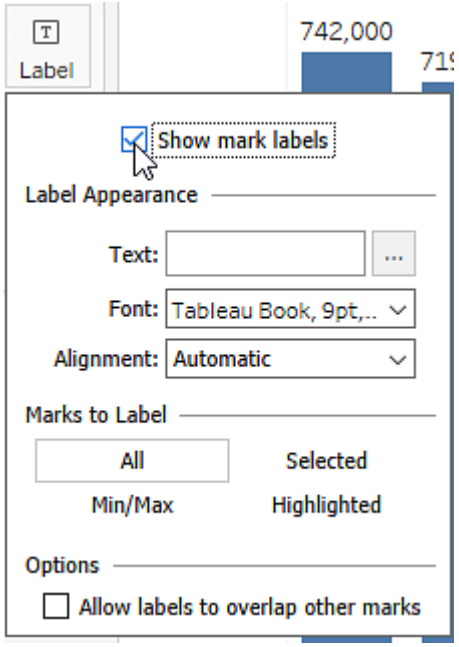


Now right-click on the newly created parameter and select the show parameter option.

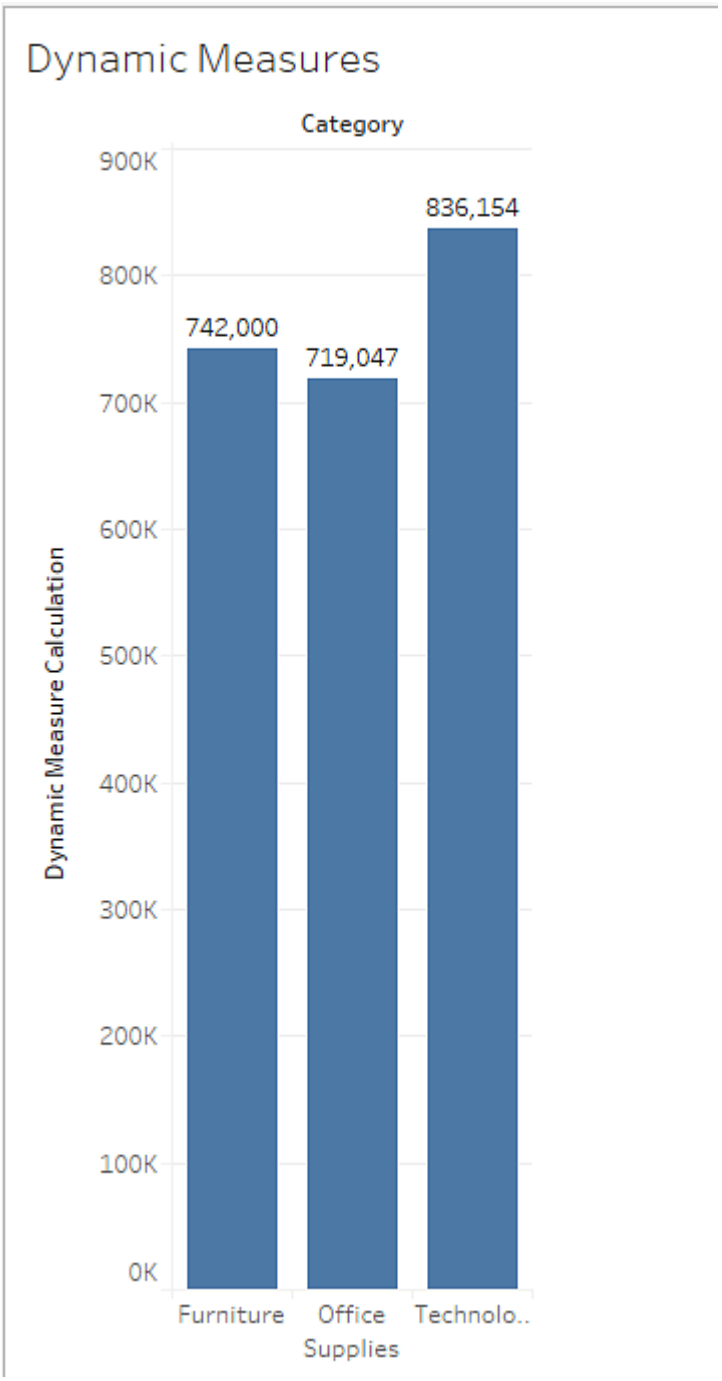




Now, select the label icon from the marks card and tick the label box.



The final result will look as follows.



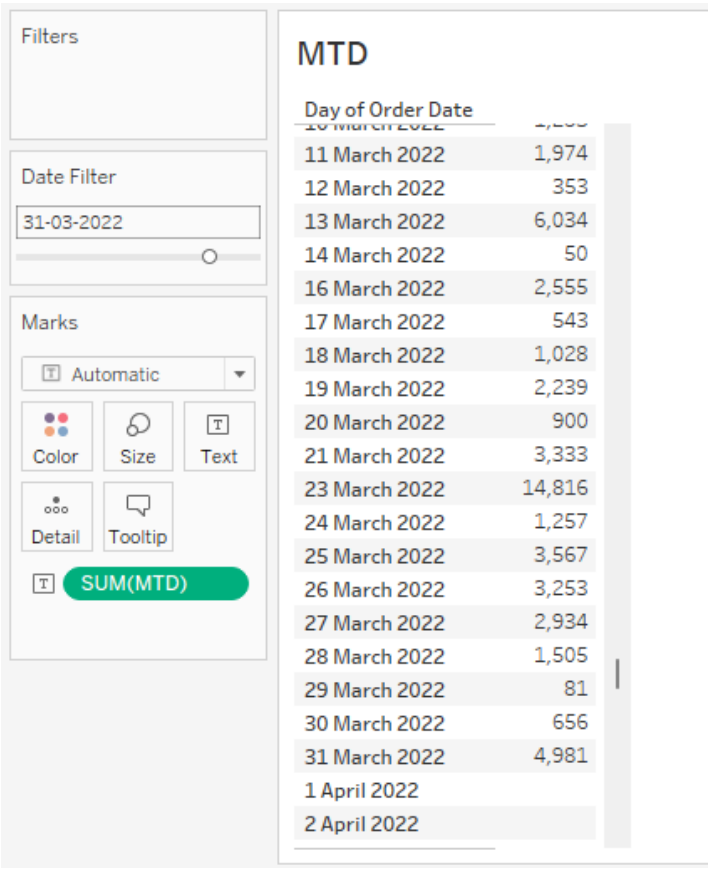
Difference between Parameters and Filters

Parameter	Filters
1. Parameters are preferred for Static Data	1. Filters are used for Dynamic Data
2. Parameters are implemented to the entire workbook	2. Filters can be implemented to specific sheets only
3. Multiple values cannot be selected while implementing Parameters	3. Multiple values can be selected while implementing Filters
4. Parameters are implemented using Calculated fields	4. Filters do not use calculated fields
5. Parameters support multiple datatypes	5. Fields cannot support multiple datatypes

Date Calculations

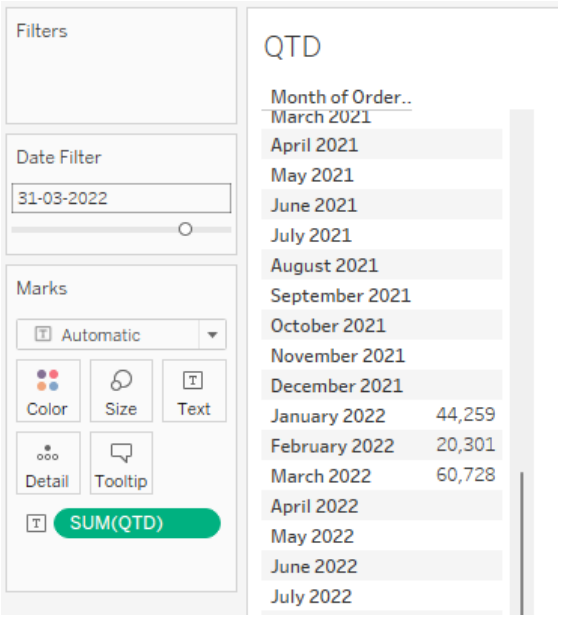
MTD: Month Selected in your date parameter till the selected date of the month.

```
IF DATETRUNC('month',[Order Date])=DATETRUNC('month',[Date Filter])
AND[Order Date]<=[Date Filter]
THEN [Sales]
END
```



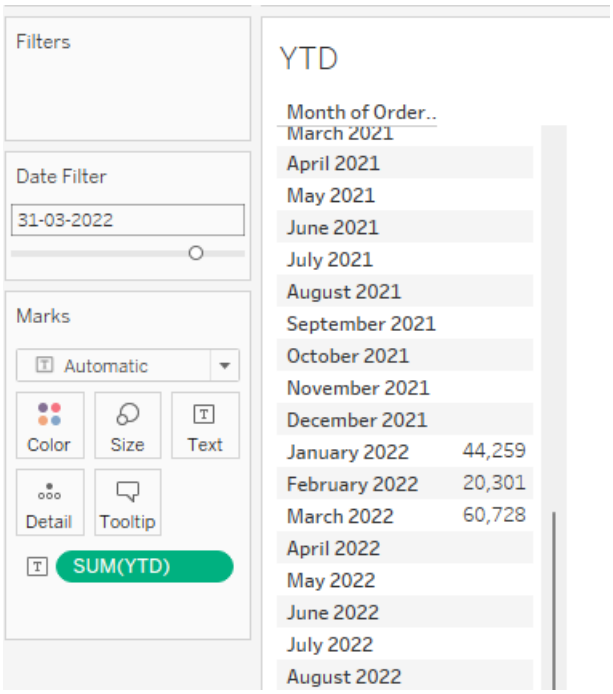
QTD: Quarter selected in your date parameter till the selected date of the month.

```
IF DATETRUNC('quarter',[Order Date])=DATETRUNC('quarter',[Date Filter])
AND [Order Date]<=[Date Filter]
THEN
[Sales]
END
```



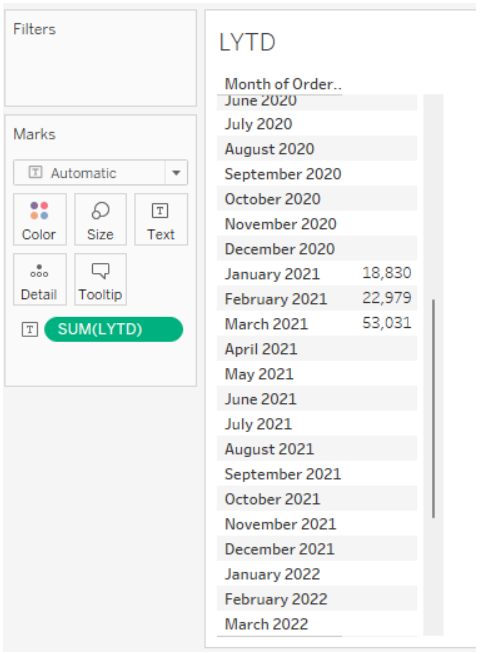
YTD: Year Selected in you date parameter till the selected date of the montl

```
IF DATETRUNC('year',[Order Date])=DATETRUNC('year',[Date Filter])
AND [Order Date]<=[Date Filter]
THEN
[Sales]
END
```



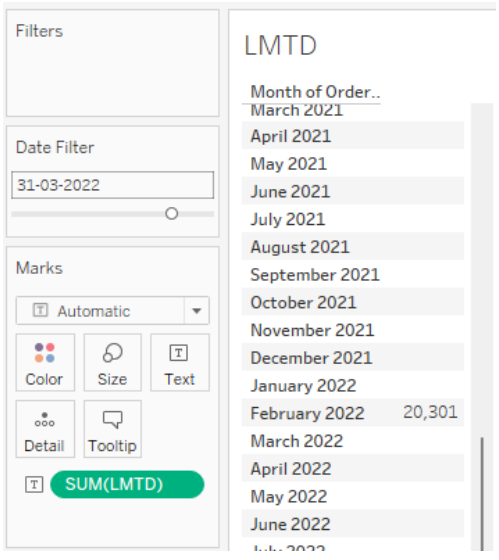
LYTD: Previous year of the year selected in your date parameter till the same day but for previous year.

```
IF DATETRUNC('year',[Order Date])=DATEADD('year',-1,DATETRUNC('year',[Date Filter]))
and [Order Date] <= DATEADD('year',-1,[Date Filter])
THEN [Sales]
END
```



LMTD: Previous month of month selected in your date parameter till the same day but for previous month.

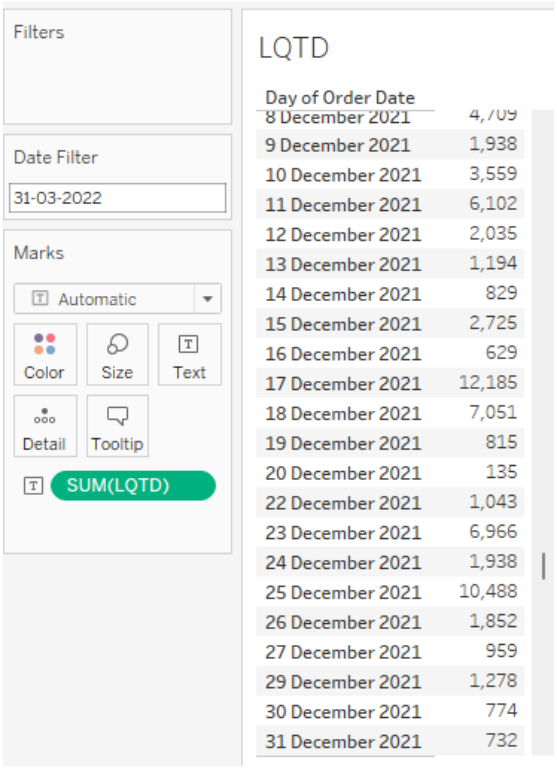
IF DATETRUNC('month',[Order Date])=DATEADD('month',-1,DATETRUNC('month',[Date Filter]))
and [Order Date] <= DATEADD('month',-1,[Date Filter])
THEN [Sales]
END



LQTD: Previous quarter of quarter selected in your date parameter till the same day but for previous quarter.

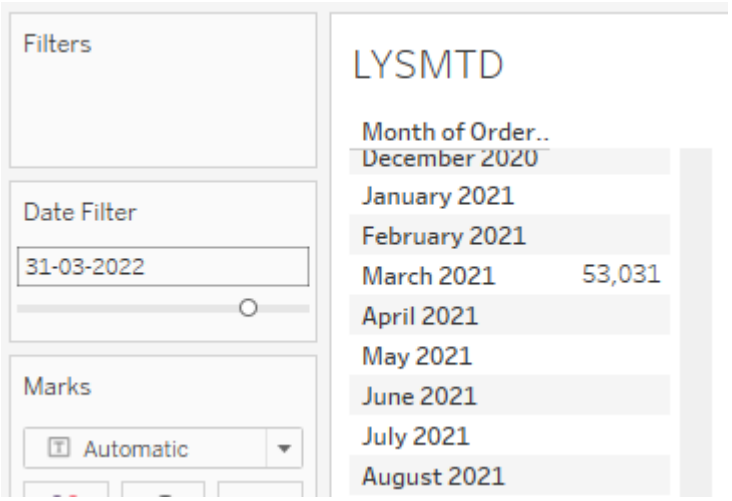
IF DATETRUNC('quarter',[Order Date])=DATEADD('quarter',-1,DATETRUNC('quarter',[Date Filter]))
and [Order Date] <= DATEADD('quarter',-1,[Date Filter])

THEN [Sales]
END



LYSMTD : Last year same month of month selected in your data parameter till the same day but for last year same month.

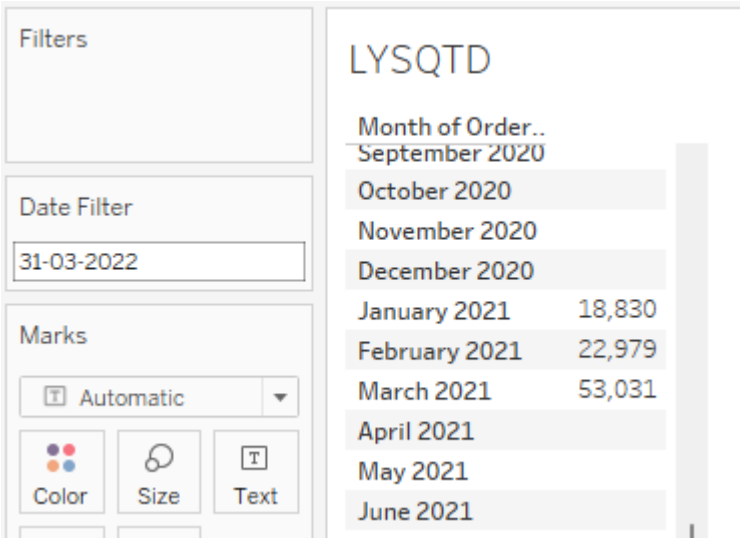
IF DATETRUNC('month',[Order Date]) = DATETRUNC('month',DATEADD('year',-1,[Date Filter]))
AND [Order Date] <= DATEADD('year',-1,[Date Filter])
THEN [Sales]
END



LYSQTD: Last year same quarter of quarter selected in your date parameter till the same but for last year same quarter.

IF DATETRUNC('month',[Order Date]) = DATETRUNC('month',DATEADD('year',-1,[Date Filter]))
AND [Order Date] <= DATEADD('year',-1,[Date Filter])

THEN [Sales]
END



Actions

Actions in Tableau are the activities performed whenever the user triggers an event. It adds more information to your charts and makes them interactive. In this blog, we'll discuss what filter actions in Tableau are, their types, and how to create these Tableau dashboard filters and use them to make your visualizations more interactive.

Filter: Use the data from one view to filter ta in another to help guide analysis.

Highlight: Call attention to marks of interest by colouring specific marks and dimming all others.

Go to URL: Create hyperlinks to external resources, such as a web page, file, or another Tableau worksheet.

Go to Sheet: Simplify navigation to other worksheets, dashboards, or stories.

Change Parameter: Let users change parameter values by directly interacting with marks on a viz.

Change Set Values: Let users change the values in a set by directly interacting with marks on a viz.

Shapes

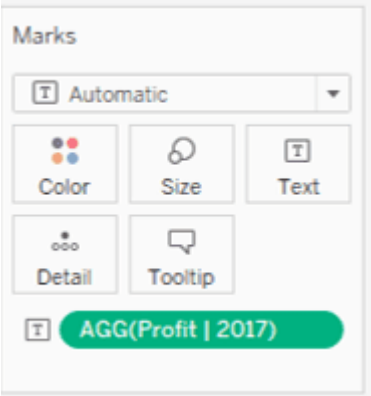
The three different ways I will dive into are:

- Default Shapes
- Custom Shapes
- Text Shapes

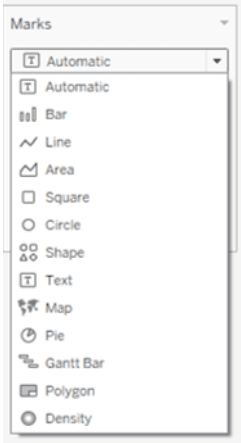
Default Shapes in Tableau

In the Marks card in Tableau, there are many different options given to developers on how they want data displayed. One of these is the Shapes option.

When you click down on the drop down in the marks card you will see different options. When you drag a field to the different areas of the Marks card, different results will show up in your worksheet because your default setting is Automatic. Tableau will determine what type of Mark is best for your field.

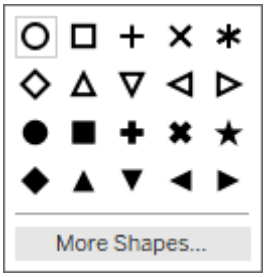


If you do not like the default selection or if you're going for a more specific representation of your data, you can select the drop down on your Marks card to choose what you want your data displayed as.



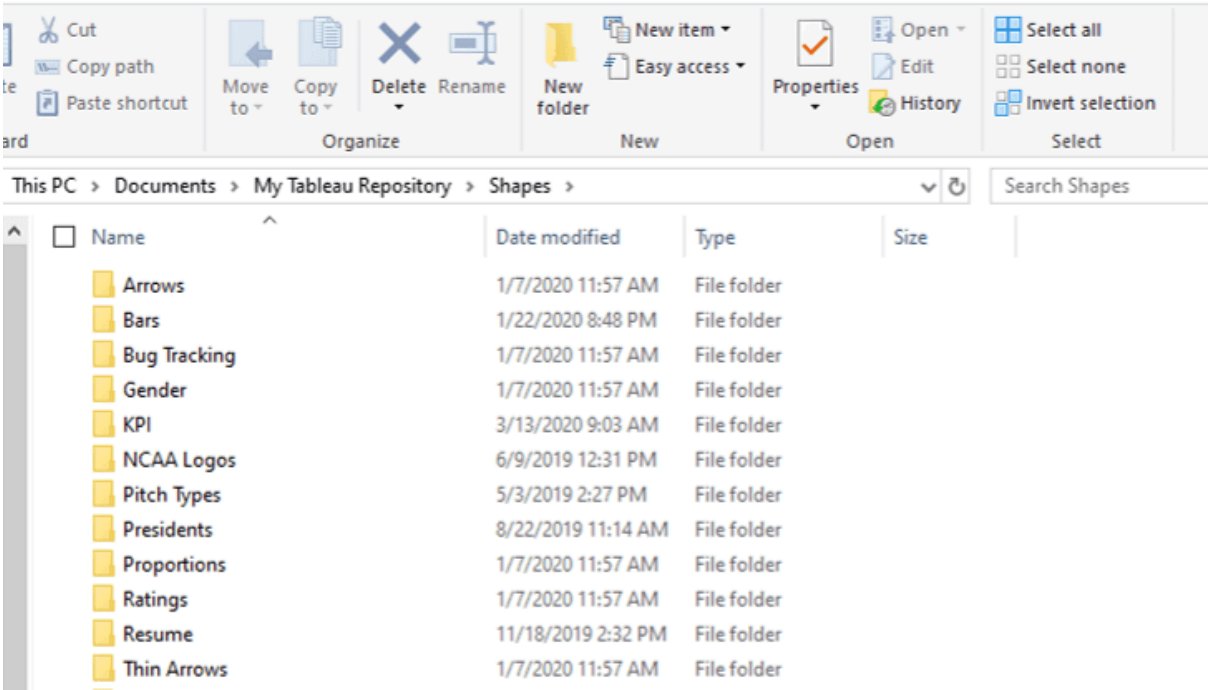
In the case above, I dragged an aggregate measure for Profits to the ma placed it on the Text mark. This gives me a sum total of profits in my worksh to click the drop down and select Circle, my sum total of profits would appear next to a blue circle. Changing the type in the Marks card will change the display of the data in your worksheet. If you choose the Shape option you will then be able to click on the Shape button and select what shape represents your data.

Once you've dragged your field onto the marks card, select Shape in the drop down. Now select whatever shape that you want to represent your data. If you drag a dimension to the shape Mark, then the different values within your Dimension will be represented with different shapes. You can see some of the default shapes that Tableau provides you below.

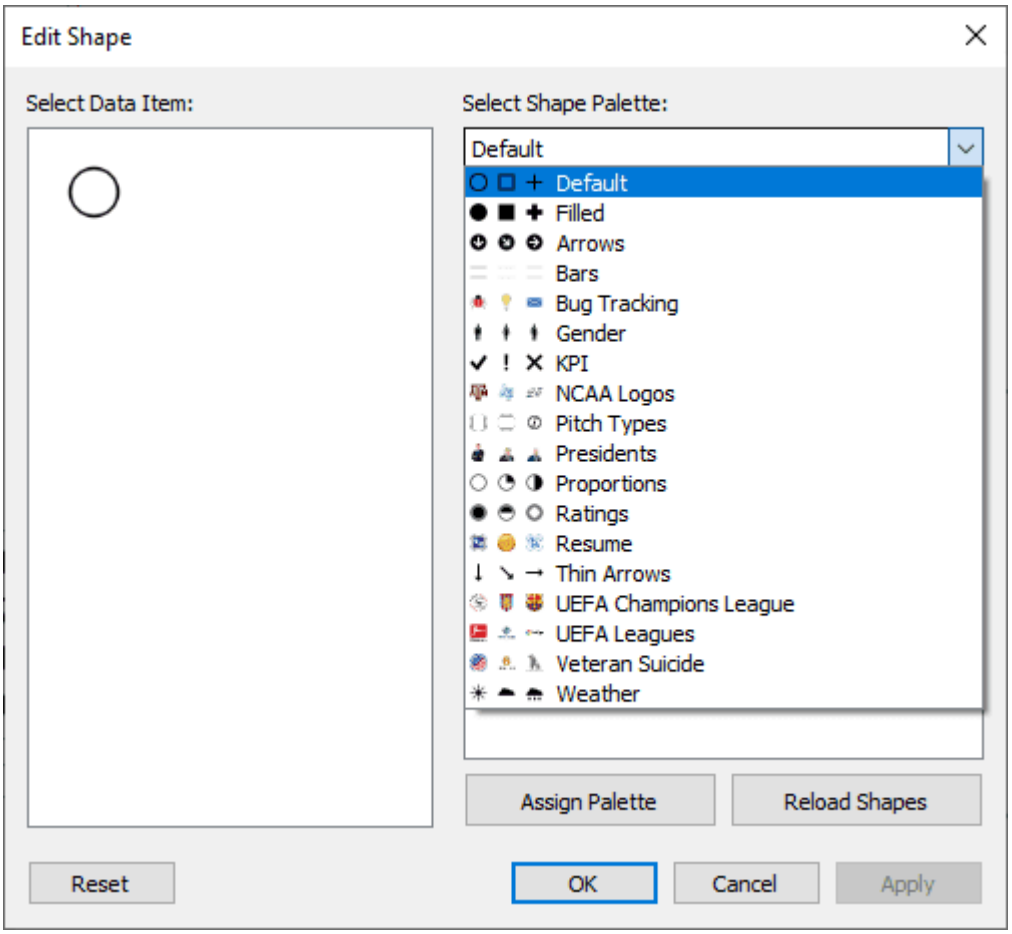


Custom Shapes in Tableau

Take a look through the default options in the More Shapes area, and you might find something you like. If you either want a different shape or have a specific shape in mind for your use case, then you are able to use custom shapes to represent your data in Tableau.



If you are not able to find a shape that you like, you can use google images t
desired image. Most of the time you will want to search via Advanced Searc
transparent image as that will render most appropriately in Tableau. Go ahead and save your desired image in the My Tableau Repository folder in Shapes. If you add a new folder under My Tableau Repository it will show up under the Shapes section of the Marks card.



