About Overstock.com:

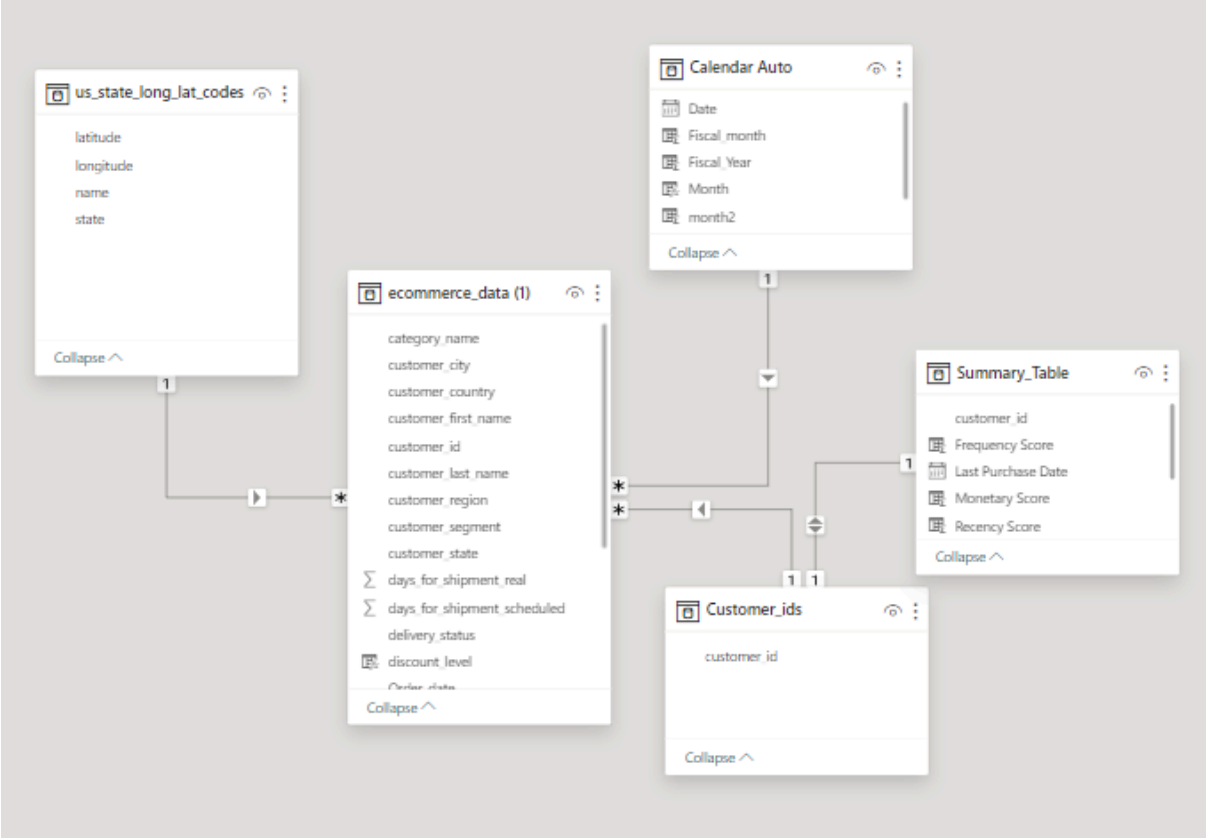
- Overstock.com is an American internet retailer headquartered in Midvale, Utah.
- Founded in 1999, Overstock initially sold surplus and returned merchandise at below-wholesale prices but later expanded to sell new merchandise.

Columns:

- **customer_id:** A unique identifier for each customer.
- **customer_first_name:** The first name of the customer.
- **customer_last_name:** The last name of the customer.
- **category_name:** The category of the product purchased (e.g., Furniture, Office Supplies, Technology).
- **product_name:** The specific name of the product purchased.
- **customer_segment:** The segment to which the customer belongs (e.g., Consumer, Corporate, Home Office).
- **customer_city:** The city where the customer resides.
- **customer_state:** The state where the customer resides.
- **customer_country:** The country where the customer resides.
- **customer_region:** The region where the customer resides.
- **delivery_status:** The status of the delivery (e.g., Delivered, Pending, Shipped).
- **order_date:** The date when the order was placed.
- **order_id:** A unique identifier for each order.
- **ship_date:** The date when the order was shipped.
- **shipping_type:** The type of shipping method used (e.g., Standard, Expedited).
- **days_for_shipment_scheduled:** The number of days scheduled for shipment.
- **days_for_shipment_real:** The actual number of days taken for shipment.
- **order_item_discount:** The discount applied to the order item.
- **sales_per_order:** The sales amount for each order.
- **order_quantity:** The quantity of items in each order.
- **profit_per_order:** The profit made on each order.

-

Data Modelling:



- Created a Calendar Table:

```
Calendar Auto =
CALENDAR
(
    MIN('ecommerce_data (1)'[Order_date]),
    MAX('ecommerce_data (1)'[Order_date])
)
```

- Fiscal Year: US : 1st October to 30th September:

```
Fiscal_Year =
IF
(
    ('Calendar Auto'[Date]) < DATE(YEAR('Calendar Auto'[Date]),10,1),YEAR
    ('Calendar Auto'[Date]),
    YEAR('Calendar Auto'[Date]) + 1
)
```

- Month Column:

```
Month = FORMAT('Calendar Auto'[Date],"MMMM")
```

- **Fiscal Month:**

```
Fiscal_month = MONTH(DATEADD('Calendar Auto'[Date],3,MONTH))
```

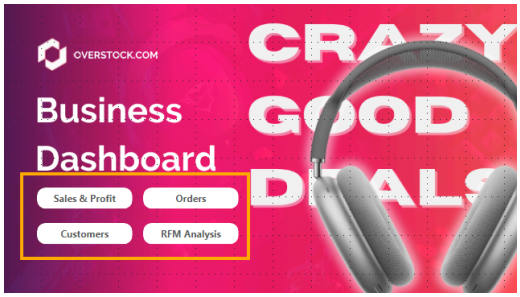
- **Quarter:**

```
Quarter = ("Q" & CEILING(DIVIDE('Calendar Auto'[Fiscal_month],3,0),1))
```

