



Data Analyst Specialist

CAI1_DAT1_G12d

DEPI Final Project

Store Sales Dataset Analysis

Group “D”

Names:

Rawan Haitham

Engy Talaat

Yara Osama

Toqa Kashef

Raneem Zaki

Supervised by:

Eng. Ahmed Samir

Week 1 (Task 1 Report):

Week 1: Build Data Model, Data Cleaning and Preprocessing

- **Tasks:**

- Data Preprocessing: Build a data model and clean and preprocess the data.
- Tools: SQL, Python (pandas, Matplotlib).

- **Deliverables:**

- Cleaned dataset ready for analysis.
- Data preprocessing notebook.

For the first week we needed to clean our data from the dataset given to us “Superstore Sales Data”. For that we used Python Jupyter and utilized Pandas and Matplotlib to find out what we needed to do with our data. As our data didn’t have much to do with it we started a simple data cleaning process and decided to focus more on our analysis questions and the story that our data can tell us later on in the visualization process.

The first step done was importing pandas, numpy, matplotlib and seaborn and code was as follows:

```
import pandas as pd  
  
import numpy as np  
  
import matplotlib.pyplot as plt  
  
import seaborn as sns
```

We then inserted our CSV file that had the data:

```
df=pd.read_csv('Superstore Sales Dataset.csv')
```

We then used the following code to find out how many distinct Order IDs we had which we found were **4922**:

```
df['Order ID'].nunique()
```

We then checked to see if there’s any Null data using the following code:

```
df.isnull().sum()
```

The results showed only the postal code

```
Row ID          0
Order ID        0
Order Date      0
Ship Date       0
Ship Mode       0
Customer ID     0
Customer Name   0
Segment          0
Country          0
City              0
State             0
Postal Code      11
Region            0
Product ID       0
Category          0
Sub-Category      0
Product Name     0
Sales              0
dtype: int64
```

It was decided after looking at the data that certain columns were not needed for our analysis, the columns were (Row ID, Customer Name and Postal Code) which is why they were removed using the following code:

```
df_cleaned = df.drop(columns=['Row ID', 'Customer Name', 'Postal Code'])
```

To get a summary on how are data currently looks like we used the following code:

```
df_cleaned.info()
```

The results were as follows, we found out the types of our columns were almost all of the Object type, and we had one Float type:

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 9800 entries, 0 to 9799

Data columns (total 15 columns):

 #   Column           Non-Null Count  Dtype  
---  --  
 0   Order ID        9800 non-null   object 
 1   Order Date      9800 non-null   object 
 2   Ship Date       9800 non-null   object 
 3   Ship Mode       9800 non-null   object 
 4   Customer ID     9800 non-null   object 
 5   Segment          9800 non-null   object 
 6   Country          9800 non-null   object 
 7   City              9800 non-null   object 
 8   State             9800 non-null   object 
 9   Region            9800 non-null   object 
 10  Product ID       9800 non-null   object 
 11  Category          9800 non-null   object 
 12  Sub-Category     9800 non-null   object 
 13  Product Name     9800 non-null   object 
 14  Sales             9800 non-null   float64 

dtypes: float64(1), object(14)
```

This is why we changed some column types such as the Ship Date and Order Date into the datetime type as well as, all the column types of object are changed to be String:

```
df_cleaned['Ship Date']=pd.to_datetime(df_cleaned['Ship Date'],
format='%d/%m/%Y')
```

```

df_cleaned['Order Date']=pd.to_datetime(df_cleaned['Order Date'],
format='%d/%m/%Y')

df['Order ID']= df['Order ID'].astype("string")

df['Ship Mode']= df['Ship Mode'].astype("string")

df['Customer ID']= df['Customer ID'].astype("string")

df['Segment']= df['Segment'].astype("string")

df['Country']= df['Country'].astype("string")

df['City']= df['City'].astype("string")

df['State']= df['State'].astype("string")

df['Region']= df['Region'].astype("string")

df['Product ID']= df['Product ID'].astype("string")

df['Category']= df['Category'].astype("string")

df['Sub-Category']= df['Sub-Category'].astype("string")

df['Product Name']= df['Product Name'].astype("string")

df['Category']= df['Category'].astype("string")

```

The results were as follows:

```

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 8655 entries, 0 to 8654

Data columns (total 15 columns):

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 9800 entries, 0 to 9799

Data columns (total 15 columns):

#   Column           Non-Null Count  Dtype  
---  --  
0   Order ID        9800 non-null   string 

```

```

1   Order Date      9800 non-null      datetime64[ns]
2   Ship Date       9800 non-null      datetime64[ns]
3   Ship Mode        9800 non-null      string
4   Customer ID     9800 non-null      string
5   Segment          9800 non-null      string
6   Country          9800 non-null      string
7   City              9800 non-null      string
8   State             9800 non-null      string
9   Region            9800 non-null      string
10  Product ID       9800 non-null      string
11  Category          9800 non-null      string
12  Sub-Category      9800 non-null      string
13  Product Name     9800 non-null      string
14  Sales             9800 non-null      float64

dtypes: datetime64[ns](2), float64(1), string(12)

```

memory usage: 1014.4 KB

We checked for any duplicates in the data but ended up finding none and the result was as follows: `np.int64(0)`