When you upload a shape to your Shapes folder under My Tableau Repository, it may not show up immediately when you click on the Shapes mark. If you face this issue, click the Reload Shapes button. The new folder and shapes should appear in your options.

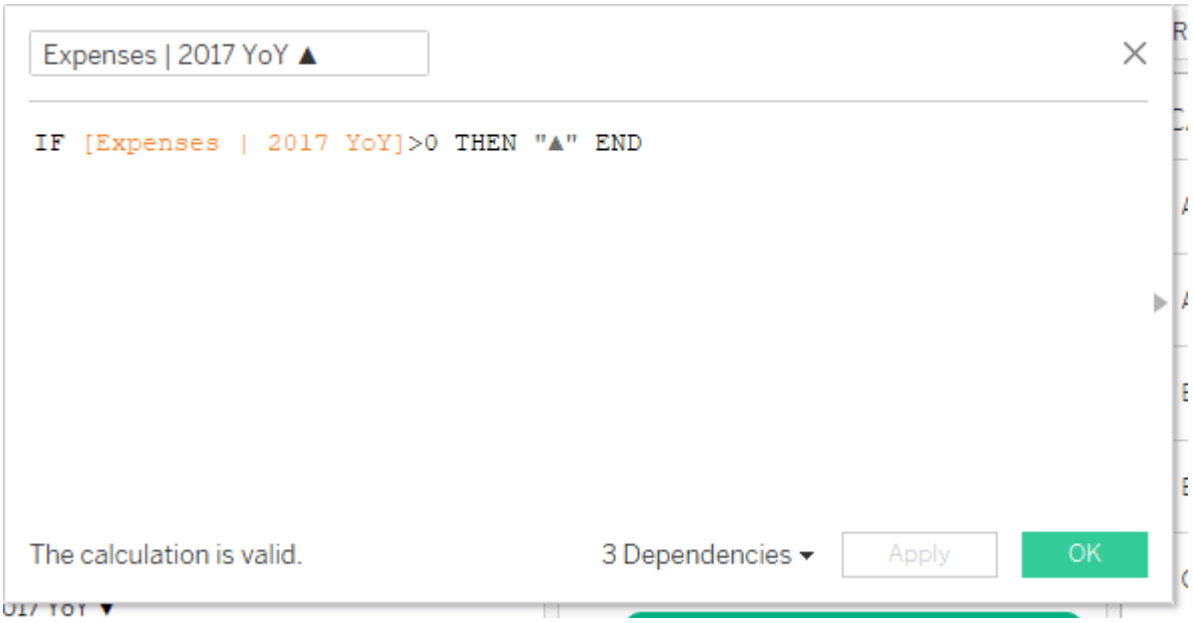
By clicking Assign Palette, your shapes will be assigned in whatever order they appear in the folder. If you want specific values of your dimension to have a certain shape, name the shapes in your folder after the dimension values and you should have a seamless transition when clicking Assign Palette.

Text Shapes

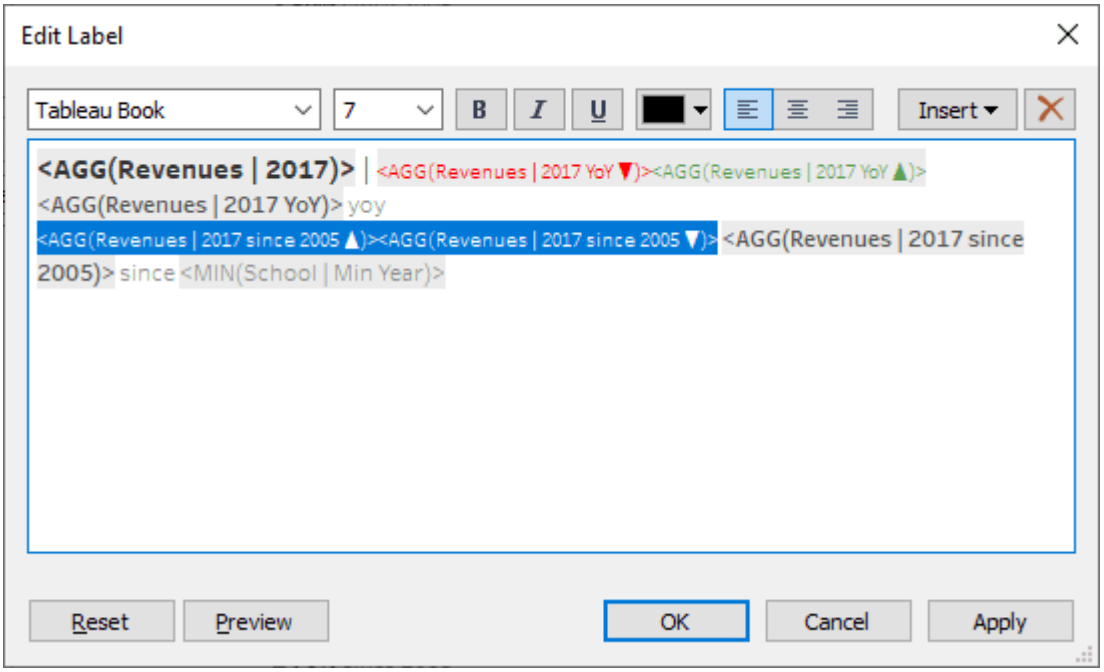
Both of the examples I have shown thus far involve the Marks card in Tableau. This next use of shapes will apply only to Text boxes, not the Marks card.

Say you want an arrow, in this example for a positive or negative change, bu
that shape to appear in either a tooltip, title, or text box. This is where we c
object to show shapes.

In the example below, you see that I have created a calculation that returns a txt object when it meets a certain criteria, in this case the YoY change for expenses being greater than 0. Because this shape is surrounded by quotes, it will appear as text. To get different shapes for your dashboard, you can visit sites like www.alt-codes.net to get different shapes.



As you see in the formula below, if a measure does not meet this criteria, then a NULL will be returned. This means that this field will not even show up as a space in the text field you've placed it in.

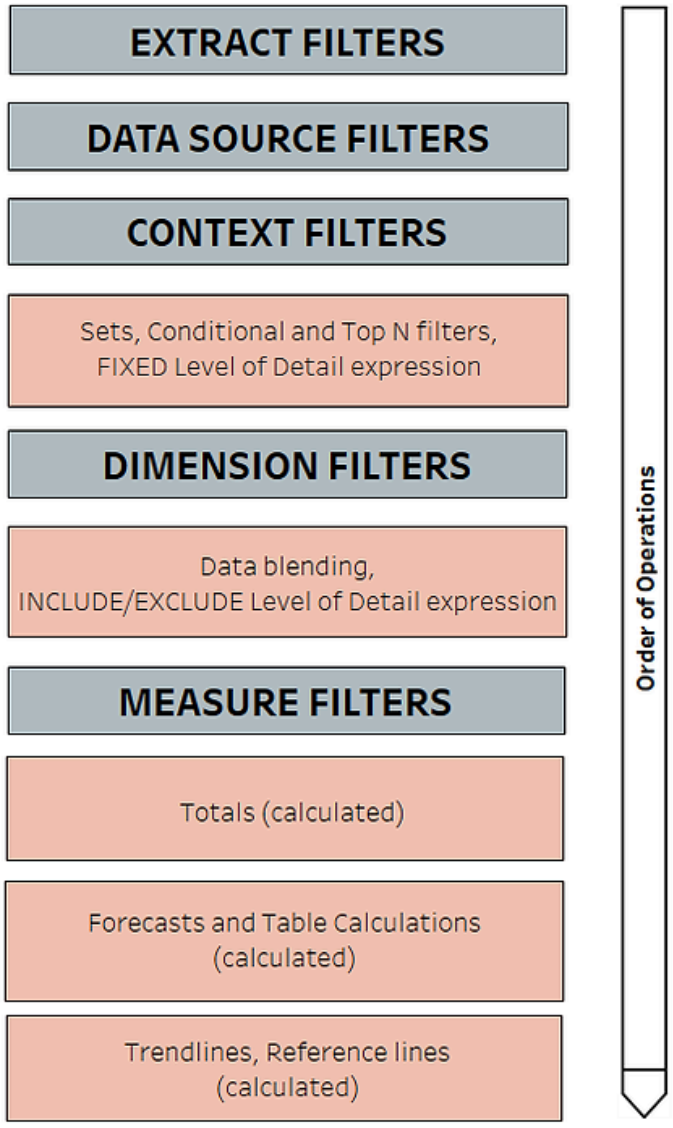


In the fields I've highlighted below, I've colored them different colors so tha
them show up, they are colored appropriately. In this example, if revenues h
then a down arrow text shape will appear and that field has been colored red. If the revenues have increased, then a green up arrow text shape will appear. The field that does not meet the criteria will be a NULL and now show up at all. This is how you can use text shapes to indicate KPIs in Tableau.

Order of Executions

Tableau's order of operation is the order in which Tableau performs actions. Anytime you build a view and add filters, those filters are executed in order established by the order of operations.

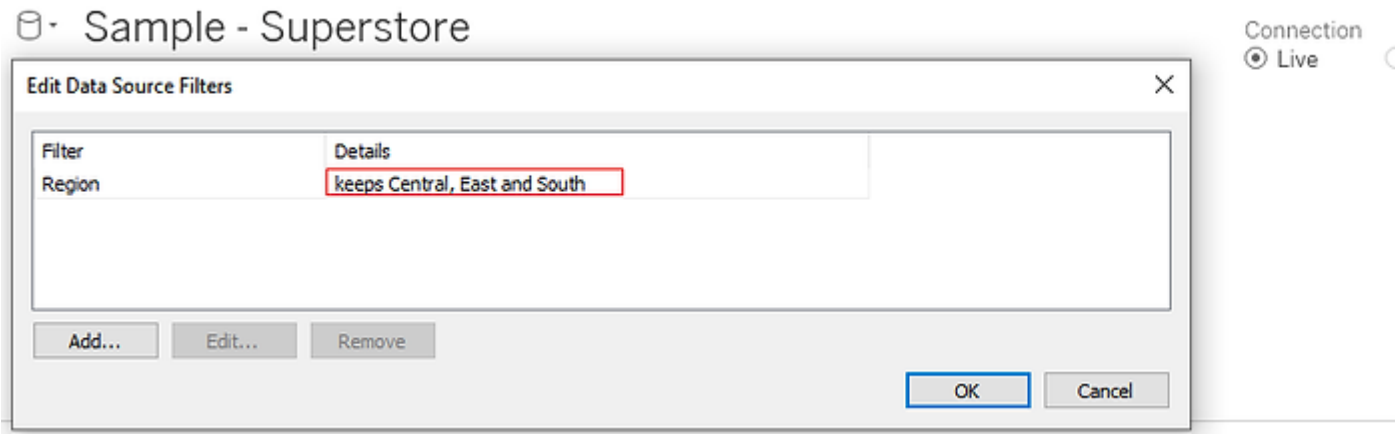
This order of operation is summarized in the figure below;



Extract/Data Source Filters

Both extract and data source filters are the first actions in Tableau's order of operations and are applied at the Tableau data source page. They are useful in restricting the data users can see when you publish a workbook or data source. A powerful way to omit fields not required in your analysis.

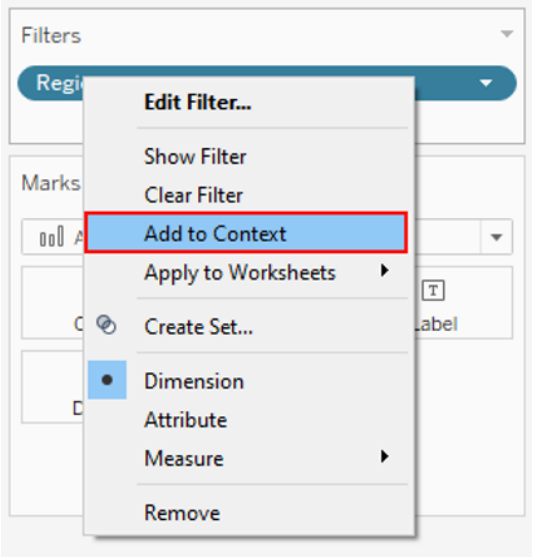
In the example below, if a data source filter is applied to exclude West region (i.e. by keeping the East, South and Central region). Then in all the analysis down the line West region will not be part of it.



(Note: To add a data source filter, while at the Tableau data source page choose 'Add' >>Add…>>Select field)

Context Filters

Context filter is the third filter in Tableau's order of operation, this filter is like a dimension filter only that it has been applied to the context by right clicking the filter and choosing 'Add to Context'.



By enabling the context filter the following happen;

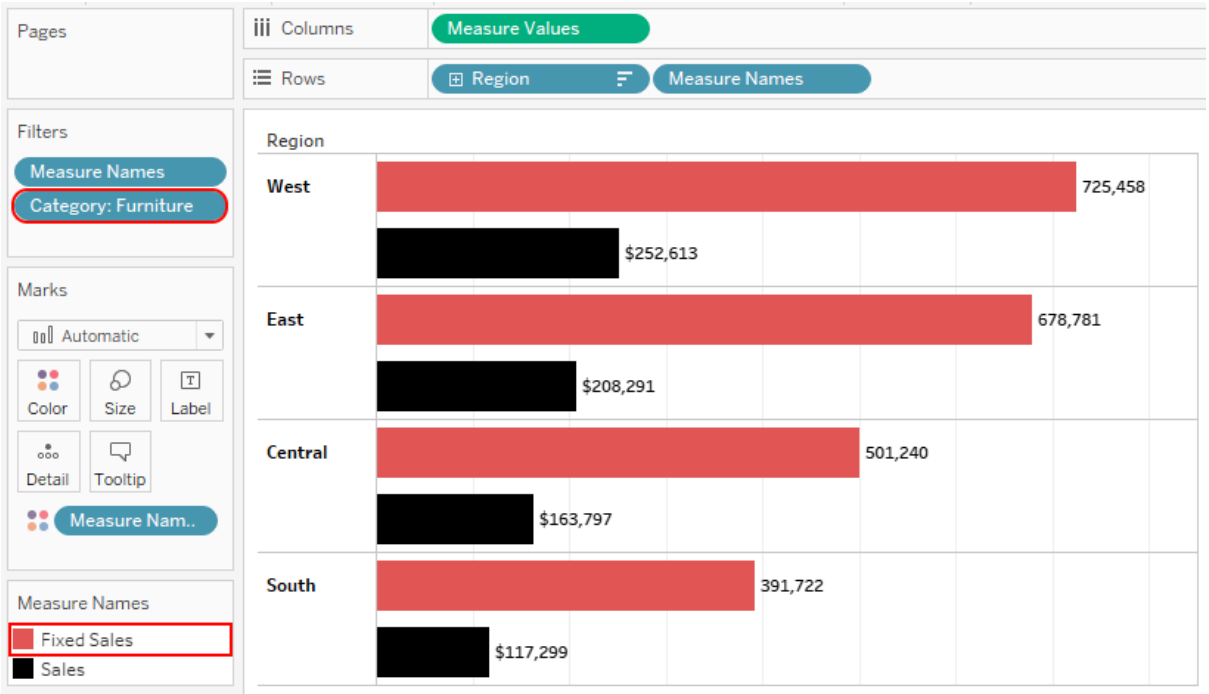
All other filters added to your view became dependent filters to the context filter – meaning they process only data that goes through the context filter.

The filter pill initially blue turns-out to grey color (indicating a context filter).

Sets, Conditional and Top N filters, FIXED Level of Detail Expressions

Next in Tableau's order of operations is sets, conditional and Top N filters and FIXED LOD expressions. All of these are computed right before dimension filters. This means level of detail expressions can be used to show whole numbers that would normally be filtered out by dimension filters which is useful when doing comparisons – computing proportion of total.

In the example below the Fixed Sales shows the total sales for all regions regardless of the dimension filter present while Sales shows the Sales for only product category 'Furniture' for all regions. Therefore, in this example, the filter 'Furniture' (dimension filter) can't filter 'Fixed Sales' since this calculation is computed before the dimension filter 'Furniture' is applied.



However, if you promote the dimension filter 'Furniture' to a context filter, then you can filter 'Fixed Sales'

This simple technique is useful when computing the proportion of total.

Dimension filters

Dimension filters are the regular filters which reduce categorical variables down. These filters can be added in Tableau's view by right clicking the dimension of interest and selecting 'Show Filter' or simply by dragging the dimension to the Filter shelf.

Data blending

Data blending takes place after dimension filters, meaning only fields that are not filtered out will be blended across the data sets.

INCLUDE/EXCLUDE LOD Expression

Include and exclude level of detail expressions are similar to Fixed LOD in the sense that they all influence the level of detail being considered in the view. Include LOD tell the calculation which fields to include in the level of detail while exclude LOD does the opposite.

Using the previous example on Fixed LOD but replacing the Fixed calculation with Include Region level of detail results in the Sales values being equal to 'Include Sales' value. This is because unlike in the case of Fixed calculation the include calculation is evaluated after the dimension filter 'Furniture' has been applied. Therefore, context filter is not required to affect the include and exclude filters.

Measure filters

Measure filters are the filters that influence continuous/numeric fields by setting the minimum and maximum values. They occur after level of detail calculations.

Measure filters can be added to Tableau's view by right clicking the measure field of interest and selecting 'Show Filter' or by dragging the measure field to the filters shelf.

Totals, Forecasts and Table Calculations, Trendlines and Reference Lines

Next in the Tableau's order of operations are totals, followed by forecasts and table calculation then trendlines and reference lines all which are computed within the view.

Axis in Tableau

Basically there are three types of axis in Tableau

1) Individual Axis
2) Blended Axis
3) Dual Axis

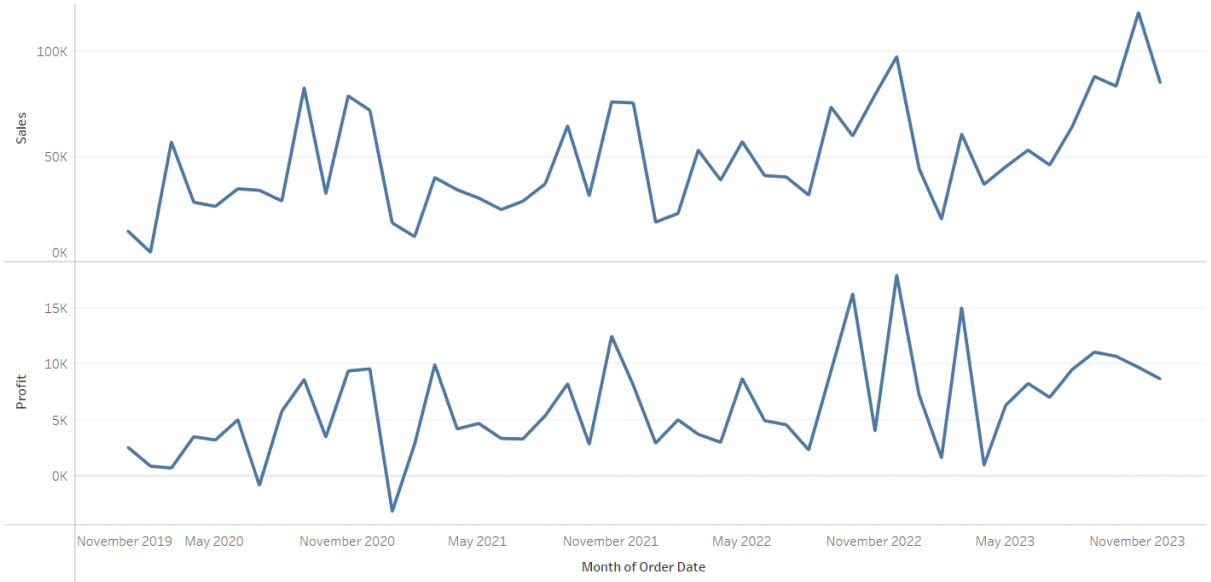
Individual Axis

In Tableau, an individual axis refers to the use of separate and independent vertical or horizontal axes for different measures within the same chart. It allows you to compare and display multiple measures with different scales or units of measurement on the same visualization. Each measure is represented by its own axis, and the values for each measure are scaled independently. This can be useful when the measures being compared have significantly different magnitudes, such as sales revenue and profit margin. An Individual Axis in Tableau is obtained by adding measure(s) into the Rows or Columns Shelf.

Steps to Create Individual Axis

1) Suppose we are using Superstore dataset
2) We can add Months in the column shelf.
3) We can add sales in row shelf and profit again in row shelf.

You will observe that there two visuals are created along side which has two individual axis this is called as individual axis.



Blended Axis:

Blended axis, also known as a dual-axis chart, is a technique in Tableau that enables you to combine multiple measures or fields into a single chart by sharing a common axis. Unlike individual axes, where each measure has its own independent axis, blended axes allow you to overlay measures on a shared axis, typically using different chart types. One measure is plotted on the primary axis, while the other measure is plotted on the secondary axis, with the scales for each measure adjusted accordingly. Blended axis

charts are useful when you want to show the relationship or correlation measures, such as sales revenue and number of customers over time. The in Tableau is beneficial for comparing two measured values against the same axis.

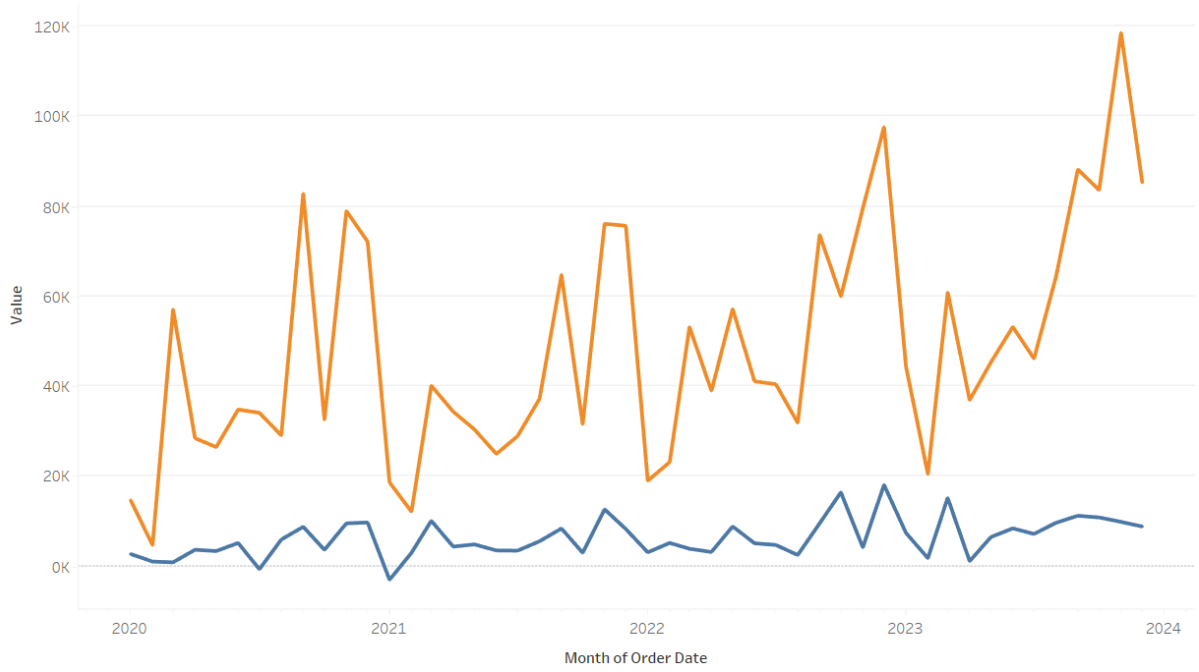
Steps to create Blended Axis

METHOD 1:

1) Drag and drop Months in the columns shelf.
2) Add Sales in row also add profit in the shelf
3) Drag and drop profit from rows into the axis of sales and your blended axis is ready.

METHOD 2:

1) Step 1 is same as above
2) You can add measure names in filters
3) Add measure values in rows and your blended axis is ready.



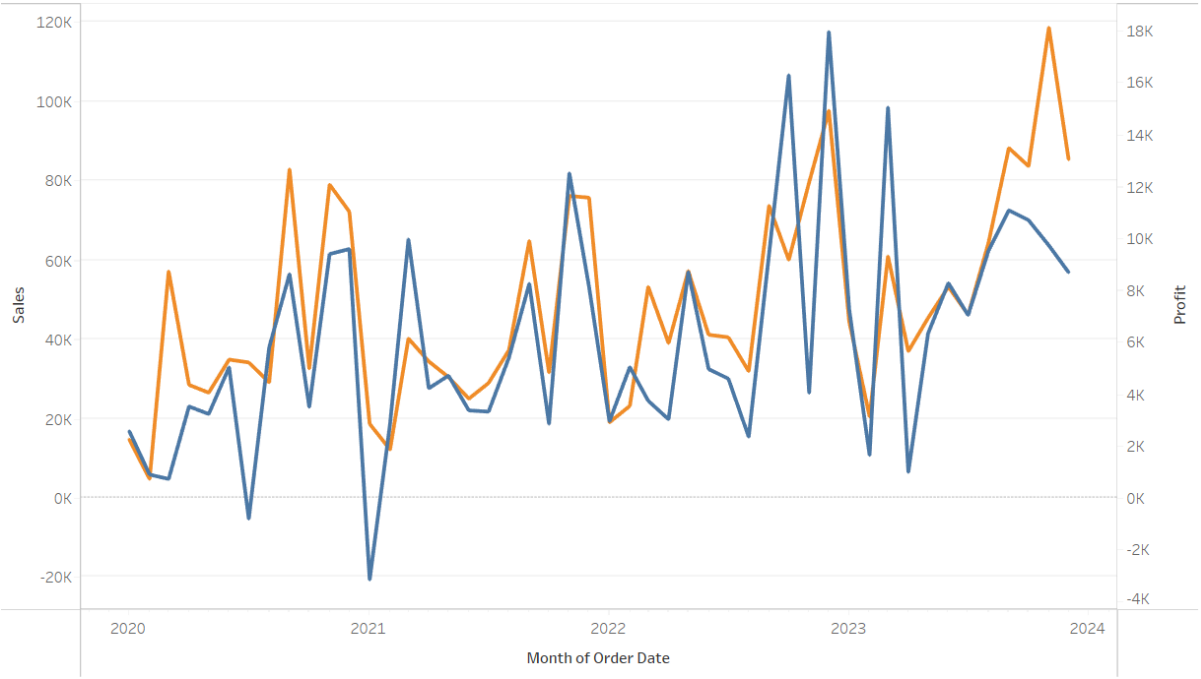
Dual Axis:

Dual axis is another way to visualize multiple measures in Tableau, but it differs from blended axis in how the scales are managed. In a dual axis chart, both measures are displayed on separate but aligned axes, allowing you to compare their values side by side. The primary and secondary axes share the same scale, making it easier to compare

the magnitudes of the measures. Dual axis charts are beneficial when compare measures with similar units or scales, such as temperature a readings.