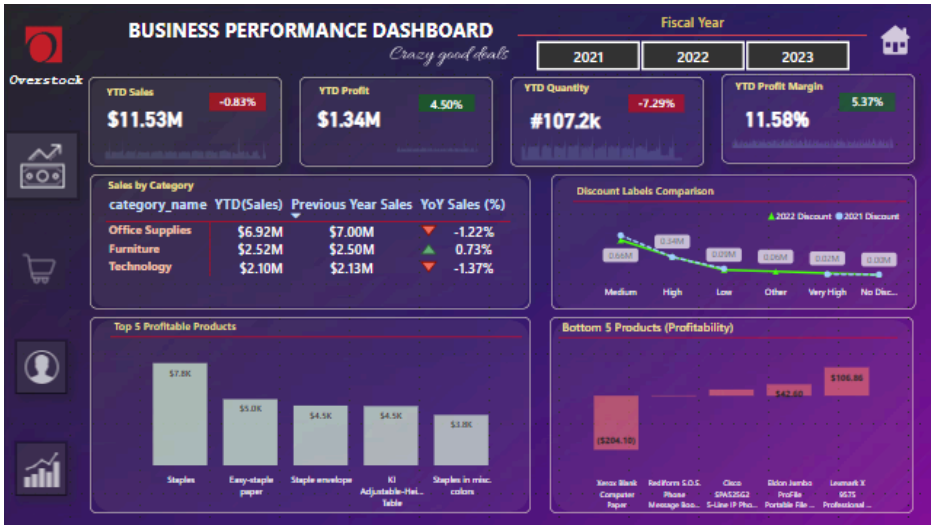
Date	Month	Fiscal_Year	month2	Fiscal_month	Quarter	Year
02/01/2021	January	2021	1	4	Q2	2021
03/01/2021	January	2021	1	4	Q2	2021
04/01/2021	January	2021	1	4	Q2	2021
05/01/2021	January	2021	1	4	Q2	2021
06/01/2021	January	2021	1	4	Q2	2021
07/01/2021	January	2021	1	4	Q2	2021
08/01/2021	January	2021	1	4	Q2	2021
09/01/2021	January	2021	1	4	Q2	2021
10/01/2021	January	2021	1	4	Q2	2021
11/01/2021	January	2021	1	4	Q2	2021
12/01/2021	January	2021	1	4	Q2	2021
13/01/2021	January	2021	1	4	Q2	2021
14/01/2021	January	2021	1	4	Q2	2021
15/01/2021	January	2021	1	4	Q2	2021

1. Landing Page:

- Tried Capturing the Colour Theme of the website.
- Used Buttons and Navigations.
- Based on the Dataset, made following segments in the dashboard:
    - Sales & Profit.
    - Orders.
    - Customers.
    - RFM Analysis.



2. Sales and Profit:



1st Component:

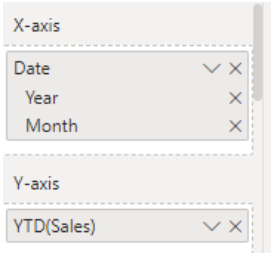
1: KPI CARD:

YTD Sales- Total Sales between Start of of the Year and last date (i.e. Current Date)

```
YTD(Sales) =
TOTALYTD
(
    SUM('ecommerce_data (1)'[sales_per_order]),'ecommerce_data
    (1)'[Order_date]
)
```

2: Line Graph:

- Shows the Changes of YTD sales from the beginning of the Year until Present, i.e.current Date.



-

3. KPI:

- Difference between YTD Sales and Previous Year same period Sales
- Then getting it divided by previous Year to get the % Change.

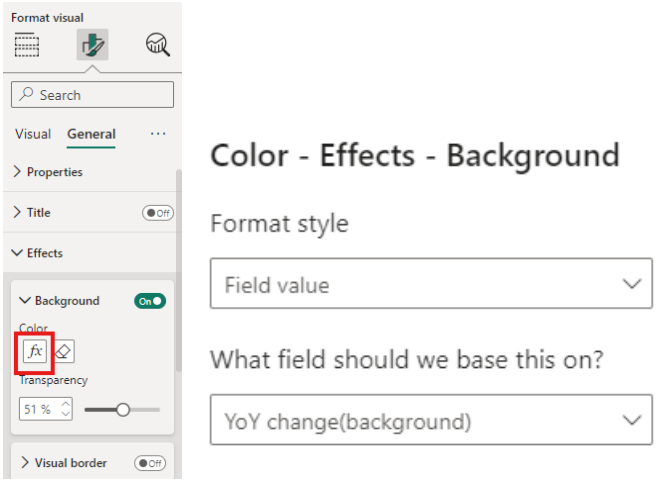
```
Previous Year Sales =

var prev =
CALCULATE
(
    SUM('ecommerce_data (1)'[sales_per_order]),
    SAMEPERIODLASTYEAR('Calendar Auto'[Date])
)
RETURN
IF
(
    isblank(prev),0,prev
)
```

```
YoY Sales (%) = ([YTD(Sales)] - [Previous Year Sales])/[Previous
Year Sales]
```

- Creating a Measure for background Colour that would change between Red and Green depending on the value, and using this Measure for conditional formatting.

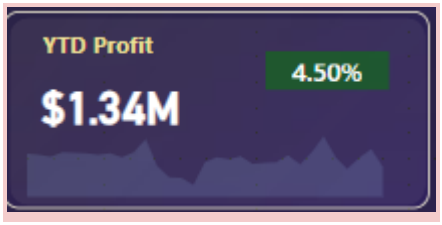
```
YoY change(background) =
IF
(
    [YoY Sales (%)] = 0, "Orange",
    IF
    (
        [YoY Sales (%)]< 0, "Red","Green"
    )
)
```



DAX Functions Used:

- Sameperiodlastyear()
- Sum()
- Isblank()
- Divide()
- If()

2nd Component:



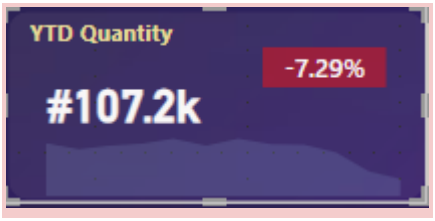
- Same as the 1st component

```
YTD(Profit) =
var ytdprof =
TOTALYTD
(
    SUM('ecommerce_data (1)'[profit_per_order]),
    'ecommerce_data (1)'[Order_date]
)
RETURN
IF
(
    ISBLANK(ytdprof),0,ytdprof
)
```

```
Previous Year Profit =
CALCULATE
(
    SUM('ecommerce_data (1)'[profit_per_order]),
    SAMEPERIODLASTYEAR('Calendar Auto'[Date])
)
```

```
YoY Profit (%) =
var num = [YTD(Profit)] - [Previous Year Profit]
var x = DIVIDE(num,[Previous Year Profit],0)
RETURN
IF
(
    ISBLANK(x) || x = 0,0,x
)
```

3rd Component:

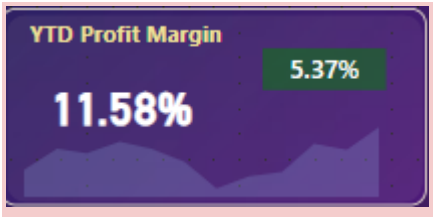



```
YTD(Quantity) =
TOTALYTD
(
    SUM('ecommerce_data (1)'[order_quantity]),
    'ecommerce_data (1)'[Order_date]
)
```

```
Previous Year Quantity =
CALCULATE
(
    SUM('ecommerce_data (1)'[order_quantity]),
    SAMEPERIODLASTYEAR('Calendar Auto'[Date])
)
```

```
YoY Quantity (%) = ([YTD(Quantity)] - [Previous Year Quantity])/[Previous Year Quantity]
```

4th Component:

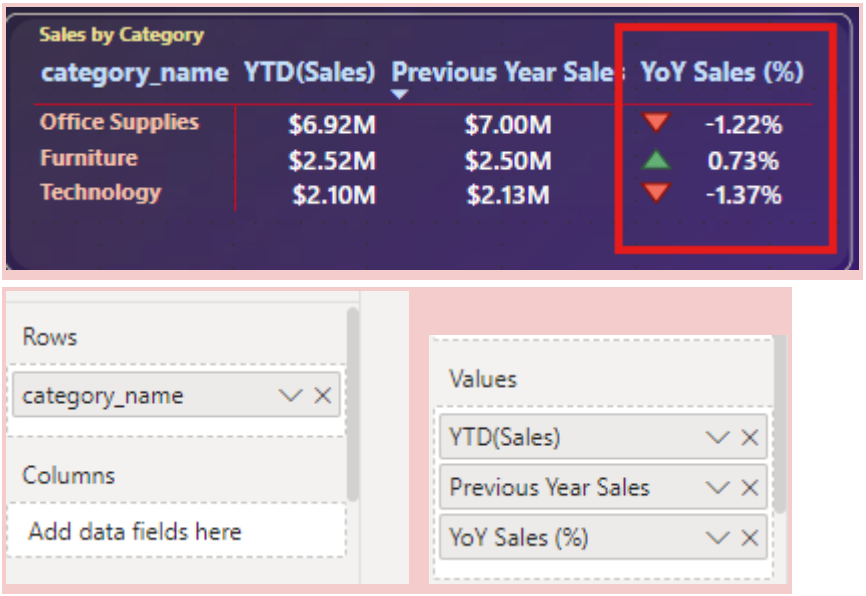


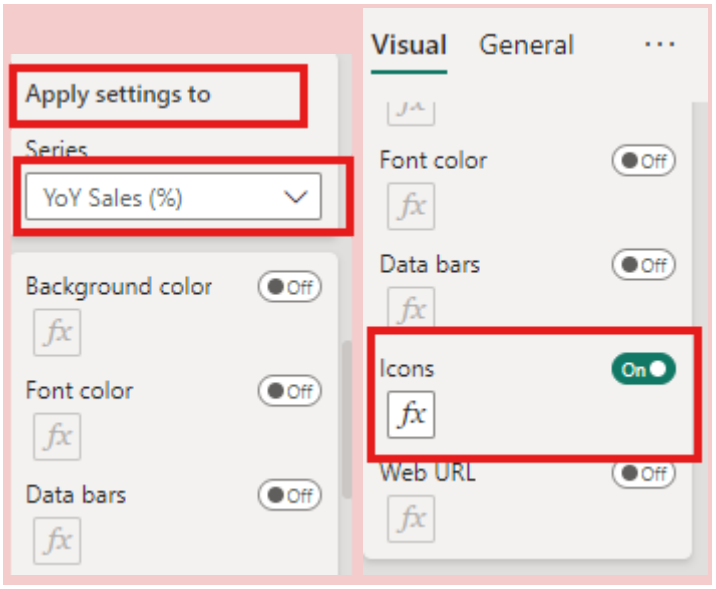

```
Profit Margin = SUM('ecommerce_data (1)'[profit_per_order])/SUM('ecommerce_data (1)'[sales_per_order])
```

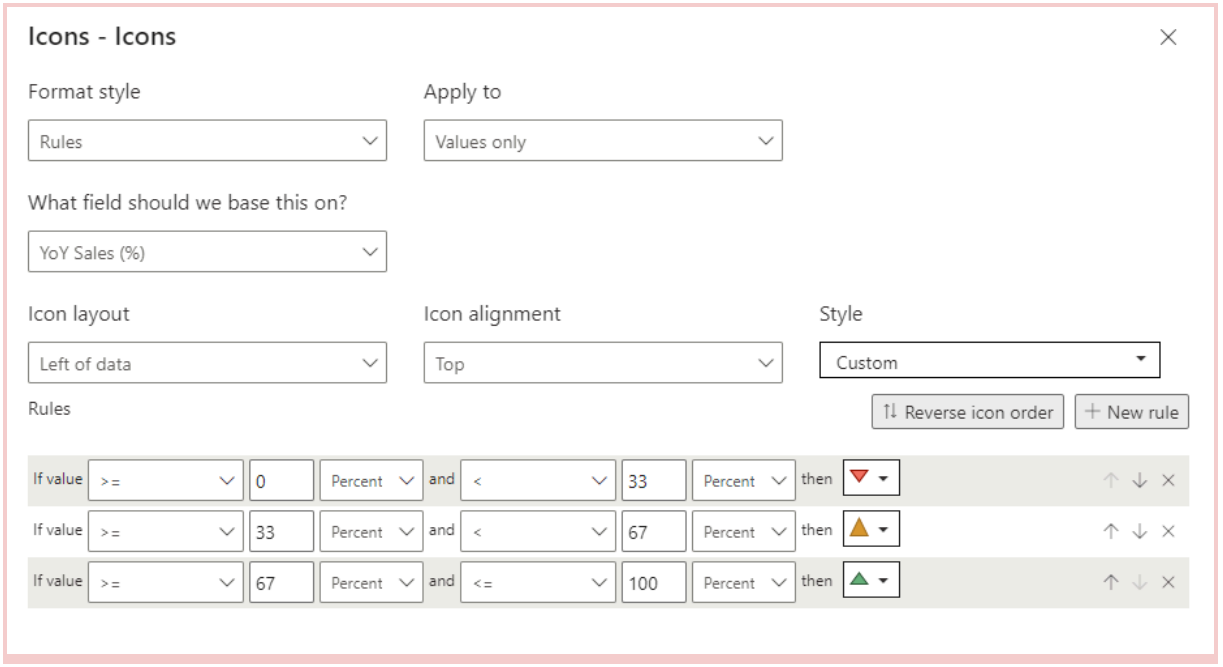
```
YTD Profit Margin =
TOTALYTD
(
    [Profit Margin],'ecommerce_data (1)'[Order_date]
)
```

```
YoY Profit Margin (%) = ([YTD Profit Margin] - [Previous
Year Profit Margin])/[Previous Year Profit Margin]
```

