Supply chain analysis

Presentation

Meet our extended team

Ahmed Essam El-dien

Ahmed Mohamed Osman

Shimaa Mohamed

Ayaa Hassan

Eman Abdelhaliem

Steps Taken

- Database Setup
- Data Exploration
- Data Cleaning and Preprocessing
- Dropping Useless Columns
- KPI Calculation

Agenda

- Objective
- Get data
- Data exploration and cleaning using SQL
- Listing the KPIs using SQL
- Data exploration and cleaning using Python
- Listing the KPIs using Python
- Dashboard design using Power BI

Objective

The primary objective of this analysis was to:

1. Clean and preprocess the dataset to handle missing or inconsistent data.
2. Calculate key performance indicators (KPIs) to measure the efficiency of the supply chain, financial performance, and customer satisfaction.
3. Provide actionable insights based on the KPIs to guide strategic business decisions.

Get data

- **We get the data from Kaggle** https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis
- We created a new database and set it up for data analysis using SQL

```
CREATE DATABASE SupplyChain
USE SupplyChain
GO
```

Project Analysis

Data exploration and cleaning using SQL

Listing the KPIs using SQL

BI Dashboard design using Power

Data exploration and cleaning using Python

Measuring Performance using Python

Data exploration and cleaning using SQL


```sql
--Exploring the data
SELECT * FROM DataCo

--Checking for duplicate values
SELECT COUNT (*) FROM DataCo AS duplicate_count
SELECT DISTINCT COUNT (*) FROM DataCo AS duplicate_count

--Checking for null values (Repeat for each column)
SELECT * FROM DataCo
WHERE Shipping_Mode iS NULL
 --Replacing missing values in Customer_Lname column
 UPDATE DataCo
 SET Customer_Lname = Customer_Fname
 WHERE Customer_Lname IS NULL

 --Replacing missing values in Customer_Zipcode column
 UPDATE DataCo
 SET Customer_City = 'Elk Grove',
 Customer_State = 'CA',
 Customer_Zipcode = '95758'
 WHERE Customer_Zipcode IS NULL AND Customer_State = '95758'
 UPDATE DataCo
 SET Customer_City = 'El Monte',
 Customer_State = 'CA',
 Customer_Zipcode = '91732'
 WHERE Customer_Zipcode IS NULL AND Customer_State = '91732'
```

Data exploration and cleaning using SQL

رواد مصر الرقمية

```sql
--Handling null values
SELECT
    COUNT(*) AS total_rows,
    SUM(CASE WHEN Order_Zipcode IS NULL THEN 1 ELSE 0 END) AS Order_Zipcode_nulls,
    SUM(CASE WHEN Product_Description IS NULL THEN 1 ELSE 0 END) AS Product_Description_nulls
FROM DataCo

--Replacing missing values in Order_Zipcode column
SELECT Order_City, Order_State, Order_Zipcode
FROM DataCo
WHERE Order_Zipcode IS NULL

WITH ZipcodeInference AS (
    SELECT Order_State, Order_City ,Order_Zipcode, COUNT(*) AS Zipcode_Count,
    ROW_NUMBER() OVER (PARTITION BY Order_State, Order_City ORDER BY COUNT(*) DESC) AS RN
    FROM DataCo
    WHERE Order_Zipcode IS NOT  NULL
    GROUP BY Order_State, Order_City ,Order_Zipcode
)
UPDATE DataCo
SET Order_Zipcode = (
    SELECT Order_Zipcode FROM ZipcodeInference
    WHERE ZipcodeInference.Order_State = DataCo.Customer_State
    AND ZipcodeInference.Order_City = DataCo.Customer_City
    AND RN = 1
    )
WHERE Order_Zipcode IS NULL
```

Data exploration and cleaning using SQL


```sql
--Deleting (Benefit_per_order) column as it hse the same values in (Order_Profit_Per_Order) column
SELECT * FROM DataCo
WHERE Order_Profit_Per_Order <> Benefit_per_order
ALTER TABLE DataCo
DROP COLUMN Benefit_per_order

--Deleting (Sales_per_customer) column as it has the same values in (Order_Item_Total) column
SELECT * FROM DataCo
WHERE Order_Item_Total <> Sales_per_customer
ALTER TABLE DataCo
DROP COLUMN Sales_per_customer

--Deleting columns with no data or useless
ALTER TABLE DataCo
DROP COLUMN Customer_Email, Customer_Password, Product_Description, Product_Image
```

Listing the KPIs using SQL


```sql
--1. Order Accuracy Rate
SELECT
CONCAT(COUNT(DISTINCT Order_Id) * 100 / (SELECT COUNT(DISTINCT Order_Id) FROM DataCo),'%')
AS "Order Accuracy Rate"
FROM DataCo
WHERE Order_Status IN ('COMPLETE', 'CLOSED')

--2. On-time Delivery Rate
SELECT
CONCAT(COUNT(DISTINCT Order_Id) * 100 / (SELECT COUNT(DISTINCT Order_Id) FROM DataCo),'%')
AS "On-time Delivery Rate"
FROM DataCo
WHERE Days_for_shipping_real <= Days_for_shipment_scheduled

--3. Perfect Order Rate
SELECT
CONCAT(COUNT(DISTINCT Order_Id) * 100 / (SELECT COUNT(DISTINCT Order_Id) FROM DataCo), '%')
AS "Perfect Order Rate"
FROM DataCo
WHERE Order_Status IN ('COMPLETE', 'CLOSED')
AND Days_for_shipping_real <= Days_for_shipment_scheduled

--4. Order Lead Time
SELECT
AVG(DATEDIFF(DAY, order_date_DateOrders, shipping_date_DateOrders))
AS "Lead Time"
FROM DataCo
```

Listing the KPIs using SQL

```sql
--5. Order Cycle Time
SELECT
AVG(DATEDIFF(DAY, order_date_DateOrders, DATEADD(DAY, Days_for_shipping_real, shipping_date_DateOrders)))
AS "Real Cycle Time"
FROM DataCo

--Cycle time with schedualed date
SELECT
AVG(DATEDIFF(DAY, order_date_DateOrders, DATEADD(DAY, Days_for_shipment_scheduled, shipping_date_DateOrders)))
AS "Scheduled Cycle time"
FROM DataCo
```

```sql
--Late Orders
SELECT Department_Name,
COUNT(Late_delivery_risk) as "Late Delivery"
FROM DataCo
WHERE Late_delivery_risk = 1
GROUP BY Department_Name
ORDER BY "Late Delivery" DESC
```

Listing the KPIs using SQL


```sql
--Real vs Scheduled Shipping Days per Shipping Mode
SELECT Shipping_Mode,
AVG(Days_for_shipment_scheduled) AS "Average Scheduled Shipping Days",
AVG(Days_for_shipping_real) AS "Average Real Shipping Days"
FROM DataCo
GROUP BY Shipping_Mode
ORDER BY "Average Scheduled Shipping Days"

--Real vs Scheduled Cycle time per Shipping mode
SELECT
Shipping_Mode,
AVG(DATEDIFF(DAY, order_date_DateOrders, DATEADD(DAY, Days_for_shipment_scheduled, shipping_date_DateOrders))
AS "Scheduled Cycle time",
AVG(DATEDIFF(DAY, order_date_DateOrders, DATEADD(DAY, Days_for_shipping_real, shipping_date_DateOrders)))
AS "Real Cycle time"
FROM DataCo
GROUP BY Shipping_Mode
ORDER BY "Scheduled Cycle time"

--Real vs Scheduled Cycle time per Customer City
SELECT
Customer_City,
AVG(DATEDIFF(DAY, order_date_DateOrders, DATEADD(DAY, Days_for_shipment_scheduled, shipping_date_DateOrders))
AS "Scheduled Cycle time",
AVG(DATEDIFF(DAY, order_date_DateOrders, DATEADD(DAY, Days_for_shipping_real, shipping_date_DateOrders)))
AS "Real Cycle time"
FROM DataCo
GROUP BY Customer_City
ORDER BY "Real Cycle time"
```

Listing the KPIs using SQL


```sql
--6. Important KPIs
SELECT
SUM(Order_Item_Quantity) as "Total Order Quantity",
COUNT(DISTINCT Order_Id) AS "Number of Orders",
FORMAT(SUM(Sales), 'C', 'en-US') AS "Total Sales without Discount",
FORMAT(SUM(Order_Item_Discount), 'C', 'en_US') AS "Total Discount",
FORMAT(SUM(Order_Item_Total), 'C', 'en_US') AS "Total Sales with Discount",
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en_US') AS "Total Profit"
FROM DataCo

--KPIs per Type
SELECT Type,
SUM(Order_Item_Quantity) as "Total Order Quantity",
COUNT(DISTINCT Order_Id) AS "Number of Orders",
FORMAT(SUM(Sales), 'C', 'en-US') AS "Total Sales without Discount",
FORMAT(SUM(Order_Item_Discount), 'C', 'en_US') AS "Total Discount",
FORMAT(SUM(Order_Item_Total), 'C', 'en_US') AS "Total Sales with Discount",
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en_US') AS "Total Profit"
FROM DataCo
GROUP BY Type
ORDER BY SUM(Order_Item_Total) DESC
```