Steps to create Dual Axis in Tableau

1) Do the similar things as in Blended Axis Step 1 and step 2
2) Click on Profit and select dual axis and dual axis chart is created.



To summarize, individual axis, blended axis, and dual axis are techniques in Tableau for displaying multiple measures or fields on a single chart. Individual axis uses separate and independent axes for each measure, blended axis shares a common axis with different chart types, and dual axis aligns two measures on separate but aligned axes with the same scale. The choice of which technique to use depends on the data and the type of analysis you want to convey effectively.

Different Types of Filters

Filters are a fundamental feature used to control the data that is displayed in visualizations. Filters allow you to narrow down the data based on specific criteria, helping you focus on the relevant information for your analysis.

Extract Filter: This filter is applied at the data source level when creating allows you to reduce the amount of data being extracted by specifying criteria that determine which data should be included.

Data Source Filter: Similar to the extract filter, the data source filter also operates at the data source level. It defines the filtering criteria that apply to the entire data source, limiting the data loaded into Tableau. This filter is useful when you want to exclude certain data from the analysis.

Context Filter: A context filter is used to create a temporary subset of data for analysis. When you apply a context filter, Tableau creates a separate query that only includes the data relevant to the filter. This can help improve performance when working with large datasets.

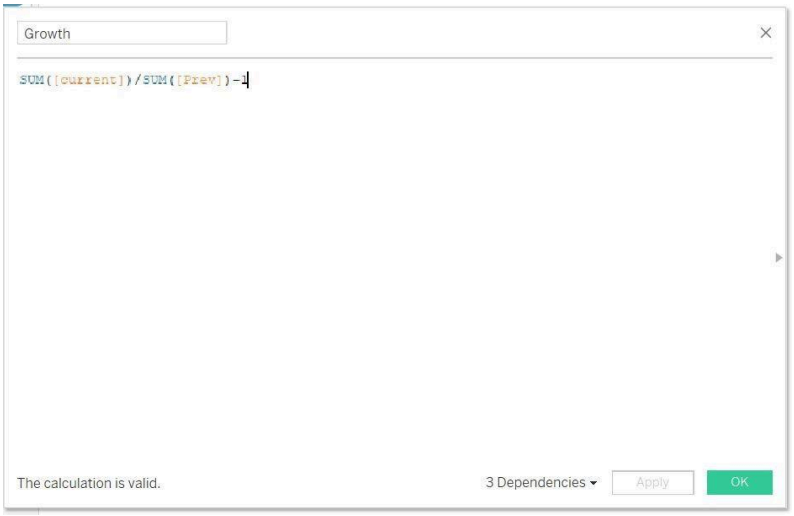
Dimension Filter: A dimension filter allows you to filter the data based on values within a specific dimension. It enables you to include or exclude specific dimension members, limiting the data displayed in your visualizations.

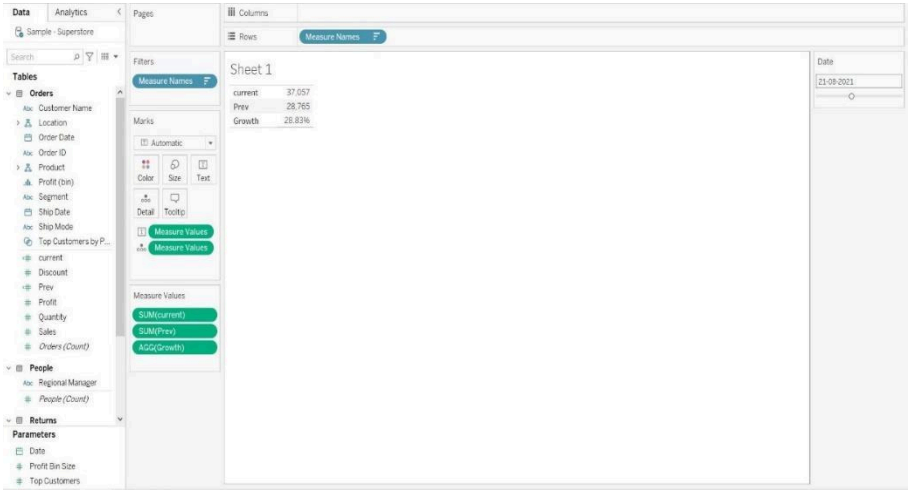
Measure Filter: A measure filter is used to filter the data based on the values of a measure. It allows you to set conditions or ranges for a measure, such as filtering sales above a certain threshold or within a specific date range.

Table Calculation Filter: Table calculation filters apply filtering based on the result of a table calculation. For example, you can filter data based on the top N values, percentiles, or ranking of a calculated field.

Conditional formatting

Profit Indication: The term "profit indication" refers to a measure or calculation that represents the profitability of a business or a specific aspect of its operations. It is typically derived from financial data, such as revenue, costs, and expenses.

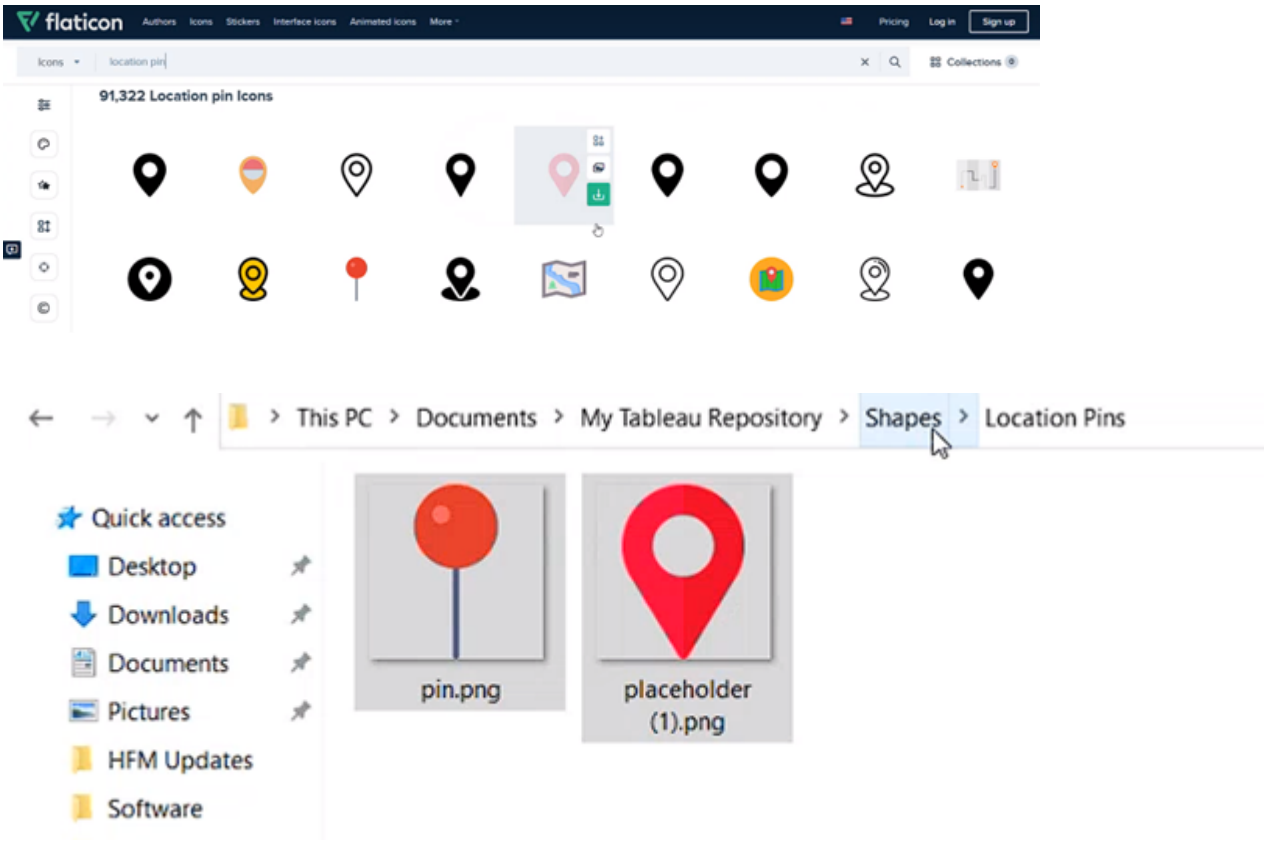


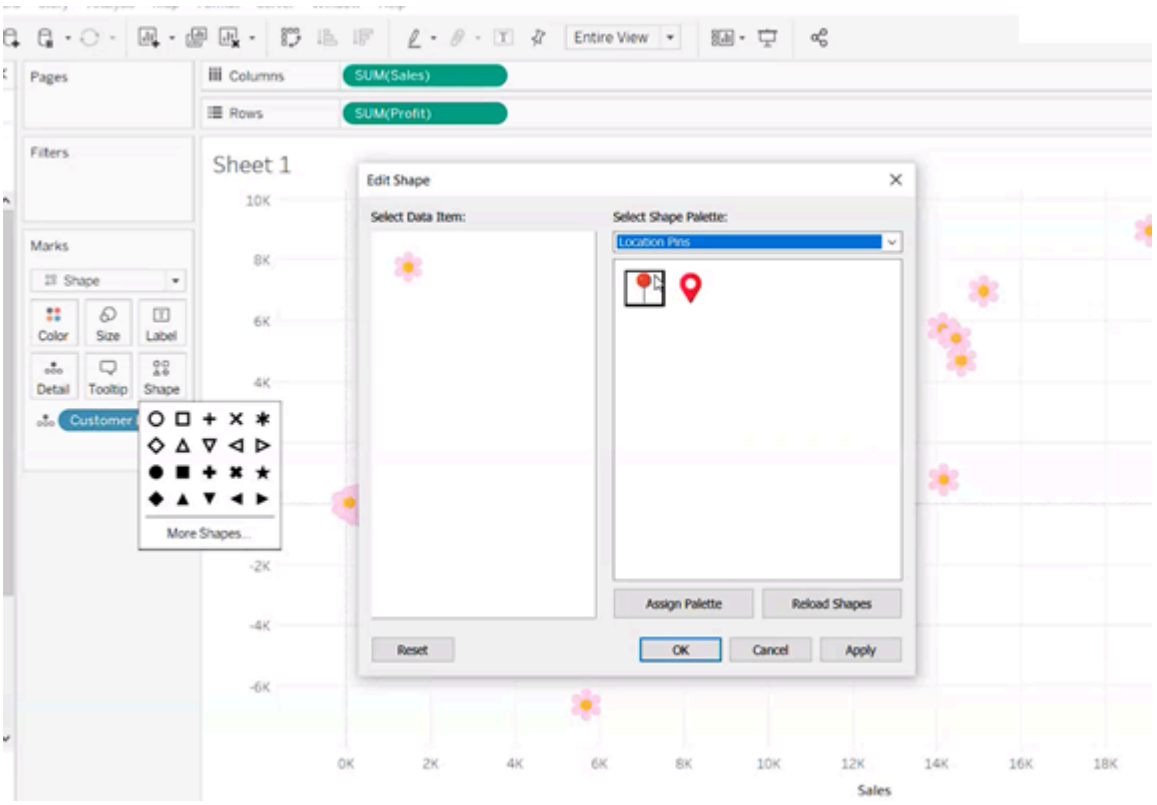


How to ADD Shapes?

1. To add shapes in Tableau, you can follow these steps:
2. Open your Tableau worksheet or dashboard.
3. In the "Marks" card, select the mark type that you want to add shapes to. For example, if you want to add shapes to a scatter plot, select the "Circle" mark type.
4. In the "Marks" card, click on the "Shape" drop-down menu and select "More Shapes".
5. In the "Edit Shape" dialog box that appears, you can choose from a variety of pre-defined shapes or create your own custom shapes by clicking on the "Import" button.
6. To apply a shape to a specific data point or group of data points, you can use the "Shape" shelf. Simply drag the field that you want to use for shaping your data to the "Shape" shelf.
7. You can also use the "Size" and "Colour" shelves to further customize your shapes and make them more meaningful.
8. Once you have added shapes to your worksheet or dashboard, you can format them as needed using the formatting options available in Tableau. For example,

you can change the colour, size, or opacity of the shapes, add border and more.





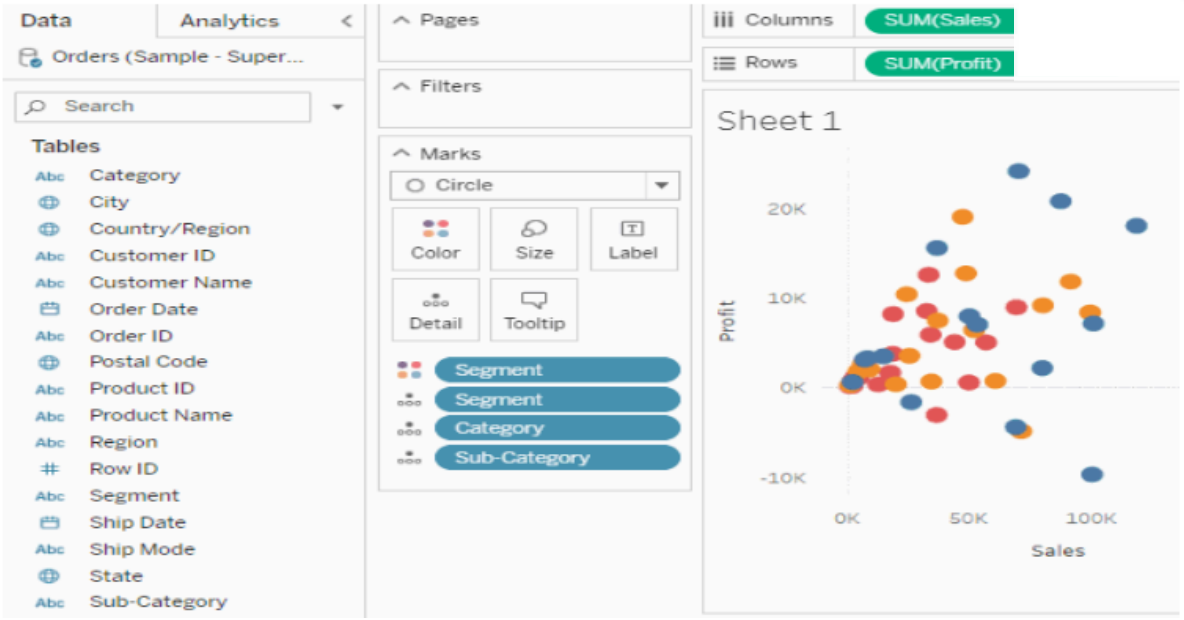
Hierarchies

When you connect to a data source, Tableau automatically separates date fields into hierarchies so you can easily break down the viz. You can also create your own custom hierarchies. For example, if you have a set of fields named Region, State, and County, you can create a hierarchy from these fields so that you can quickly drill down between levels in the viz.

Create a hierarchy

To create a hierarchy:

1. In the Data pane, drag a field and drop it directly on top of another field.
2. Note: When you want to create a hierarchy from a field inside a folder, right-click (control-click on a Mac) the field and then select Create Hierarchy.
3. When prompted, enter a name for the hierarchy and click OK.
4. Drag additional fields into the hierarchy as needed. You can also re-order fields in the hierarchy by dragging them to a new position.

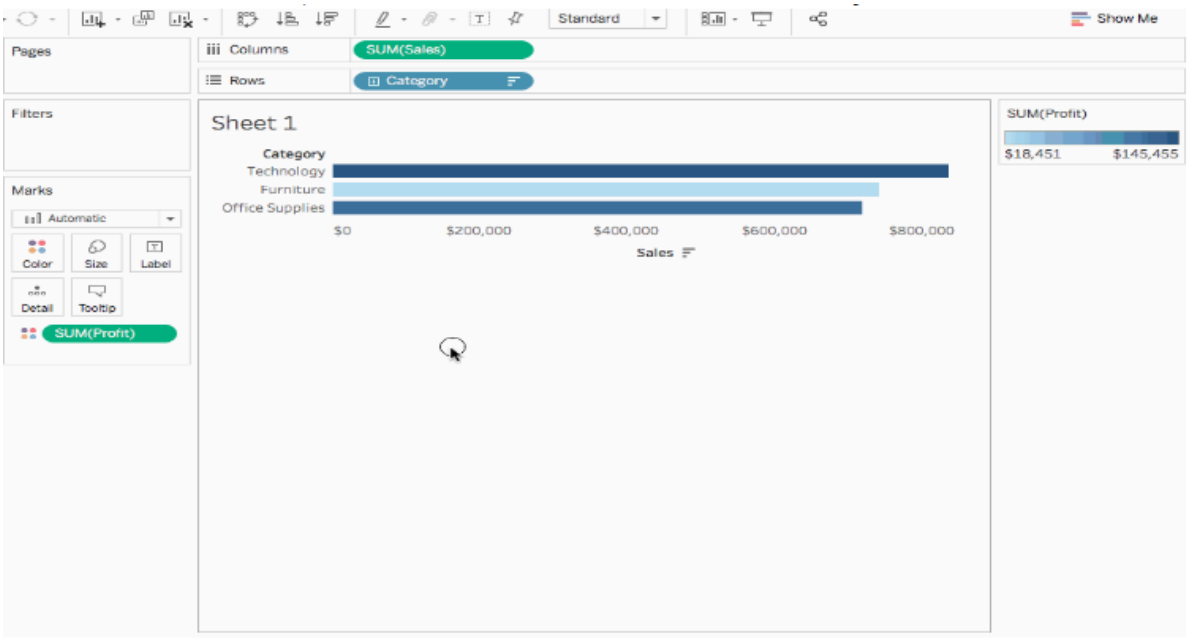


Drill up or down in a hierarchy

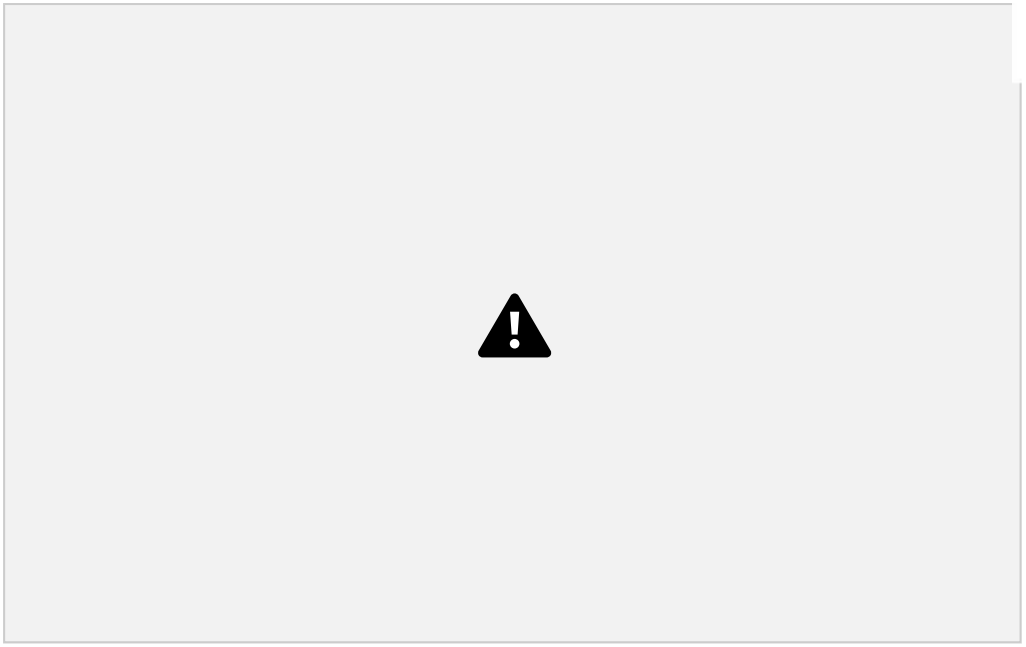
When you add a field from a hierarchy to the visualization, you can quickly drill up or down in the hierarchy to add or subtract more levels of detail.

To drill up or down in a hierarchy in Tableau Desktop or in web authoring:

In the visualization, click the + or - icon on the hierarchy field.



When you are editing or viewing the visualization on the web, you have the option of clicking the + or - icon next to a field label.



Remove a hierarchy

To remove a hierarchy:

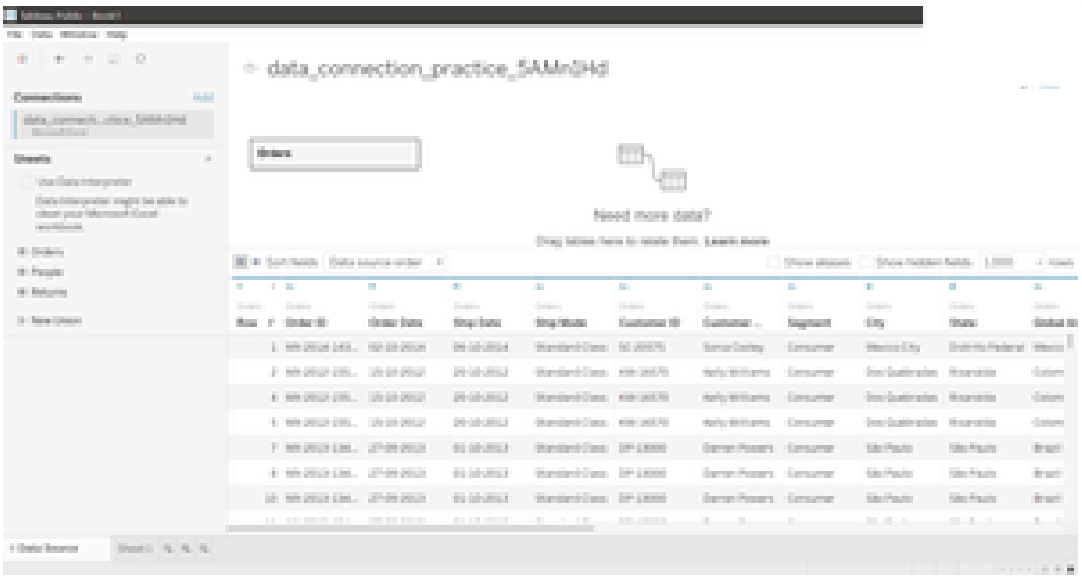
In the Data pane, right-click (control-click on a Mac) the hierarchy and select Remove Hierarchy.
The fields in the hierarchy are removed from the hierarchy and the hierarchy disappears from the Data pane.

Bins

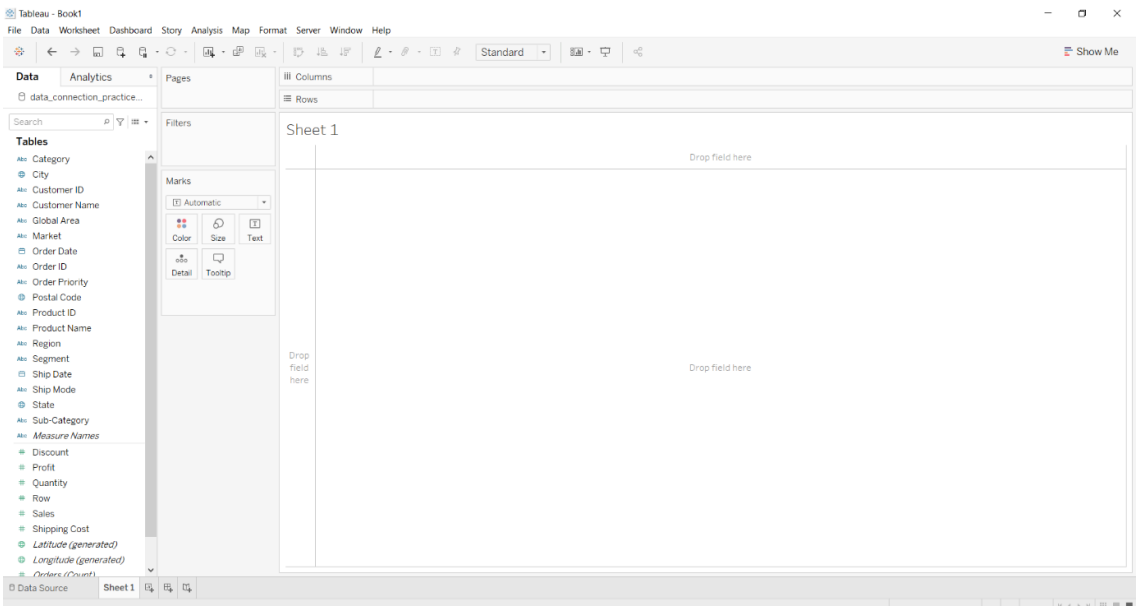
Tableau bins are containers of equal size that store data values like or fitting in bin size. Also, we'll say that bins a group of data into groups of equal intervals or size making it a scientific distribution of data. In Tableau, data from any discrete field are often taken to form bins. Although, Tableau users mostly use measure fields to form numeric bins.

Step-by-step Approach:

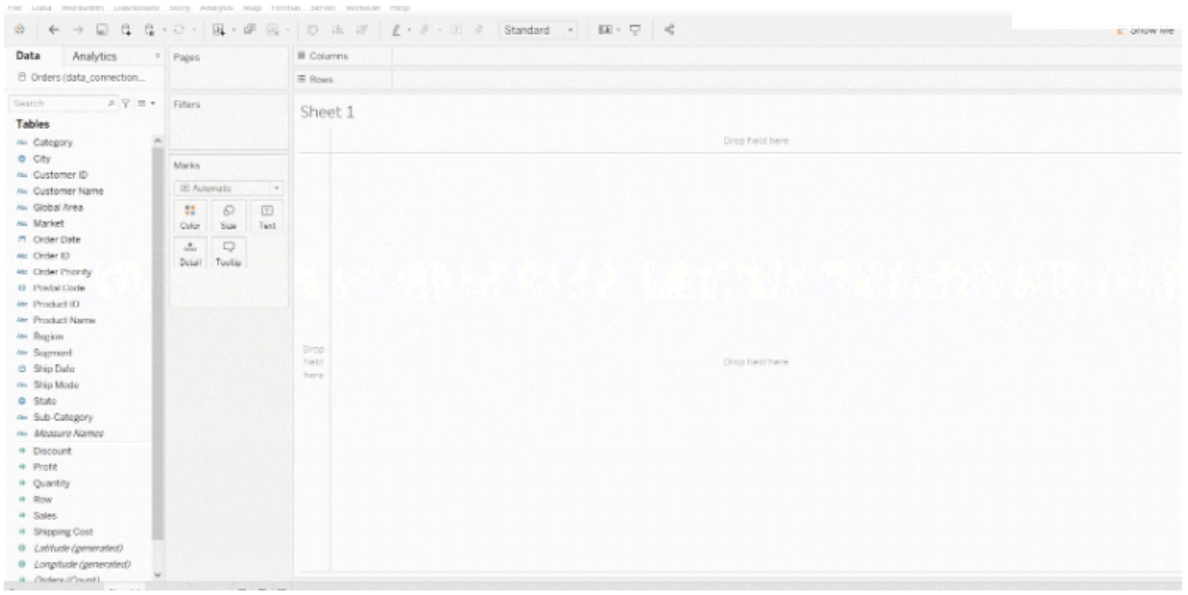
1. Open Tableau tool and connect a dataset into it.
2. Drag and drop the one sheet of connected dataset.



