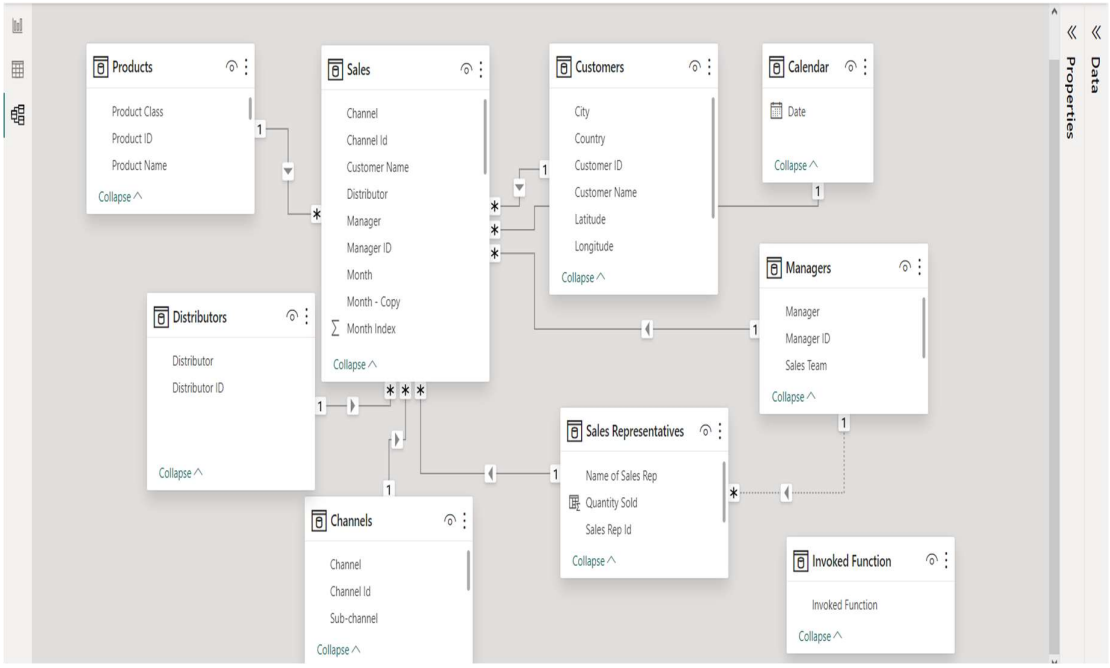
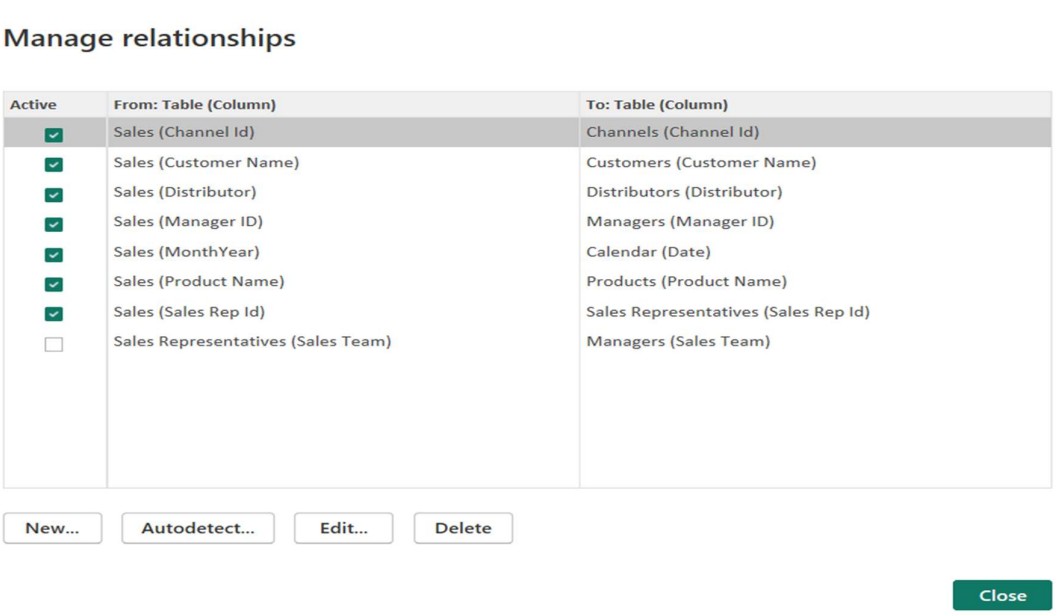
Pharma Data Analysis Answers

1. Schema Design: Given the provided dataset, create a Power BI data model with appropriate tables and relationships, considering the Distributor, Customer Name, City, and other relevant columns.



2. Relationships: Establish the necessary relationships between the tables in your data model. For instance, connect the "Sales" table to the "Customers" table.

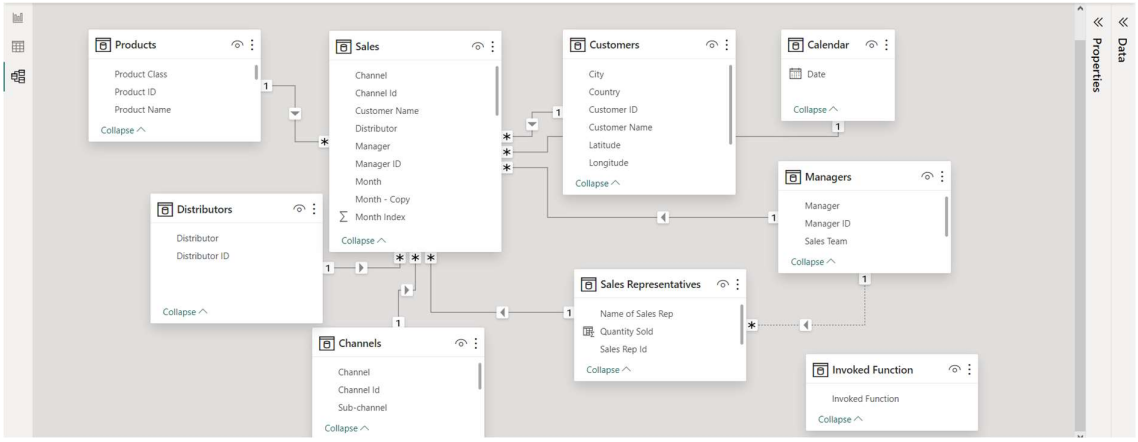


3. Role-Playing Dimensions: In your data model, demonstrate how you'd handle role-playing dimensions for "Sales Rep" and "Manager."

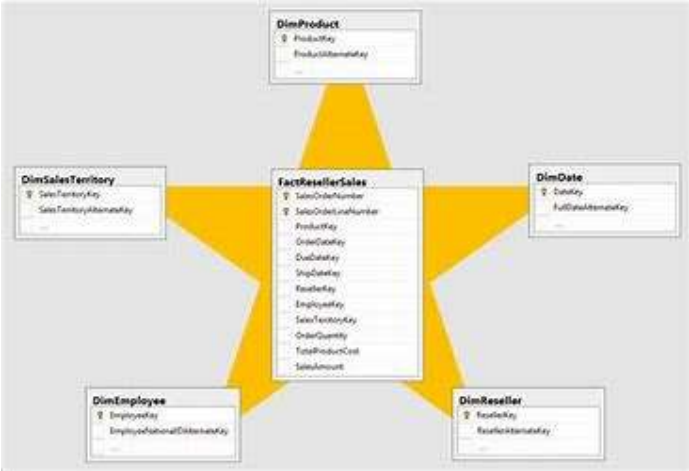
   To handle role-playing dimensions for

   A) Sales Rep, I duplicated the Sales table, removed all columns except Name of the Sales Rep and Sales Team, added an index column for Sales Rep Id, added a new calculated column for Quantity Sold, added a new conditional column in Sales Table using the Ids in the Sales Rep table, established a relationship with the Sales table based on Sales Rep Id.