Listing the KPIs using SQL


```sql
--KPIs per Customer Segment
SELECT Customer_Segment,
SUM(Order_Item_Quantity) as "Total Order Quantity",
COUNT(DISTINCT Order_Id) AS "Number of Orders",
FORMAT(SUM(Sales), 'C', 'en-US') AS "Total Sales without Discount",
FORMAT(SUM(Order_Item_Discount), 'C', 'en_US') AS "Total Discount",
FORMAT(SUM(Order_Item_Total), 'C', 'en_US') AS "Total Sales with Discount",
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en_US') AS "Total Profit"
FROM DataCo
GROUP BY Customer_Segment
ORDER BY SUM(Order_Item_Total) DESC

--KPIs per Department
SELECT Department_Name,
SUM(Order_Item_Quantity) as "Total Order Quantity",
COUNT(DISTINCT Order_Id) AS "Number of Orders",
FORMAT(SUM(Sales), 'C', 'en-US') AS "Total Sales without Discount",
FORMAT(SUM(Order_Item_Discount), 'C', 'en_US') AS "Total Discount",
FORMAT(SUM(Order_Item_Total), 'C', 'en_US') AS "Total Sales with Discount",
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en_US') AS "Total Profit"
FROM DataCo
GROUP BY Department_Name
ORDER BY SUM(Order_Item_Total) DESC
```

Listing the KPIs using SQL

```sql
--KPIs per Category Name
SELECT Category_Name,
SUM(Order_Item_Quantity) as "Total Order Quantity",
COUNT(DISTINCT Order_Id) AS "Number of Orders",
FORMAT(SUM(Sales), 'C', 'en-US') AS "Total Sales without Discount",
FORMAT(SUM(Order_Item_Discount), 'C', 'en_US') AS "Total Discount",
FORMAT(SUM(Order_Item_Total), 'C', 'en_US') AS "Total Sales with Discount",
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en_US') AS "Total Profit"
FROM DataCo
GROUP BY Category_Name
ORDER BY SUM(Order_Item_Total) DESC

--KPIs per Region
SELECT Order_Region,
SUM(Order_Item_Quantity) as "Total Order Quantity",
COUNT(DISTINCT Order_Id) AS "Number of Orders",
FORMAT(SUM(Sales), 'C', 'en-US') AS "Total Sales without Discount",
FORMAT(SUM(Order_Item_Discount), 'C', 'en_US') AS "Total Discount",
FORMAT(SUM(Order_Item_Total), 'C', 'en_US') AS "Total Sales with Discount",
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en_US') AS "Total Profit"
FROM DataCo
GROUP BY Order_Region
ORDER BY SUM(Order_Item_Total) DESC
```

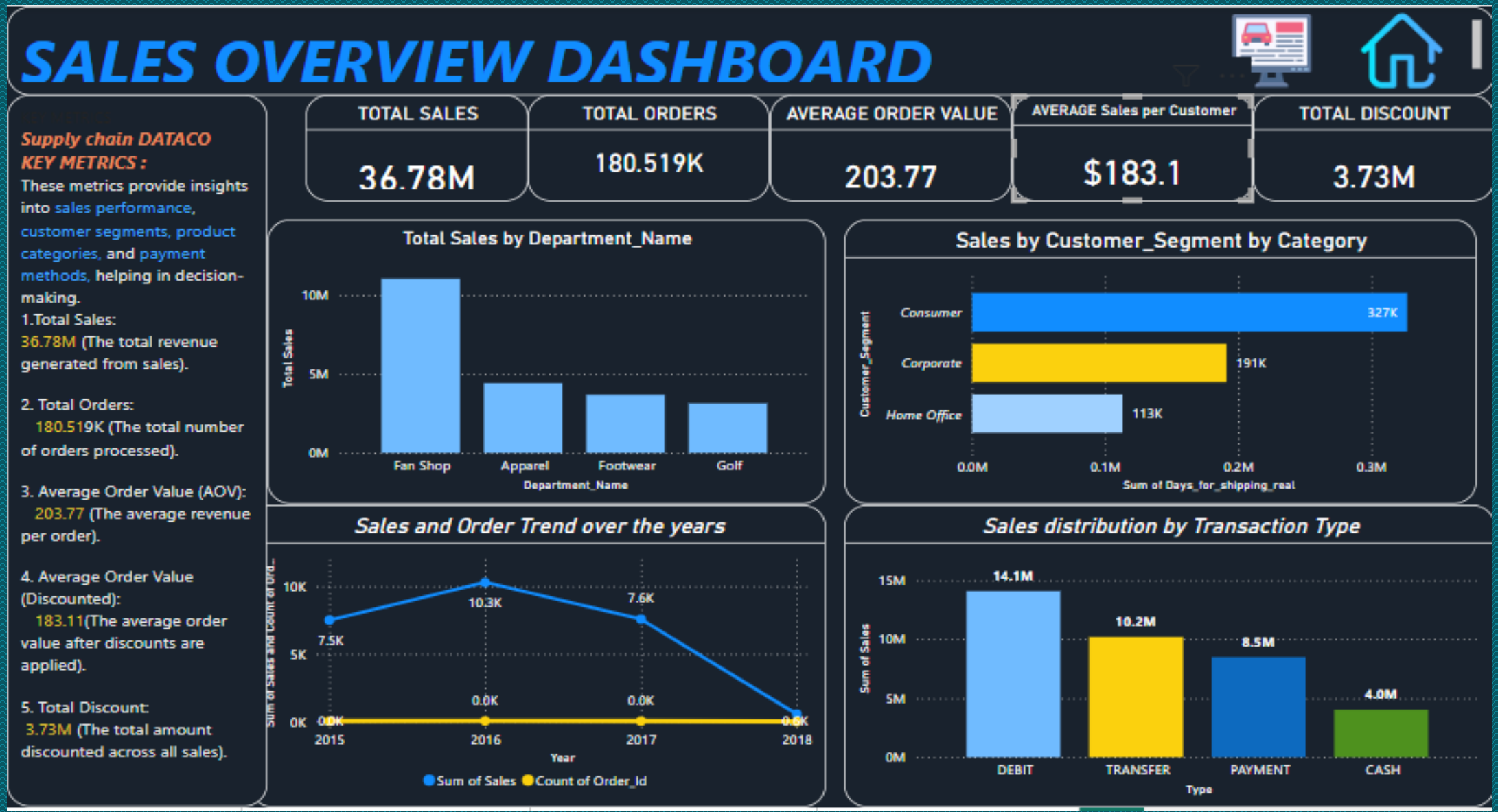
Listing the KPIs using SQL


```sql
--7.Average Order Value (AOV)
SELECT
FORMAT(SUM(Order_Item_Total) / COUNT(Order_Item_Id), 'C', 'en-US') AS "Average Order Value"
FROM DataCo

--8.Lost Sales
SELECT FORMAT(SUM(Order_Item_Total), 'C', 'en-US') AS "Lost Sales"
FROM DataCo
WHERE Order_Status = 'CANCELED'
```