```
df.duplicated().sum()
```

Finally we wanted to check for any outliers using Matplotlib and we actually discovered there were outliers and we removed them.

Steps:

1. **Calculate Q1 (25th percentile) and Q3 (75th percentile)** for the "Sales" column.
2. **Compute the IQR (Q3 - Q1).**
3. **Determine the lower and upper bounds** for acceptable data points as:
 - o Lower bound = $Q1 - 1.5 * IQR$
 - o Upper bound = $Q3 + 1.5 * IQR$
4. **Filter out the outliers** that fall outside this range.

5. Remove the outliers from the dataset.

After applying the IQR method, 1,145 outliers were detected and removed from the dataset. The cleaned dataset now contains 8,655 rows.

To ensure the quality of the data and prevent skewing the analysis, outliers in the "Sales" column were identified and removed. The Interquartile Range (IQR) method was used to detect outliers. This method identifies data points that fall significantly below the 25th percentile (Q1) or above the 75th percentile (Q3). The following steps were carried out:

1. The 25th percentile (Q1) and 75th percentile (Q3) of the "Sales" data were calculated.
2. The IQR was determined by subtracting Q1 from Q3.
3. Outliers were defined as any value below $Q1 - 1.5 * IQR$ or above $Q3 + 1.5 * IQR$.
4. A total of 1,145 outliers were removed, reducing the dataset to 8,655 entries from the original 9,800.

This procedure ensures that extreme values do not disproportionately influence the analysis, leading to more reliable insights.

Visualization:

- A scatter plot was generated where:
 - **X-axis:** Represents the index of each data point (row number in the dataset).
 - **Y-axis:** Represents the sales value.
 - **Red points** indicate outliers, while **blue points** represent non-outlier data.

Key Observations:

- Outliers were identified as extreme sales values either far lower or far higher than the general sales data range.
- The majority of the data falls within the non-outlier range, with a few points (outliers) marked in red, indicating exceptionally high or low sales transactions.

```
Q1 = df['Sales'].quantile(0.25)
Q3 = df['Sales'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
outliers    =    df[ (df['Sales']    <    lower_bound)    |    (df['Sales']    >
upper_bound) ]

non_outliers = df[ (df['Sales']    >=    lower_bound)    &    (df['Sales']    <=
upper_bound) ]



plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)

plt.scatter(non_outliers.index,    non_outliers['Sales'],    color='blue',
label='Non-Outliers')

plt.scatter(outliers.index,          outliers['Sales'],        color='red',
label='Outliers')

plt.xlabel('Index')

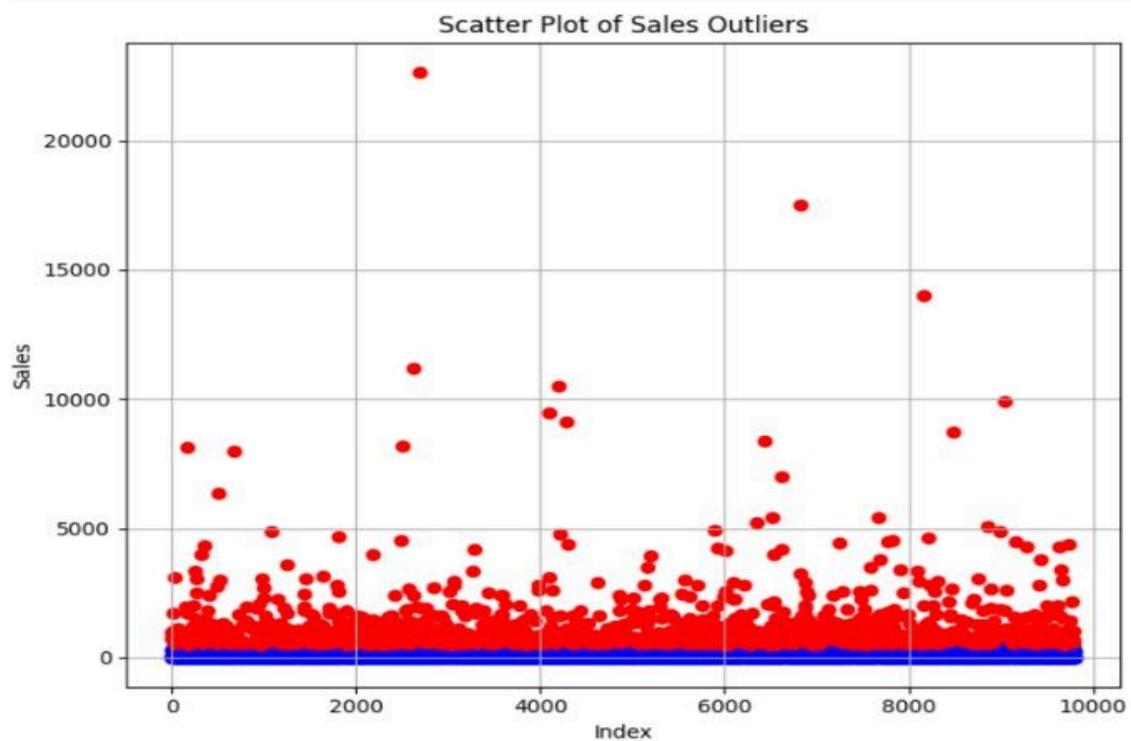
plt.ylabel('Sales')