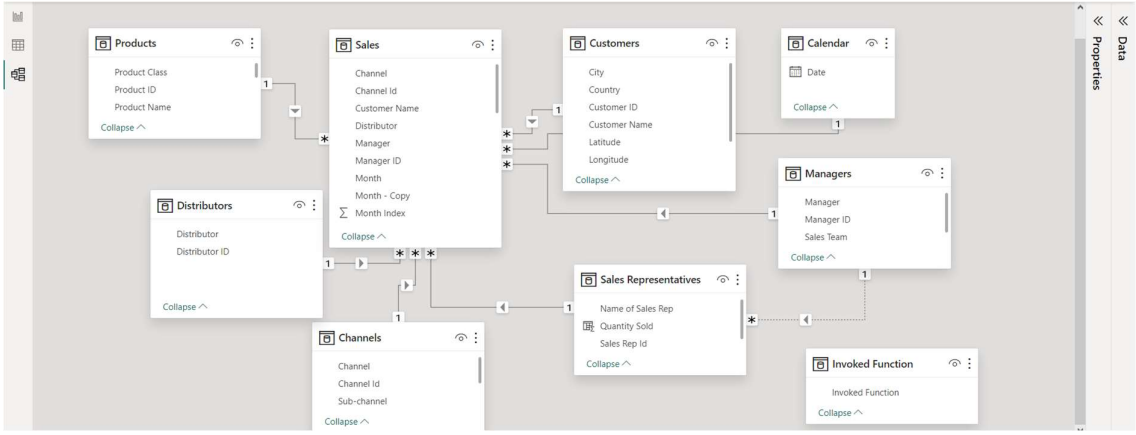
   B) Manager, I duplicated the Sales table, removed all columns except Name of the Manager and Sales Team, added an index column for Manager Id, added a new conditional column in Sales Table using the Ids in the Manager table, established a relationship with the Sales table based on Manager Id.



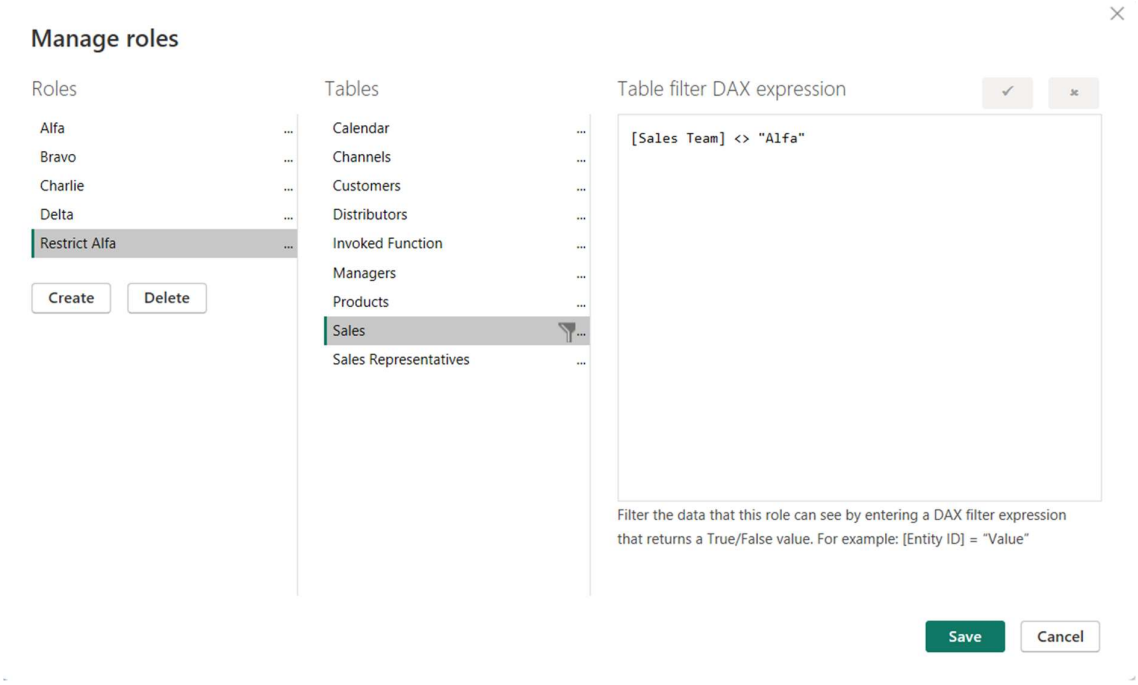
4. Schemas: Build a star schema based on the data and explain how your schema design helps optimize report performance.



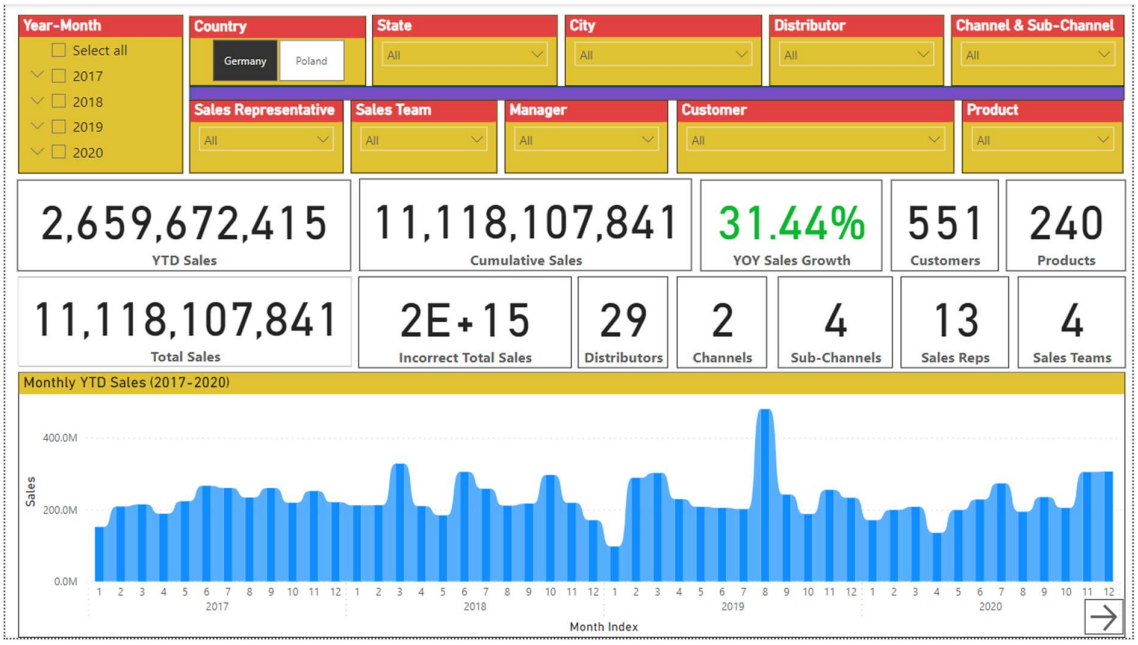
A **star schema** is a type of database schema that is used to organize data in a way that optimizes query performance. It is called a star schema because the diagram of the schema looks like a star, with a central fact table and multiple dimension tables radiating out from it.
The star schema is designed to reduce the number of joins required to retrieve data, which can improve query performance. By organizing data into dimensions and facts, the star schema can help to minimize the amount of data that needs to be scanned to answer a query.
In addition to reducing the number of joins, the star schema can also help to improve query performance by reducing the amount of data that needs to be aggregated. By pre-aggregating data in the fact table, the star schema can help to minimize the amount of data that needs to be processed to answer a query.