Listing the KPIs using SQL

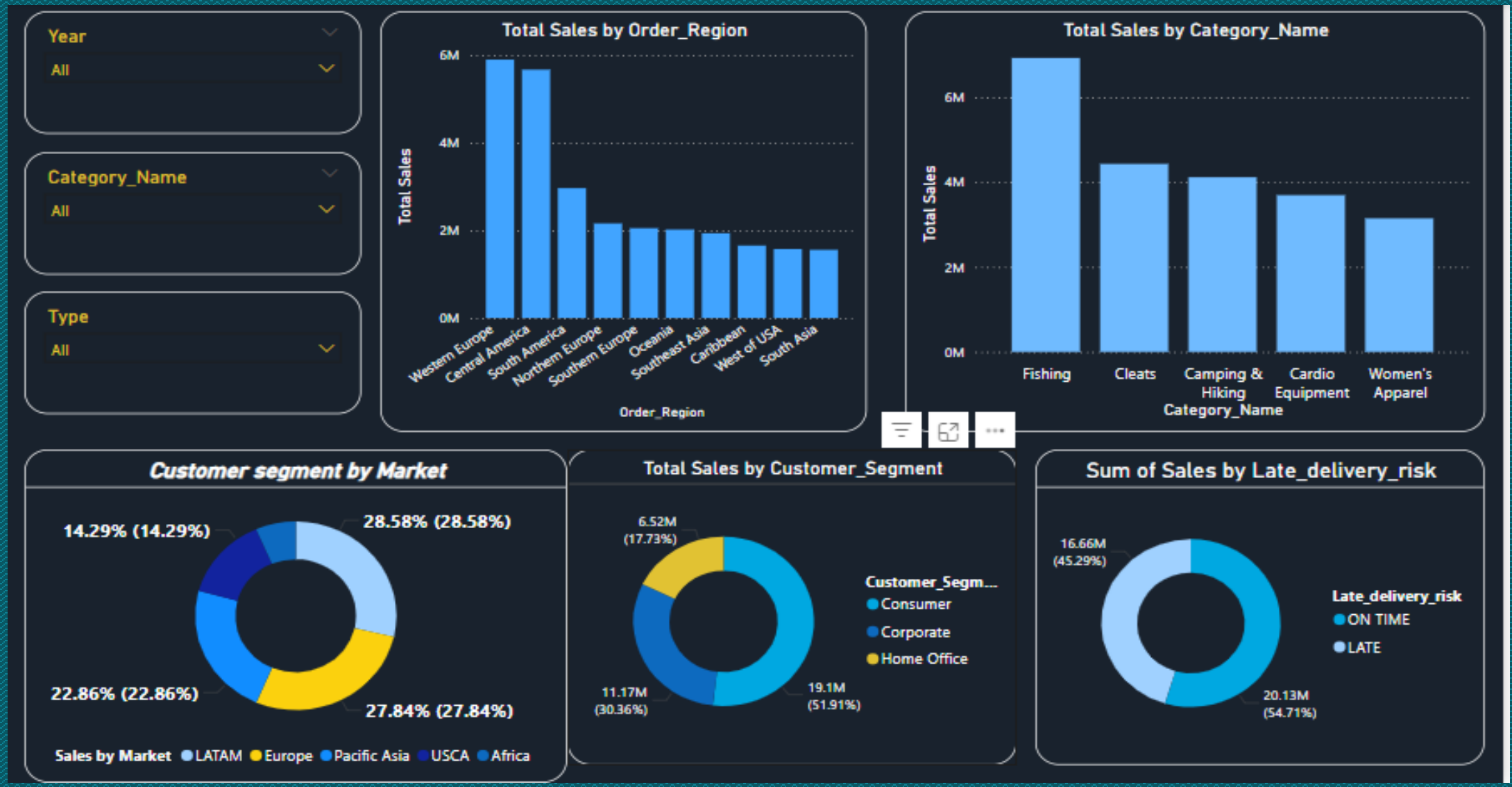

```sql
--9.Return Rate
SELECT
COUNT(DISTINCT Order_Id) AS "Total Returned Orders",
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en-US') AS "Total Returned Money",
CONCAT(COUNT(DISTINCT Order_Id) * 100 / (SELECT COUNT(DISTINCT Order_Id) FROM DataCo), '%')
AS "Return Rate"
FROM DataCo
WHERE Order_Profit_Per_Order < 0
AND Delivery_Status <> 'Shipping Canceled'

--Return Rate per Category Name
SELECT TOP(5)
Category_Name,
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en-US') AS "Total Returned Money",
CONCAT(COUNT(DISTINCT Order_Id) * 100 / (SELECT COUNT(DISTINCT Order_Id) FROM DataCo), '%')
AS "Return Rate"
FROM DataCo
WHERE Order_Profit_Per_Order < 0
AND Delivery_Status <> 'Shipping Canceled'
GROUP BY Category_Name
ORDER BY "Return Rate" DESC

--Return Rate per Shipping Mode
SELECT
Shipping_Mode,
FORMAT(SUM(Order_Profit_Per_Order), 'C', 'en-US') AS "Total Returned Money",
CONCAT(COUNT(DISTINCT Order_Id) * 100 / (SELECT COUNT(DISTINCT Order_Id) FROM DataCo), '%')
AS "Return Rate"
FROM DataCo
WHERE Order_Profit_Per_Order < 0
AND Delivery_Status <> 'Shipping Canceled'
GROUP BY Shipping_Mode
ORDER BY SUM(Order_Profit_Per_Order)
```