plt.title('Scatter Plot of Sales Outliers')

plt.grid()

plt.tight_layout()

plt.show()
```



Finally we used the following code to compare between the number of row before and after removing the outliers:

```
original_rows = df_cleaned.shape[0]  
non_outliers = non_outliers.shape[0]  
original_rows, non_outliers
```

The output:

(9800, 8655)

The result showed that there were about 1,145 rows that's not in the range.

Task 2:

Week 2: Analysis Questions Phase

- **Tasks:**
 - Determine Data Analysis Questions: Determine all possible analysis questions that can be deducted from the given dataset and would be of interest to the organization's decision makers, e.g., what is the impact on products category and regions on sales performance?
 - Tools: SQL, Python (pandas, Matplotlib).
- **Deliverables:**
 - Set of analysis questions that can be answered via the dataset.

x

Key Areas for Analysis:

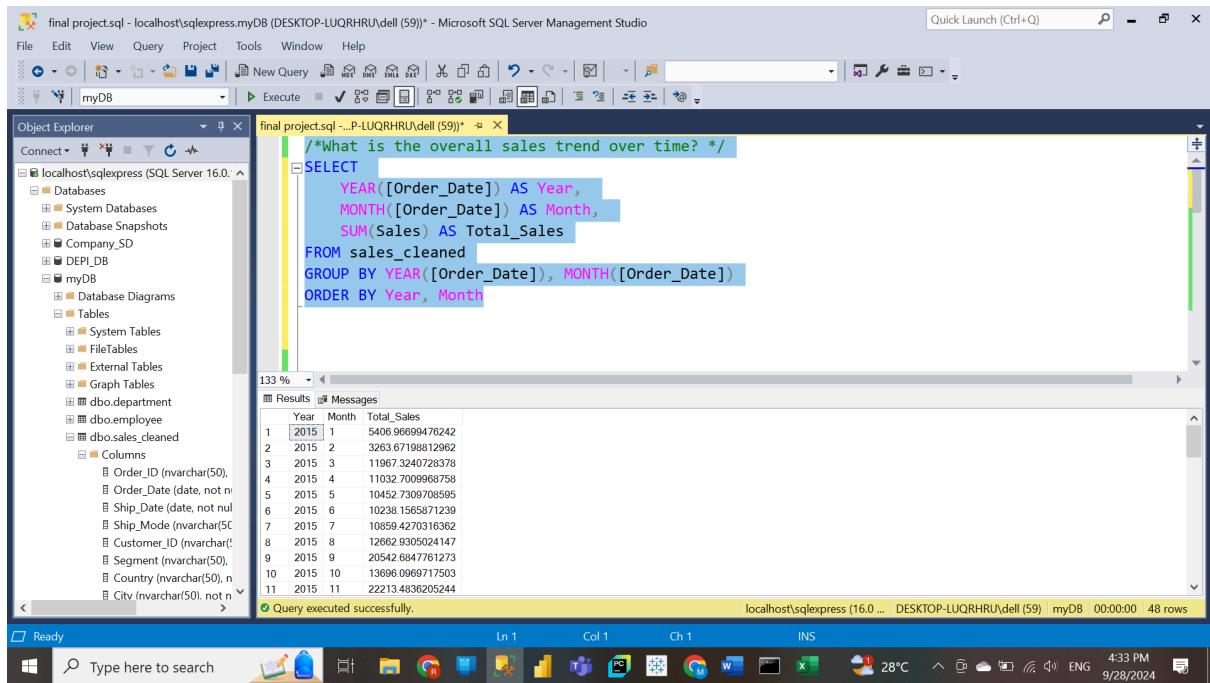
We can categorize potential analysis questions into various business areas such as **sales performance, customer behavior, regional analysis, product performance, and logistics**.

Analysis Questions:

1. Sales Performance Analysis:

1. What is the overall sales trend over time?

To see the overall sales trend over time, we first needed to Convert the Order Date to a format that allows aggregation (month/year), then we grouped by time period using the following SQL code:



final project.sql - localhost\sqlexpress.myDB (DESKTOP-LUQRHRU\dell (59)) - Microsoft SQL Server Management Studio