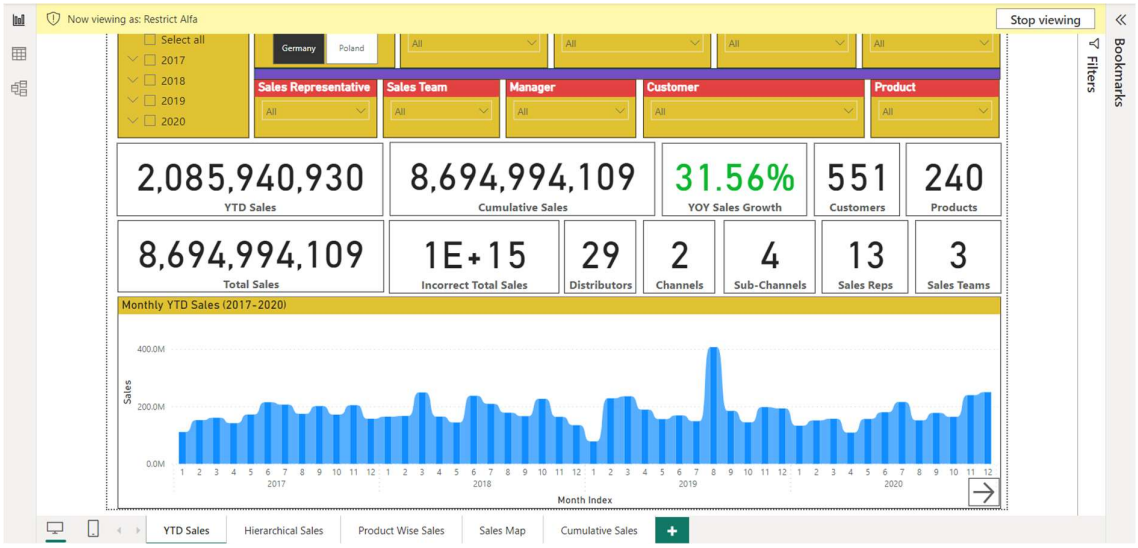
5. Row-Level Security: Set up row-level security in your data model, restricting access for a specific sales team. Show which measures are affected.



Viewing without selecting any role



Viewing as ‘Restrict Alfa’.

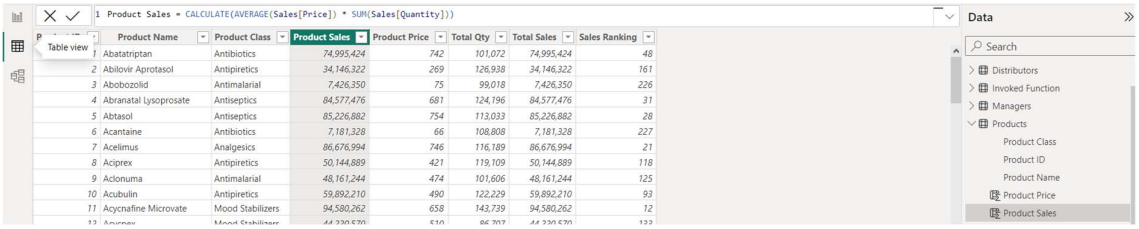


After applying Row Level Security to restrict team Alfa, all the measures are affected including YTD Sales, Cumulative Sales, Total Sales, YOY Sales Growth, Customers, Products, Distributors, Channels, Sub-channels, Sales Reps and Sales Teams.

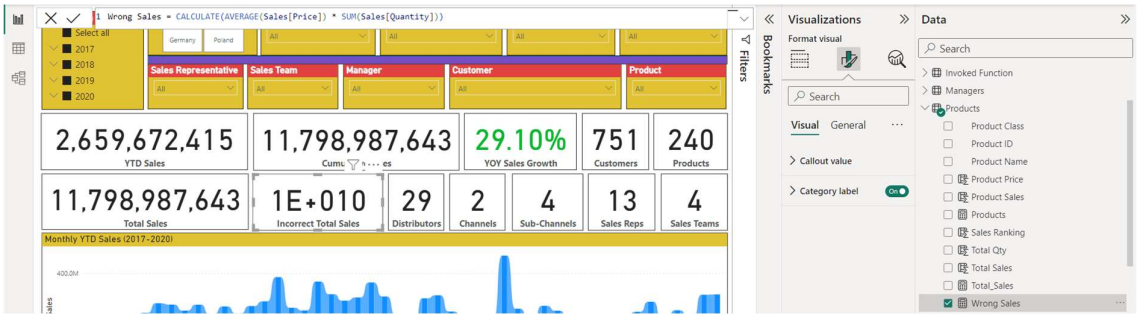
6. Calculated Columns vs. Measures: Calculate the total sales for each product both as a calculated column and a measure. Compare the results and explain the differences.

   We can calculate the total sales for each product only by using a calculated column as a calculated column performs the same calculation for each row. We can see the product wise sales in the Product Sales Calculated Column shown below.

```
Product Sales = CALCULATE(AVERAGE(Sales[Price]) * SUM(Sales[Quantity]))
```

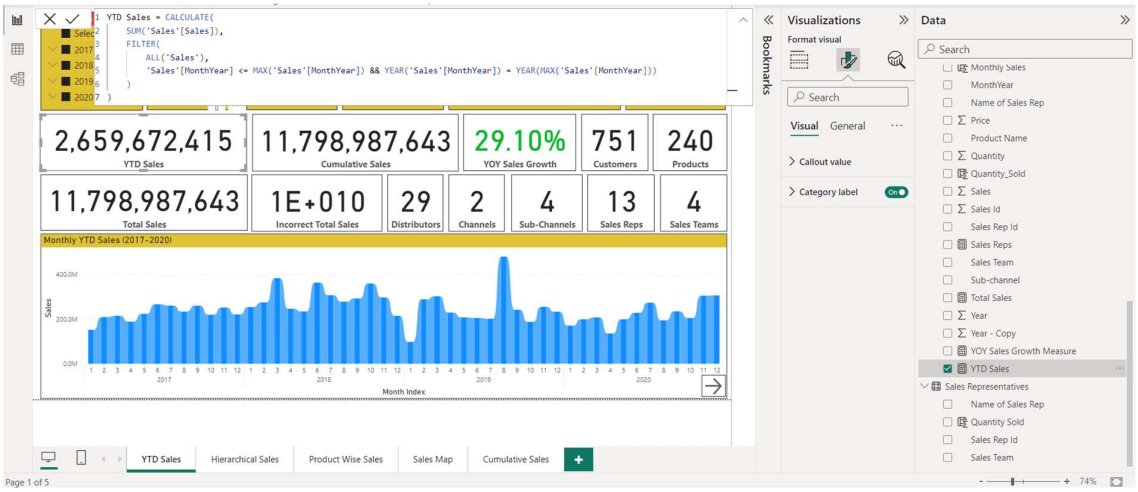


   However, a measure performs the same calculation for the whole products data and hence results in an incorrect answer as shown below in the **Incorrect Total Sales card.**

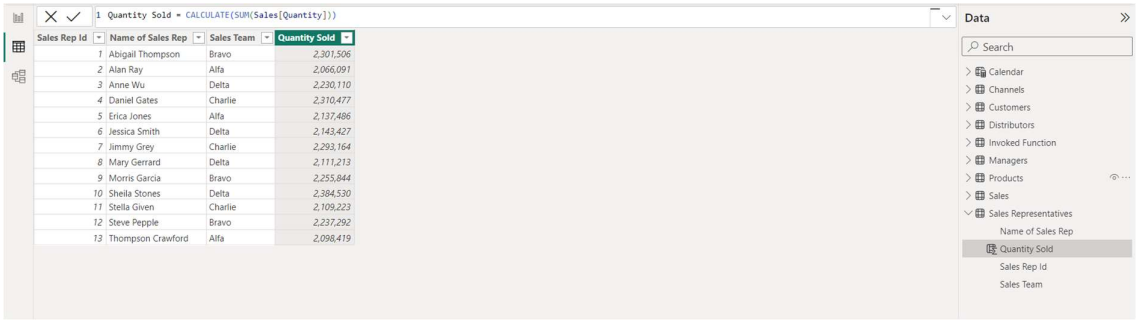


7. Time Intelligence: Using DAX, create a measure that calculates the year-to-date (YTD) sales for each month.

```
YTD Sales = CALCULATE(
    SUM('Sales'[Sales]),
    FILTER(
        ALL('Sales'),
        'Sales'[MonthYear] <= MAX('Sales'[MonthYear]) && YEAR('Sales'[MonthYear]) =
YEAR(MAX('Sales'[MonthYear]))
    )
)
```