• Dashboard design using Power BI•

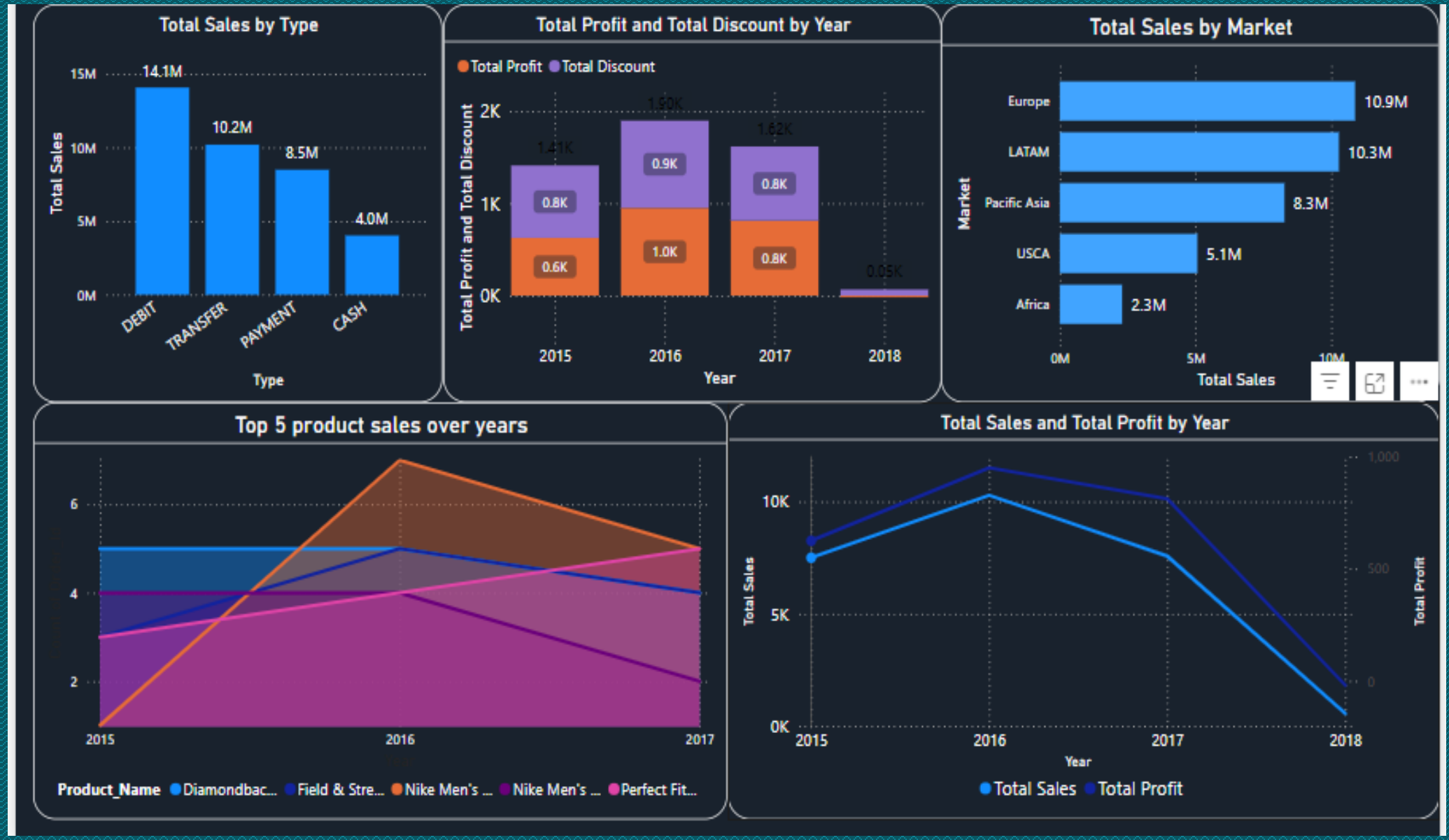


Dashboard design using Power BI

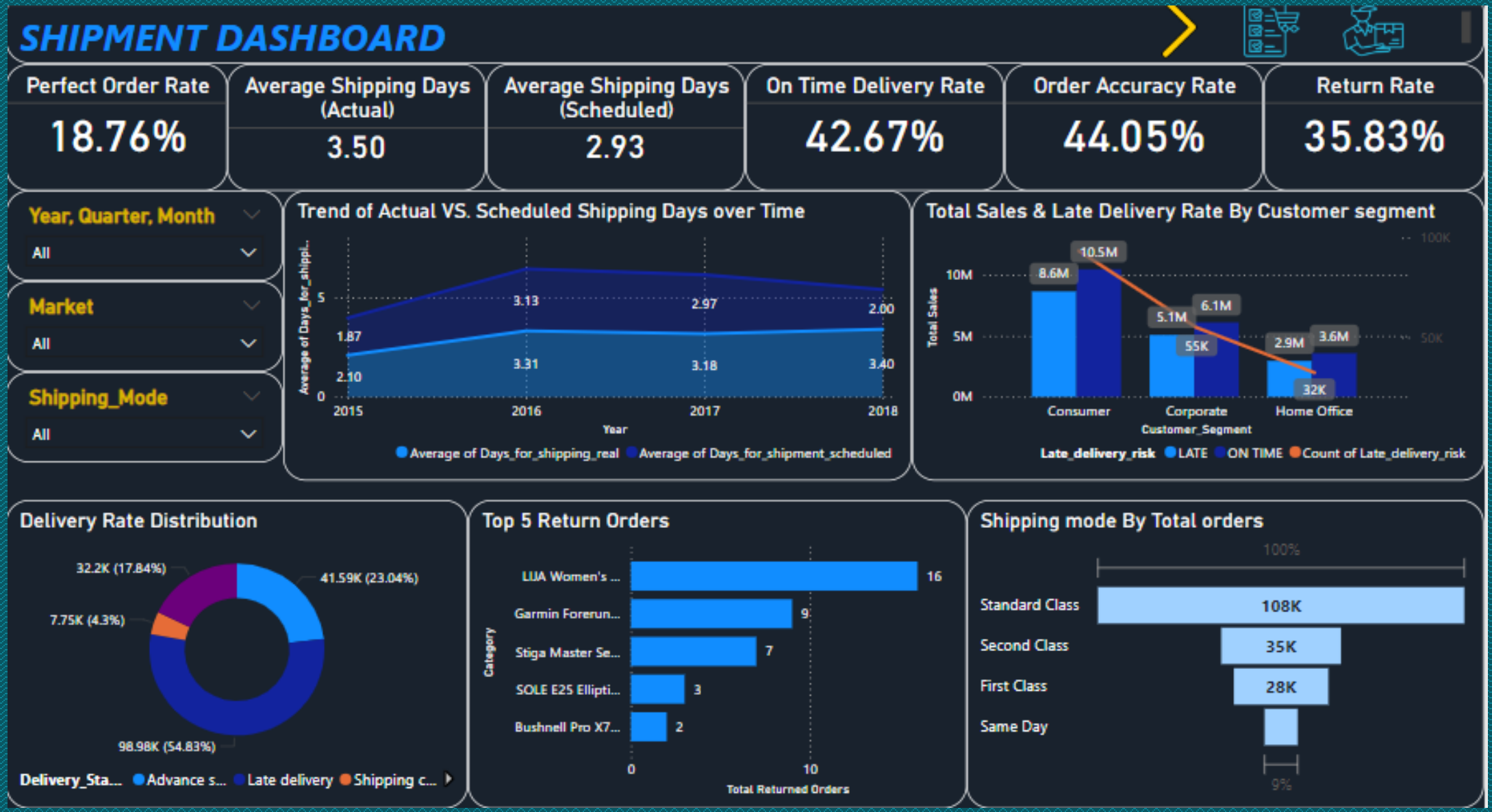
رواد مصر الرقمية



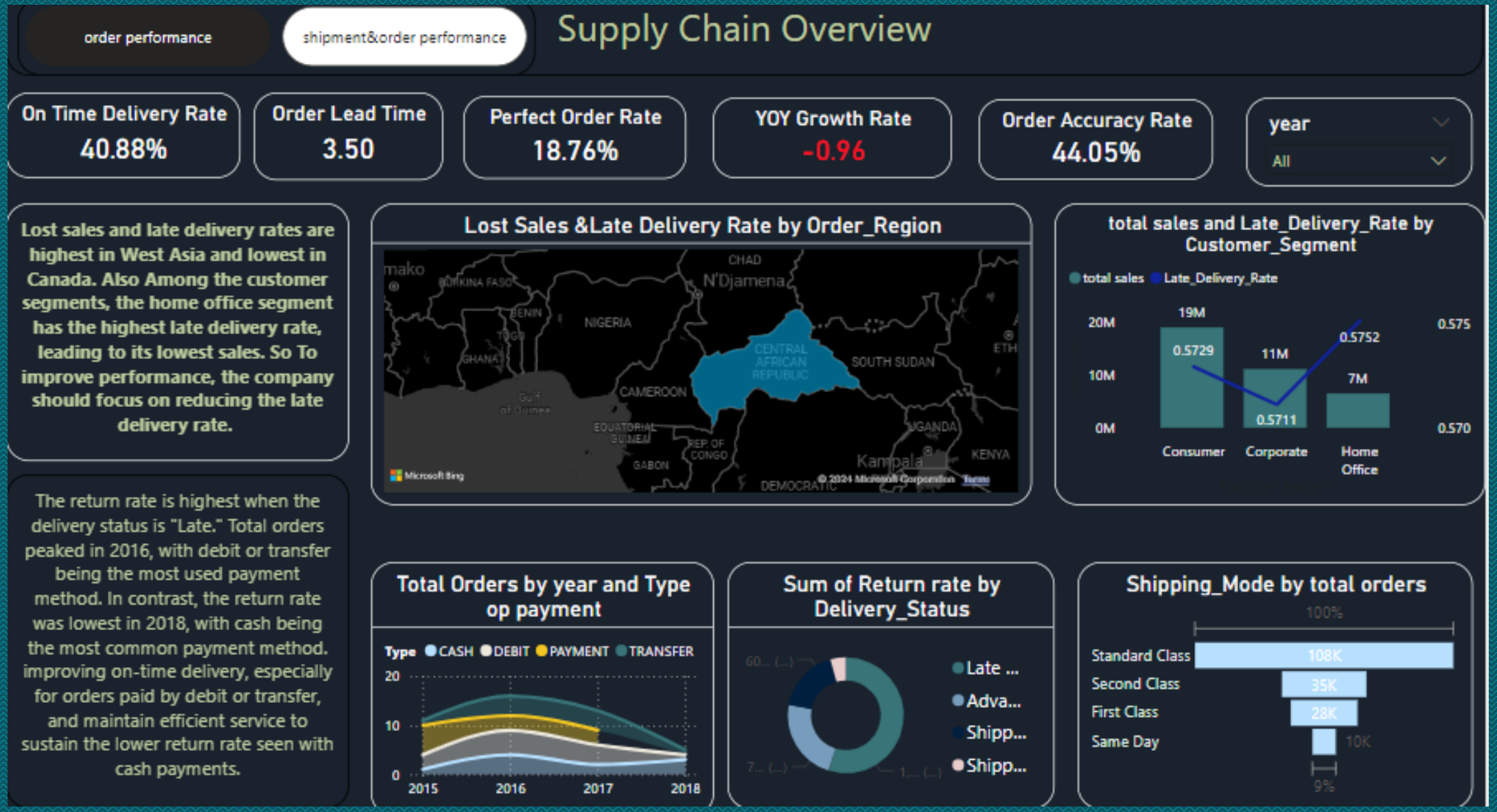
Dashboard design using Power BI



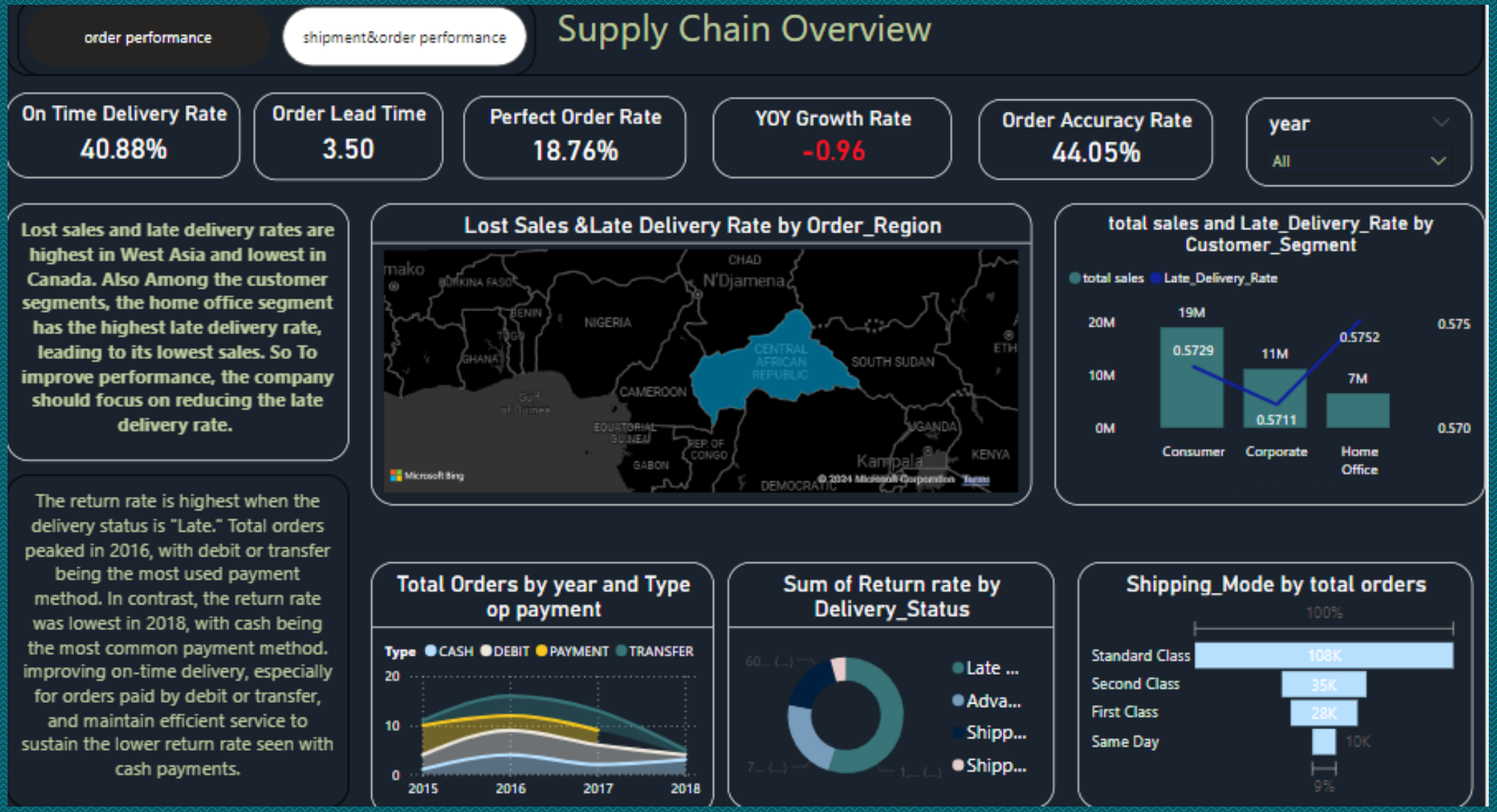
Dashboard design using Power BI