3. Click on *sheet1* to open the tableau worksheet.
4. On clicking *Sheet1* you will get whole dataset attributes on left side and a worksheet for work.



5. To create bins in tableau follow the given instructions:
   - Select an attribute
   - Right click on it
   - Click on create
   - Click on bins
   - Modify the suggested values (optional)
   - Click ok.

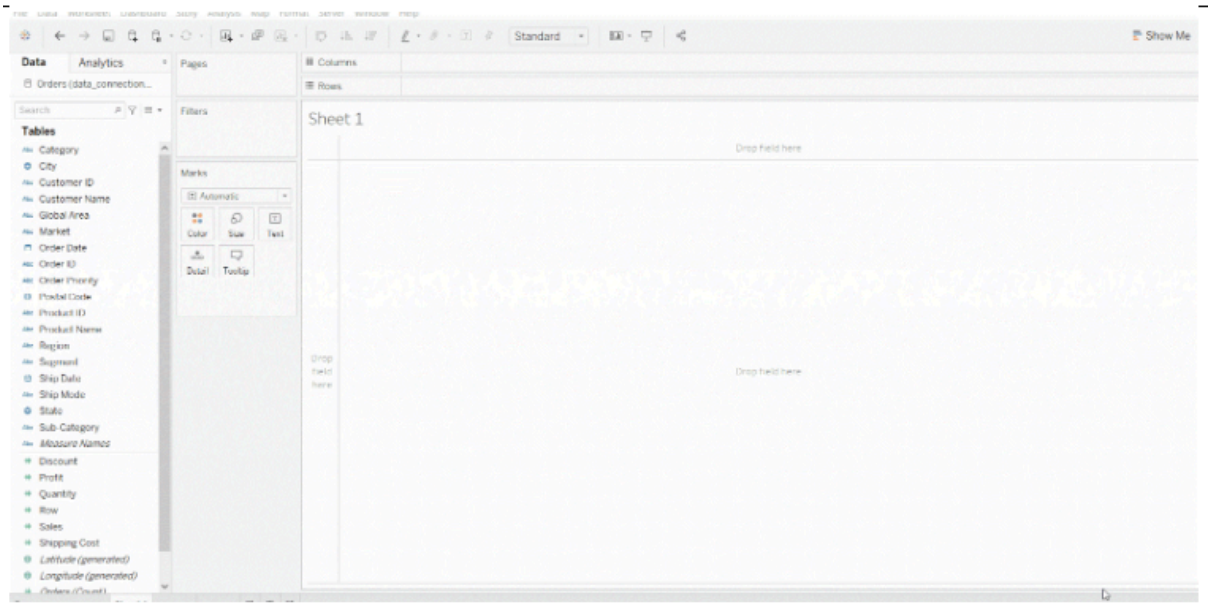


Below are some examples demonstrating the use of bins in Tableau:

Example 1:

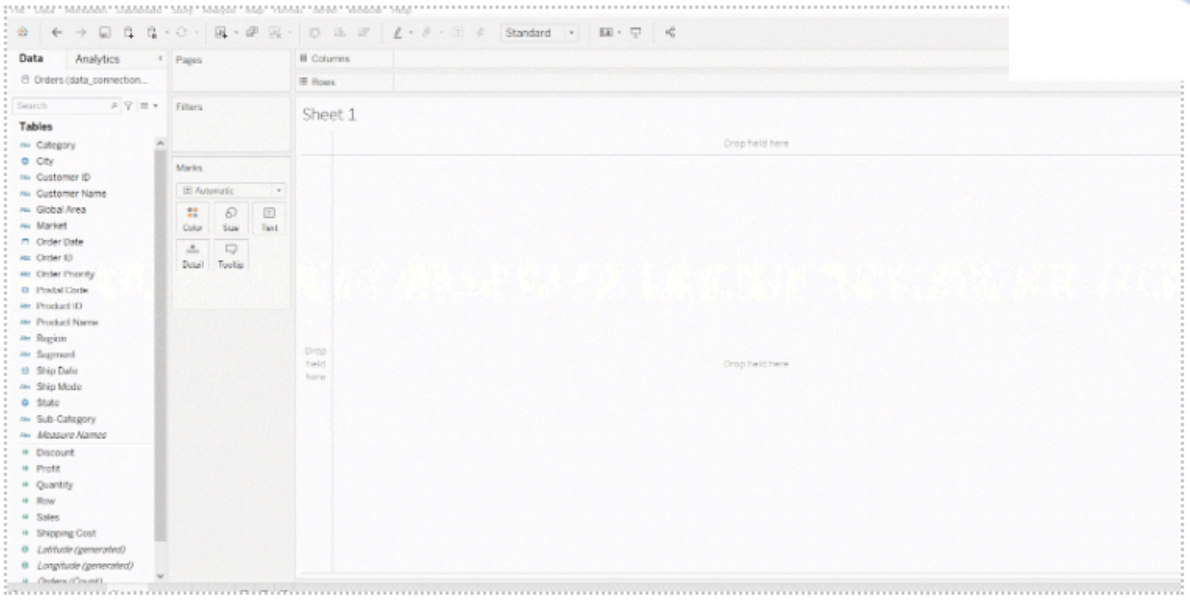
In this example, we use a short trick for creating a bin of an attribute.

- Here, we drag and drop an attribute into rows.
- Click on Show Me and select histogram.
- A bin of that drag and drop attribute in automatically created.



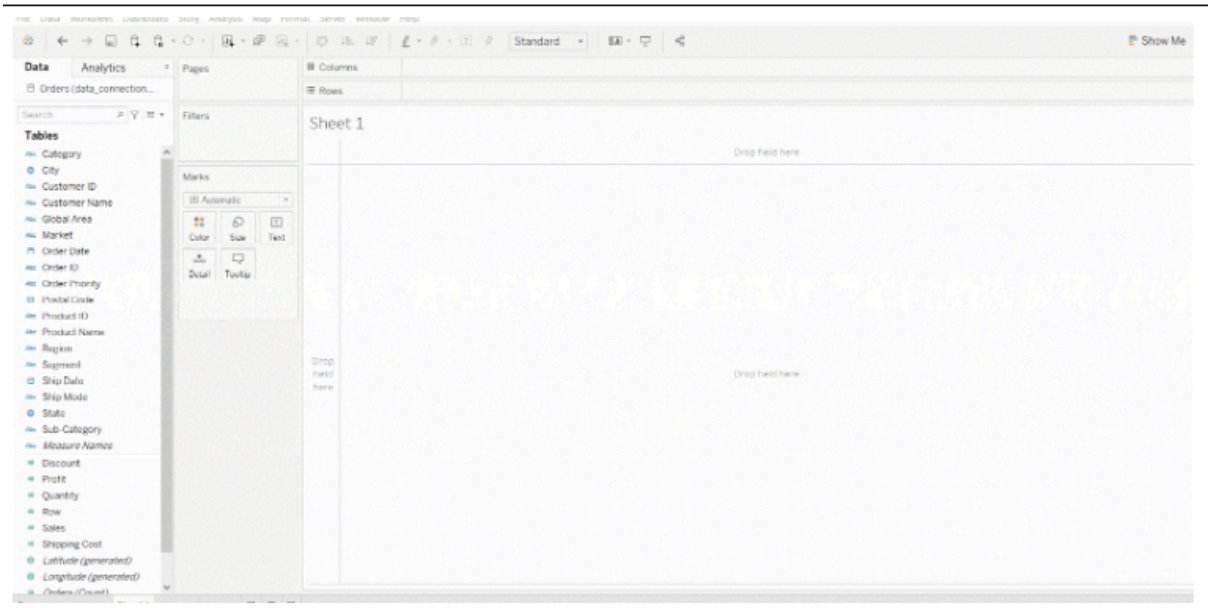
Example 2:

In this example, we use bin of an attribute and demonstrate the further operations that can be performed on it.



Example 3:

In this example, we will learn how to delete a bin of an attribute. One can simply delete it by right clicking on it and then click delete.



LOD (Level of Details)

Level of Detail expressions (also known as LOD expressions) allow you to compute values at the data source level and the visualisation level. However, LOD expressions give you even more control on the level of granularity you want to compute.

There are three types of LOD supported:

- **Exclude:** This LOD type is used to exclude specified dimension from the view.
- **Include:** This LOD type is used to include a specified dimension along with other dimensions within the view.
- **Fixed:** Unlike exclude and include, fixed doesn't depend on what is on the view rather focuses more on calculations. Thus it produces a fixed value for the specified dimensions.

Use case 1

The LOD (Level of Detail) analysis focuses on customer retention, specifically whether a customer has placed a repeated order after a certain period of time. In this analysis, we will be using a Fixed LOD approach.

Here's a step-by-step guide to perform the analysis:

Step 1: Add the dimensions such as Customer Name and Day of Order Date to your dataset.

Step 2: Calculate the Minimum Date using the following formula with Fixed LOD:

Min Date = {FIXED [Customer Name]:MIN([Order Date])}

This formula calculates the minimum order date for each customer across the entire dataset.

Step 3: Calculate the Second Highest Date after the Minimum Date using the following formula:

Second Date = { FIXED [Customer Name] : MAX(IF [Order Date] > [Minimum Date] THEN [Order Date] END) }

This formula finds the maximum order date for each customer that is greater than the minimum date.

Step 4: Calculate the Month Difference between the Minimum Date and the Second Date:

Month Difference = DATEDIFF('month', [Minimum Date], [Second Date])

This formula calculates the number of months between the minimum date and the second date for each customer.

Customer Name	Day of Order Date	Day of Min date	Day of second date	month diff	
Aaron Bergman	18 February 2018	18 February 2018	7 March 2018	1	Less than 6 Month
	7 March 2018	18 February 2018	7 March 2018	1	Less than 6 Month
	10 November 2020	18 February 2018	7 March 2018	1	Less than 6 Month
Aaron Hawkins	22 April 2018	22 April 2018	13 May 2018	1	Less than 6 Month
	13 May 2018	22 April 2018	13 May 2018	1	Less than 6 Month
	25 October 2018	22 April 2018	13 May 2018	1	Less than 6 Month
	31 December 2018	22 April 2018	13 May 2018	1	Less than 6 Month
	27 December 2019	22 April 2018	13 May 2018	1	Less than 6 Month
	20 March 2020	22 April 2018	13 May 2018	1	Less than 6 Month
	18 December 2021	22 April 2018	13 May 2018	1	Less than 6 Month
Aaron Smayling	27 July 2018	27 July 2018	28 March 2020	20	More Than 6 month
	28 March 2020	27 July 2018	28 March 2020	20	More Than 6 month
	25 September 2020	27 July 2018	28 March 2020	20	More Than 6 month
	2 January 2021	27 July 2018	28 March 2020	20	More Than 6 month
	1 August 2021	27 July 2018	28 March 2020	20	More Than 6 month
	4 September 2021	27 July 2018	28 March 2020	20	More Than 6 month
	3 October 2021	27 July 2018	28 March 2020	20	More Than 6 month
Adam Bellavance	18 September 2019	18 September 2019	13 March 2020	6	Less than 6 Month
	13 March 2020	18 September 2019	13 March 2020	6	Less than 6 Month
	29 August 2020	18 September 2019	13 March 2020	6	Less than 6 Month
	1 September 2020	18 September 2019	13 March 2020	6	Less than 6 Month
	7 May 2021	18 September 2019	13 March 2020	6	Less than 6 Month
	16 September 2021	18 September 2019	13 March 2020	6	Less than 6 Month
	5 November 2021	18 September 2019	13 March 2020	6	Less than 6 Month
	6 November 2021	18 September 2019	13 March 2020	6	Less than 6 Month
Adam Hart	16 November 2018	16 November 2018	21 December 2019	13	More Than 6 month

Step 5: Apply conditional formatting based on the calculated Month Difference. You can customize the formatting based on your requirements. For example, you can highlight customers with a month difference greater than a certain threshold or apply different colors based on the range of month differences.

IF [month diff]<=6 THEN 'Less than 6 Month'

ELSEIF [month diff]>6 THEN 'More Than 6 month'

ELSE 'No order'

END

condition	
Less than 6 Month	46.03%
More Than 6 month	52.46%
No order	1.51%

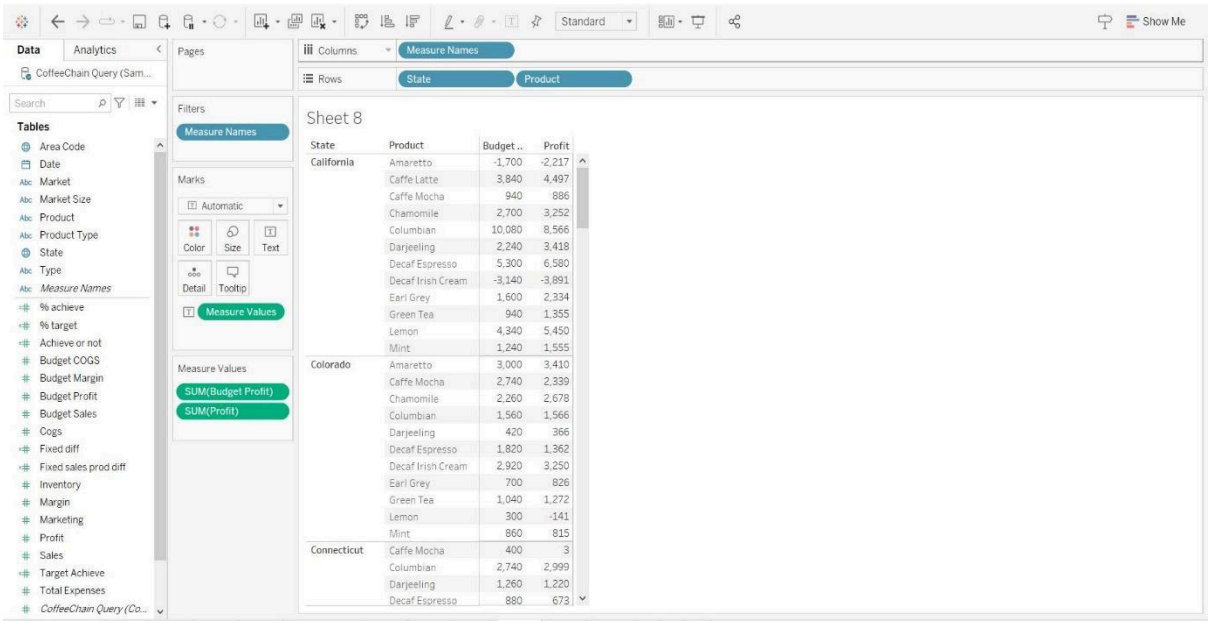
By following these steps, you can analyze customer retention using the Fixe
approach, identifying customers who have placed repeated orders after a p
period of time and understanding the month difference between those orders.

Use Case 2

Use Case (Fixed) : Actual vs Target

Step 1: We have to calculate budget profit and profit of each product in each state.

Step 2: Just drag state and product in rows and budget profit and profit in column.

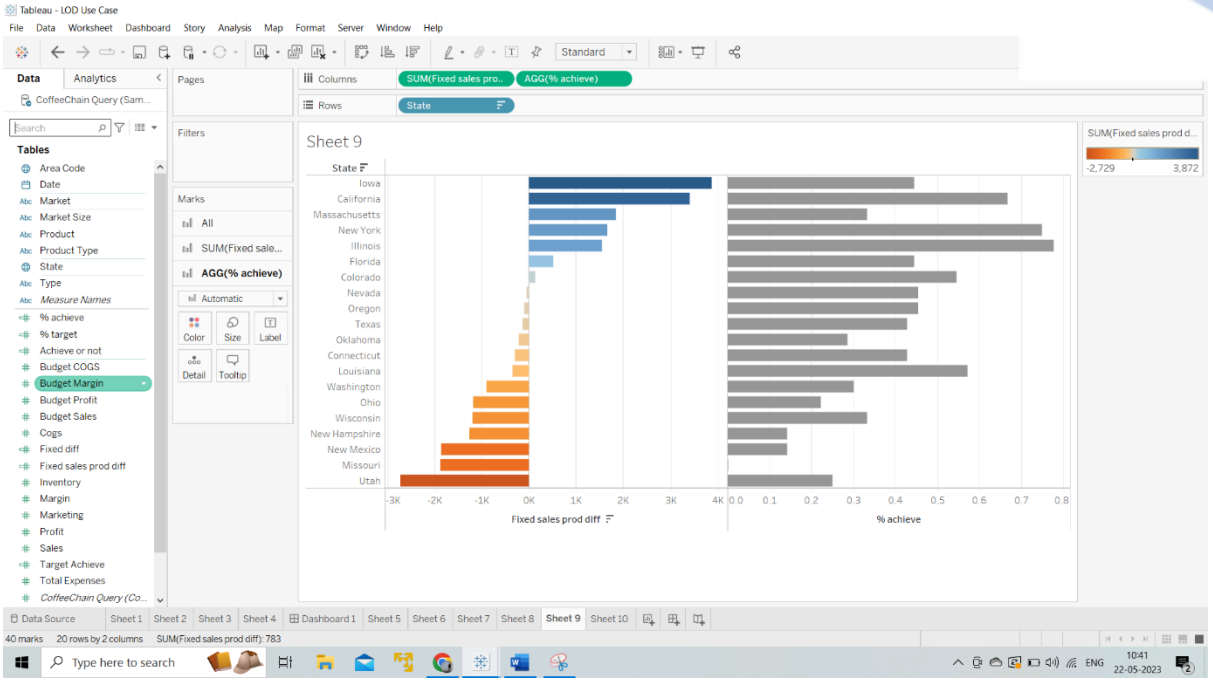


Step 3: We have to check the difference between profit and budget profit.

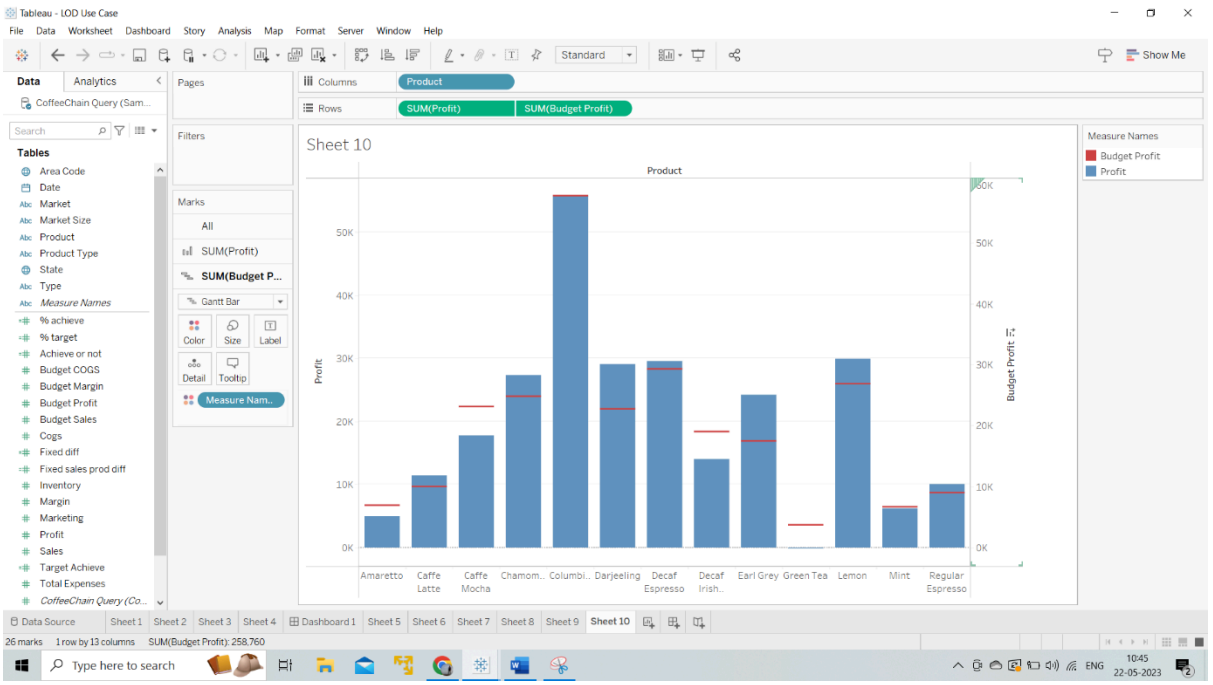
Step 4: We have to calculate the budget is achieved or not.

Step 5: We have to calculate percentage of how many products achieved their target.

Step 6: We have to check state wise budget target achieved or not and how many percent of products achieved their target.



Step 7: We have to check product wise difference between profit and budget profit by using bar and Gantt bar chart with Dual axis.



Use Case 3

Problem Statement: How can we determine the number of days in each month that are highly profitable, profitable, or unprofitable?

Solution: Step 1: To better understand the profitability, we create a calculat
determines whether the sum of profits is greater than zero. If it is, the calcu
assigns a value of 1; otherwise, it assigns a value of 0.

Step 2: To categorize the days into highly profitable, profitable, and loss, we create another calculation. If the sum of profits is greater than 10, it falls into the highly profitable category. If the sum of profits is greater than zero, it falls into the profitable category. If the sum of profits is less than zero, it falls into the loss category. This calculation is then fixed with the order date.

Step 3: We place these calculations in the rows of our visualization. We select the discrete year and month to be displayed in the column. Using an area chart, we can visualize the distribution of highly profitable, profitable, and unprofitable days over time.

Step 4: To highlight the boundaries of the graph, we duplicate all the calculations and convert them into dual axes. This allows us to visually differentiate the categories and draw attention to the areas of high profitability or loss.

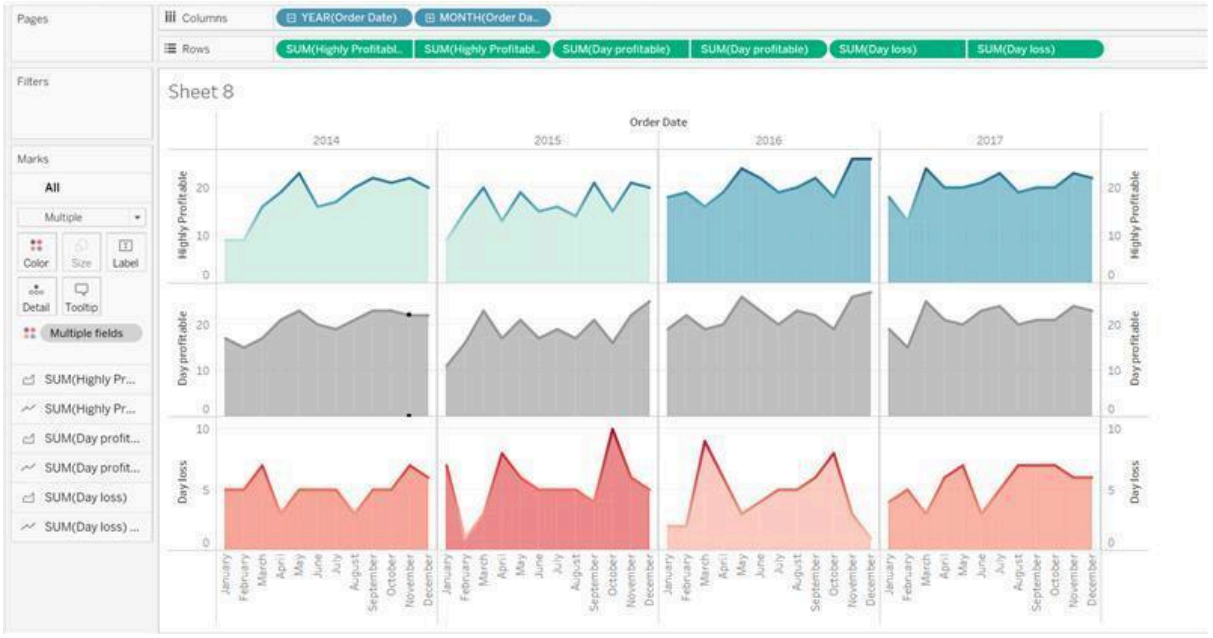


Table Calculation

What is a table calculation?

A table calculation is a transformation you apply to the values in a visualization. Table calculations are a special type of calculated field that computes on the local data in Tableau. They are calculated based on what is currently in the visualization and do not consider any measures or dimensions that are filtered out of the visualization.

You can use table calculations for a variety of purposes, including:

- Transforming values to rankings
- Transforming values to show running totals
- Transforming values to show percent of total

For any Tableau visualization, there is a virtual table that is determined by the dimensions in the view. This table is not the same as the tables in your data source. Specifically, the virtual table is determined by the dimensions within the "level of detail," which means the dimensions on any of the following shelves or cards in a Tableau worksheet:

The basics: addressing and partitioning

When you add a table calculation, you must use all dimensions in the level of detail either for partitioning (scoping) or for addressing (direction).

The dimensions that define how to group the calculation (the scope of data it is performed on) are called **partitioning fields**. The table calculation is performed separately within each partition.

The remaining dimensions, upon which the table calculation is performed, are called **addressing fields**, and determine the direction of the calculation.