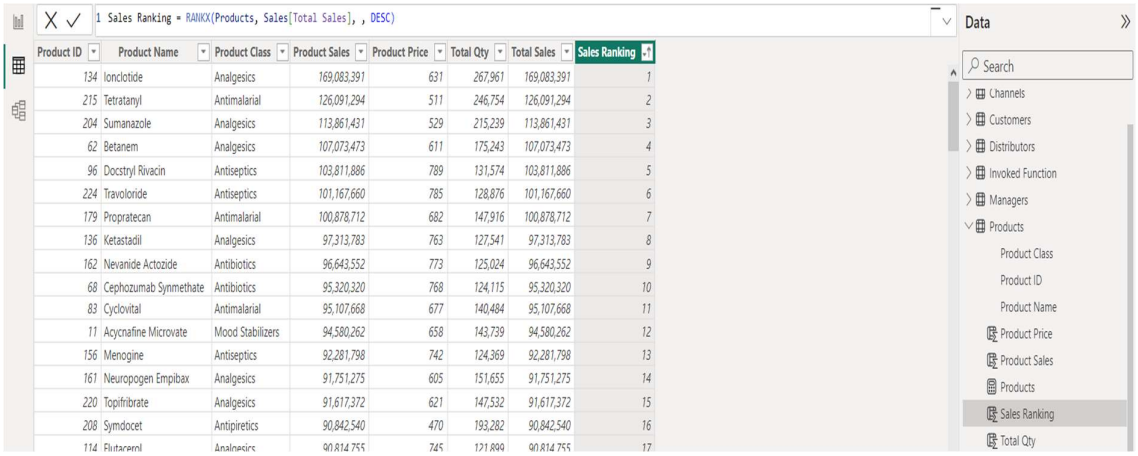
8. Filter Context vs. Row Context: Write a DAX calculation that shows the total quantity sold by each sales rep. Explain how filter and row contexts apply.



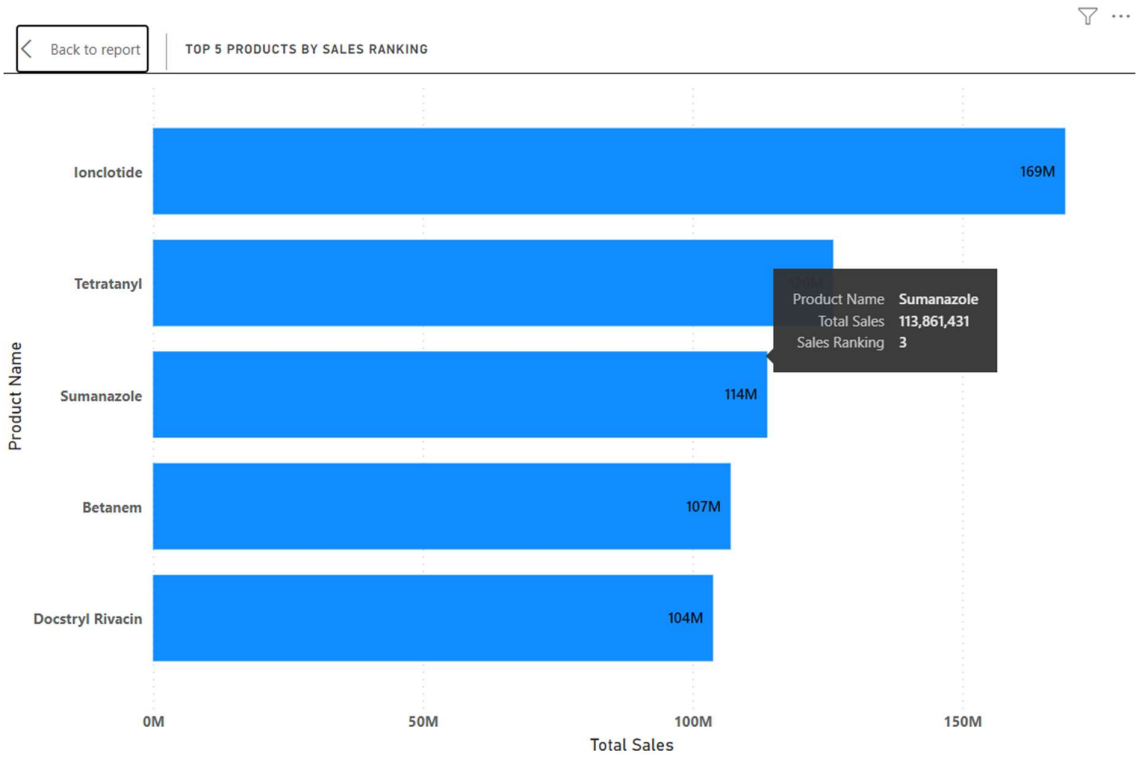
- The **CALCULATE** function modifies the filter context of the expression by applying additional filters to the data model. In this case, it is not applying any additional filters, so it is simply returning the sum of the **Quantity** column.

- The formula which I have used is using only **row context**. The **CALCULATE** function is not modifying the filter context of the expression by applying additional filters to the data model. Instead, it is simply returning the sum of the **Quantity** column for each row of the **Sales** table.

9. Ranking: Create a DAX measure that ranks products by sales. Display the top 5 products by rank in a visual.

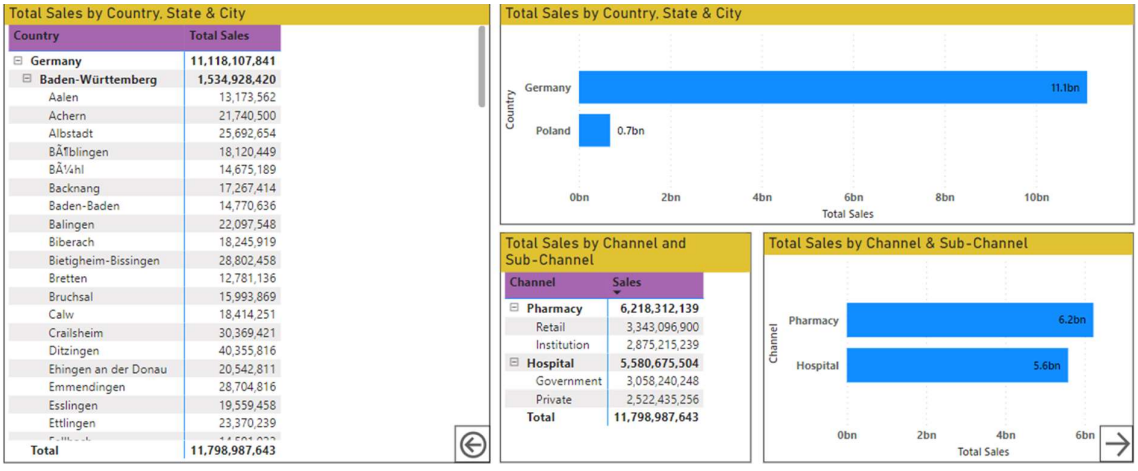



```
Sales Ranking = RANKX(Products, Sales[Total Sales], , DESC)
```