```

/*
 *What is the overall sales trend over time? */
SELECT
    YEAR([Order_Date]) AS Year,
    MONTH([Order_Date]) AS Month,
    SUM(Sales) AS Total_Sales
FROM sales_cleaned
GROUP BY YEAR([Order_Date]), MONTH([Order_Date])
ORDER BY Year, Month

```

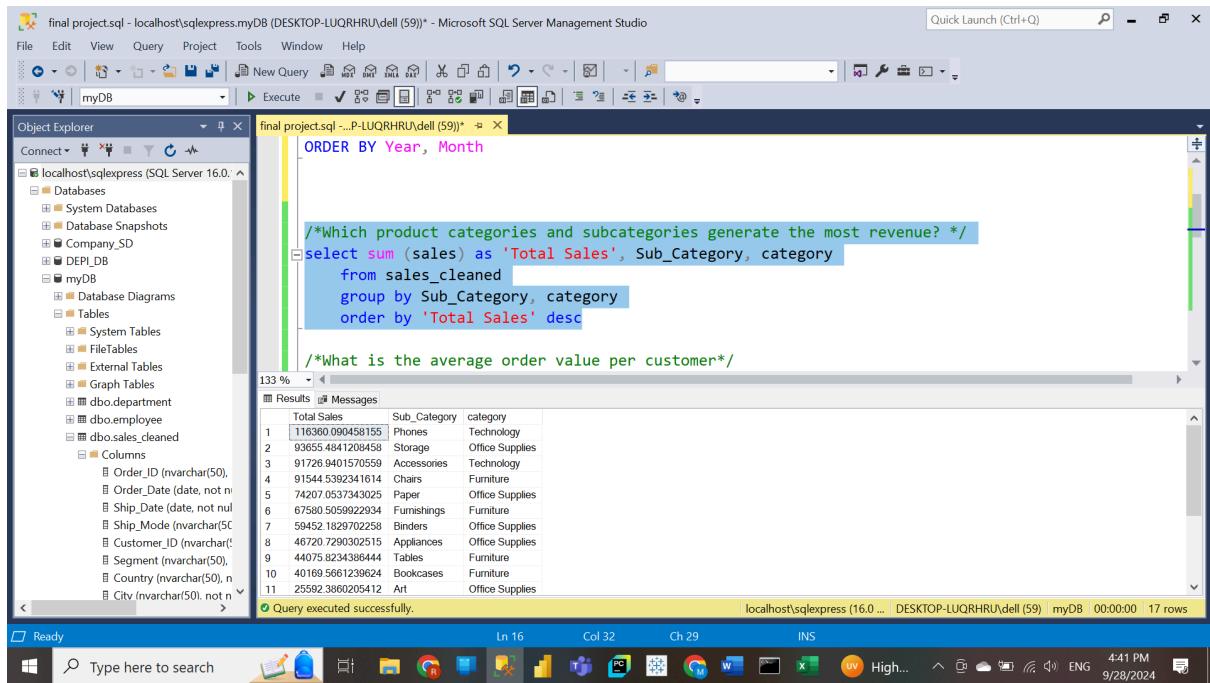
Results

Year	Month	Total_Sales
2015	1	5406.96699476242
2015	2	3263.67198812962
2015	3	11967.3240728378
2015	4	11032.700968758
2015	5	10452.7309708595
2015	6	10238.1565871239
2015	7	10859.4270316362
2015	8	12662.9305024147
2015	9	20542.6847761273
2015	10	13696.0969717503
2015	11	22213.4836205244

Query executed successfully.

2. Which product categories and subcategories generate the most revenue?

To see which category and sub-category generated the most sales we used the following code:



final project.sql - localhost\sqlexpress.myDB (DESKTOP-LUQRHRU\dell (59)) - Microsoft SQL Server Management Studio

```

ORDER BY Year, Month

/*
 *Which product categories and subcategories generate the most revenue? */
select sum (sales) as 'Total Sales', Sub_Category, category
    from sales_cleaned
    group by Sub_Category, category
    order by 'Total Sales' desc