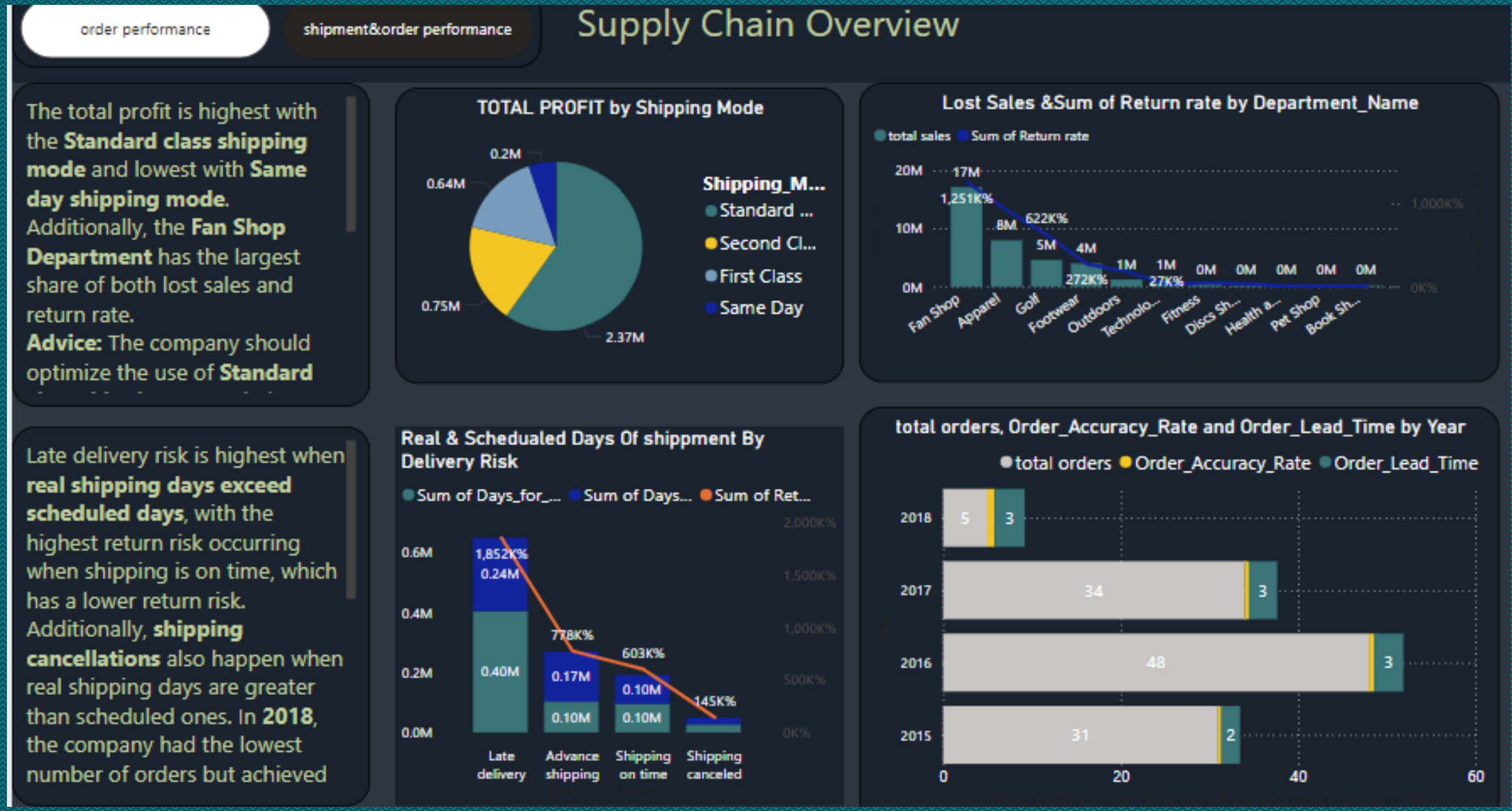
Dashboard design using Power BI



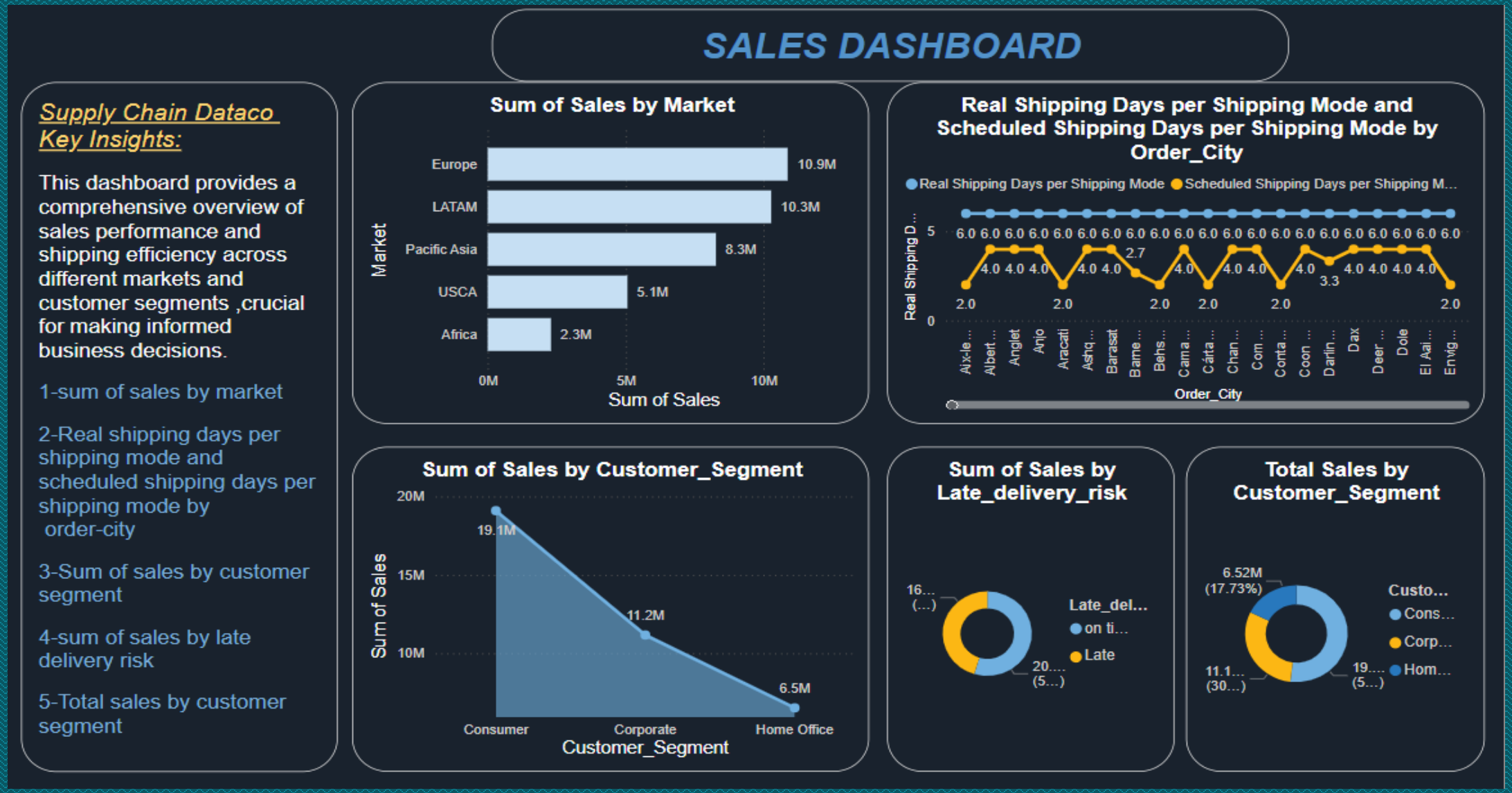
Dashboard design using Power BI



Dashboard design using Power BI



Dashboard design using Power BI



Data exploration and cleaning using Python

رواد مصر الرقمية

Firstly, prepare libraries that we need.