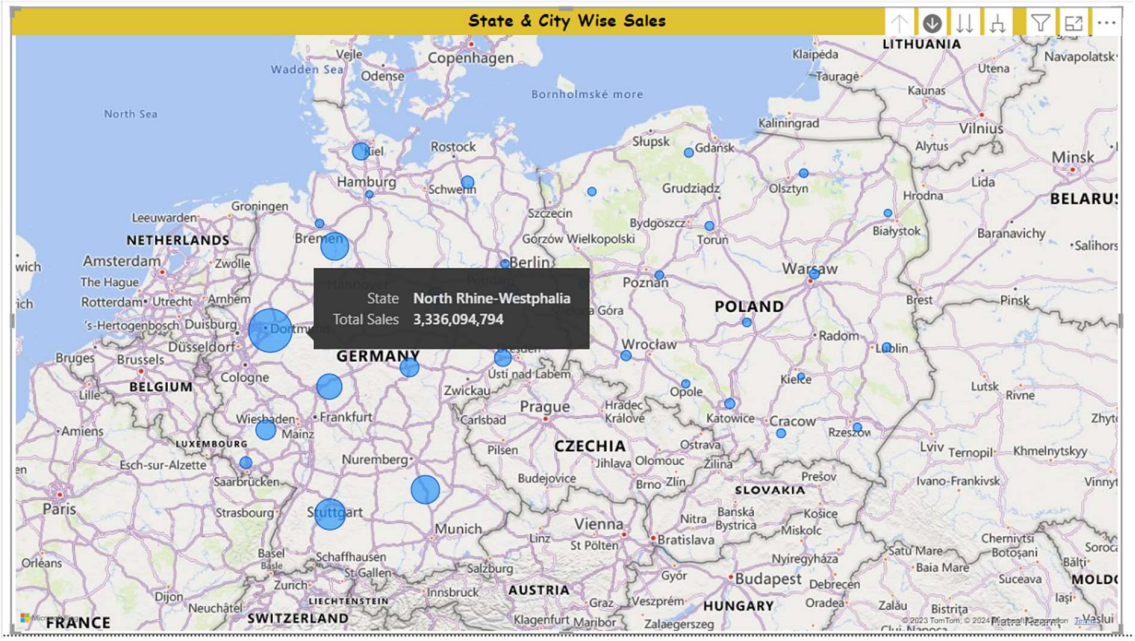
10. Parent-Child Hierarchies: If there's a hierarchy in your data, such as categories and subcategories, create a DAX measure to summarize sales at the subcategory level.

Since we cannot use SUMMARIZE() or SUMMARIZECOLUMNS() in a measure or a calculated column, I have used a rather simple way. I have created a matrix and added the hierarchical fields (Country-State-City) in one matrix and (Channel-Subchannel) in another and used a simple measure which is Total Sales = SUM(Sales[Sales]) in the values. The Matrix automatically generates the category and sub-category wise sales and allows to expand/collapse on demand.

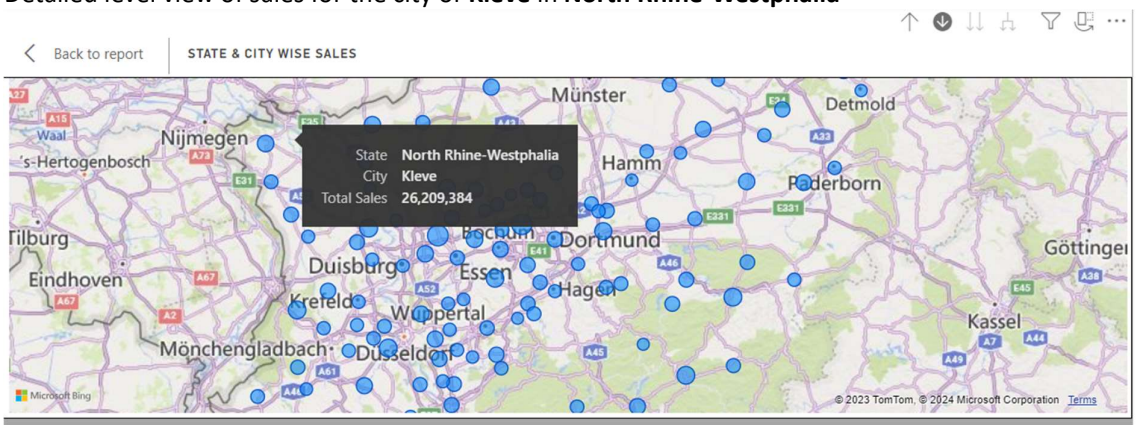


11. Drill-Through: Build a report where users can drill through from a summary to detailed data. For example, starting from a map, drill through to a table of individual sales for a specific city.

Summary level view of sales for the state of **North Rhine-Westphalia.**



Detailed level view of sales for the city of **Kleve** in **North Rhine-Westphalia**



State, City	Total Sales
North Rhine-Westphalia, Cologne	28,307,927
North Rhine-Westphalia, Gelsenkirchen	26,532,373
North Rhine-Westphalia, LÃ¼denscheid	26,339,532
North Rhine-Westphalia, Kleve	26,209,384
North Rhine-Westphalia, Dorsten	26,177,593
North Rhine-Westphalia, Lage	26,009,596
North Rhine-Westphalia, Oberhausen	25,860,721
North Rhine-Westphalia, Bocholt	25,826,483
North Rhine-Westphalia, Kerpen	25,575,623
North Rhine-Westphalia, HÃ¼ckelhoven	25,378,116
North Rhine-Westphalia, Lippstadt	25,290,286
North Rhine-Westphalia, Coesfeld	24,862,159

I have right clicked on the city named Kleeve and utilized the show as a table to look at the detailed level data in a table.

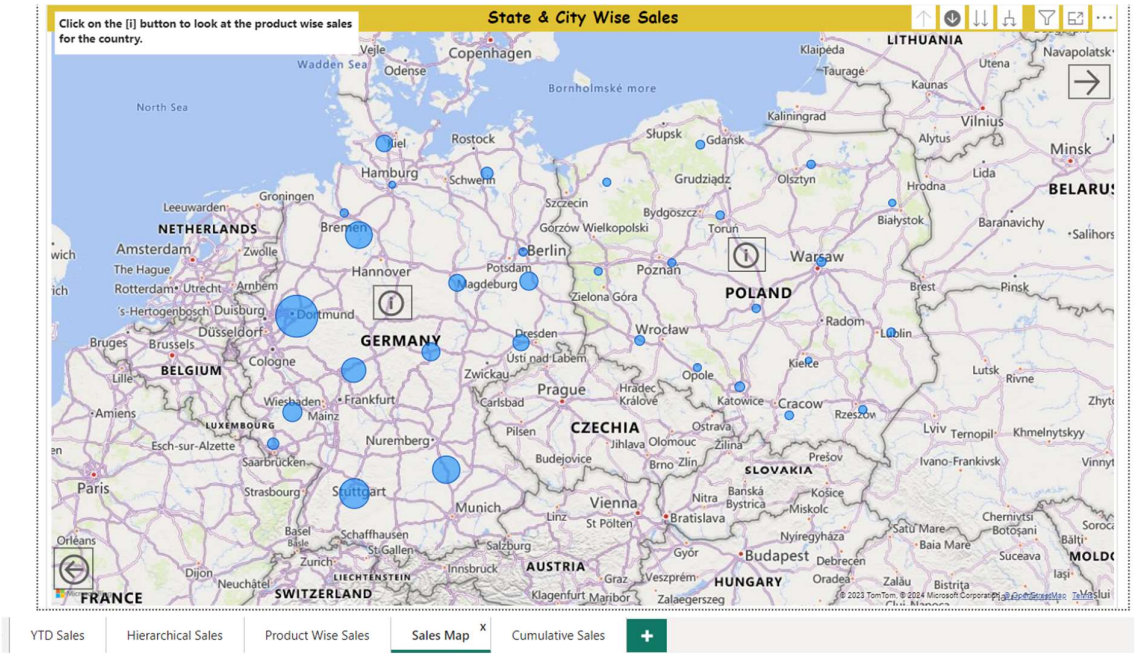
12. Custom Visuals: Use a custom visual in your report to visualize sales data in a unique way. Explain why you chose this custom visual.