/*
 *What is the average order value per customer*/

```

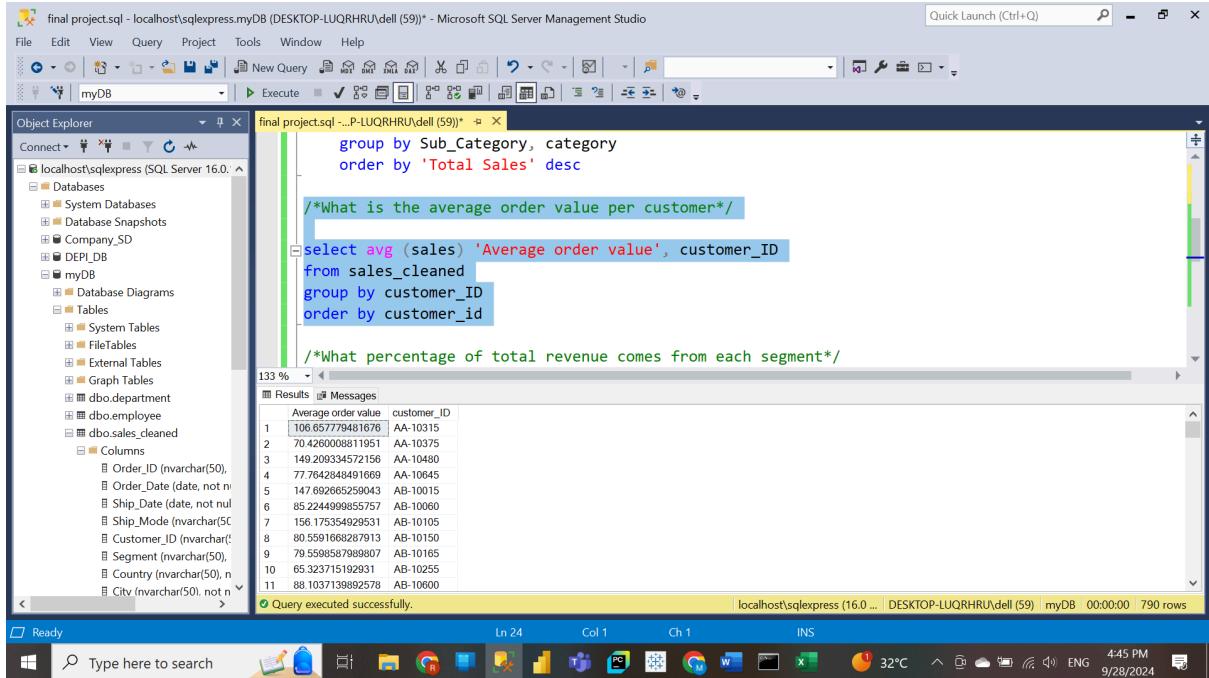
Results

Total Sales	Sub_Category	category
116360.090458155	Phones	Technology
93655.4841209458	Storage	Office Supplies
91726.9401570559	Accessories	Technology
91544.5392341614	Chairs	Furniture
74207.0537343025	Paper	Office Supplies
67580.5059922394	Furnishings	Furniture
59452.1829702258	Binders	Office Supplies
46720.7290302515	Appliances	Office Supplies
44075.8234386444	Tables	Furniture
40169.5661239624	Bookcases	Furniture
25592.3860205412	Art	Office Supplies

Query executed successfully.

3. What is the average order value per customer?

To see the averages sale value for each customer we used the following SQL code:



final project.sql - localhost\sqlexpress.myDB (DESKTOP-LUQRHRU\dell (59)) - Microsoft SQL Server Management Studio

```

group by Sub_Category, category
order by 'Total Sales' desc

/*What is the average order value per customer*/
select avg (sales) 'Average order value', customer_ID
from sales_cleaned
group by customer_ID
order by customer_id