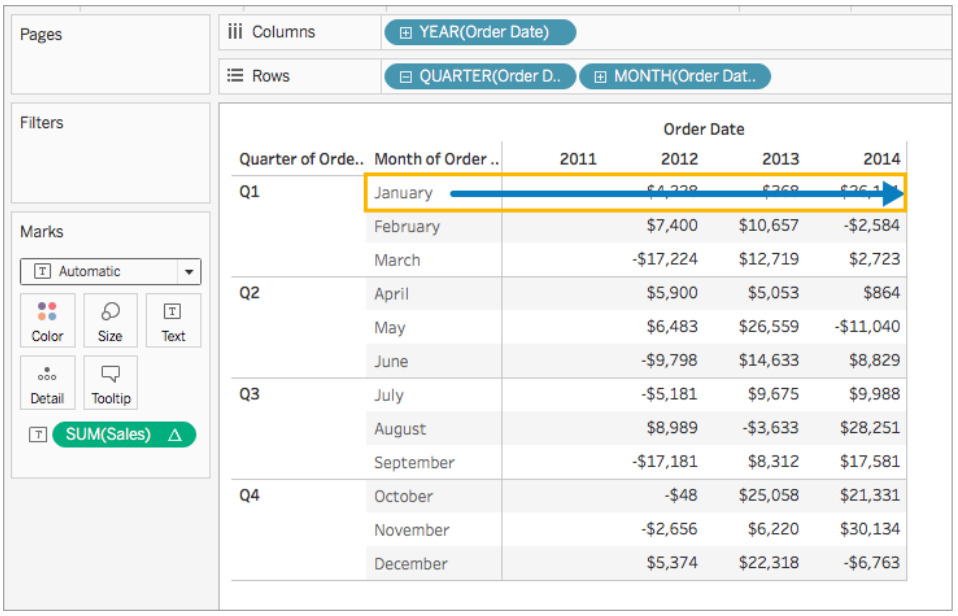
Partitioning fields break the view up into multiple sub-views (or sub-tables the table calculation is applied to the marks within each such partition. The which the calculation moves (for example, in calculating a running sum, or computing the difference between values) is determined by the addressing fields. So when you order the fields in the Specific Dimensions section of the Table Calculation dialog box from top to bottom, you are specifying the direction in which the calculation moves through the various marks in the partition.

When you add a table calculation using the Compute Using options, Tableau identifies some dimensions as addressing and others as partitioning automatically, as a result of your selections. But when you use Specific Dimensions, then it's up to you to determine which dimensions are for addressing and which for partitioning.

Table (across)

Computes across the length of the table and restarts after every partition.

For example, in the following table, the calculation is computed across columns (YEAR(Order Date)) for every row (MONTH(Order Date)).



		Order Date			
Quarter of Orde..	Month of Order ..	2011	2012	2013	2014
Q1	January				
	February		$7,400	$10,657	-$2,584
	March		-$17,224	$12,719	$2,723
Q2	April		$5,900	$5,053	$864
	May		$6,483	$26,559	-$11,040
	June		-$9,798	$14,633	$8,829
Q3	July		-$5,181	$9,675	$9,988
	August		$8,989	-$3,633	$28,251
	September		-$17,181	$8,312	$17,581
Q4	October		-$48	$25,058	$21,331
	November		-$2,656	$6,220	$30,134
	December		$5,374	$22,318	-$6,763

Table (down)

Computes down the length of the table and restarts after every partition.

For example, in the following table, the calculation is computed down rows (MONTH(Order Date)) for every column (YEAR(Order Date)).

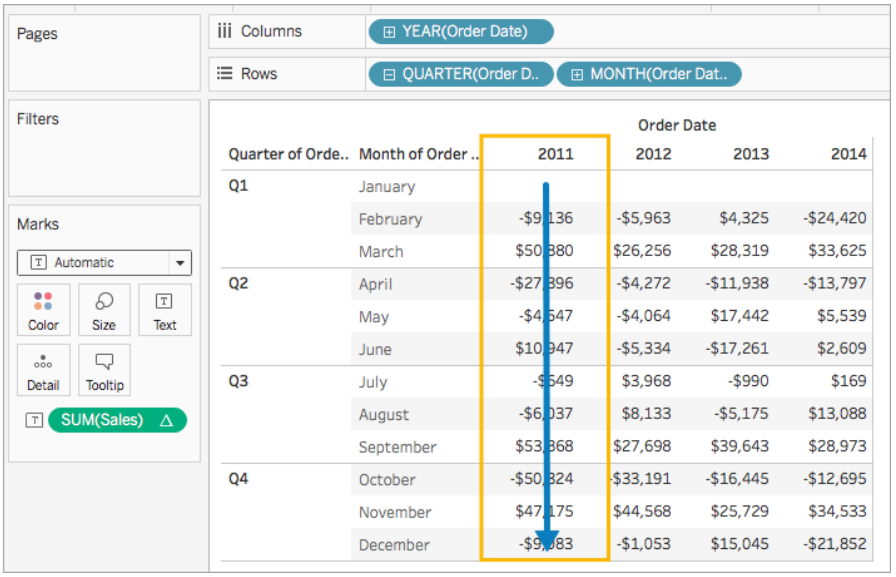


Table (across then down)

Computes across the length of the table, and then down the length of the table.

For example, in the following table, the calculation is computed across columns (YEAR(Order Date)), down a row (MONTH(Order Date)), and then across columns again for the entire table.

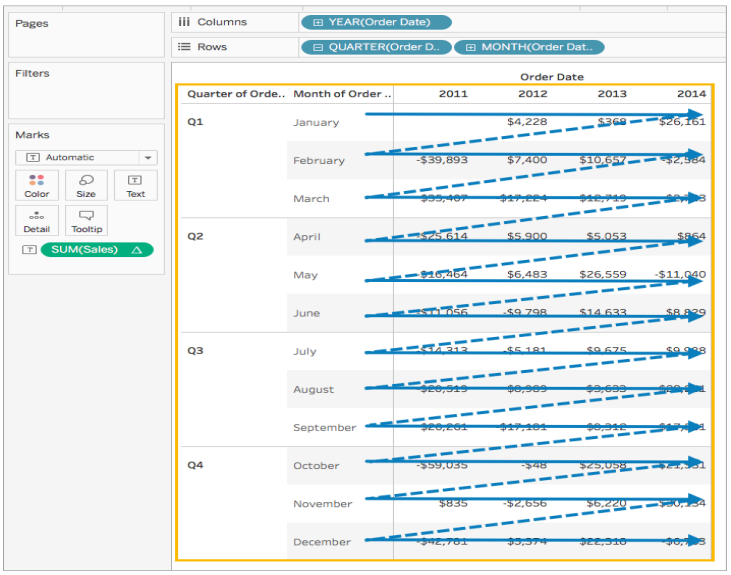
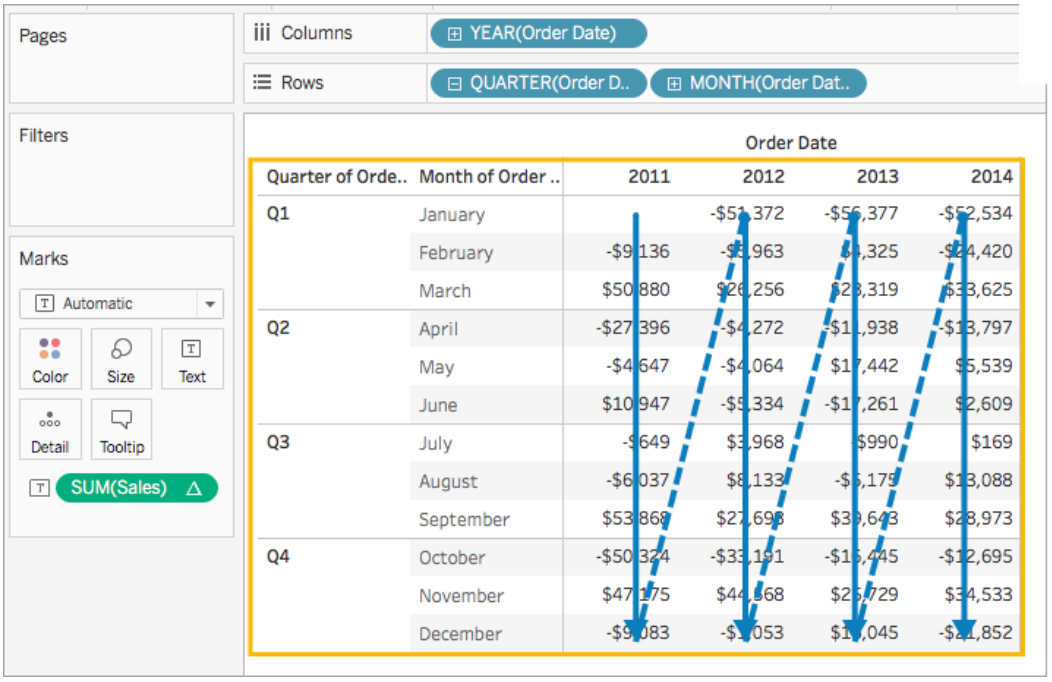


Table (down then across)

Computes down the length of the table, and then across the length of the table.

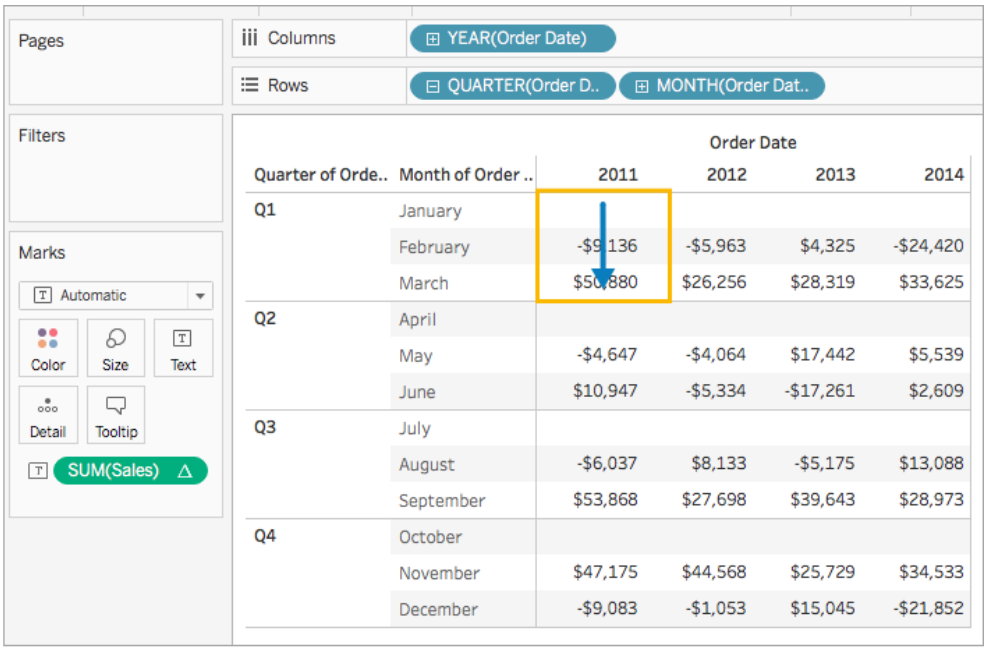
For example, in the following table, the calculation is computed down rows (MONTH(Order Date)), across a column (YEAR(Order Date), and then down rows again.



Pane (down)

Computes down an entire pane.

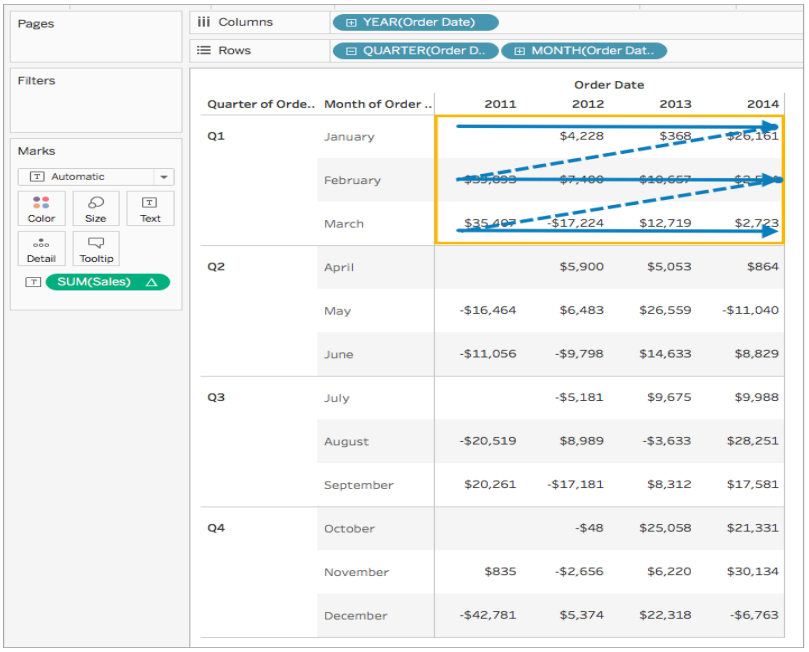
For example, in the following table, the calculation is computed down rows (MONTH(Order Date)) for a single pane.



Pane (across then down)

Computes across an entire pane and then down the pane.

For example, in the following table, the calculation is computed across colu
(YEAR(Order Date)) for the length of the pane, down a row (MONTH(Order
then across columns for the length of the pane again.

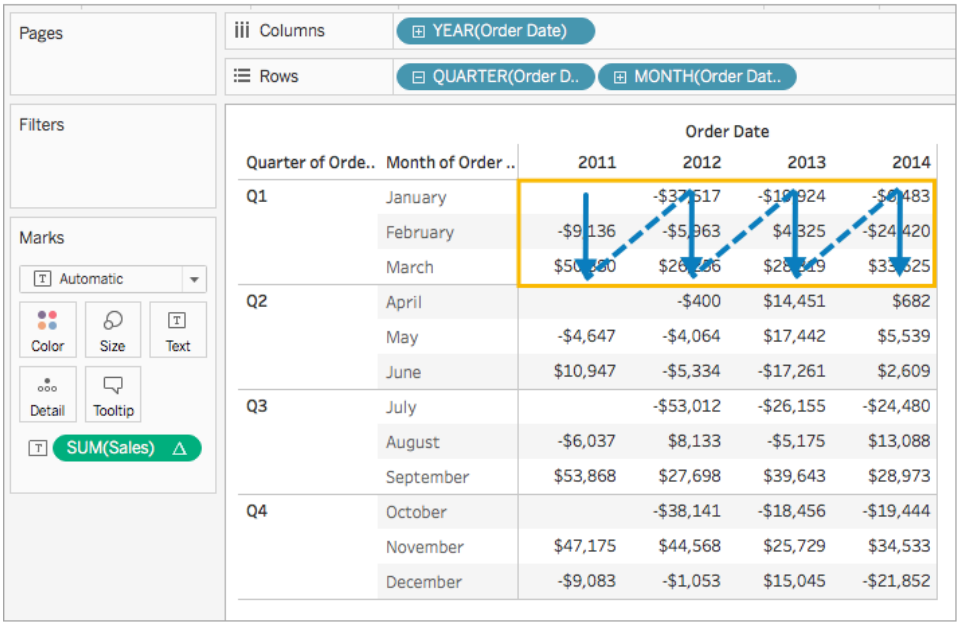


		Order Date			
Quarter of Orde..	Month of Order ..	2011	2012	2013	2014
Q1	January		$4,228	$368	$26,161
	February	[illegible]	[illegible]	[illegible]	[illegible]
	March	$35,407	-$17,224	$12,719	$2,723
Q2	April		$5,900	$5,053	$864
	May	-$16,464	$6,483	$26,559	-$11,040
	June	-$11,056	-$9,798	$14,633	$8,829
Q3	July		-$5,181	$9,675	$9,988
	August	-$20,519	$8,989	-$3,633	$28,251
	September	$20,261	-$17,181	$8,312	$17,581
Q4	October		-$48	$25,058	$21,331
	November	$835	-$2,656	$6,220	$30,134
	December	-$42,781	$5,374	$22,318	-$6,763

Pane (down then across)

Computes down an entire pane and then across the pane.

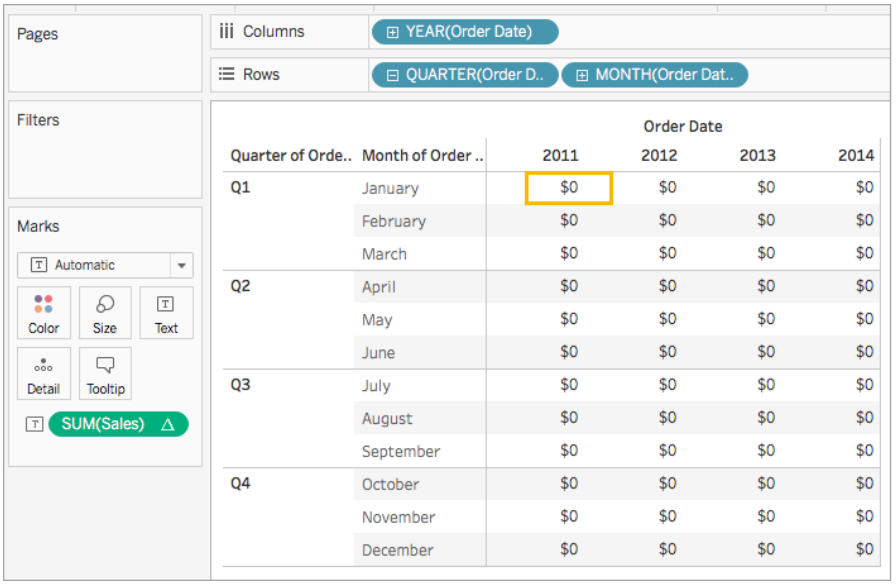
For example, in the following table, the calculation is computed down rows (MONTH(Order Date)) for the length of the pane, across a column (YEAR(Order Date)), and then down the length of the pane agian.



		Order Date			
Quarter of Orde..	Month of Order ..	2011	2012	2013	2014
Q1	January		-$37,517	-$19,924	-$6,483
	February	-$9,136	-$5,963	$4,325	-$24,420
	March	$50,280	$26,206	$28,249	$33,625
Q2	April		-$400	$14,451	$682
	May	-$4,647	-$4,064	$17,442	$5,539
	June	$10,947	-$5,334	-$17,261	$2,609
Q3	July		-$53,012	-$26,155	-$24,480
	August	-$6,037	$8,133	-$5,175	$13,088
	September	$53,868	$27,698	$39,643	$28,973
Q4	October		-$38,141	-$18,456	-$19,444
	November	$47,175	$44,568	$25,729	$34,533
	December	-$9,083	-$1,053	$15,045	-$21,852

Cell

Computes within a single cell.



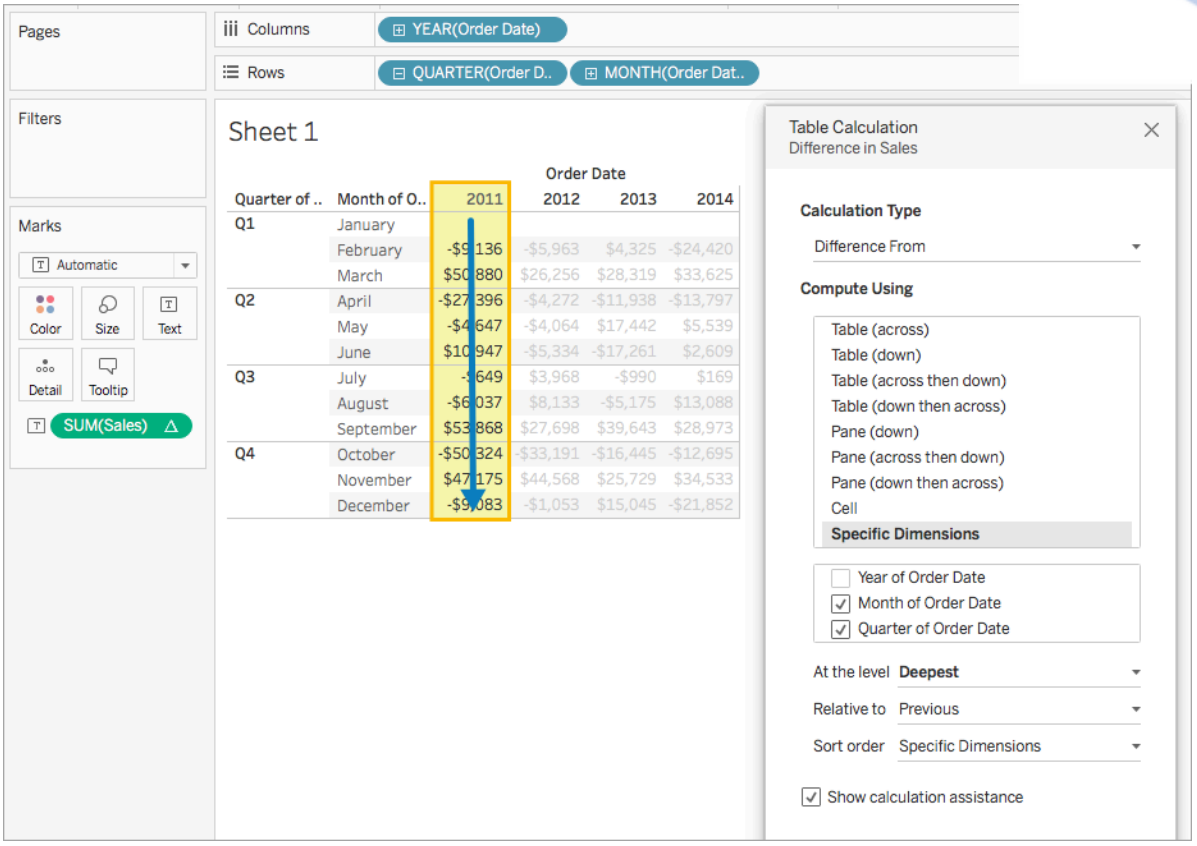
		Order Date			
Quarter of Orde..	Month of Order ..	2011	2012	2013	2014
Q1	January	$0	$0	$0	$0
	February	$0	$0	$0	$0
	March	$0	$0	$0	$0
Q2	April	$0	$0	$0	$0
	May	$0	$0	$0	$0
	June	$0	$0	$0	$0
Q3	July	$0	$0	$0	$0
	August	$0	$0	$0	$0
	September	$0	$0	$0	$0
Q4	October	$0	$0	$0	$0
	November	$0	$0	$0	$0
	December	$0	$0	$0	$0

Specific Dimensions

Computes only within the dimensions you specify.

For example, in the following visualization the dimensions, Month of Order Date and Quarter of Order Date, are the addressing fields (since they are selected), and Year of Order Date is the partitioning field (since it is not selected). So the calculation transforms the difference from each month across all quarters within a year. The calculation starts over for every year.

Note that if all dimensions are selected, then the entire table is in scope.



At the level

The **At the level** option is only available when you select **Specific Dimensions** in the Table Calculations dialog box, and when more than one dimension is selected in the field immediately below the **Compute Using** options —that is, when more than one dimension is defined as an addressing field.

This option is not available when you're defining a table calculation with **Compute Using**, because those values establish partitions by position. But with **Specific Dimensions**, because the visual structure and the table calculation are not necessarily aligned, the **At the level** option is available to let you fine-tune your calculation.

Use this setting to set a break (that is, restart of the calculation) in the view, based on a particular dimension. How is this different from just using that dimension for partitioning? In fact, it is partitioning, but it's partitioning by position rather than by value, which is how partitioning is defined with the **Compute Using** options.

The choices available from the At the level drop-down list in the example above are:

Create a table calculation

To learn how to create a table calculation, follow along with the steps in the example below. To learn how to create quick table calculations, see Quick Table Calculations.

Step 1: Build the visualization

1. Open Tableau and connect to the **Sample-Superstore** saved data so

2. Navigate to a new worksheet.

3. From the **Data** pane, under Dimensions, drag **Order Date** to the **Rows** shelf.

   The dimension updates to YEAR(Order Date).

4. On the Rows shelf, right-click **YEAR(Order Date)** and select **Quarter**.

5. On the Rows shelf, click the **+** icon on **QUARTER(Order Date)**.

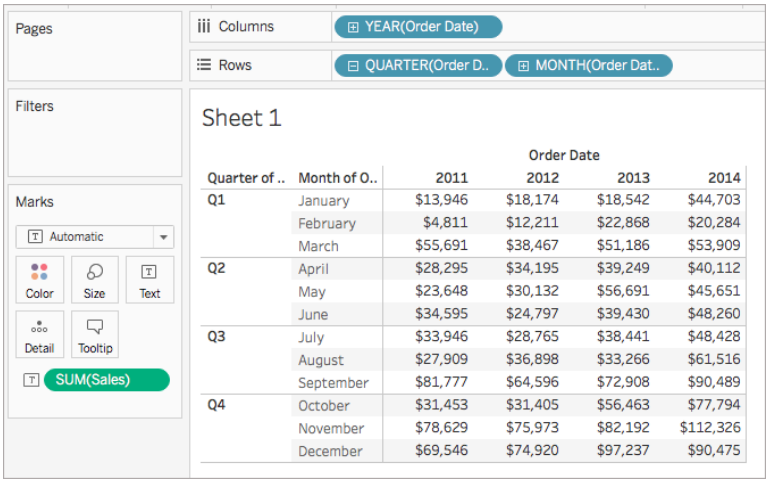
   MONTH(Order Date) is added to the shelf.

6. From the **Data** pane, under Dimensions, drag **Order Date** to the **Columns** shelf.

   The dimension updates to YEAR(Order Date) again.

7. From the **Data** pane, under Measures, drag **Sales** to **Text** on the Marks card.

   The updates to look like this:



		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

Step 2: Add the table calculation

1. On the Marks card, right-click SUM(Sales) and select **Add Table Calculation**.

2. In the Table Calculation dialog box that opens, do the following:

   - For **Calculation Type**: select **Difference From**.

     For more information about the types of table calculations you can use in Tableau, and how you can configure them, see Table Calculation Types.

   - For **Compute Using**, select **Table (across)**.

For more information about these options, see The basics: ad partitioning section.

Note that as you select how to compute the calculation, the visualization updates with visual indicators to guide you.

- When finished, click the X in the top corner of the Table Calculation dialog box to exit it.

The calculation is applied to the values in the visualization.