    I wasn't able to use a custom visual as it requires me to login and have a Pro license which I cannot afford just for the sake of a few custom visuals.

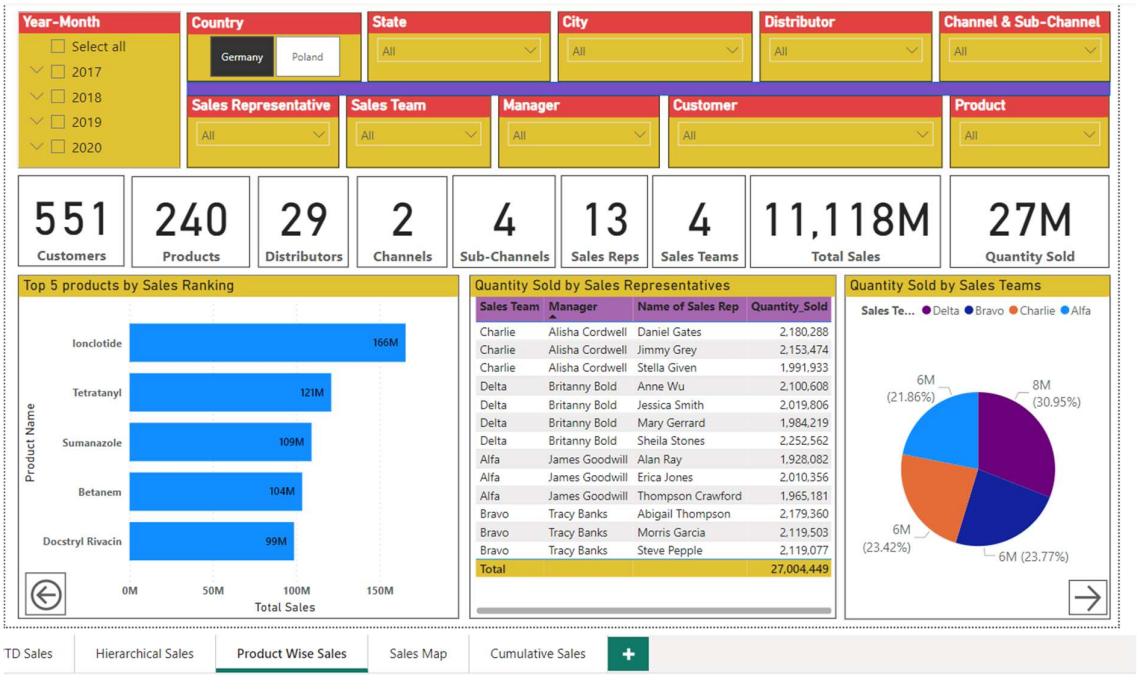
13. Bookmarks and Buttons: Create a report with bookmarks and buttons that allow users to navigate between different pages or states within the report.

    In the Sales Map Tab, I have created two **Bookmark Information action buttons** for Germany and Poland which on clicking would take to the **Product Wise Sales** for the respective Countries.

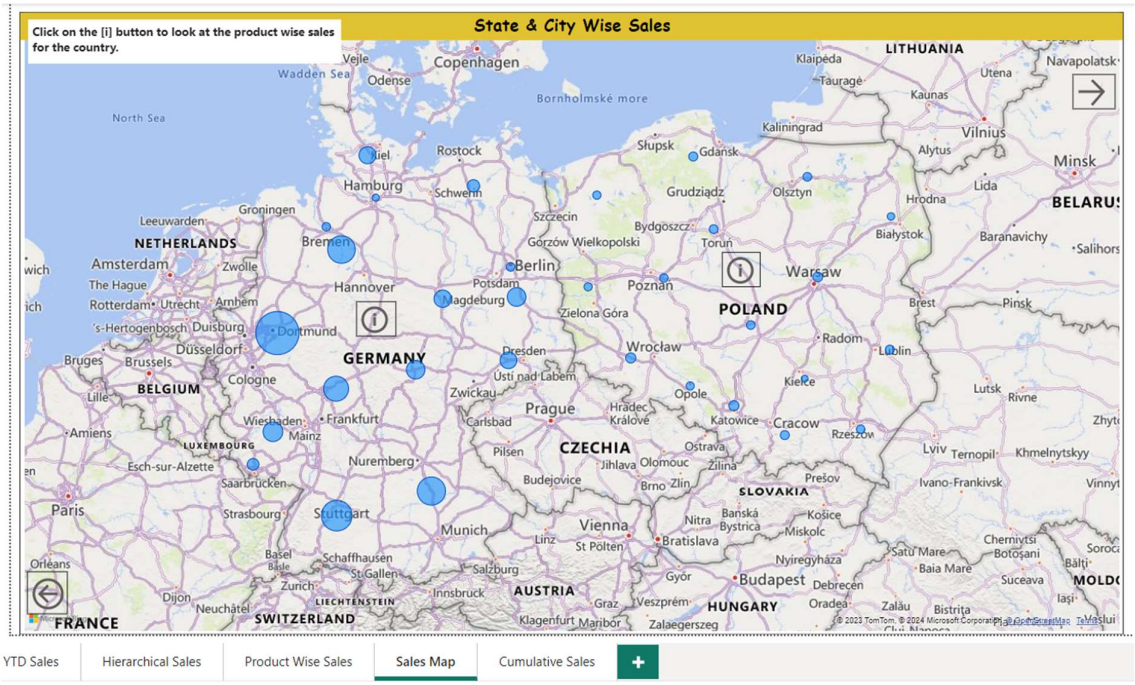
Before Clicking the Germany Info button



After Clicking on Germany Info button, we can see the product wise sales for Germany. We can also navigate back to the page by clicking the right arrow button configured for navigation.



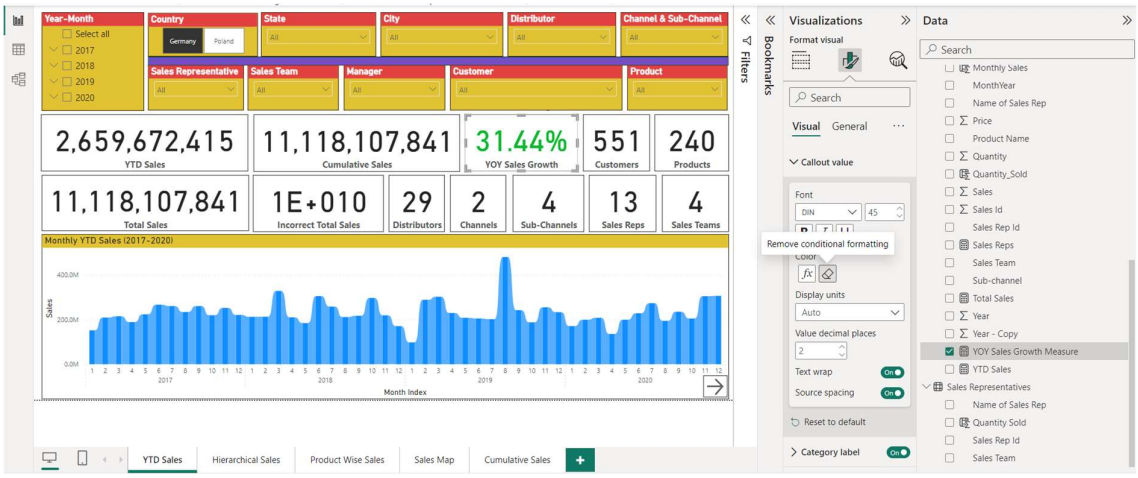
After clicking the right arrow button.