/*What percentage of total revenue comes from each segment*/

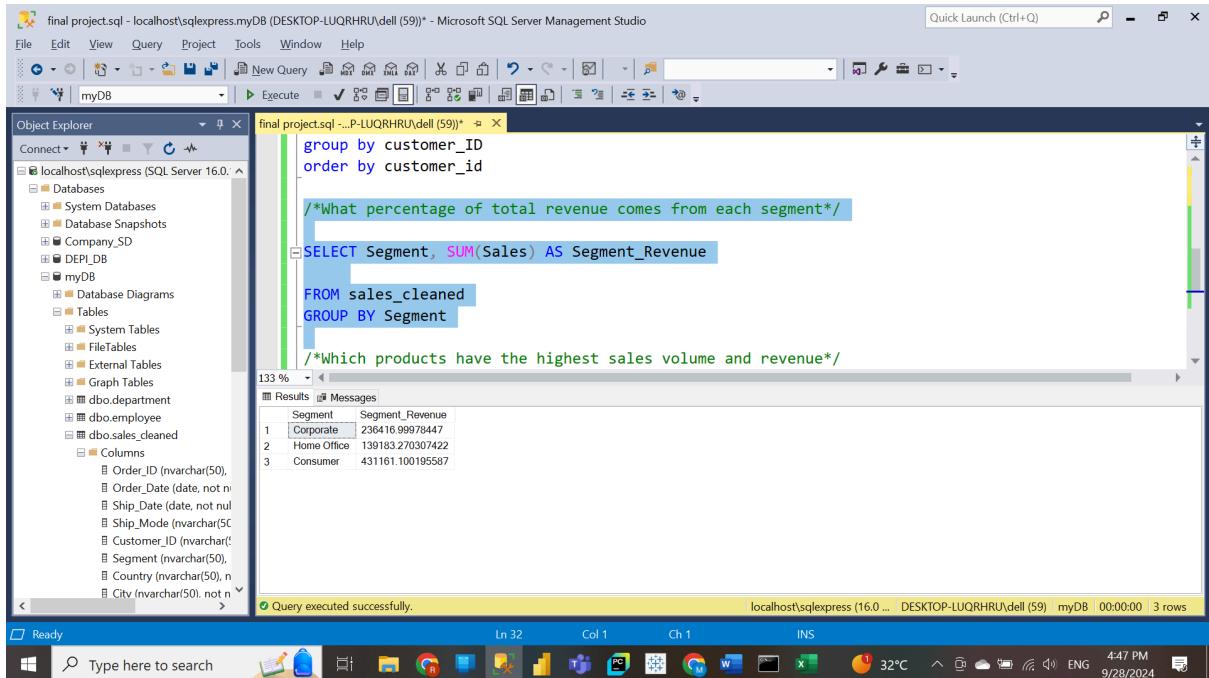
```

Results

Average order value	customer_ID
106.657779481676	AA-10315
70.4260008811951	AA-10375
149.209334572156	AA-10480
77.7642848491669	AA-10645
147.692665259043	AB-10015
85.224499855757	AB-10060
156.175354929531	AB-10105
80.55916682267913	AB-10150
79.559586799807	AB-10165
65.323715192931	AB-10255
88.1037139892578	AB-10600

Query executed successfully.

4. What percentage of total revenue comes from each segment (Consumer, Corporate, Home Office)?



final project.sql - localhost\sqlexpress.myDB (DESKTOP-LUQRHRU\dell (59)) - Microsoft SQL Server Management Studio

```

group by customer_ID
order by customer_id

/*What percentage of total revenue comes from each segment*/
SELECT Segment, SUM(Sales) AS Segment_Revenue
FROM sales_cleaned
GROUP BY Segment

/*Which products have the highest sales volume and revenue*/

```

Results

Segment	Segment_Revenue
Corporate	236416.99978447
Home Office	139183.270307422
Consumer	431161.100195567

Query executed successfully.

5. Which products have the highest sales volume and revenue?

```

final project.sql - localhost\sqlexpress.myDB (DESKTOP-LUQRHRU\dell (59)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
myDB
Execute
SELECT Segment, SUM(Sales) AS Segment_Revenue
FROM sales_cleaned
GROUP BY Segment
/*Which products have the highest sales volume and revenue*/
select top 1 sales, product_name
from sales_cleaned

```

Segment	Segment_Revenue
Corporate	23641699978447
Home Office	139183270307422
Consumer	431161100195587

Query executed successfully.

2. Customer Behavior Analysis:

7. Who are the top 10 customers in terms of total sales and order frequency?

To see the top 10 customers based on their total sales and order frequency, the calculates their total sales by summing the Sales column and counts the number of orders placed was created on SQL to help us answer this question and it was as follows:

```

--who are the top 10 customers in terms of total sales and order frequency?
SELECT TOP 10 Customer_ID , Customer_Name , sum(Sales) AS total_sales , COUNT(Order_ID) AS order_frequency
FROM db.Sales
GROUP BY Customer_ID , Customer_Name
ORDER BY total_sales DESC , order_frequency DESC

```

Customer_ID	Customer_Name	total_sales	order_frequency
SM-2030	Sean Miller	25043.050460078	15
TC-2090	Tamir Chand	19032.200000041	12
RG-2100	Rebecca Bush	15117.238987002	18
TA-2138	Tan Arshook	14956.618844038	10
AB-10105	Aden Botkin	14473.571243763	20
KL-16645	Ken Lonsdale	14175.228950517	29
SC-20095	Seril Chand	14142.3341379166	22
HL-15040	Hunter Lopez	12873.29762735	11
SE-20110	Seril Engle	12209.4381030323	19
CC-12370	Christopher Conant	12129.0715373755	11

As we can see the customer with the highest total sales is SM-20320, Sean Miller, with a total sales of (25043.05). They also placed 15 orders.

8. Which customer segments (Consumer, Corporate, Home Office) contribute most to sales?

We will know the most customer segment contributed to sales by group the data by segment, calculate the total revenue and order count for each segment using SQL

SQLQuery1.sql - (local)\test (LAPTOP-FHTIGHHT)\Tops (55) - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

Object Explorer

SQLQuery1.sql (LAPTOP-FHTIGHHT\Tops (55))