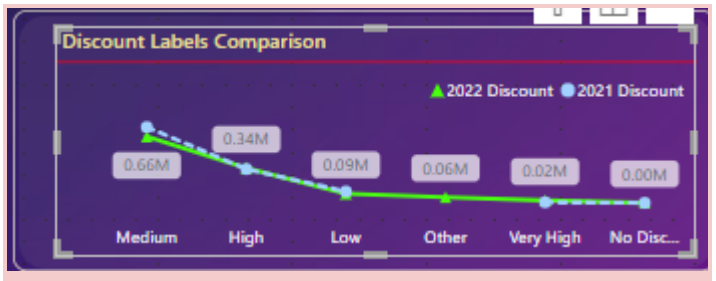
5th Component:

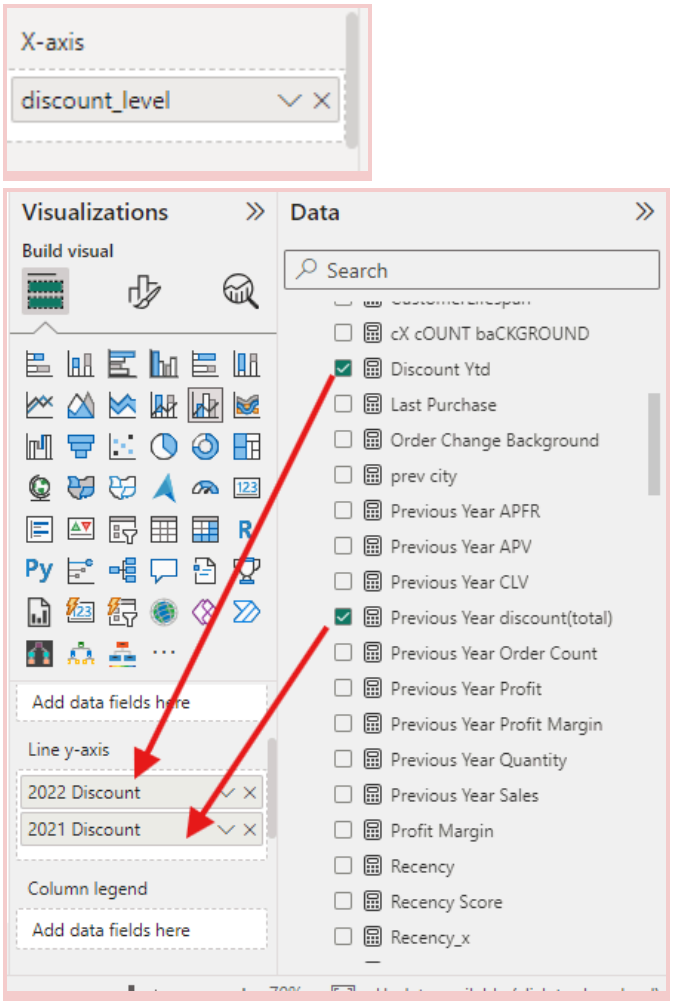




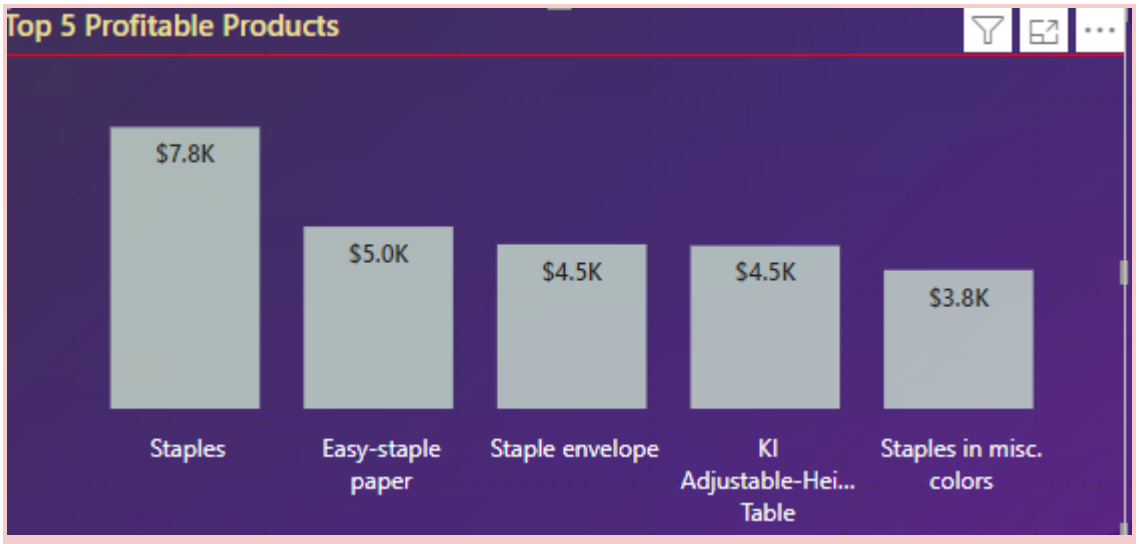


6th Component:

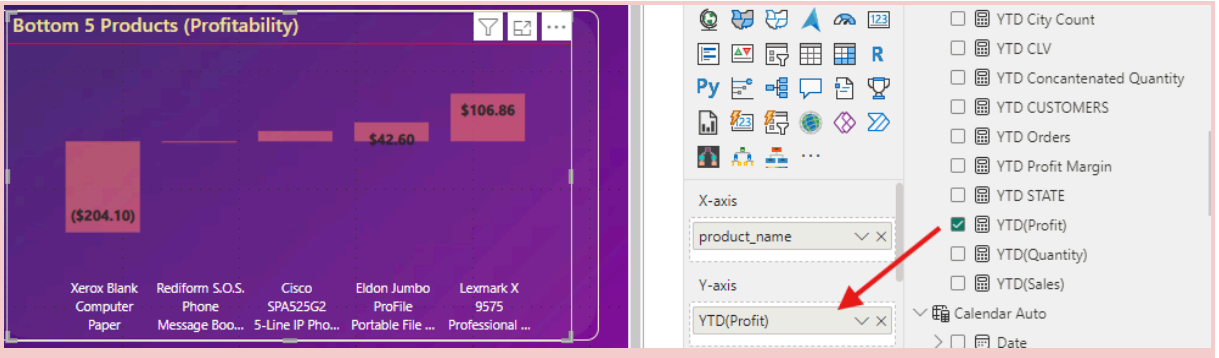




7th Component:



8th Component:



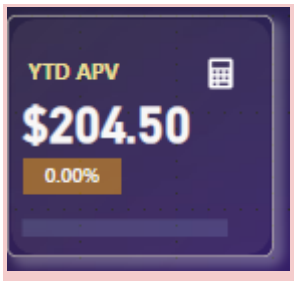
9th Component:

3. Orders:

1st Component:

- Same as before.
- YTD Orders.
- Previous Year YTD of Orders using (SameperiodLastYear)
- Finding YoY % Change.

2nd Component:



Total Revenue Per Customer:

```
Total Revenue Per Customer =
CALCULATE
(
    SUM('ecommerce_data (1)'[sales_per_order]),
    ALLEXCEPT('ecommerce_data (1)','ecommerce_data (1)'
    [customer_id])
)
```

Total Orders Per Customer:

```
Total Orders Per Customer =
CALCULATE
(
    COUNT('ecommerce_data (1)'[order_id]),
    ALLEXCEPT('ecommerce_data (1)','ecommerce_data (1)'
    [customer_id])
)
```

AVERAGE PURCHASE VALUE(APV):

```
Average Purchase Value (APV) =
DIVIDE
(
    [Total Revenue Per Customer],
    [Total Orders Per Customer]
)
```

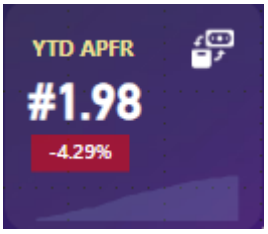
Significance of APV for an E-commerce Business

- Higher APV indicates that customers are spending more per transaction, which can suggest higher-value products or effective upselling strategies.
- Targeted Campaigns: For example, if the APV is high, a business might focus on premium products or services.
- Upselling and Cross-selling: APV can inform strategies to increase order value through upselling (encouraging the purchase of a higher-end product) and cross-selling (offering complementary products).
- Customer Segmentation:Identifying High-Value Customers:
    - APV can be used to segment customers.
    - Identifying and nurturing high-APV customers can lead to increased loyalty and repeat purchases.

3rd Component:

Average Purchase Frequency(APFR):

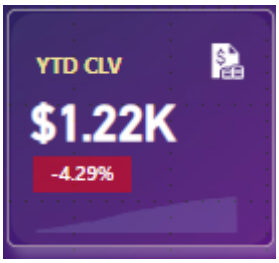
```
Average Purchase Frequency Rate (APFR) =
var num = COUNT('ecommerce_data (1)'[order_id])
var deno = DISTINCTCOUNT('ecommerce_data (1)'[customer_id])
RETURN
DIVIDE
(
    num,
    deno,
    0
)
```



Then in a similar way, we calculated the YDT APFR, YTD APFR(Previous Year) using Sameperiodlastyear.

4th Component:

Customer Lifetime Value(CLV):

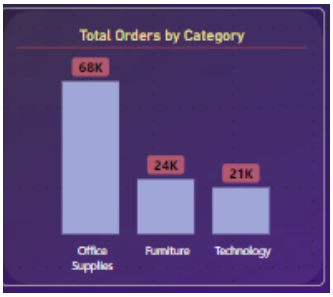


Customer Lifespan for Industry : 3 years (Standard)

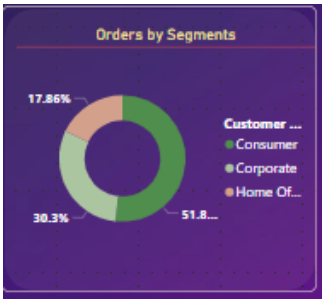
```
Customer Lifetime Value (CLV) =
[Average Purchase Value (APV)] * [Average Purchase Frequency
Rate (APFR)] * [CustomerLifespan]
```

Then in a similar way, we calculated the YDT CLV, YTD CLV(Previous Year) using Sameperiodlastyear.