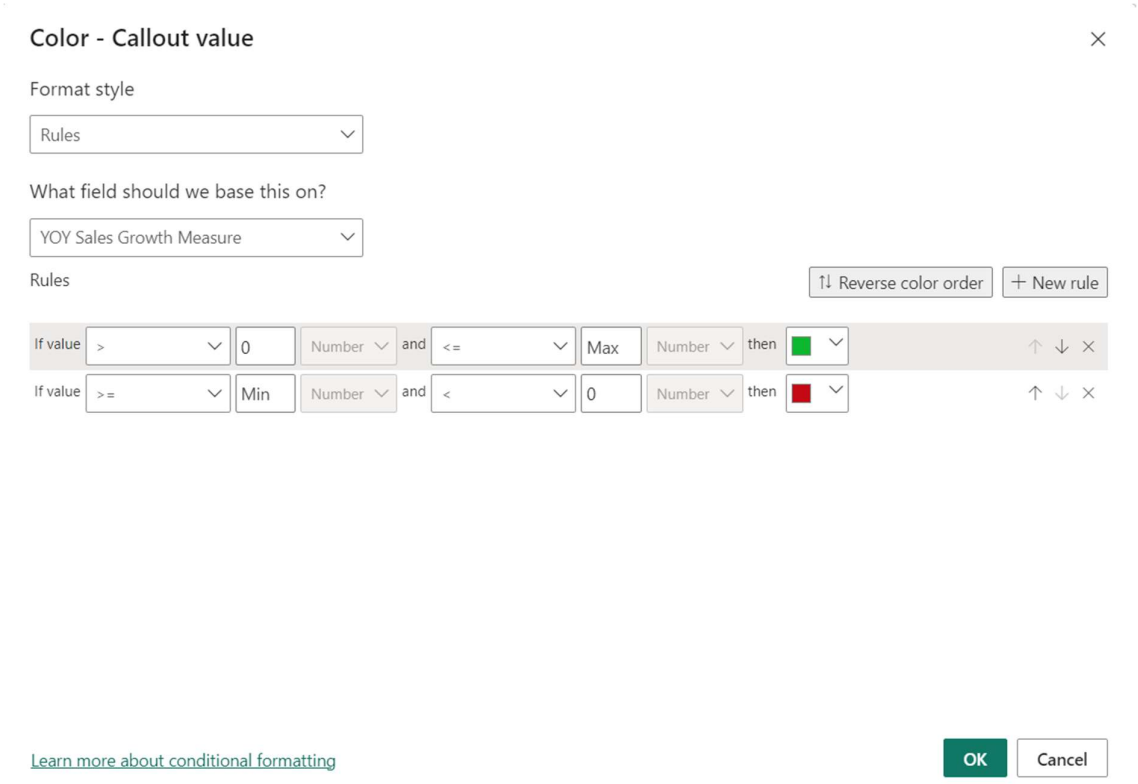


The report has the right arrow buttons for the first four tabs which would take us to the next tabs and left arrow buttons for the last 4 tabs which would take us to the previous tabs.

14. Conditional Formatting: Apply conditional formatting to a measure so that it changes color when sales exceed a certain target value.

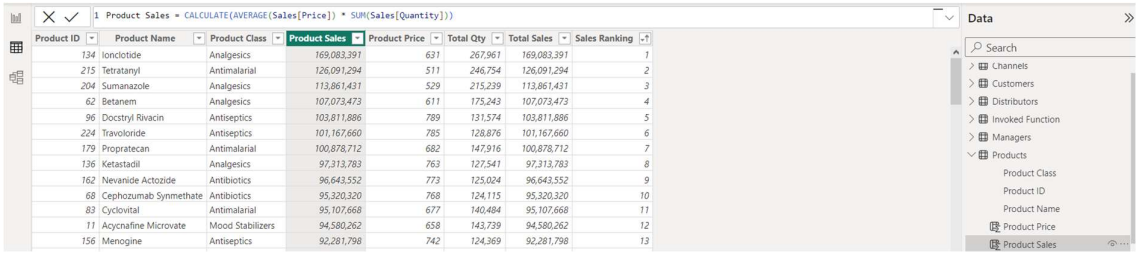
I have applied conditional formatting to the YOY Sales Growth Measure.





The Y-O-Y Growth figure is displayed in Green if it is above 0 i.e.(positive) else it is shown in red if it is equal to or less than 0 (i.e. negative).

15. Calculated Columns: Add a calculated column to your data model that calculates the total cost of each product (Quantity x Price).



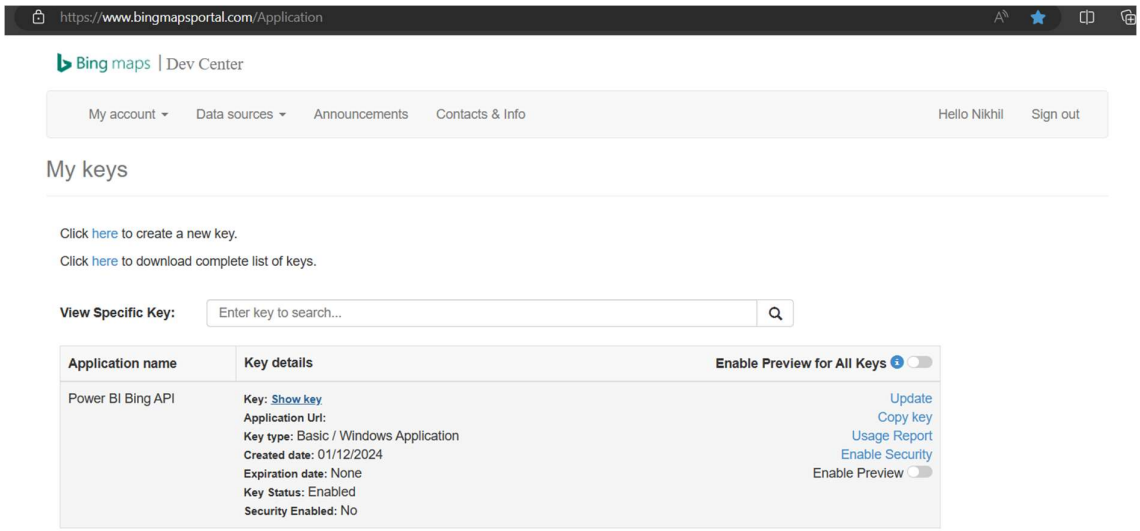

```
Product Sales = CALCULATE(AVERAGE(Sales[Price]) * SUM(Sales[Quantity]))
```

The calculated column for Product Sales in the Product Table multiplies the average of the Price from the related Sales table with the Sum of the quantity from the related Sales table to generate the Total Sales for each product.

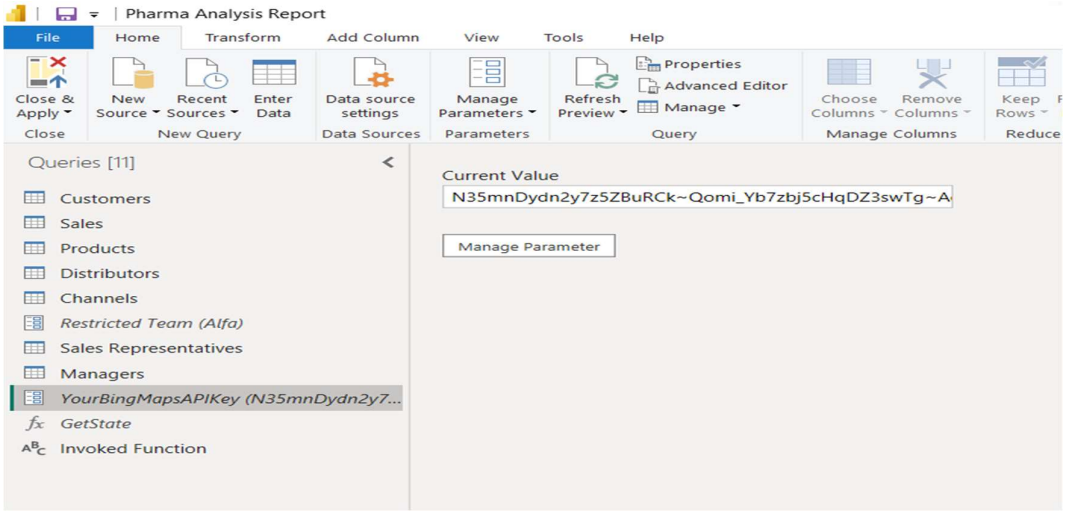
16. New Column from Example: Use the "New Column from Example" feature to add a column that categorizes cities into regions based on a predefined mapping.