```
-- Which customer segments (Consumer, Corporate, Home Office) contribute most to sales--  
SELECT Segment, sum(Sales) as totalSales, count(*) as orderCount  
FROM Sales  
GROUP BY Segment  
ORDER BY totalSales DESC
```

Results

Segment	totalSales	orderCount
Consumer	1140000.52976102	5101
Corporate	683494.071592331	2853
Home Office	424982.177076221	1746

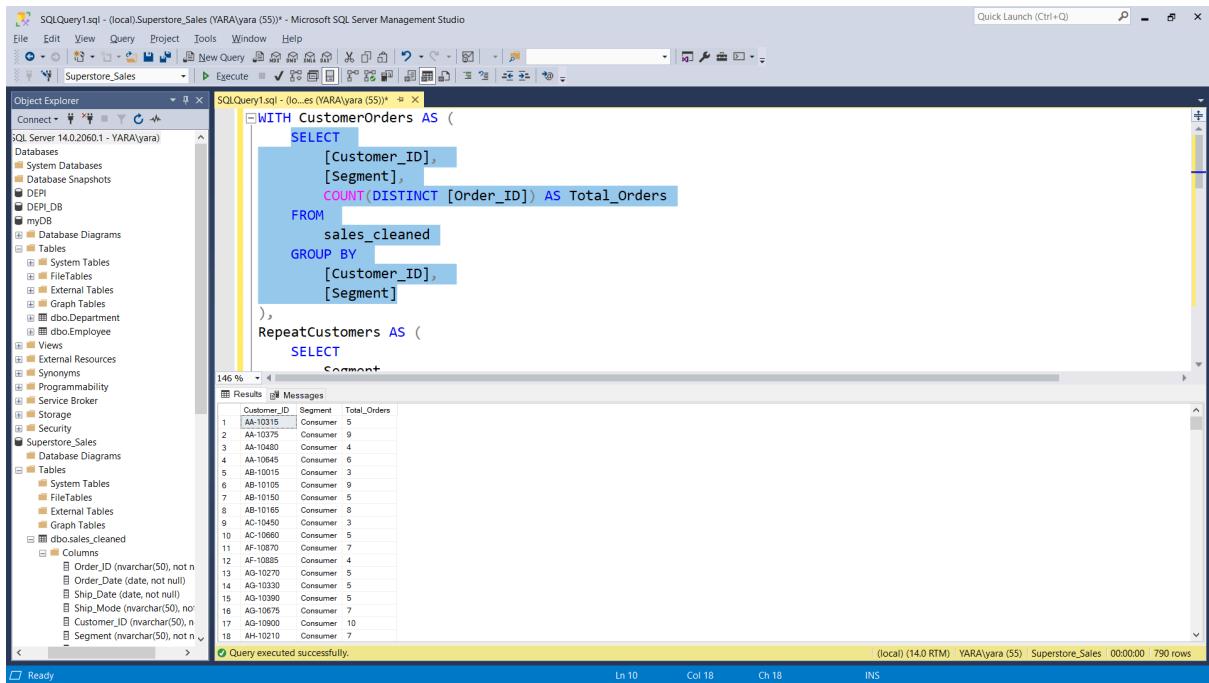
Query executed successfully.

As we can see the customer segment that contributed most to sales is the consumer with total sales of (1148060.53) and order count of 5101

9. What is the repeat purchase rate by customer segment?

This question wanted to see how many customers made more than one order in each of the three segments (Corporate, Home Office and Consumer). So in order to write the code for this on SQL it needed to be done in multiple steps.

We had to use the “WITH” CTE to reach a final conclusion, starting with a normal SELECT function to find the total orders done by each distinct customer in all the segments and the code was as follows:



```

SQLQuery1.sql - (local).Superstore_Sales (YARA\yara (55)) - Microsoft SQL Server Management Studio
Object Explorer
SQL Server 14.0.2000.1 - YARA\yara
Databases
System Databases
Database Snapshots
DEPI
DEPI_DB
myDB
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Department
dbo.Employee
News
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Superstore_Sales
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.sales_cleaned
Columns
Order_ID (nvarchar(50), not null)
Order_Date (date, not null)
Ship_Mode (nvarchar(50), not null)
Ship_Date (date, not null)
Customer_ID (nvarchar(50), not null)
Segment (nvarchar(50), not null)
CustomerOrders AS (
    SELECT
        [Customer_ID],
        [Segment],
        COUNT(DISTINCT [Order_ID]) AS Total_Orders
    FROM
        sales_cleaned
    GROUP BY
        [Customer_ID],
        [Segment]
),
RepeatCustomers AS (
    SELECT
        Segment
)
146 % < 4
Results Messages
Customer_ID Segment Total_Orders
1 AA-10315 Consumer 5
2 AA-10375 Consumer 9
3 AA-10480 Consumer 4
4 AA-10445 Consumer 6
5 AB-10015 Consumer 3
6 AB-10190 Consumer 9
7 AB-10150 Consumer 5
8 AB-10165 Consumer 8
9 AC-10450 Consumer 3
10 AC-10660 Consumer 5
11 AF-10870 Consumer 7
12 AF-10860 Consumer 4
13 AG-10270 Consumer 5
14 AG-10330 Consumer 5
15 AH-10875 Consumer 5
16 AG-10675 Consumer 7
17 AG-10900 Consumer 10
18 AH-10210 Consumer 7
790 rows
Query executed successfully.

```

However we needed to find the total repeat customers in each segment not the individual customers which is why the following code was written that gets the total repeat customers and compares it with the total customers in each segment in addition to the percentage of repeated purchases in each segment.

SQLQuery1.sql - (local).Superstore_Sales (YARA\yara (55)) - Microsoft SQL Server Management Studio