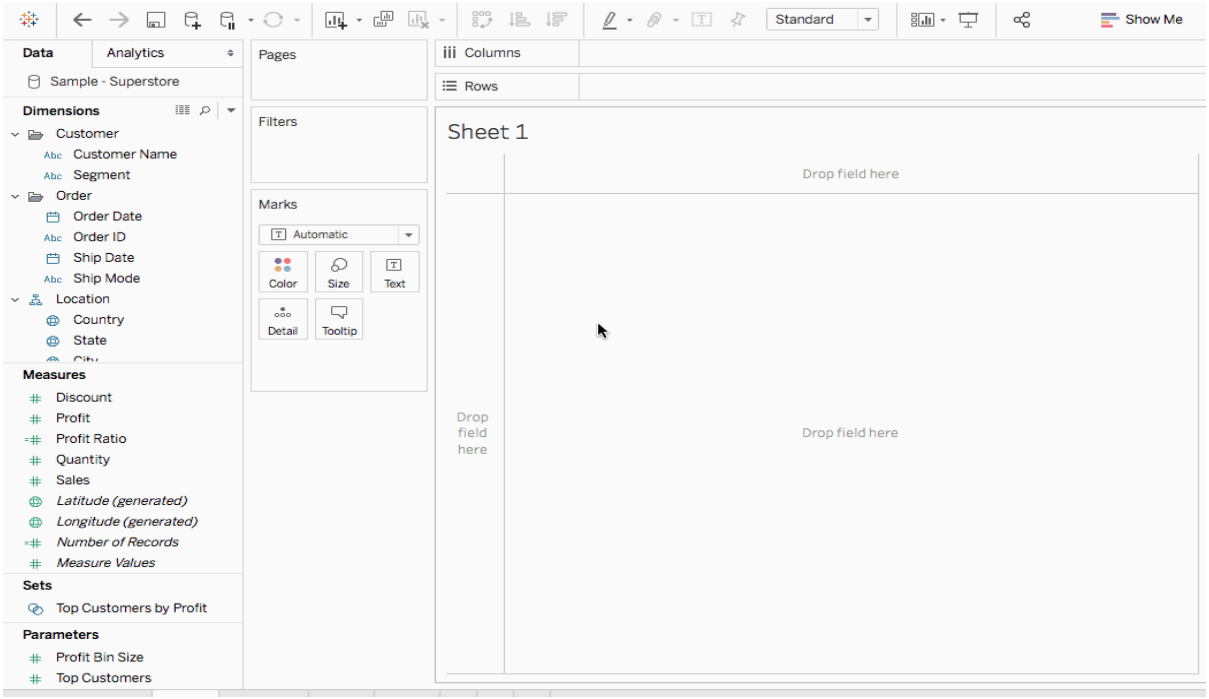


Table calculation functions:

FIRST( )

Returns the number of rows from the current row to the first row in the partition. For example, the view below shows quarterly sales. When FIRST() is computed within the Date partition, the offset of the first row from the second row is -1.

Year of Order Date	Quarter of Order Date	Region: Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

$160,877	0
$197,213	-1
$302,678	-2
$297,208	-3
$180,609	-4
$195,785	-5
$116,613	-6

Example

When the current row index is 3, FIRST() = -2.

INDEX( )

Returns the index of the current row in the partition, without any sorting with regard to value. The first row index starts at 1. For example, the table below shows quarterly sales. When INDEX() is computed within the Date partition, the index of each row is 1, 2, 3, 4..., etc.

Year of Order Date	Quarter of Order Date	Region: Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

	INDEX()
$160,877	1
$197,213	2
$302,678	3
$297,208	4
$180,609	5
$195,785	6
$116,613	7

Example

For the third row in the partition, INDEX() = 3.

LAST( )

Returns the number of rows from the current row to the last row in the partition. For example, the table below shows quarterly sales. When LAST() is computed within the Date partition, the offset of the last row from the second row is 5.

Year of Order Date	Quarter of Order Date	Region: Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

$160,877	6
$197,213	5
$302,678	4
$297,208	3
$180,609	2
$195,785	1
$116,613	0

Example

When the current row index is 3 of 7, LAST() = 4.

LOOKUP(expression, [offset])

Returns the value of the expression in a target row, specified as a relative offset from the current row. Use FIRST() + n and LAST() - n as part of your offset definition for a target relative to the first/last rows in the partition. If offset is omitted, the row to compare to can be set on the field menu. This function returns NULL if the target row cannot be determined.

The view below shows quarterly sales. When LOOKUP (SUM(Sales), 2) is computed within the Date partition, each row shows the sales value from 2 quarters into the future.

Year of Order Date	Quarter of Order Date	Region: Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3 +2	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

Year of Order Date	Quarter of Order Date	Region: Central	East	South	West
2009	Q1	$302,678	$165,201	$283,806	$206,512
	Q2	$297,208	$226,983	$214,845	$230,291
	Q3	$180,609	$180,123	$273,943	$251,145
	Q4	$195,785	$224,882	$251,391	$195,976
2010	Q1	$116,613	$50,363	$194,601	$102,731
	Q2				
	Q3				

Example

LOOKUP(SUM([Profit]), FIRST()+2) computes the SUM(Profit) in the third row of partition.

MODEL_EXTENSION_BOOL (model_name, arguments, expression)

Returns the boolean result of an expression as calculated by a named model deployed on a TabPy external service.

Model_name is the name of the deployed analytics model you want to use.

Each argument is a single string that sets the input values that the deployed model accepts, and is defined by the analytics model.

Use expressions to define the values that are sent from Tableau to the analytics model. Be sure to use aggregation functions (SUM, AVG, etc.) to aggregate the results.

When using the function, the data types and order of the expressions must match that of the input arguments.

Example

MODEL_EXTENSION_BOOL ("isProfitable","inputSales", "inputCosts", SUM([Sales]), SUM([Costs]))

MODEL_EXTENSION_INT (model_name, arguments, expression)

Returns an integer result of an expression as calculated by a named model deployed on a TabPy external service.

Model_name is the name of the deployed analytics model you want to use.

Each argument is a single string that sets the input values that the deployed model accepts, and is defined by the analytics model.

Use expressions to define the values that are sent from Tableau to the analytics model. Be sure to use aggregation functions (SUM, AVG, etc.) to aggregate the results.

When using the function, the data types and order of the expressions must match that of the input arguments.

Example

MODEL_EXTENSION_INT ("getPopulation", "inputCity", "inputState", MAX([City]), MAX ([State]))

MODEL_EXTENSION_REAL (model_name, arguments, expression)

Returns a real result of an expression as calculated by a named model deployed on a TabPy external service.

Model_name is the name of the deployed analytics model you want to use.

Each argument is a single string that sets the input values that the deployed model accepts, and is defined by the analytics model.

Use expressions to define the values that are sent from Tableau to the analytics model. Be sure to use aggregation functions (SUM, AVG, etc.) to aggregate the results.

When using the function, the data types and order of the expressions must
the input arguments.

Example
MODEL_EXTENSION_REAL ("profitRatio", "inputSales", "inputCosts", SUM([Sales]), SUM([Costs]))

MODEL_EXTENSION_STRING (model_name, arguments, expression)

Returns the string result of an expression as calculated by a named model deployed on a TabPy external service.

Model_name is the name of the deployed analytics model you want to use.

Each argument is a single string that sets the input values that the deployed model accepts, and is defined by the analytics model.

Use expressions to define the values that are sent from Tableau to the analytics model. Be sure to use aggregation functions (SUM, AVG, etc.) to aggregate the results.

When using the function, the data types and order of the expressions must match that of the input arguments.

Example
MODEL_EXTENSION_STR ("mostPopulatedCity", "inputCountry", "inputYear", MAX ([Country]), MAX([Year]))

MODEL_PERCENTILE(target_expression, predictor_expression(s))

Returns the probability (between 0 and 1) of the expected value being less than or equal to the observed mark, defined by the target expression and other predictors. This is the Posterior Predictive Distribution Function, also known as the Cumulative Distribution Function (CDF).

This function is the inverse of MODEL_QUANTILE. For information on predictive modeling functions, see How Predictive Modeling Functions Work in Tableau.

Example
The following formula returns the quantile of the mark for sum of sales, adjusted for count of orders.

MODEL_PERCENTILE(SUM([Sales]), COUNT([Orders]))

MODEL_QUANTILE(quantile, target_expression, predictor_expression(s))

Returns a target numeric value within the probable range defined by the target expression and other predictors, at a specified quantile. This is the Posterior Predictive Quantile.

This function is the inverse of MODEL_PERCENTILE. For information on predictive modeling functions, see How Predictive Modeling Functions Work in Tableau.

Example
The following formula returns the median (0.5) predicted sum of sales, adjusted for count of orders.

MODEL_QUANTILE(0.5, SUM([Sales]), COUNT([Orders]))

PREVIOUS_VALUE(expression)

Returns the value of this calculation in the previous row. Returns the given expression if the current row is the first row of the partition.

Example

SUM([Profit]) * PREVIOUS_VALUE(1) computes the running product of SUM(Profit).

RANK(expression, ['asc' | 'desc'])

Returns the standard competition rank for the current row in the partition. Identical values are assigned an identical rank. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is descending.

With this function, the set of values (6, 9, 9, 14) would be ranked (4, 2, 2, 1).

Nulls are ignored in ranking functions. They are not numbered and they do not count against the total number of records in percentile rank calculations.

For information on different ranking options, see Rank calculation.

Example

The following image shows the effect of the various ranking functions (RANK, RANK_DENSE, RANK_MODIFIED, RANK_PERCENTILE, and RANK_UNIQUE) on a set of values. The data set contains information on 14 students (StudentA through StudentN); the Age column shows the current age of each student (all students are between 17 and 20 years of age). The remaining columns show the effect of each rank function on the set of age values, always assuming the default order (ascending or descending) for the function.

Student	Age	RANKofAge	RANK_DENSEofAge	RANK_MODIFIEDofAge	RANK_PERCENTILEofAge	RANK_UNIQUEofAge
StudentA	19	4	2	7	79%	4
StudentB	18	8	3	12	50%	8
StudentC	19	4	2	7	79%	5
StudentD	18	8	3	12	50%	9
StudentE	17	13	4	14	14%	13
StudentF	18	8	3	12	50%	10
StudentG	19	4	2	7	79%	6
StudentH	20	1	1	3	100%	1
StudentI	19	4	2	7	79%	7
StudentJ	20	1	1	3	100%	2
StudentK	20	1	1	3	100%	3
StudentL	17	13	4	14	14%	14
StudentM	18	8	3	12	50%	11
StudentN	18	8	3	12	50%	12

RANK_DENSE(expression, ['asc' | 'desc'])

Returns the dense rank for the current row in the partition. Identical values are assigned an identical rank, but no gaps are inserted into the number sequence. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is descending.

With this function, the set of values (6, 9, 9, 14) would be ranked (3, 2, 2, 1).

Nulls are ignored in ranking functions. They are not numbered and they do against the total number of records in percentile rank calculations.

For information on different ranking options, see Rank calculation.

RANK_MODIFIED(expression, ['asc' | 'desc'])

Returns the modified competition rank for the current row in the partition. Identical values are assigned an identical rank. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is descending.

With this function, the set of values (6, 9, 9, 14) would be ranked (4, 3, 3, 1).

Nulls are ignored in ranking functions. They are not numbered and they do not count against the total number of records in percentile rank calculations.

For information on different ranking options, see Rank calculation.

RANK_PERCENTILE(expression, ['asc' | 'desc'])

Returns the percentile rank for the current row in the partition. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is ascending.

With this function, the set of values (6, 9, 9, 14) would be ranked (0.00, 0.67, 0.67, 1.00).

Nulls are ignored in ranking functions. They are not numbered and they do not count against the total number of records in percentile rank calculations.

For information on different ranking options, see Rank calculation.

RANK_UNIQUE(expression, ['asc' | 'desc'])

Returns the unique rank for the current row in the partition. Identical values are assigned different ranks. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is descending.

With this function, the set of values (6, 9, 9, 14) would be ranked (4, 2, 3, 1).

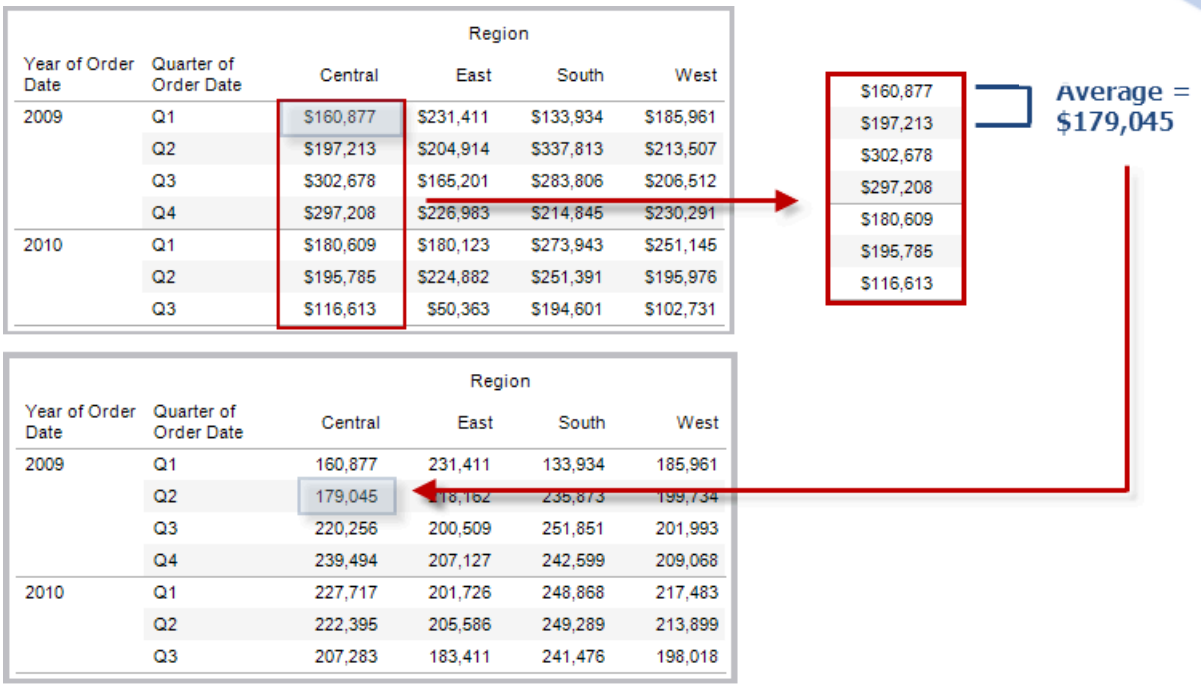
Nulls are ignored in ranking functions. They are not numbered and they do not count against the total number of records in percentile rank calculations.

For information on different ranking options, see Rank calculation.

RUNNING_AVG(expression)

Returns the running average of the given expression, from the first row in the partition to the current row.

The view below shows quarterly sales. When RUNNING_AVG(SUM([Sales]) is computed within the Date partition, the result is a running average of the sales values for each quarter.



Example

RUNNING_AVG(SUM([Profit])) computes the running average of SUM(Profit).

RUNNING_COUNT(expression)

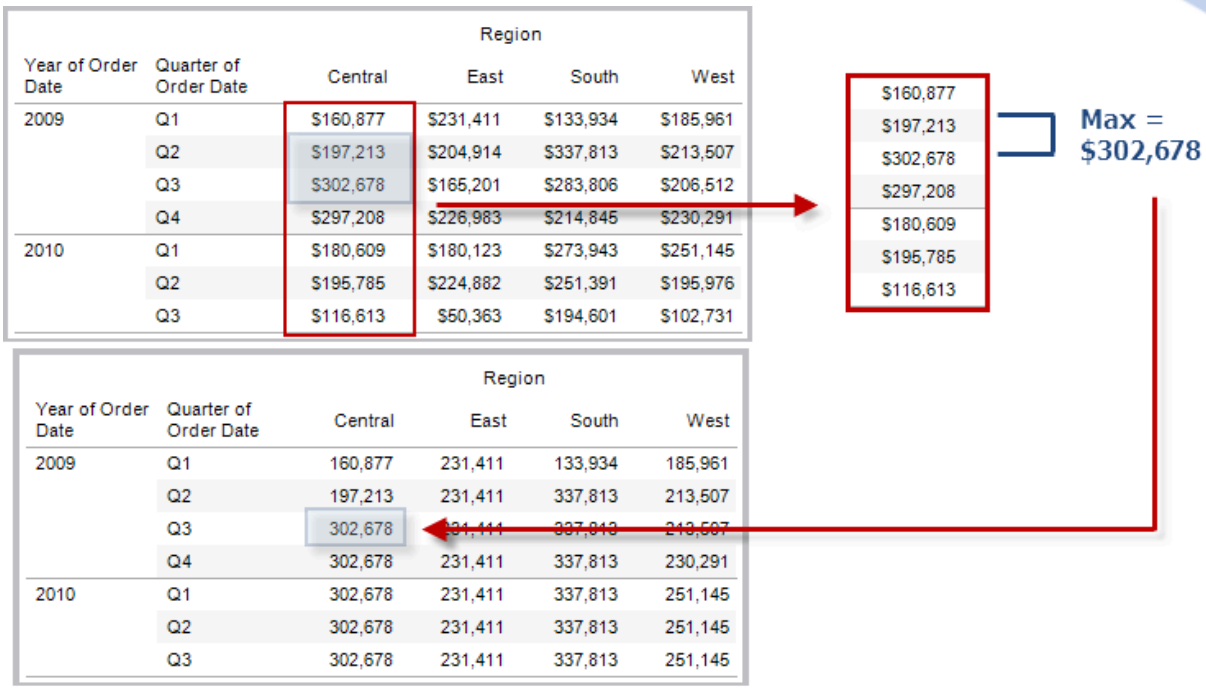
Returns the running count of the given expression, from the first row in the partition to the current row.

Example

RUNNING_COUNT(SUM([Profit])) computes the running count of SUM(Profit).

RUNNING_MAX(expression)

Returns the running maximum of the given expression, from the first row in the partition to the current row.

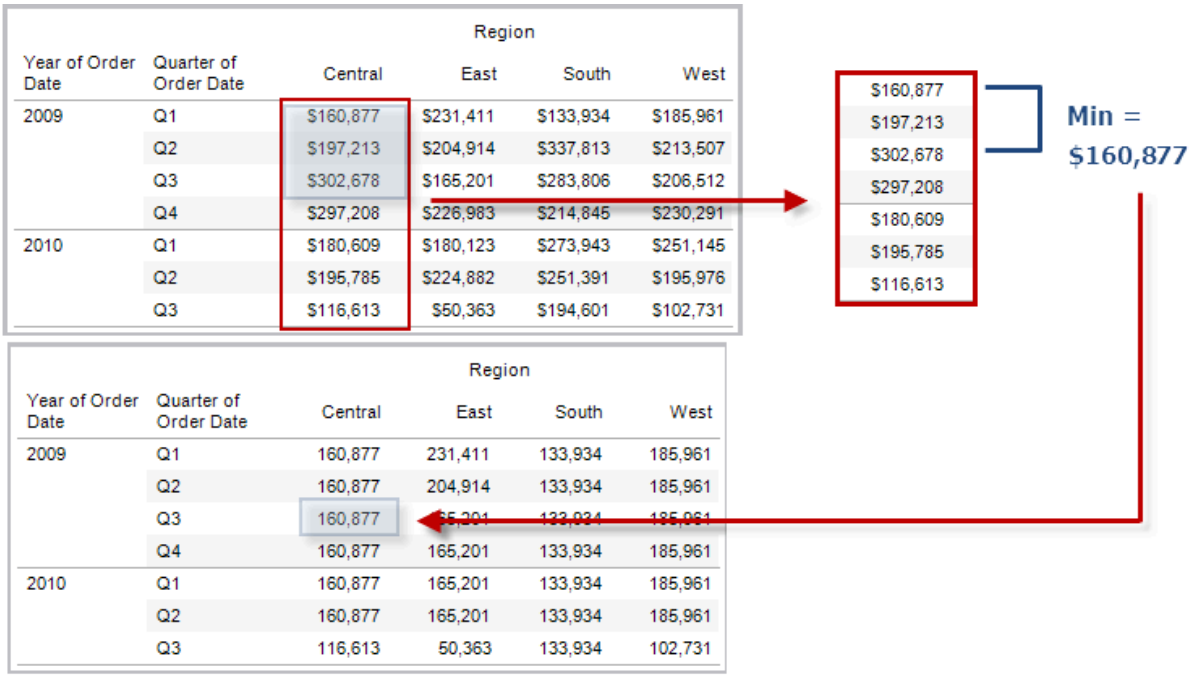


Example

RUNNING_MAX(SUM([Profit])) computes the running maximum of SUM(Profit).

RUNNING_MIN(expression)

Returns the running minimum of the given expression, from the first row in the partition to the current row.

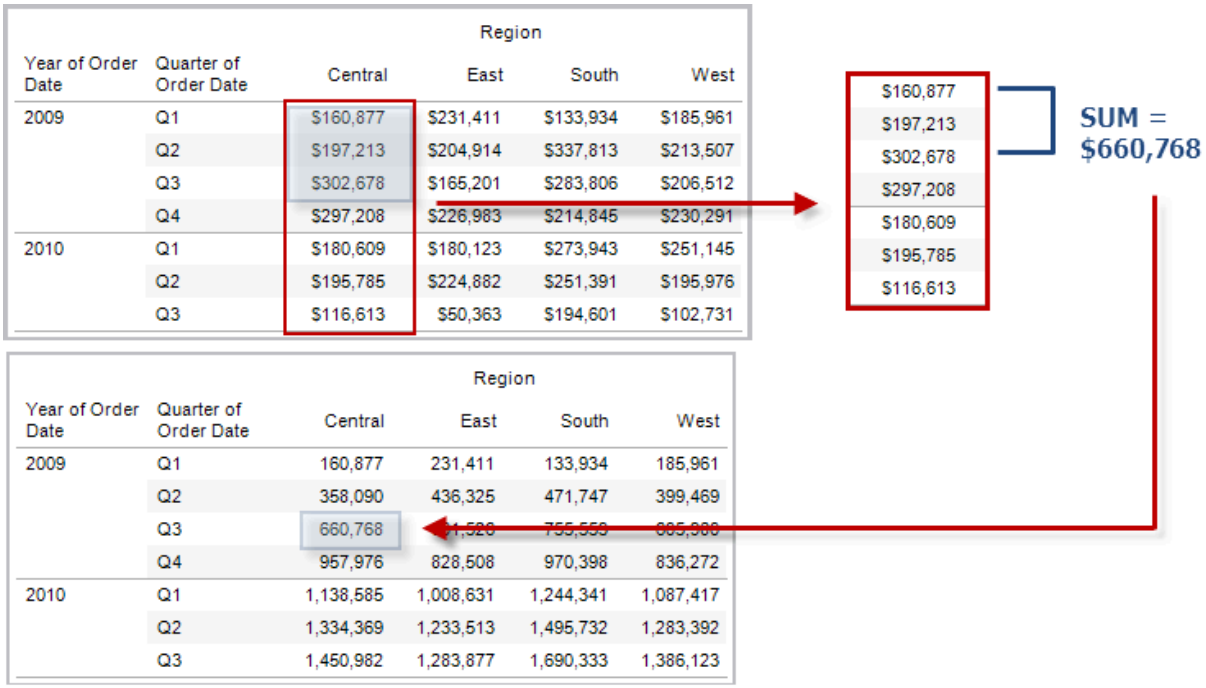


Example

RUNNING_MIN(SUM([Profit])) computes the running minimum of SUM(Profit).

RUNNING_SUM(expression)

Returns the running sum of the given expression, from the first row in the partition to the current row.



		Region			
Year of Order Date	Quarter of Order Date	Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

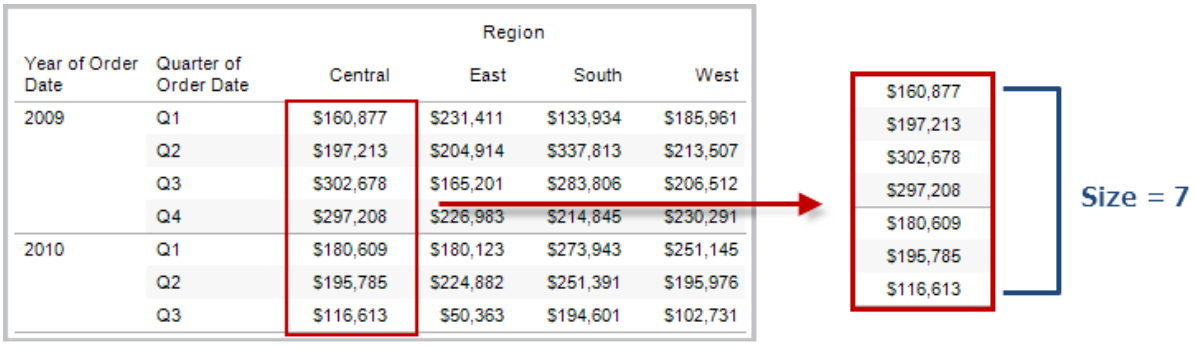
		Region			
Year of Order Date	Quarter of Order Date	Central	East	South	West
2009	Q1	160,877	231,411	133,934	185,961
	Q2	358,090	436,325	471,747	399,469
	Q3	660,768	[illegible]	[illegible]	[illegible]
	Q4	957,976	828,508	970,398	836,272
2010	Q1	1,138,585	1,008,631	1,244,341	1,087,417
	Q2	1,334,369	1,233,513	1,495,732	1,283,392
	Q3	1,450,982	1,283,877	1,690,333	1,386,123

Example

RUNNING_SUM(SUM([Profit])) computes the running sum of SUM(Profit)

SIZE()

Returns the number of rows in the partition. For example, the view below shows quarterly sales. Within the Date partition, there are seven rows so the Size() of the Date partition is 7.



		Region			
Year of Order Date	Quarter of Order Date	Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

Example

SIZE() = 5 when the current partition contains five rows.

SCRIPT_BOOL

Returns a Boolean result from the specified expression. The expression is p directly to a running analytics extension service instance.

In R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.).

In Python expressions, use _arg*n* (with a leading underscore).

Examples
In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT_BOOL("is.finite(.arg1)", SUM([Profit]))

The next example returns True for store IDs in Washington state, and False otherwise. This example could be the definition for a calculated field titled IsStoreInWA.

SCRIPT_BOOL('grepl(".*_WA", .arg1, perl=TRUE)',ATTR([Store ID]))

A command for Python would take this form:

SCRIPT_BOOL("return map(lambda x : x > 0, _arg1)", SUM([Profit]))

SCRIPT_INT

Returns an integer result from the specified expression. The expression is passed directly to a running analytics extension service instance.

In R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.)

In Python expressions, use _arg*n* (with a leading underscore).

Examples
In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT_INT("is.finite(.arg1)", SUM([Profit]))

In the next example, k-means clustering is used to create three clusters:

SCRIPT_INT('result <- kmeans(data.frame(.arg1,.arg2,.arg3,.arg4), 3);result$cluster;', SUM([Petal length]), SUM([Petal width]),SUM([Sepal length]),SUM([Sepal width]))

A command for Python would take this form:

SCRIPT_INT("return map(lambda x : int(x * 5), _arg1)", SUM([Profit]))

SCRIPT_REAL

Returns a real result from the specified expression. The expression is passed directly to a running analytics extension service instance. In

R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.)