The column from examples feature is not that smart to detect the State of a city from examples. Therefore, I have used a different approach which is wonderful.

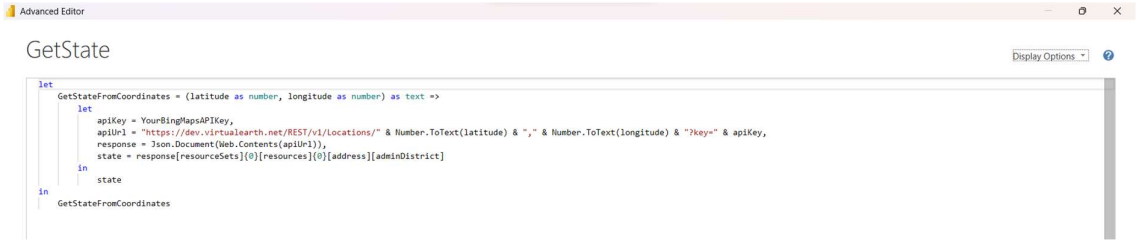
1. Obtained the Bing Map API Key using bingmapsportal.com



2. Created a parameter and stored my Bing Map API Key in this parameter named ‘Your Bing Maps API Key’.

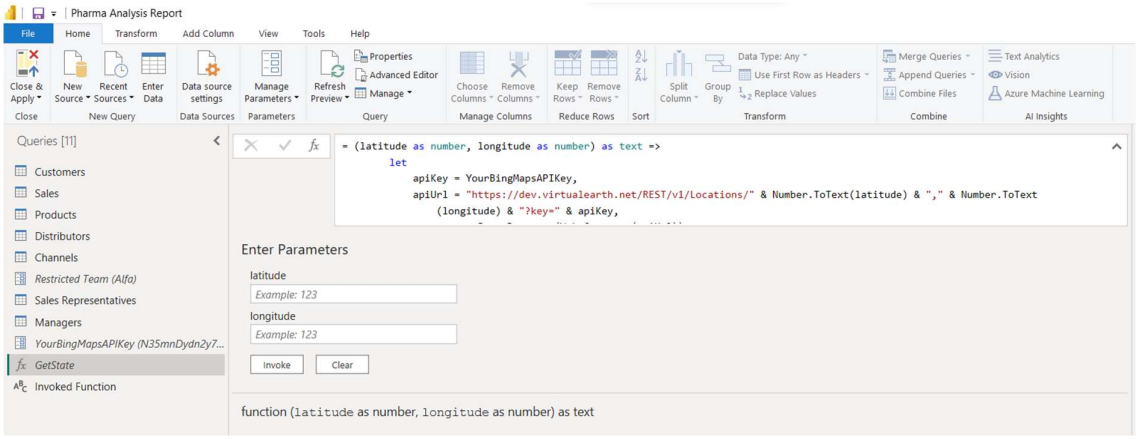


3. Defined a custom function named ‘Get State’ using Advanced Editor with help from ChatGpt.

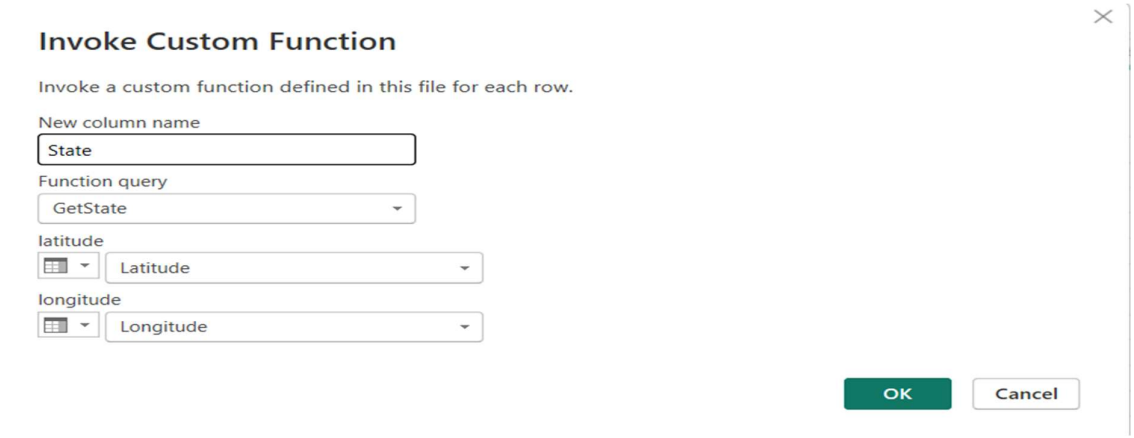


The Get State function uses the Latitude and Longitude information from the Customers table ,sends it in a request url to Bing Maps API and fetches the State from the response obtained

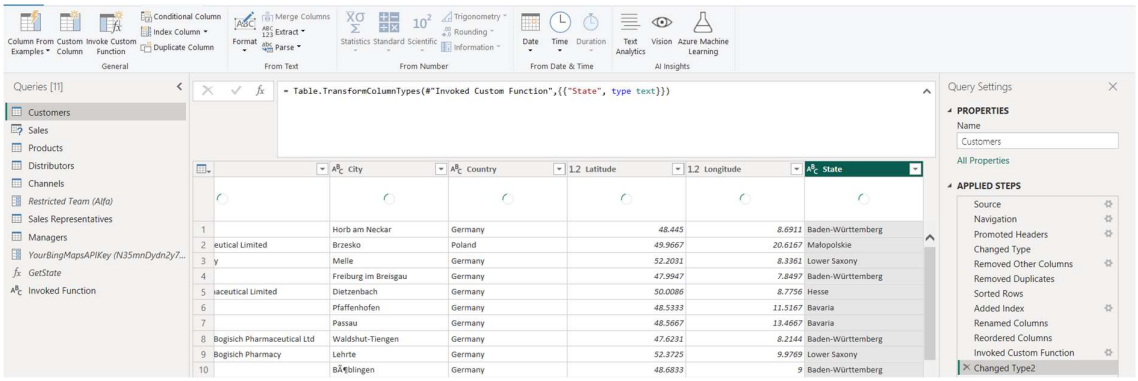
.



4. Invoked the Get State function using the Invoke Custom Function option.



After following these steps, I could obtain a column showing the states belonging to the latitudes and longitudes of the cities mentioned in the data set.



This couldn't have been possible using column from examples. Column from examples works best if we have a numeric or logical pattern.

17. Time-Based Calculations: Create a measure that calculates the year-over-year (YoY) growth in sales for each month.

```
YOY Sales Growth Measure =
VAR CurrentYearSales = SUM('Sales'[Sales])
VAR PreviousYearSales = CALCULATE(
    SUM('Sales'[Sales]),
    SAMEPERIODLASTYEAR('Calendar'[Date])
)
RETURN
    IF(
        NOT ISBLANK(PreviousYearSales),
        (CurrentYearSales - PreviousYearSales) / PreviousYearSales,
        0
    )
```

18. Cumulative Total: Develop a measure to show the cumulative total of sales over time and visualize it in a line chart.

```
Cumulative YTD Sales = CALCULATE(SUM('Sales'[Sales]),
                        FILTER(ALL('Calendar'[Date]),
                               'Calendar'[Date] <= MAX('Calendar'[Date])
                        )
                )
```