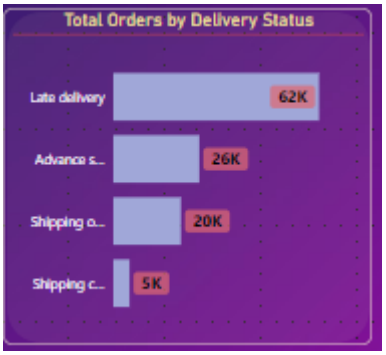
5th Component:



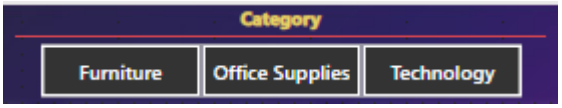
6th Component:



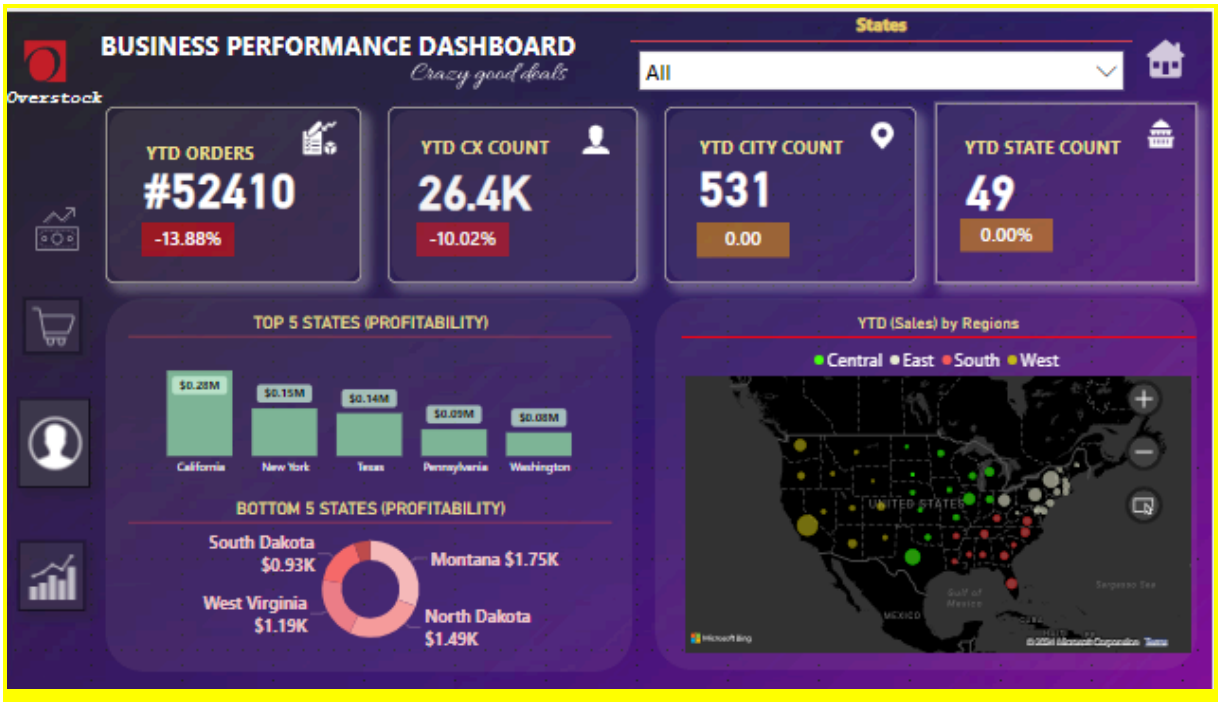
7th Component:



8th Component:



4. Customers:



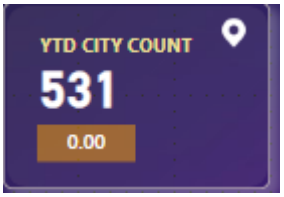
1st Component:

2nd Component:

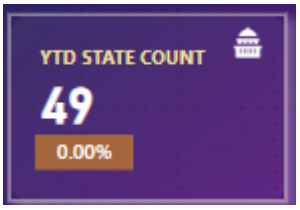

```
YTD CUSTOMERS =
TOTALYTD
(
    DISTINCTCOUNT('ecommerce_data (1)'[customer_id]),
    'Calendar Auto'[Date]
)

yOy Customer Count (%) =
VAR PREV =
CALCULATE
(
    DISTINCTCOUNT('ecommerce_data (1)'[customer_id]),
    SAMEPERIODLASTYEAR('Calendar Auto'[Date])
)
var CUR = [YTD CUSTOMERS]
VAR DIFF = CUR - PREV
VAR DIV = DIVIDE(DIFF,PREV,0)
return
DIV
```

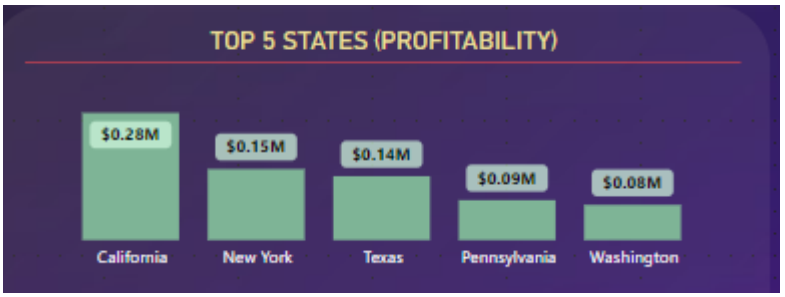
3rd Component:



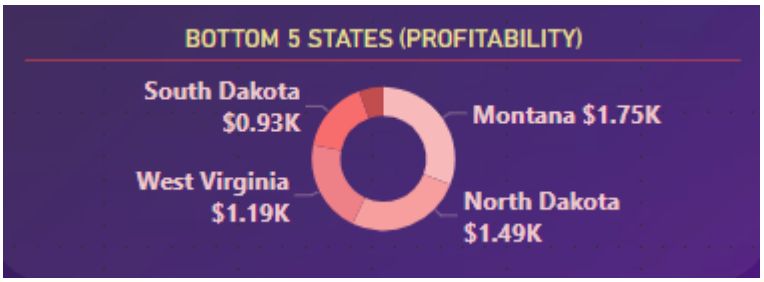
4th Component:



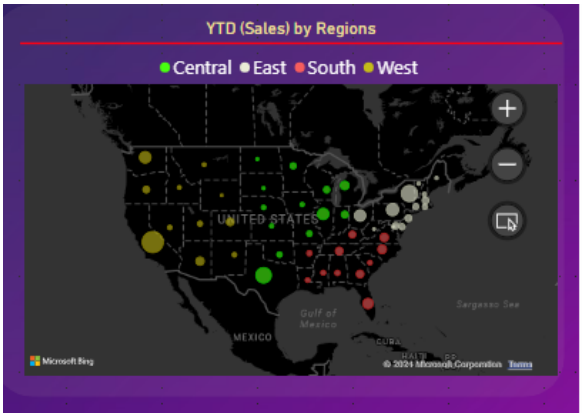
5th Component:



6th Component:



7th Component:



8th Component:

Slicer Component, CX ID

5. RFM:

- The Dataset had a list of all purchases done by Customers.
- Thus, let's say Customerid x might have multiple entries on different dates with different order ids.
- RFM Recency, Frequency, Monetary Analysis: required for each customer.
- something like a groupby customerid.
- Thus, we would need to create a separate table by selecting specific tables and summarizing them by customer id and then doing calculations on them.