In Python expressions, use _arg*n* (with a leading underscore).

Examples
In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT_REAL("is.finite(.arg1)", SUM([Profit]))

The next example converts temperature values from Celsius to Fahrenheit.

SCRIPT_REAL('library(udunits2);ud.convert(.arg1, "celsius", "degree_fahrenheit")',AVG([Temperature]))

A command for Python would take this form:

SCRIPT_REAL("return map(lambda x : x * 0.5, _arg1)", SUM([Profit]))

SCRIPT_STR

Returns a string result from the specified expression. The expression is passed directly to a running analytics extension service instance.

In R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.)

In Python expressions, use _arg*n* (with a leading underscore).

Examples
In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT_STR("is.finite(.arg1)", SUM([Profit]))

The next example extracts a state abbreviation from a more complicated string (in the original form 13XSL_CA, A13_WA):

SCRIPT_STR('gsub(".*_", "", .arg1)',ATTR([Store ID]))

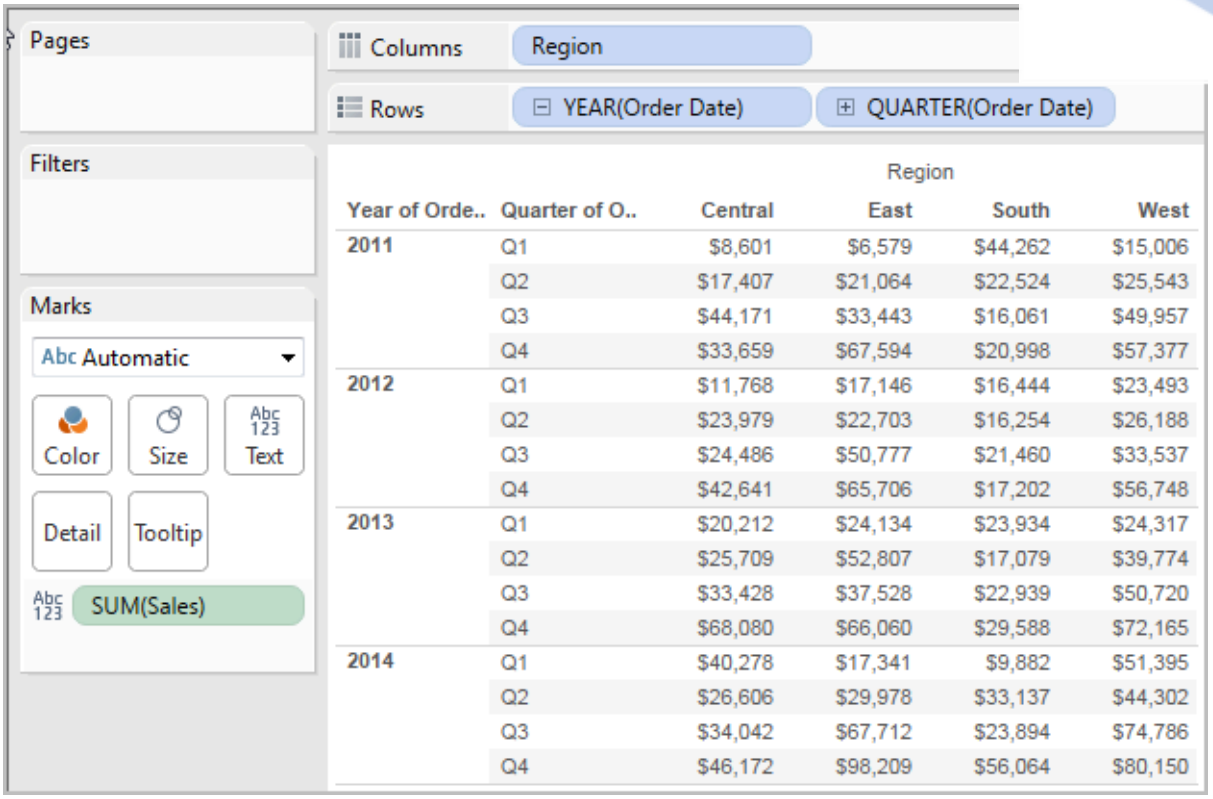
A command for Python would take this form:

SCRIPT_STR("return map(lambda x : x[:2], _arg1)", ATTR([Region]))

TOTAL(expression)

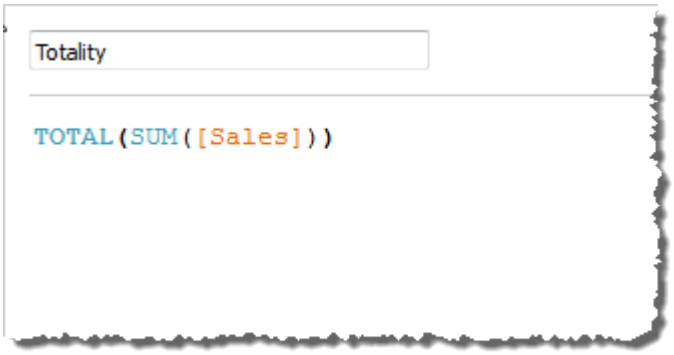
Returns the total for the given expression in a table calculation partition.

Example
Assume you are starting with this view:

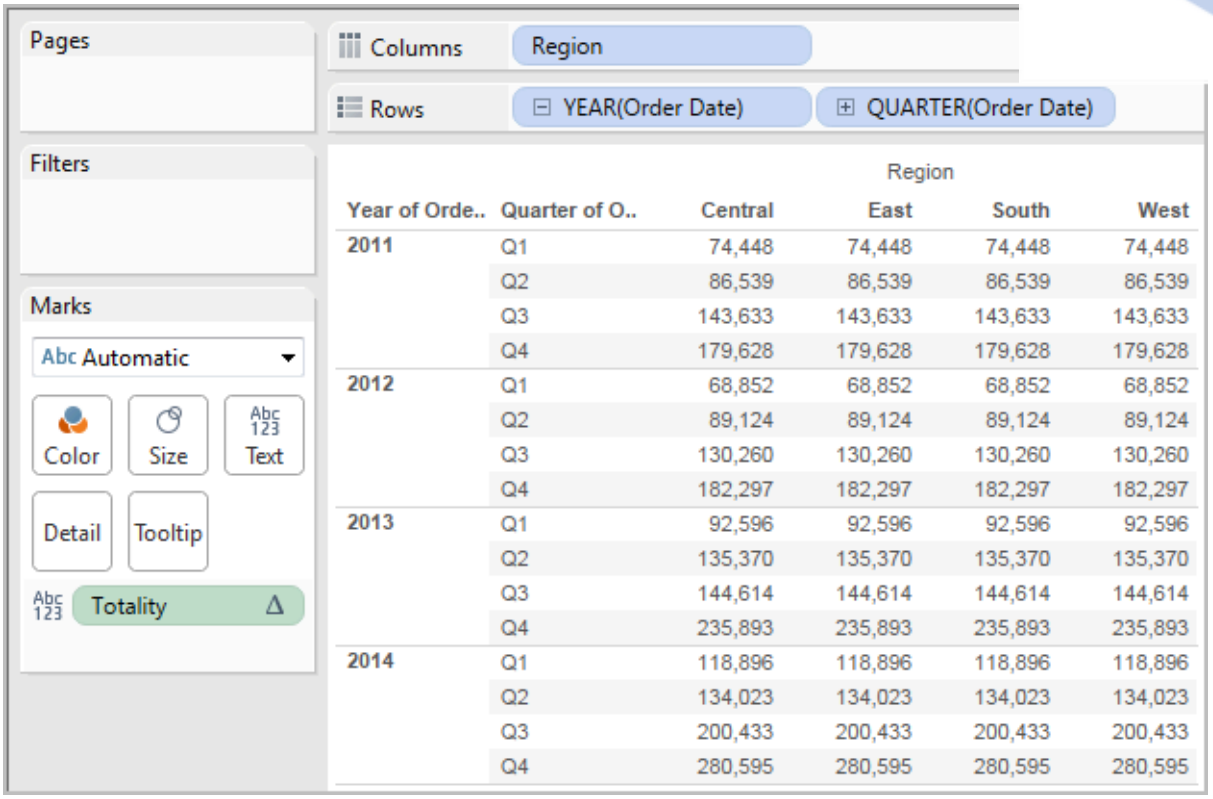


		Region			
Year of Orde..	Quarter of O..	Central	East	South	West
2011	Q1	$8,601	$6,579	$44,262	$15,006
	Q2	$17,407	$21,064	$22,524	$25,543
	Q3	$44,171	$33,443	$16,061	$49,957
	Q4	$33,659	$67,594	$20,998	$57,377
2012	Q1	$11,768	$17,146	$16,444	$23,493
	Q2	$23,979	$22,703	$16,254	$26,188
	Q3	$24,486	$50,777	$21,460	$33,537
	Q4	$42,641	$65,706	$17,202	$56,748
2013	Q1	$20,212	$24,134	$23,934	$24,317
	Q2	$25,709	$52,807	$17,079	$39,774
	Q3	$33,428	$37,528	$22,939	$50,720
	Q4	$68,080	$66,060	$29,588	$72,165
2014	Q1	$40,278	$17,341	$9,882	$51,395
	Q2	$26,606	$29,978	$33,137	$44,302
	Q3	$34,042	$67,712	$23,894	$74,786
	Q4	$46,172	$98,209	$56,064	$80,150

You open the calculation editor and create a new field which you name Totality:

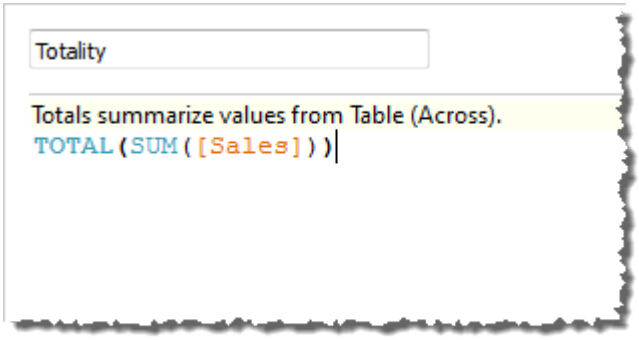


You then drop Totality on Text, to replace SUM(Sales). Your view changes such that it sums values based on the default Compute Using value:



		Region			
Year of Orde..	Quarter of O..	Central	East	South	West
2011	Q1	74,448	74,448	74,448	74,448
	Q2	86,539	86,539	86,539	86,539
	Q3	143,633	143,633	143,633	143,633
	Q4	179,628	179,628	179,628	179,628
2012	Q1	68,852	68,852	68,852	68,852
	Q2	89,124	89,124	89,124	89,124
	Q3	130,260	130,260	130,260	130,260
	Q4	182,297	182,297	182,297	182,297
2013	Q1	92,596	92,596	92,596	92,596
	Q2	135,370	135,370	135,370	135,370
	Q3	144,614	144,614	144,614	144,614
	Q4	235,893	235,893	235,893	235,893
2014	Q1	118,896	118,896	118,896	118,896
	Q2	134,023	134,023	134,023	134,023
	Q3	200,433	200,433	200,433	200,433
	Q4	280,595	280,595	280,595	280,595

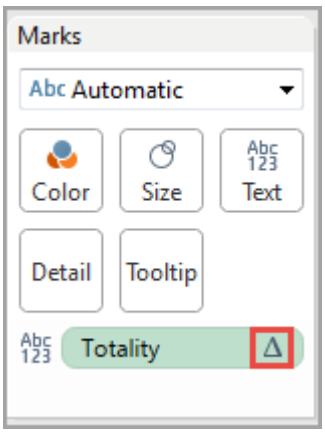
This raises the question, What is the default Compute Using value? If you right-click (Control-click on a Mac) Totality in the Data pane and choose Edit, there is now an additional bit of information available:



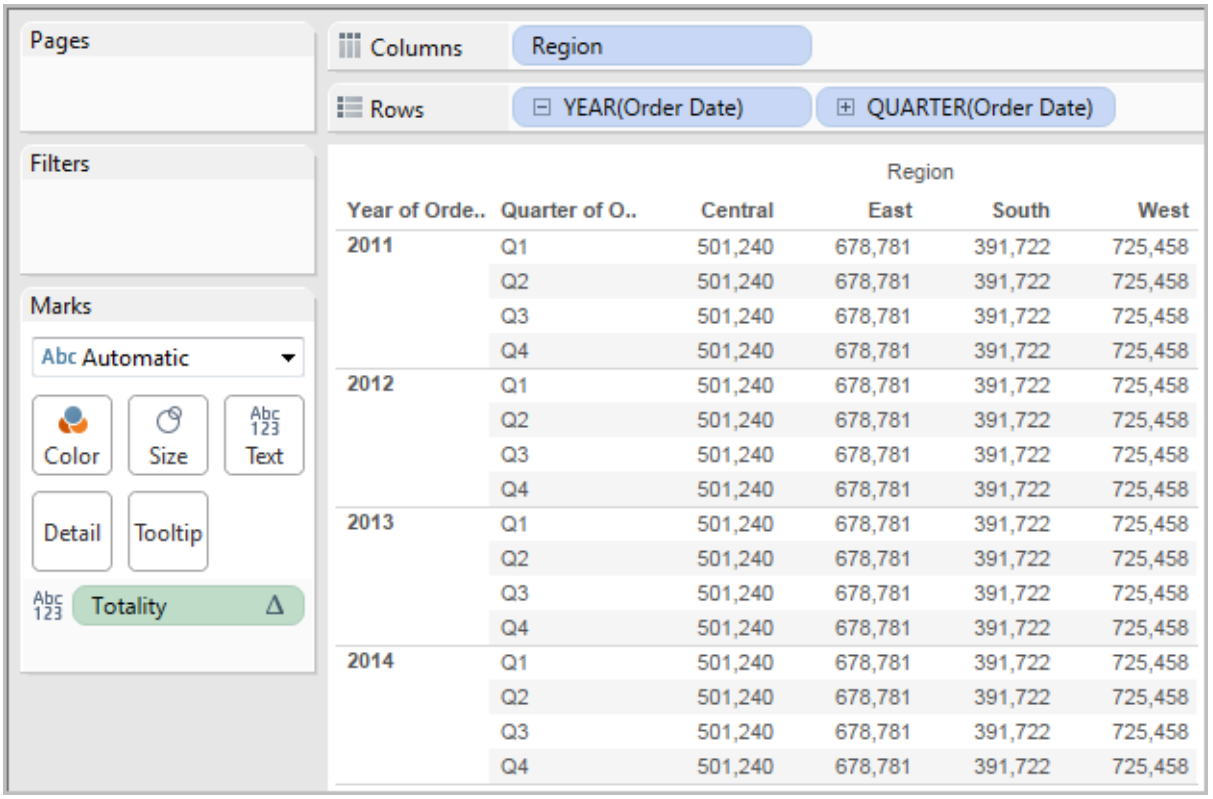
The default Compute Using value is Table (Across). The result is that Totality is summing the values across each row of your table. Thus, the value that you see across each row is the sum of the values from the original version of the table.

The values in the 2011/Q1 row in the original table were $8601, $6579, $44262, and $15006. The values in the table after Totality replaces SUM(Sales) are all $74,448, which is the sum of the four original values.

Notice the triangle next to Totality after you drop it on Text:



This indicates that this field is using a table calculation. You can right-click the field and choose Edit Table Calculation to redirect your function to a different Compute Using value. For example, you could set it to Table (Down). In that case, your table would look like this:

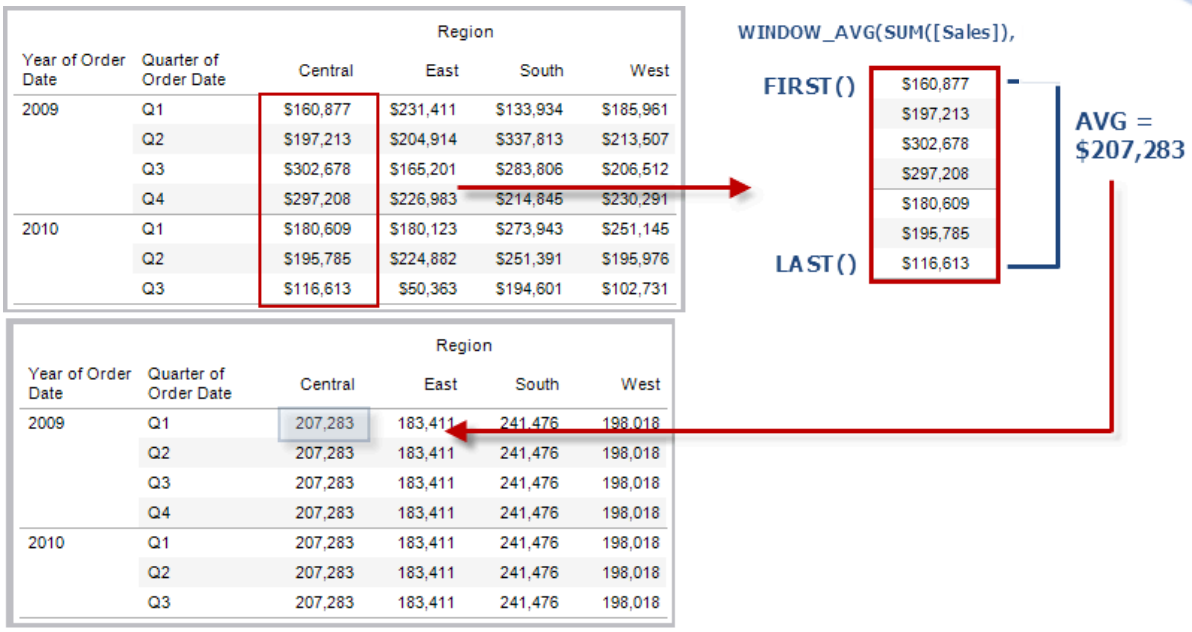


		Region			
Year of Orde..	Quarter of O..	Central	East	South	West
2011	Q1	501,240	678,781	391,722	725,458
	Q2	501,240	678,781	391,722	725,458
	Q3	501,240	678,781	391,722	725,458
	Q4	501,240	678,781	391,722	725,458
2012	Q1	501,240	678,781	391,722	725,458
	Q2	501,240	678,781	391,722	725,458
	Q3	501,240	678,781	391,722	725,458
	Q4	501,240	678,781	391,722	725,458
2013	Q1	501,240	678,781	391,722	725,458
	Q2	501,240	678,781	391,722	725,458
	Q3	501,240	678,781	391,722	725,458
	Q4	501,240	678,781	391,722	725,458
2014	Q1	501,240	678,781	391,722	725,458
	Q2	501,240	678,781	391,722	725,458
	Q3	501,240	678,781	391,722	725,458
	Q4	501,240	678,781	391,722	725,458

WINDOW_AVG(expression, [start, end])

Returns the average of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

For example, the view below shows quarterly sales. A window average within the Date partition returns the average sales across all dates.



Example

WINDOW_AVG(SUM([Profit]), FIRST()+1, 0) computes the average of SUM(Profit) from the second row to the current row.

WINDOW_CORR(expression1, expression2, [start, end])

Returns the Pearson correlation coefficient of two expressions within the window. The window is defined as offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If start and end are omitted, the entire partition is used.

The Pearson correlation measures the linear relationship between two variables. Results range from -1 to +1 inclusive, where 1 denotes an exact positive linear relationship, as when a positive change in one variable implies a positive change of corresponding magnitude in the other, 0 denotes no linear relationship between the variance, and −1 is an exact negative relationship.

There is an equivalent aggregation fuction: CORR. See Tableau Functions (Alphabetical)(Link opens in a new window).

Example

The following formula returns the Pearson correlation of SUM(Profit) and SUM(Sales) from the five previous rows to the current row.

WINDOW_CORR(SUM[Profit]), SUM([Sales]), -5, 0)

WINDOW_COUNT(expression, [start, end])

Returns the count of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

Example

WINDOW_COUNT(SUM([Profit]), FIRST()+1, 0) computes the count of SUM(Profit) second row to the current row

WINDOW_COVAR(expression1, expression2, [start, end])

Returns the *sample covariance* of two expressions within the window. The window is defined as offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end arguments are omitted, the window is the entire partition.

Sample covariance uses the number of non-null data points n - 1 to normalize the covariance calculation, rather than n, which is used by the population covariance (with the WINDOW_COVARP function). Sample covariance is the appropriate choice when the data is a random sample that is being used to estimate the covariance for a larger population.

There is an equivalent aggregation fuction: COVAR. See Tableau Functions (Alphabetical)(Link opens in a new window).

Example

The following formula returns the sample covariance of SUM(Profit) and SUM(Sales) from the two previous rows to the current row.

WINDOW_COVAR(SUM([Profit]), SUM([Sales]), -2, 0)

WINDOW_COVARP(expression1, expression2, [start, end])

Returns the *population covariance* of two expressions within the window. The window is defined as offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If start and end are omitted, the entire partition is used.

Population covariance is sample covariance multiplied by (n-1)/n, where n is the total number of non-null data points. Population covariance is the appropriate choice when there is data available for all items of interest as opposed to when there is only a random subset of items, in which case sample covariance (with the WINDOW_COVAR function) is appropriate.

There is an equivalent aggregation fuction: COVARP. Tableau Functions (Alphabetical)(Link opens in a new window).

Example

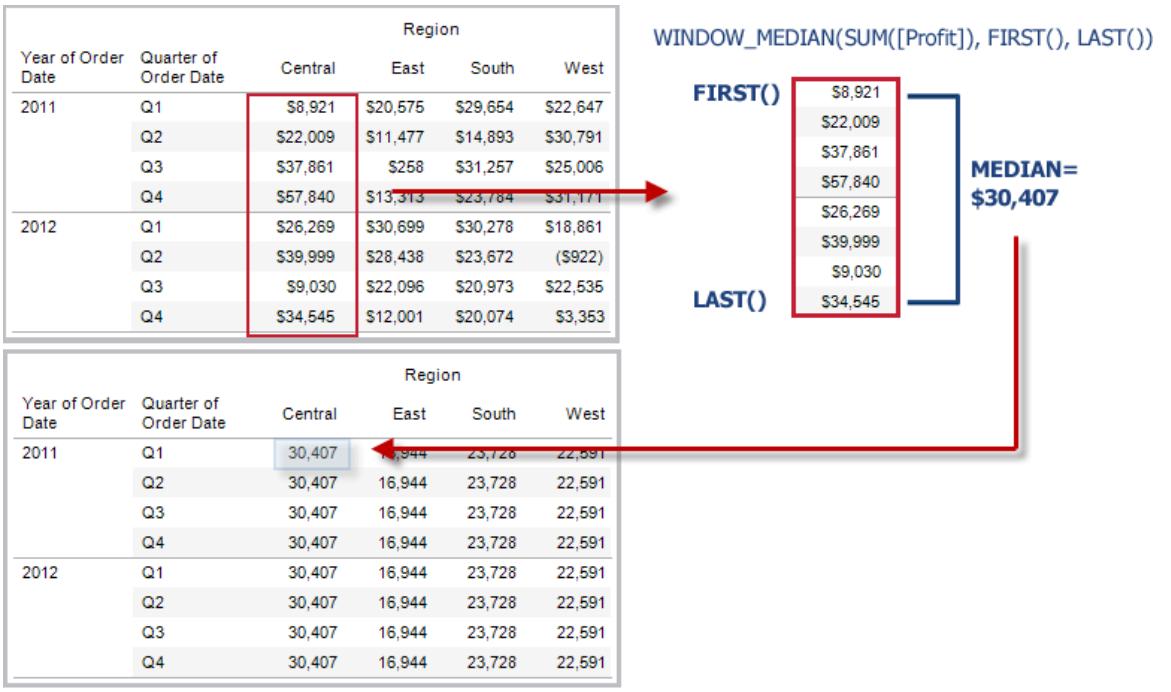
The following formula returns the population covariance of SUM(Profit) and SUM(Sales) from the two previous rows to the current row.

WINDOW_COVARP(SUM([Profit]), SUM([Sales]), -2, 0)

WINDOW_MEDIAN(expression, [start, end])

Returns the median of the expression within the window. The window is de
means of offsets from the current row. Use FIRST()+n and LAST()-n for offs
first or last row in the partition. If the start and end are omitted, the entire partition is used.

For example, the view below shows quarterly profit. A window median within the Date partition returns the median profit across all dates.



Year of Order Date	Quarter of Order Date	Region: Central	East	South	West
2011	Q1	$8,921	$20,575	$29,654	$22,647
	Q2	$22,009	$11,477	$14,893	$30,791
	Q3	$37,861	$258	$31,257	$25,006
	Q4	$57,840	$13,313	$23,784	$31,171
2012	Q1	$26,269	$30,699	$30,278	$18,861
	Q2	$39,999	$28,438	$23,672	($922)
	Q3	$9,030	$22,096	$20,973	$22,535
	Q4	$34,545	$12,001	$20,074	$3,353

Year of Order Date	Quarter of Order Date	Region: Central	East	South	West
2011	Q1	30,407	16,944	23,728	22,591
	Q2	30,407	16,944	23,728	22,591
	Q3	30,407	16,944	23,728	22,591
	Q4	30,407	16,944	23,728	22,591
2012	Q1	30,407	16,944	23,728	22,591
	Q2	30,407	16,944	23,728	22,591
	Q3	30,407	16,944	23,728	22,591
	Q4	30,407	16,944	23,728	22,591

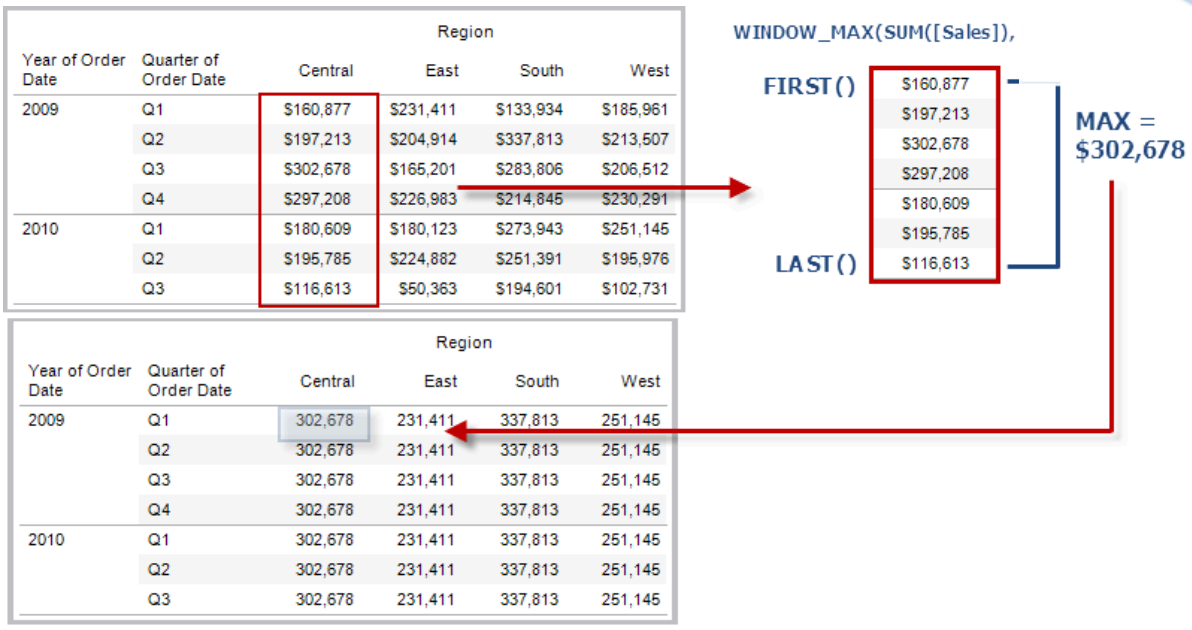
Example

WINDOW_MEDIAN(SUM([Profit]), FIRST()+1, 0) computes the median of SUM(Profit) from the second row to the current row.