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Load and clean the data

[2]:
```
df = pd.read_csv('DataCoSupplyChainDataset.csv', parse_dates=['order date (DateOrders)'],encoding='latin1')
df = df.sort_values('order date (DateOrders)')
```

Columns name

[3]:
```
df.columns
```

[3]:
```
Index(['Type', 'Days for shipping (real)', 'Days for shipment (scheduled)',
       'Benefit per order', 'Sales per customer', 'Delivery Status',
       'Late_delivery_risk', 'Category Id', 'Category Name', 'Customer City',
       'Customer Country', 'Customer Email', 'Customer Fname', 'Customer Id',
       'Customer Lname', 'Customer Password', 'Customer Segment',
       'Customer State', 'Customer Street', 'Customer Zipcode',
       'Department Id', 'Department Name', 'Latitude', 'Longitude', 'Market',
       'Order City', 'Order Country', 'Order Customer Id',
       'order date (DateOrders)', 'Order Id', 'Order Item Cardprod Id',
       'Order Item Discount', 'Order Item Discount Rate', 'Order Item Id',
       'Order Item Product Price', 'Order Item Profit Ratio',
       'Order Item Quantity', 'Sales', 'Order Item Total',
       'Order Profit Per Order', 'Order Region', 'Order State', 'Order Status',
       'Order Zipcode', 'Product Card Id', 'Product Category Id',
       'Product Description', 'Product Image', 'Product Name', 'Product Price',
       'Product Status', 'shipping date (DateOrders)', 'Shipping Mode'],
      dtype='object')
```

Drop Critical information and unnecessary columns.

[4]:
```
df.drop(['Product Description','Customer Password','Customer Email',
         'Customer Email','Order Zipcode', 'Customer Zipcode','Customer Lname',
         'Product Image','Customer Fname'], axis = 1, inplace = True)
```

Data exploration and cleaning using Python

رواد مصر الرقمية

Create year column from order date columns.

[8]:
```
df['year'] = df['order date (DateOrders)'].dt.year
```

Create month column from order date columns.

[9]:
```
df['month'] = df['order date (DateOrders)'].dt.month
```

Get Descriptive statistics of each column.

[10]:
```
df.describe()
```

[10]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk	Category Id	Customer Id	Department Id	Latitude	Longitude	...
count	180519.000000	180519.000000	180519.000000	180519.000000	180519.000000	180519.000000	180519.000000	180519.000000	180519.000000	180519.000000	...
mean	3.497654	2.931847	21.974989	183.107609	0.548291	31.851451	6691.379495	5.443460	29.719955	-84.915675	...
min	0.000000	0.000000	-4274.979980	7.490000	0.000000	2.000000	1.000000	2.000000	-33.937553	-158.025986	...
25%	2.000000	2.000000	7.000000	104.379997	0.000000	18.000000	3258.500000	4.000000	18.265432	-98.446312	...
50%	3.000000	4.000000	31.520000	163.990005	1.000000	29.000000	6457.000000	5.000000	33.144863	-76.847908	...
75%	5.000000	4.000000	64.800003	247.399994	1.000000	45.000000	9779.000000	7.000000	39.279617	-66.370583	...
max	6.000000	4.000000	911.799988	1939.989990	1.000000	76.000000	20757.000000	12.000000	48.781933	115.263077	...
std	1.623722	1.374449	104.433526	120.043670	0.497664	15.640064	4162.918106	1.629246	9.813646	21.433241	...

8 rows × 29 columns

Data exploration and cleaning using Python

Summarize Net Sales with discount and Sales in mean and total table group by year.

For more accurate need group rows using order id to get one value represent each order.

[11]:
```
grouped_df = df.groupby('Order Id').first().reset_index()
pivot_table = grouped_df.pivot_table(values=['Sales','Order Item Total'], index=['year'], aggfunc='mean')
pivot_table2 = df.pivot_table(values=['Sales','Order Item Total'], index=['year'], aggfunc='sum')
pivot_table.columns = ['Net Sales', 'Sales']
pivot_table2.columns = ['Net Sales', 'Sales']
print("Pivot Table: Mean of Net Sales and Sales Grouped by Year")
print(pivot_table)
print("Pivot Table2: Total of Net Sales and Sales Grouped by Year")
print(pivot_table2)
```

```
Pivot Table: Mean of Net Sales and Sales Grouped by Year
      Net Sales       Sales
year
2015  177.220489  197.191756
2016  177.907361  198.013097
2017  220.643208  245.478559
2018  140.344819  156.217671
Pivot Table2: Total of Net Sales and Sales Grouped by Year
        Net Sales         Sales
year
2015  1.108954e+07  1.234083e+07
2016  1.105600e+07  1.230382e+07
2017  1.061091e+07  1.180844e+07
2018  2.979521e+05  3.316501e+05
```

Measuring Performance using Python

رواد مصر الرقمية

Calculate On-Time Shipping Rate

[13]:
```
df['on_time_shipping'] = df['Days for shipping (real)'] <= df['Days for shipment (scheduled)']
on_time_shipping_rate = df['on_time_shipping'].mean() * 100
print(f"On-Time Shipping Rate: {on_time_shipping_rate:.2f}%")
```

```
On-Time Shipping Rate: 42.72%
```

Calculate Average Delay Rate in Shipping

[14]:
```
df['shipping_delay_status'] = (df['Days for shipping (real)'] > df['Days for shipment (scheduled)'])
average_delay_rate = df['shipping_delay_status'].mean() * 100
print(f"Delay in Shipping Rate: {average_delay_rate:.2f}%")
```

```
Delay in Shipping Rate: 57.28%
```

Calculate Shipping Variability

[15]:
```
df['shipping_delay'] = (df['Days for shipping (real)'] - df['Days for shipment (scheduled)'])
average_delay = df['shipping_delay'].mean()
shipping_variability = df['shipping_delay'].std()
print(f"Shipping Variability: {shipping_variability:1f} days")
```

```
Shipping Variability: 1.490966 days
```

Measuring Performance using Python

Grouping by Item name and summing the quantities and total costs

[16]:
```
grouped_item = df.groupby('Product Name').agg({'Order Item Quantity': 'sum', 'Order Item Total': 'sum'}).reset_index()
grouped_item
```

[16]:

	Product Name	Order Item Quantity	Order Item Total
0	Adult dog supplies	492	37318.299847
1	Baby sweater	207	10957.400143
2	Bag Boy Beverage Holder	845	19009.969910
3	Bag Boy M330 Push Cart	208	14981.660008
4	Bowflex SelectTech 1090 Dumbbells	10	5171.899902
...	...	...	...
113	adidas Kids' F5 Messi FG Soccer Cleat	781	24626.510213
114	adidas Men's F10 Messi TRX FG Soccer Cleat	939	50354.210668
115	adidas Men's Germany Black Crest Away Tee	859	19279.919983
116	adidas Youth Germany Black/Red Away Match Soc	969	61037.900009
117	insta-bed Neverflat Air Mattress	60	8011.650148

118 rows × 3 columns

Measuring Performance using Python

رواد مصر الرقمية

Sorting the DataFrame based on Quantity and selecting the top 10 items

[17]:
```
top_10_df = grouped_item.sort_values(by='Order Item Quantity', ascending=False).head(10)
top_10_df
```

[17]:

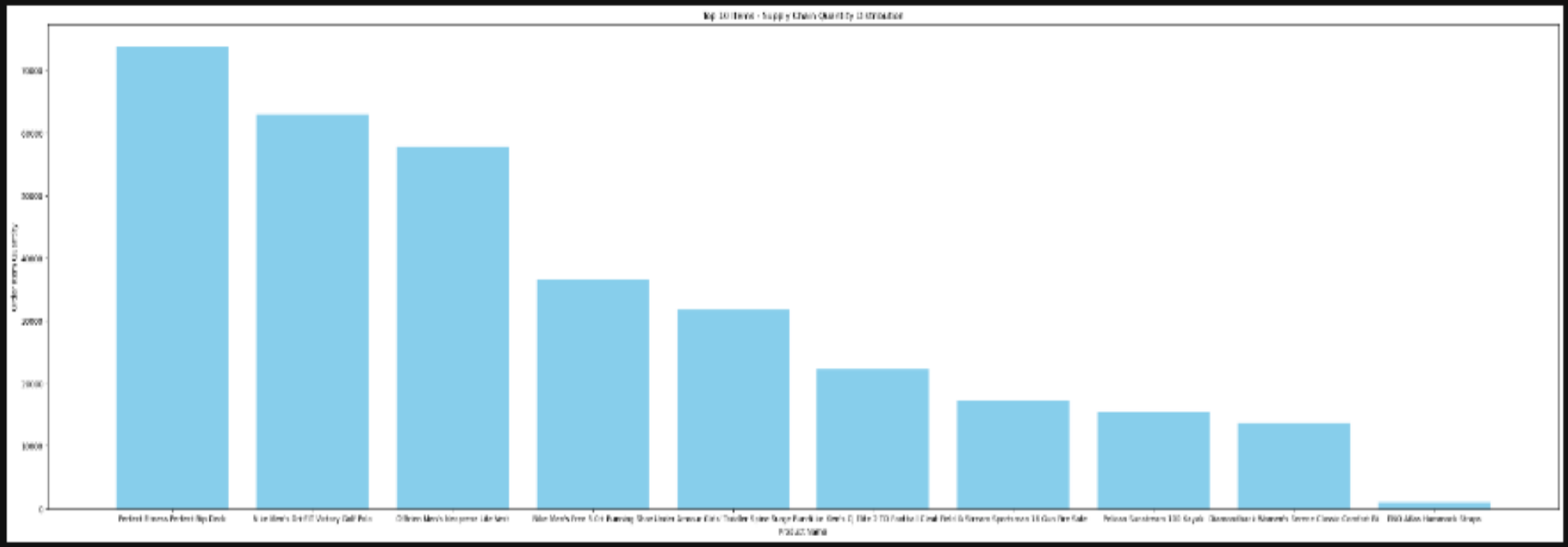
	Product Name	Order Item Quantity	Order Item Total
71	Perfect Fitness Perfect Rip Deck	73698	3.973180e+06
59	Nike Men's Dri-FIT Victory Golf Polo	62956	2.828708e+06
67	O'Brien Men's Neoprene Life Vest	57803	2.596454e+06
61	Nike Men's Free 5.0+ Running Shoe	36680	3.295693e+06
102	Under Armour Girls' Toddler Spine Surge Runni	31735	1.140771e+06
56	Nike Men's CJ Elite 2 TD Football Cleat	22246	2.598494e+06
24	Field & Stream Sportsman 16 Gun Fire Safe	17325	6.226935e+06
70	Pelican Sunstream 100 Kayak	15500	2.785518e+06
21	Diamondback Women's Serene Classic Comfort Bi	13729	3.700784e+06
22	ENO Atlas Hammock Straps	998	2.687578e+04

Measuring Performance using Python

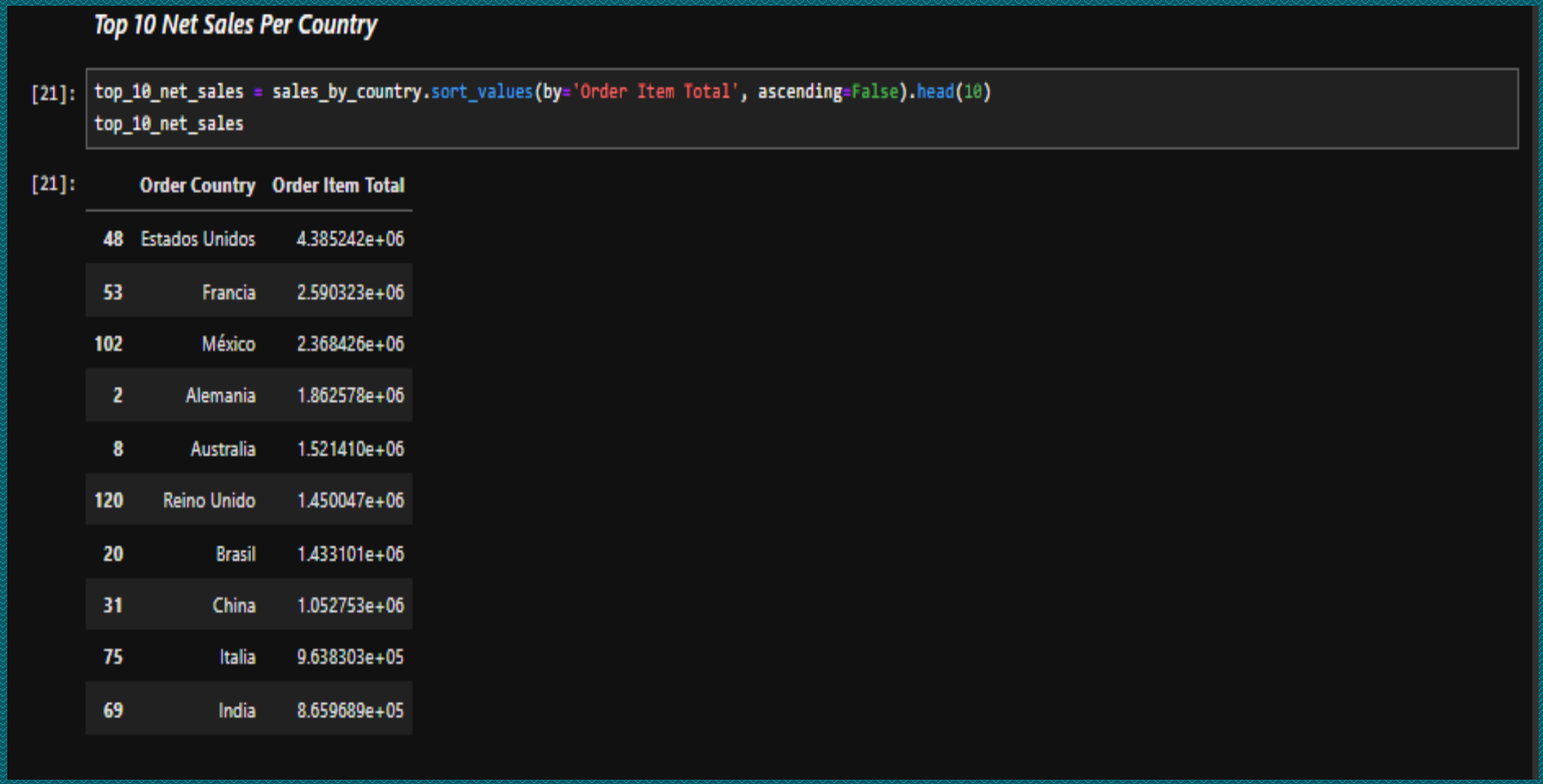
رواد مصر الرقمية

Visualizing the Pivot table of top 10 product

[19]:
```
plt.figure(figsize=(40, 10))
plt.bar(top_10_df['Product Name'], top_10_df['Order Item Quantity'], color='skyblue')
plt.xlabel('Product Name')
plt.ylabel('Order Item Quantity')
plt.title('Top 10 Items - Supply Chain Quantity Distribution')
plt.show()
```



Measuring Performance using Python



Top 10 Net Sales Per Country

[21]:
```
top_10_net_sales = sales_by_country.sort_values(by='Order Item Total', ascending=False).head(10)
top_10_net_sales
```

[21]:

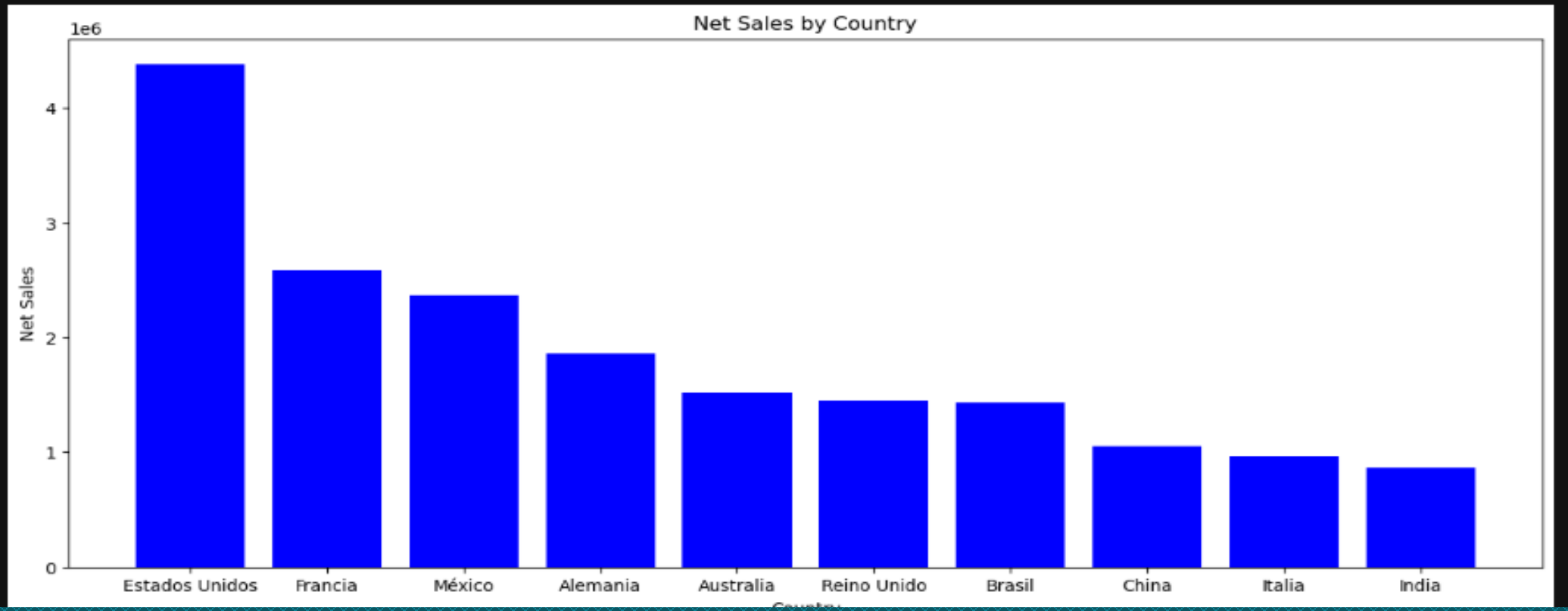
	Order Country	Order Item Total
48	Estados Unidos	4.385242e+06
53	Francia	2.590323e+06
102	México	2.368426e+06
2	Alemania	1.862578e+06
8	Australia	1.521410e+06
120	Reino Unido	1.450047e+06
20	Brasil	1.433101e+06
31	China	1.052753e+06
75	Italia	9.638303e+05
69	India	8.659689e+05