- Because, for Recency: we would require the last purchase date for each customer.

```
Summary_Table =
ADDCOLUMNS
(
    SUMMARIZE
    (
        'ecommerce_data (1)','ecommerce_data (1)'[customer_id]
    ),
    "Last Purchase Date", CALCULATE(MAX('ecommerce_data (1)'[Order_date])),
    "Total Orders Count", CALCULATE(DISTINCTCOUNT('ecommerce_data (1)'
    [order_id])),
    "Total Spendings", CALCULATE(SUM('ecommerce_data (1)'[sales_per_order]))
)
```

We used **Summarise() and Addcolumns():**

This created a Summary Table:

customer_id	Last Purchase Date	Total Orders Count	Total Spendings
C_ID_41345	12/2/2022 12:00:00 AM	1	129.9900055
C_ID_66821	9/25/2022 12:00:00 AM	1	129.9900055
C_ID_70856	10/12/2022 12:00:00 AM	1	129.9900055
C_ID_69381	2/8/2022 12:00:00 AM	1	129.9900055
C_ID_47191	4/1/2022 12:00:00 AM	1	129.9900055
C_ID_49454	11/27/2021 12:00:00 AM	1	129.9900055
C_ID_46827	3/27/2021 12:00:00 AM	1	129.9900055
C_ID_72538	11/14/2021 12:00:00 AM	1	129.9900055
C_ID_72520	12/4/2021 12:00:00 AM	1	129.9900055
C_ID_74851	8/31/2021 12:00:00 AM	1	129.9900055
C_ID_61616	8/20/2022 12:00:00 AM	1	129.9900055
C_ID_68351	7/12/2022 12:00:00 AM	1	129.9900055
C_ID_47946	12/11/2021 12:00:00 AM	1	129.9900055
C_ID_50561	1/10/2021 12:00:00 AM	1	129.9900055
C_ID_66909	3/30/2022 12:00:00 AM	1	129.9900055
C_ID_55716	3/30/2021 12:00:00 AM	1	129.9900055

Mark they have been categorised for each customer, no repetition .

Now we have to find Recency: Recency - Total No of Days since last purchase.This has to be done with respect to Current Date() i.e today. But we took the reference date as Max Order Date which is 31st Dec 2022.

```
Reference Date = CALCULATE(MAX('ecommerce_data (1)'[Order_date]),ALL
('ecommerce_data (1)'))
```

Recency = Difference between Last Purchase Date and Reference Date in Days. Used **DatedDiff()**

```
Recency_days = DATEDIFF(Summary_Table[Last Purchase Date],[Reference Date],
DAY)
```

Recency_Score: The Old Purchases are less frequent so we would score them less and the recent purchases must be given higher scores since they are recent and customers are more likely to purchase again. We used Switch()

```
Recency Score =
SWITCH(
    TRUE(),
    Summary_Table[Recency_days] <= 90, 5,
    Summary_Table[Recency_days] <= 180, 4,
    Summary_Table[Recency_days] <= 365, 3,
    Summary_Table[Recency_days] <= 540, 2,
    1
)
```

Frequency Score:

This is based on the number of Purchases done by the CX,
i.e How frequently he makes the purchase.

Higher no of Purchases -> Higher scores.
We used Switch() based on the Total Order Counts column.

```
Frequency Score =
SWITCH
(
    TRUE(),
    Summary_Table[Total Orders Count] >= 5,5,
    Summary_Table[Total Orders Count] >= 4 && Summary_Table[Total Orders Count] < 5, 4,
    Summary_Table[Total Orders Count] >= 3 && Summary_Table[Total Orders Count] < 4,3,
    Summary_Table[Total Orders Count] >= 2  && Summary_Table[Total Orders Count] < 3, 2,
    Summary_Table[Total Orders Count] >= 1 && Summary_Table[Total Orders Count] < 2,1,
    0
)
```

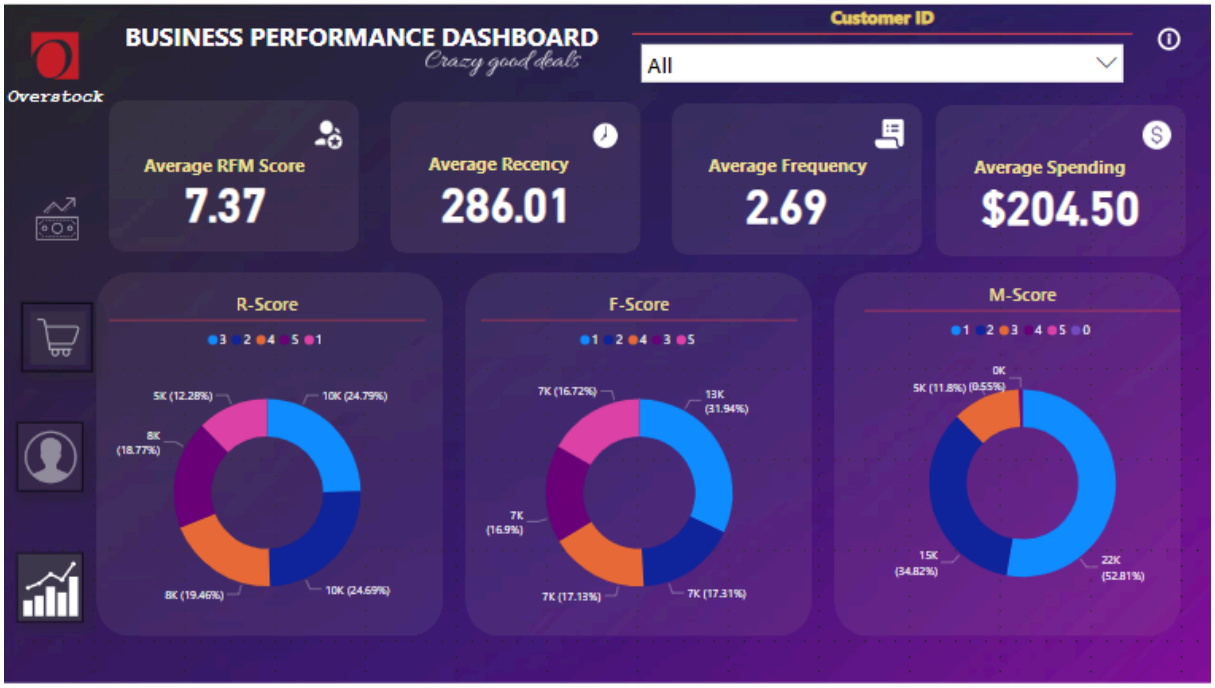
Monetary Score:

This is based on the total Spendings done by the Customer.
Higher Spendings => Higher Score
Used Switch() on Total Spendings Column

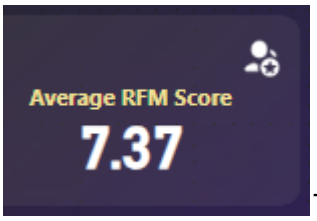
```
Monetary Score =
SWITCH(
    TRUE(),
    Summary_Table[Total Spendings] >= 3000, 5,
    Summary_Table[Total Spendings] >= 2000 && Summary_Table[Total Spendings] <
    3000,4,
    Summary_Table[Total Spendings] >= 1000 && Summary_Table[Total Spendings] <
    2000,3,
    Summary_Table[Total Spendings] >= 500 && Summary_Table[Total Spendings] <
    1000, 2,
    Summary_Table[Total Spendings] >= 10 && Summary_Table[Total Spendings] <
    500 , 1,
    0
)
```

Table becomes :

customer_id	Last Purchase Date	Total Orders Count	Total Spendings	Recency_days	Recency Score	Frequency Score	Monetary Score
C_ID_41345	12/2/2022 12:00:00 AM	1	129.9900055	29	5	1	1
C_ID_66821	9/25/2022 12:00:00 AM	1	129.9900055	97	4	1	1
C_ID_70856	10/12/2022 12:00:00 AM	1	129.9900055	80	5	1	1
C_ID_69381	2/8/2022 12:00:00 AM	1	129.9900055	326	3	1	1
C_ID_47191	4/1/2022 12:00:00 AM	1	129.9900055	274	3	1	1
C_ID_49454	11/27/2021 12:00:00 AM	1	129.9900055	399	2	1	1
C_ID_46827	3/27/2021 12:00:00 AM	1	129.9900055	644	1	1	1
C_ID_72538	11/14/2021 12:00:00 AM	1	129.9900055	412	2	1	1
C_ID_72520	12/4/2021 12:00:00 AM	1	129.9900055	392	2	1	1
C_ID_74851	8/31/2021 12:00:00 AM	1	129.9900055	487	2	1	1
C_ID_61616	8/20/2022 12:00:00 AM	1	129.9900055	133	4	1	1
C_ID_68351	7/12/2022 12:00:00 AM	1	129.9900055	172	4	1	1
C_ID_47946	12/11/2021 12:00:00 AM	1	129.9900055	385	2	1	1
C_ID_50561	1/10/2021 12:00:00 AM	1	129.9900055	720	1	1	1
C_ID_66909	3/30/2022 12:00:00 AM	1	129.9900055	276	3	1	1
C_ID_55716	3/30/2021 12:00:00 AM	1	129.9900055	641	1	1	1



1st Component:



This involves adding Recency Score, Frequency Score, Monetary Score and then finding its average score i.e sum of Total RFM Scores for all customers divided by total number of customers.

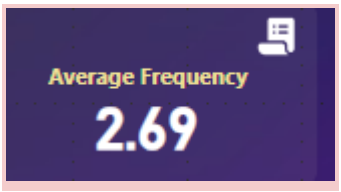
Thus we preferred using **Averagex():**

```
Average RFM Score = AVERAGEX
(
    Summary_Table,Summary_Table[Recency Score] + Summary_Table
    [Frequency Score] + Summary_Table[Monetary Score]
)
```

2nd Component:

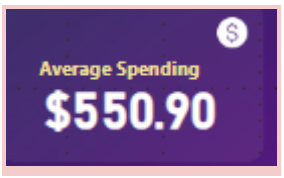

```
Average Recency =
AVERAGE
(
    Summary_Table[Recency_days]
)
```

3rd Component:



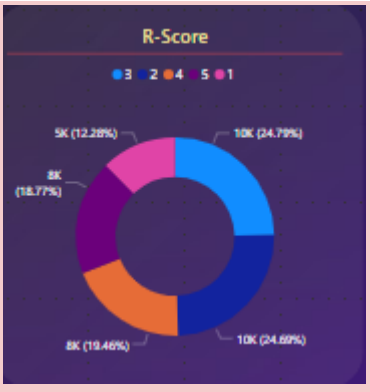

```
Average Frequency = AVERAGE(Summary_Table[Total Orders Count])
```

4th Component:

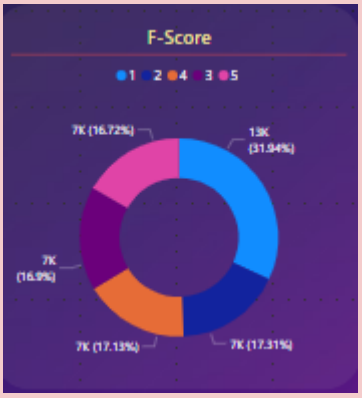



```
Average Spending = AVERAGE(Summary_Table[Total Spendings])
```

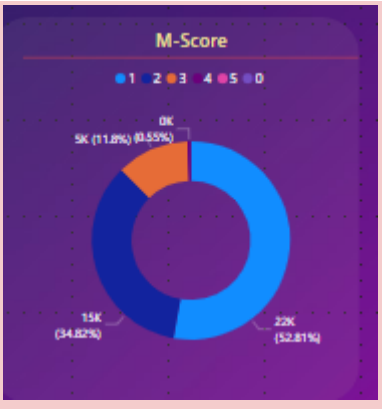
5th Component:



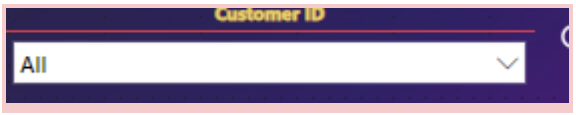
6th Component:



7th Component:



7th Component:



Slicer with customer ids.