WINDOW_MAX(expression, [start, end])

Returns the maximum of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

For example, the view below shows quarterly sales. A window maximum within the Date partition returns the maximum sales across all dates.



		Region			
Year of Order Date	Quarter of Order Date	Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

		Region			
Year of Order Date	Quarter of Order Date	Central	East	South	West
2009	Q1	302,678	231,411	337,813	251,145
	Q2	302,678	231,411	337,813	251,145
	Q3	302,678	231,411	337,813	251,145
	Q4	302,678	231,411	337,813	251,145
2010	Q1	302,678	231,411	337,813	251,145
	Q2	302,678	231,411	337,813	251,145
	Q3	302,678	231,411	337,813	251,145

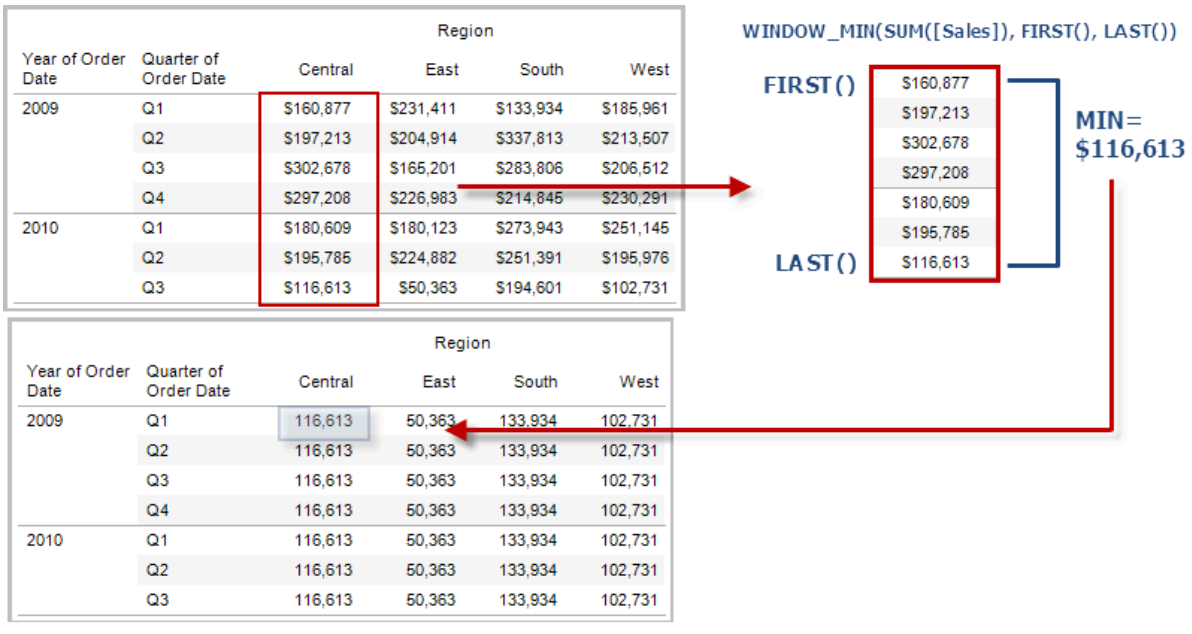
Example

WINDOW_MAX(SUM([Profit]), FIRST()+1, 0) computes the maximum of SUM(Profit) from the second row to the current row.

WINDOW_MIN(expression, [start, end])

Returns the minimum of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

For example, the view below shows quarterly sales. A window minimum within the Date partition returns the minimum sales across all dates.



		Region			
Year of Order Date	Quarter of Order Date	Central	East	South	West
2009	Q1	$160,877	$231,411	$133,934	$185,961
	Q2	$197,213	$204,914	$337,813	$213,507
	Q3	$302,678	$165,201	$283,806	$206,512
	Q4	$297,208	$226,983	$214,845	$230,291
2010	Q1	$180,609	$180,123	$273,943	$251,145
	Q2	$195,785	$224,882	$251,391	$195,976
	Q3	$116,613	$50,363	$194,601	$102,731

		Region			
Year of Order Date	Quarter of Order Date	Central	East	South	West
2009	Q1	116,613	50,363	133,934	102,731
	Q2	116,613	50,363	133,934	102,731
	Q3	116,613	50,363	133,934	102,731
	Q4	116,613	50,363	133,934	102,731
2010	Q1	116,613	50,363	133,934	102,731
	Q2	116,613	50,363	133,934	102,731
	Q3	116,613	50,363	133,934	102,731

Example

WINDOW_MIN(SUM([Profit]), FIRST()+1, 0) computes the minimum of SUM(Profit) from the second row to the current row.

WINDOW_PERCENTILE(expression, number, [start, end])

Returns the value corresponding to the specified percentile within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

Example

WINDOW_PERCENTILE(SUM([Profit]), 0.75, -2, 0) returns the 75th percentile for SUM(Profit) from the two previous rows to the current row.

WINDOW_STDEV(expression, [start, end])

Returns the sample standard deviation of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

Example

WINDOW_STDEV(SUM([Profit]), FIRST()+1, 0) computes the standard deviation of SUM(Profit) from the second row to the current row.

WINDOW_STDEVP(expression, [start, end])

Returns the biased standard deviation of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

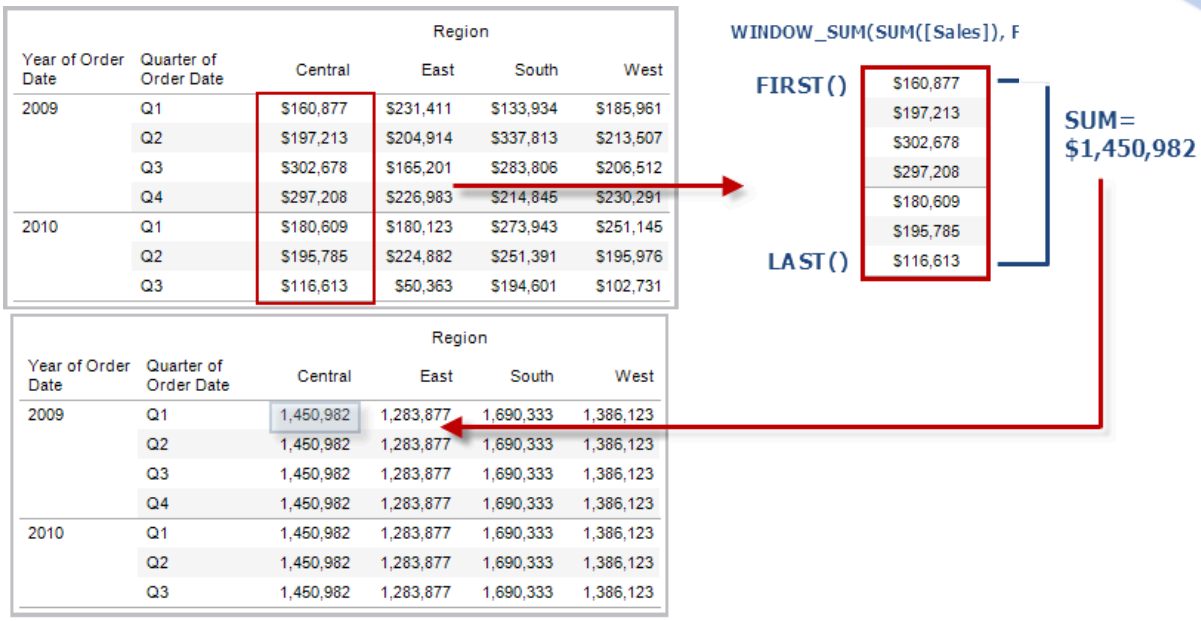
Example

WINDOW_STDEVP(SUM([Profit]), FIRST()+1, 0) computes the standard deviation of SUM(Profit) from the second row to the current row.

WINDOW_SUM(expression, [start, end])

Returns the sum of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

For example, the view below shows quarterly sales. A window sum computed within the Date partition returns the summation of sales across all quarters.



Example

WINDOW_SUM(SUM([Profit]), FIRST()+1, 0) computes the sum of SUM(Profit) from the second row to the current row.

WINDOW_VAR(expression, [start, end])

Returns the sample variance of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

Example

WINDOW_VAR((SUM([Profit])), FIRST()+1, 0) computes the variance of SUM(Profit) from the second row to the current row.

WINDOW_VARP(expression, [start, end])

Returns the biased variance of the expression within the window. The window is defined by means of offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

Example

WINDOW_VARP(SUM([Profit]), FIRST()+1, 0) computes the variance of SUM(Profit) from the second row to the current row.

Tableau File Extensions

1. Tableau Workbook (.twb)

The Tableau Workbook file type is the one that you will use the most when working in Tableau. This file type has the extension .twb and is set as default for the users. As we know, a workbook in Tableau is a file that contains sheets, dashboards, etc. So, this particular Tableau file type contains information about worksheets and dashboards present within a workbook. All the information regarding the fields, aggregation types, styles, formatting, filters, etc is present in these files.

One important thing to note about this Tableau file type is that we can only create them if we are using live data connections and share them only with the users having access to the same live data connection. The .twb files also contain metadata related to the existing data connection. However, a .twb file does not contain actual data concerning the workbook.

To create .twb file, go to the data source control panel of the active data connection and then go to File option (present on the toolbar) and select Save As. Then, you can select the file type as Tableau Workbook from the Save As Type drop-down list.

2. Tableau Packaged Workbook (.twbx)

The Tableau Packaged Workbook file type has both information about the constituents of a workbook and the data extracted from the data source. The data extract taken from the source is present in the form of a .tde file. Tableau Packaged Workbook files have an extension .twbx. You can use a .twbx file type in place of a .twb (Tableau Workbook) file when you wish to share a workbook with a user who does not have access to the live data connection. Thus, in this case, you need to have a file which contains data extracted from the source along with the other information about the workbook.

The Tableau Packaged Workbook files can also contain information about attached images or customized geocoding. To create a .twbx file, go to File then select Save As and then select the .twbx option from the drop-down list given there.

3. Tableau Data Source (.tds)

The Tableau Data Source files are files that contain all the necessary information regarding a data connection made in Tableau. When we set up a fresh connection to a data source we make a lot of modifications in it as per our requirements such as setting data types, aggregations, custom fields, etc. The Tableau Data Source files contain all the

required information on setting up a data connection along with the meta specific modifications made by the users.

The .tds file helps in saving information on data connections with custom fields and table joins. However, this Tableau file type only saves the information needed to establish a connection with a data source but not the actual data. Thus, such files can be used to share information between users having access to the same data source.

To create a .tds file, go to the Data tab on the toolbar. Then choose a data source that you wish to connect to and select Add To Saved Data Source option. After this, save that file as Tableau Data Source file.

4. Tableau Packaged Data Source (.tdsx)

A Tableau Packaged Data Source file is a file that contains information of a data source connection along with the data extracted from that source. The extracted data is saved as a .tde file and the information on data source as .tds file (like we saw above). The extracted data can be from any local file such as a text file, extract files (.hyper or .tde), Excel files, Access files, etc.

However, the extension of a Tableau Packaged Data Source file has the extension as .tdsx. The Tableau Packaged Data Source files are used when we want to share data and other relevant information about a data source with a user who does not have access to the data source and its data.

To create a .tdsx file, go to the Data tab on the toolbar. Then choose a data source that you wish to connect to and select Add To Saved Data Source option. After this, save that file as Tableau Packaged Data Source file.

5. Tableau Data Extract (.tde)

The Tableau Data Extract files have the extension .tde. These Tableau file types only contain a local copy of the entire or a subset of data from its source. It is important to note here that the .tde files do not contain a file path or information about the data source, workbooks, dashboards, etc. Tableau Data Extract files are important and useful as they are highly compressed and optimized to improve Tableau's performance (especially when you are using a slow data connection). You can use .tde files for offline work as well. One noted drawback of such Tableau file types is that the data in it cannot

be refreshed automatically as and when it refreshes at the source. Howeve a few step process to refresh the data present as an extract in your .tde files

To create a Tableau Data Extract (.tde) file, go to the Data tab present on the top left of the Tableau toolbar. Select a data source and click on the Extract Data option. After this, you can either select fields from the data source that you wish to extract or just click on Extract to create a data extract file (.tde) of the entire data set present at the data source.

6. Tableau Bookmark (.tbm)

Files with the extension .tbm are Tableau Bookmark files. These Tableau file types are most commonly used to save worksheets and share them with others so that they can use it in their workbooks without having to create a new worksheet from scratch.

To create a .tbm file, go to the Windows option present on the toolbar. From there, select Bookmark and then click on Create Bookmark. This will create a .tbm file of the active worksheet.

However, with the newer version of Tableau that was released, the use of Bookmark files has gone down. In Tableau versions 8.1. and later, we can directly copy and paste worksheets from one workbook to the other without having to create a .tbm file for it.

7. Tableau Map Source (.tms)

A Tableau Map Source file contains information about maps and its elements for use in Tableau. The extension of such files is .tms. As per the default settings, Tableau will fetch map details like background and other layers from a certain map server or provider. In Tableau, you have the option to add map details from a WMS server of your choice or a custom map from Mapbox. Once you create a map file (.tms) of your preference, Tableau will fetch map details from that file instead of the default one and load map images and information accordingly. You can also share these .tms users in your group for others to use.

To create a Tableau Map Source (.tms) file, click on Map from the toolbar. Then go to Background Maps and select WMS Server from Map Services> Add. After adding the map server of your choice, you can export it to your local desktop by selecting an Export option from the WMS Server connections window. To use this map in future, add the .tms file into the Tableau Repository in the Map sources directory.

8. Tableau Preference (.tps)

A Tableau Preference file contains all the information related to a customized color palette. You can create a custom color palette or a theme and save it as a .tps file so that you can use it all over the workbook uniformly at once. The Tableau Preference files have the extension as .tps and exist in XML format. These Tableau Preference files are present in My Tableau Repository.

Sorting in Tableau

In Tableau, sorting refers to the process of arranging data in a specific order within a visualization. Sorting allows you to present your data in a more organized and meaningful way, making it easier for viewers to understand and analyze.

Tableau provides various options for sorting data, including data source order, alphabetic sorting, field sorting, manual sorting, and nested sorting. Let's take a closer look at each type:

Data Source Order: This is the default sorting option in Tableau. It preserves the original order of the data as it appears in the underlying data source. The data is displayed in the order it was imported or extracted.

Alphabetic Sorting: This type of sorting arranges data in alphabetical order based on the selected field. For example, if you have a field named "City," you can sort the data alphabetically to display cities in ascending or descending order.

Field Sorting: Field sorting allows you to sort the data based on a specific field's values. You can sort the field in ascending or descending order. This is useful when you want to sort your data based on a numerical field, such as sales amounts or dates.

Manual Sorting: Manual sorting gives you the ability to define a custom sort order by manually rearranging the values in a field. This is helpful when you want to prioritize or emphasize certain values in the visualization. You can drag and drop the values to rearrange them in the desired order.

Nested Sorting: Nested sorting allows you to sort data based on multiple fields. You can specify the primary sort field, and within each value of the primary sort field, you can further sort the data using a secondary sort field. This type of sorting helps in creating more complex sorting hierarchies.

To apply sorting in Tableau, you can right-click on a field or axis in your visualization and select the "Sort" option. From there, you can choose the desired sorting type and configure the sort order.

By using these sorting options effectively, you can organize and present your data in a way that best suits your analysis and visualization needs in Tableau.

Dashboard Objects

Different dashboard objects are as follows

Vertical Container: A vertical container is a layout object that helps organize other dashboard objects vertically. It allows you to group and stack multiple objects, such as worksheets, filters, or images, in a vertical arrangement.

Horizontal Container: Similar to a vertical container, a horizontal container is a layout object that organizes other dashboard objects horizontally. It enables you to place and align objects side by side within the container.

Text: The text object allows you to add static text to your dashboard. You can use it to provide titles, subtitles, descriptions, or any other textual information to enhance the understanding of the data being presented.

Extension: Extensions are add-on components or custom functionalities developed by Tableau partners or the Tableau community. They extend the capabilities

of Tableau by providing additional visualizations, integrations, or features t added to your dashboards.

Ask Data: Ask Data is a natural language processing (NLP) feature in Tableau that allows users to ask questions about their data using plain English sentences. It enables users to explore data and receive immediate visual responses directly within a dashboard.

Data Story: Data Stories in Tableau provide a guided narrative flow for your data analysis. It allows you to combine visualizations, text, and media to create a compelling story that highlights key insights and findings from your data.

Image: The image object lets you add pictures, logos, or other visual elements to your dashboard. You can insert images from local files or web URLs to provide additional context or branding.

Blank: The blank object is an empty space or placeholder on the dashboard where you can add custom objects, shapes, or annotations. It provides flexibility in designing your dashboard layout.

Web Page: The web page object allows you to embed an entire web page within your dashboard. You can display web content, such as live web applications, interactive reports, or external websites, directly within your Tableau dashboard.

Navigation: Navigation objects, like buttons or hyperlinks, enable users to interact with the dashboard and navigate between different views or sections. They provide a way to create intuitive user experiences and guide users through the content.

Download: Download objects allow users to export or download the data or visualizations displayed in the dashboard. It provides options to save the data as CSV, PDF, or other file formats for further analysis or sharing.

Add Filters: Add filters objects are interactive controls that allow use

filters directly within the dashboard. Users can select specific values or ranges to filter the data and explore different perspectives of the information presented.

These dashboard objects offer a wide range of options for designing interactive, informative, and visually appealing dashboards in Tableau. They provide flexibility and customization to meet your specific data presentation and analysis requirements.

Visualizations in Tableau

Butterfly chart

1) Ensure you have two measures: In order to make a butterfly chart, you need at least two measures to compare. Let's consider "Sales 2021" and "Sales 2022" as our measures.

2) Calculate the sales for each year: Create calculated fields for "Sales 2021" and "Sales 2022" using the following formulas:

Sales 2021: SUM(IF YEAR([Order Date]) = 2021 THEN [Sales] END)

Sales 2022: SUM(IF YEAR([Order Date]) = 2022 THEN [Sales] END)

3) Add the measures to rows: Drag both measures onto the Rows shelf, which will create two bar charts side by side.

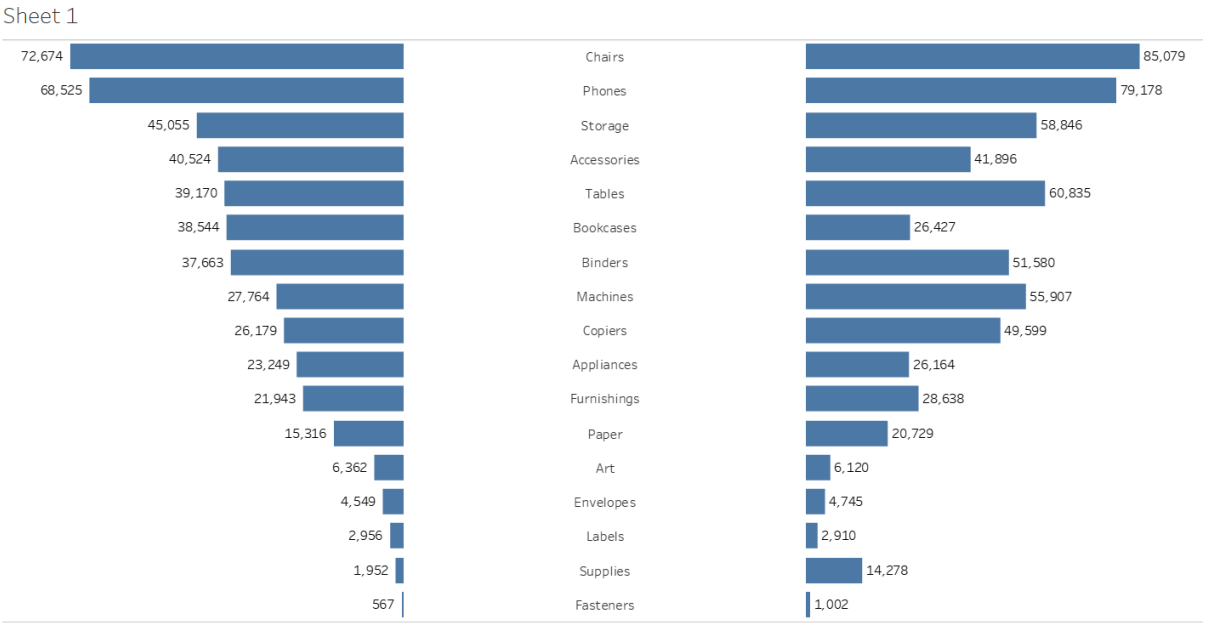
4) Reverse the axis and hide it: Right-click on the axis of the first bar chart, select "Edit Axis," and check the "Reversed" option. Then, hide the axis by unchecking "Show Header."

5) Add a placeholder for category dimensions: In the middle of the two bar charts, add a zero (0) as a placeholder. This will act as a reference point for the category dimensions.

6) Display category dimension using the placeholder: Convert the 0 from au
text by right-clicking on the zero and selecting "Convert to" > "Dimension (L
Then, drag the category dimension to the Text shelf to display it at the start.

7) Sort the chart: Sort the chart by the category dimension in ascending or descending order to ensure a proper arrangement of the bars.

8) Finalize the chart: Customize the appearance of the chart by adjusting colors, formatting labels, adding legends, or any other desired formatting options.



Sunburst Chart

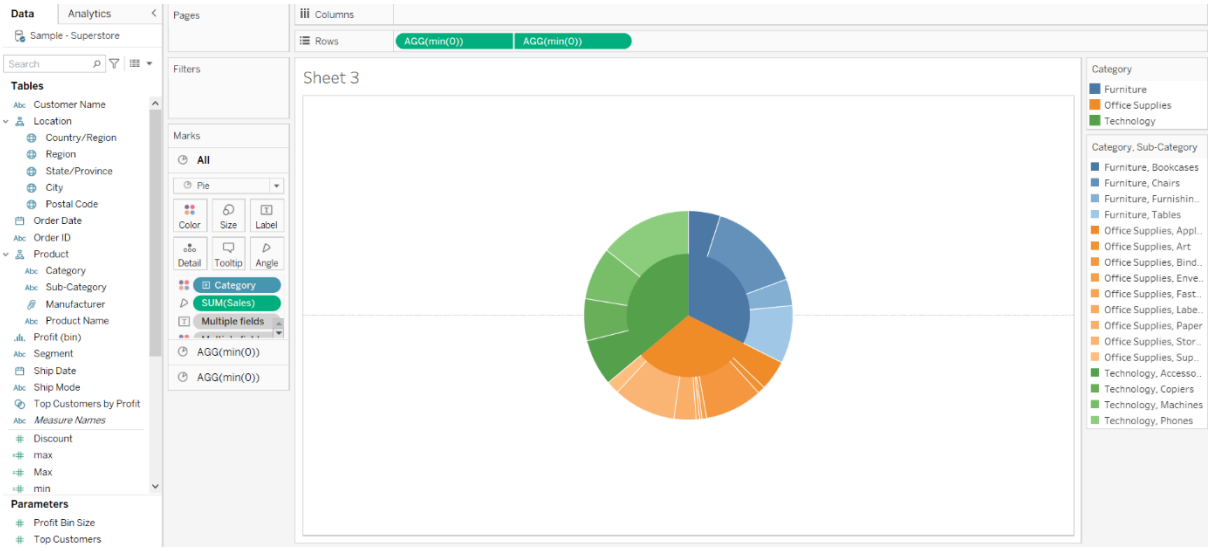
Sunburst chart is used to visualize hierarchical data through a series of rings that are partitioned into different categories. Each level of the hierarchy is represented by one ring with the inner most ring being the top of the hierarchy.

1.Set the chart type of the mark card as a pie chart and assign the category variable to represent the colors. This will create a simple pie chart, with the sales values determining the angles.

2.Include the "min(0)" function twice in the row section for the two charts that will be displayed on the same page. Remember that the first "min(0)" function has already been applied to the category variable.

3.For the second "min(0)" function, assign the subcategory variable to repr
colors, using the same color palette as the one used for the category variabl
chart.

4.Combine the first and second "min(0)" functions as dual axes and adjust their sizes accordingly. Make the subcategory "min(0)" function larger to appear outside the chart, while keeping the category "min(0)" function smaller.



Gradient Bar Chart:

A gradient bar chart in Tableau is often used to visualize the distribution or comparison of a measure across different categories. It provides a visual representation of the magnitude or intensity of the measure using a color gradient.

Steps:

1. Place the subcategory field in the Rows shelf and the sum of sales field in the Columns shelf.

2. Change the mark type to Line to create the bar chart effect.

3. Add a constant value of 0 to the Columns shelf to set the minimum value.

4. Place the Measure Names field in the Filters shelf, Colors shelf, and Path shelf.

5. Assign colors to the Measure Names field to create the gradient effect.

6. Adjust the line size to visualize the bars as desired.

7. Set the background color to black for a more impactful visualization.

8. Format the grid lines in the chart for better readability.