Measuring Performance using Python

رواد مصر الرقمية

Visualizing Net Sales by Country

[22]:
```
plt.figure(figsize=(15, 6))
plt.bar(top_10_net_sales['Order Country'], top_10_net_sales['Order Item Total'], color='blue')
plt.xlabel('Country')
plt.ylabel('Net Sales')
plt.title('Net Sales by Country')
plt.show()
```



Measuring Performance using Python

رواد مصر الرقمية

Quantity Shipped Over Time

Create Actual Delivery Date by adding Actual Shipping Days to Shipping Date

[23]:
```
# Converting DateOrders to datetime
df['shipping date (DateOrders)'] = pd.to_datetime(df['shipping date (DateOrders)'])

# Ensure 'Days for shipping' is of integer type
df['Days for shipping (real)'] = df['Days for shipping (real)'].astype(int)

# Adding Delivery Date by adding Days for shipping to DateOrders
df['Actual Delivery Date'] = df['shipping date (DateOrders)'] + pd.to_timedelta(df['Days for shipping (real)'], unit='D')
```

Calculating Actual Cycle Time as the difference between Actual Delivery Date and Order Date

[24]:
```
df['Actual Cycle Time (Days)'] = (df['Actual Delivery Date'] - df['order date (DateOrders)']).dt.days
```

Grouping by Shipping Mode and calculating average cycle time

[25]:
```
grouped = df.groupby('Shipping Mode')['Actual Cycle Time (Days)'].mean().reset_index()
```

Measuring Performance using Python

Sorting by mean of Actual delivery days

[26]:
```
sorted_grouped = grouped.sort_values(by='Actual Cycle Time (Days)')
sorted_grouped
```

[26]:

	Shipping Mode	Actual Cycle Time (Days)
1	Same Day	0.478279
0	First Class	4.000000
2	Second Class	7.981656
3	Standard Class	7.991815

Pivot table for mean of Actual Shipping Days grouping by Customer Segment and Shipping Mode

[27]:
```
actual_seg = df.pivot_table(values='Actual Cycle Time (Days)', index=['Customer Segment','Shipping Mode'], aggfunc='mean').reset_index()
```

Measuring Performance using Python

Sort pivot table by Customer Segment and Schedual Shipping Days

[28]:
```
sorted_actual = actual_seg.sort_values(by=['Customer Segment','Actual Cycle Time (Days)'], ascending=True)
sorted_actual
```

[28]:

	Customer Segment	Shipping Mode	Actual Cycle Time (Days)
1	Consumer	Same Day	0.486014
0	Consumer	First Class	4.000000
2	Consumer	Second Class	7.965105
3	Consumer	Standard Class	8.008329
5	Corporate	Same Day	0.484034
4	Corporate	First Class	4.000000
7	Corporate	Standard Class	7.945246
6	Corporate	Second Class	7.993059
9	Home Office	Same Day	0.442999
8	Home Office	First Class	4.000000
10	Home Office	Second Class	8.009252
11	Home Office	Standard Class	8.022687

Measuring Performance using Python

Creating Schedualed Delivery Date by adding Schedualed Shipping Days to Shipping Date

[29]:
```
# Converting DateOrders to datetime
df['shipping date (DateOrders)'] = pd.to_datetime(df['shipping date (DateOrders)'])

# Ensure 'Days for shipping' is of integer type
df['Days for shipment (scheduled)'] = df['Days for shipment (scheduled)'].astype(int)

# Adding Delivery Date by adding Days for shipping to DateOrders
df['Schedualed Delivery Date'] = df['shipping date (DateOrders)'] + pd.to_timedelta(df['Days for shipment (scheduled)'], unit='D')
```

Calculating Schedualed Cycle Time as the difference between Schedualed Delivery Date and Order Date

[30]:
```
df['Schedualed Cycle Time (Days)'] = (df['Schedualed Delivery Date'] - df['order date (DateOrders)']).dt.days
```

Grouping by Shipping Mode and calculating average Schedualed cycle time ¶

[31]:
```
grouped = df.groupby('Shipping Mode')['Schedualed Cycle Time (Days)'].mean().reset_index()
```

Measuring Performance using Python

Sorting by mean of Schedualed delivery days

[32]:
```
Schedualed_sorted_grouped = grouped.sort_values(by='Schedualed Cycle Time (Days)')
Schedualed_sorted_grouped
```

[32]:

	Shipping Mode	Schedualed Cycle Time (Days)
1	Same Day	0.000000
0	First Class	3.000000
2	Second Class	5.990828
3	Standard Class	7.995907

Pivot table for mean of Schedualed Shipping Days grouping by Customer Segment and Shipping Mode

[33]:
```
schedualed_seg = df.pivot_table(values='Schedualed Cycle Time (Days)', index=['Customer Segment','Shipping Mode'], aggfunc='mean').reset_index()
```

Measuring Performance using Python

Sort pivot table by Customer Segment and Schedual Shipping Days

[34]:
```
sorted_schedual = schedualed_seg.sort_values(by=['Customer Segment','Schedualed Cycle Time (Days)'], ascending=True)
sorted_schedual
```

[34]:

	Customer Segment	Shipping Mode	Schedualed Cycle Time (Days)
1	Consumer	Same Day	0.000000
0	Consumer	First Class	3.000000
2	Consumer	Second Class	5.982553
3	Consumer	Standard Class	8.004164
5	Corporate	Same Day	0.000000
4	Corporate	First Class	3.000000
6	Corporate	Second Class	5.996530
7	Corporate	Standard Class	7.972623
9	Home Office	Same Day	0.000000
8	Home Office	First Class	3.000000
10	Home Office	Second Class	6.004626
11	Home Office	Standard Class	8.011344

Measuring Performance using Python

Pivot table for mean of Net Sales grouping by Customer Segment and Shipping Mode

[35]:
```
pivot_seg = df.pivot_table(values='Order Item Total', index=['Customer Segment','Shipping Mode'], aggfunc='mean').reset_index()
```

Sort Pivot table by Customer Segment and Net Sales

[36]:
```
sorted_pivot_seg = pivot_seg.sort_values(by=['Customer Segment','Order Item Total'], ascending=True)
sorted_pivot_seg
```

[36]:

	Customer Segment	Shipping Mode	Order Item Total
1	Consumer	Same Day	179.686130
2	Consumer	Second Class	183.066929
3	Consumer	Standard Class	183.912382
0	Consumer	First Class	184.239123
5	Corporate	Same Day	177.646207
6	Corporate	Second Class	181.855328
4	Corporate	First Class	183.484108
7	Corporate	Standard Class	183.881321
10	Home Office	Second Class	180.340911
9	Home Office	Same Day	180.718626
8	Home Office	First Class	180.781003
11	Home Office	Standard Class	182.679490

Results and Insights

- Cleaning the data improved its usability for analysis.
- KPIs provided clear insights into supply chain efficiency.
- Key observations:
  - Moderate On-time Delivery Rate
  - Notable Return Rate in specific categories
  - Significant Lost Sales due to cancellations

Key Performance Indicators (KPIs)

- Order Accuracy Rate: 44%
- On-time Delivery Rate: 42%
- Perfect Order Rate: 18%
- Average Order Value (AOV): $183.11
- Lost Sales: $668,244.99
- Return Rate: 35%

Description

The dataset used in this analysis is from the DataCo Supply Chain. It includes various attributes like orders, customers, products, shipping, and financial performance. Key focus areas: customer details, product pricing, shipping times, and financial metrics

Recommendations

Optimize Shipping Processes:

- Address shipping bottlenecks and improve supplier coordination.

Reduce Order Cancellations:

-Investigate cancellation reasons and enhance customer communication.

Improve Product Quality:

- Focus on categories with high return rates.

Enhance Customer Segmentation:

- Use insights to tailor marketing campaigns to high-performing segments.

Thank You

24Slides