```

WITH CustomerOrders AS (
    SELECT
        [Customer_ID],
        [Segment],
        COUNT(DISTINCT [Order_ID]) AS Total_Orders
    FROM
        sales_cleaned
    GROUP BY
        [Customer_ID],
        [Segment]
),
RepeatCustomers AS (
    SELECT
        Segment
    FROM
        CustomerOrders
    WHERE
        Total_Orders > 1
    GROUP BY
        Segment
),
TotalCustomers AS (
    SELECT
        Segment,
        Repeat_Customers,
        Total_Customers,
        Repeat_Purchase_Rate_Percentage
    FROM
        CustomerOrders
    WHERE
        Total_Orders > 1
    GROUP BY
        Segment,
        Total_Customers
)

```

Results

Segment	Repeat_Customers	Total_Customers	Repeat_Purchase_Rate_Percentage
Corporate	229	234	97.8632478632479
Home Office	147	148	99.3243243243243
Consumer	401	408	98.2843137254902

Query executed successfully.

SQLQuery1.sql - (local).Superstore_Sales (YARA\yara (55)) - Microsoft SQL Server Management Studio

```

    ),
RepeatCustomers AS (
    SELECT
        Segment,
        COUNT([Customer_ID]) AS Repeat_Customers
    FROM
        CustomerOrders
    WHERE
        Total_Orders > 1
    GROUP BY
        Segment
),
TotalCustomers AS (
    SELECT
        Segment,
        Repeat_Customers,
        Total_Customers,
        Repeat_Purchase_Rate_Percentage
    FROM
        CustomerOrders
    WHERE
        Total_Orders > 1
    GROUP BY
        Segment,
        Total_Customers
)

```

Results

Segment	Repeat_Customers	Total_Customers	Repeat_Purchase_Rate_Percentage
Corporate	229	234	97.8632478632479
Home Office	147	148	99.3243243243243
Consumer	401	408	98.2843137254902

Query executed successfully.

```

GROUP BY
    Segment
),
TotalCustomers AS (
    SELECT
        Segment,
        COUNT([Customer_ID]) AS Total_Customers
    FROM
        CustomerOrders
    GROUP BY
        Segment
)
SELECT
    tc.Segment,
    tc.Repeat_Customers,
    tc.Total_Customers,
    Repeat_Purchase_Rate_Percentage
FROM
    TotalCustomers tc
    
```

Segment	Repeat_Customers	Total_Customers	Repeat_Purchase_Rate_Percentage
Corporate	229	234	97.8632478632479
Home Office	147	148	99.3243243243243
Consumer	401	408	98.2843137254902

```

    )
SELECT
    tc.Segment,
    tc.Repeat_Customers,
    tc.Total_Customers,
    (CAST(rc.Repeat_Customers AS FLOAT) / tc.Total_Customers) * 100 AS Repeat_Purchase_Rate_Percentage
FROM
    TotalCustomers tc
    LEFT JOIN
        RepeatCustomers rc
    ON
        tc.Segment = rc.Segment;
    
```

Segment	Repeat_Customers	Total_Customers	Repeat_Purchase_Rate_Percentage
Corporate	229	234	97.8632478632479
Home Office	147	148	99.3243243243243
Consumer	401	408	98.2843137254902

As we can see the home office segment had the highest percentage of repeat customers (99.3%), followed by the consumer segment (98.28%) and the corporate segment was in third place with (97.86%).

10. What are the average sales per order for different customer segments?

To get the average sales per order we must get first the total sales and number of orders by SQL:

SQLQuery1.sql - (local)\test (LAPTOP-FHITGHHT\Topaz (55)) - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

Object Explorer

SQLQuery1.sql - (LAPTOP-FHITGHHT\Topaz (55))

```
--What are the average sales per order for different customer segments--  
SELECT Segment, sum(Sales) as totalSales, count(Order_ID) as #of_orders,  
       sum(Sales)/count(Order_ID) as Avg_sales_per_order  
FROM fact_Sales  
group BY Segment
```

Results

Segment	totalSales	#of_orders	Avg_sales_per_order
Corporate	68849.071692331	2953	23.1591645201
Home Office	424982.177076221	1746	243.42330749367
Consumer	114060.52579102	5101	22.505777251719

Query executed successfully.

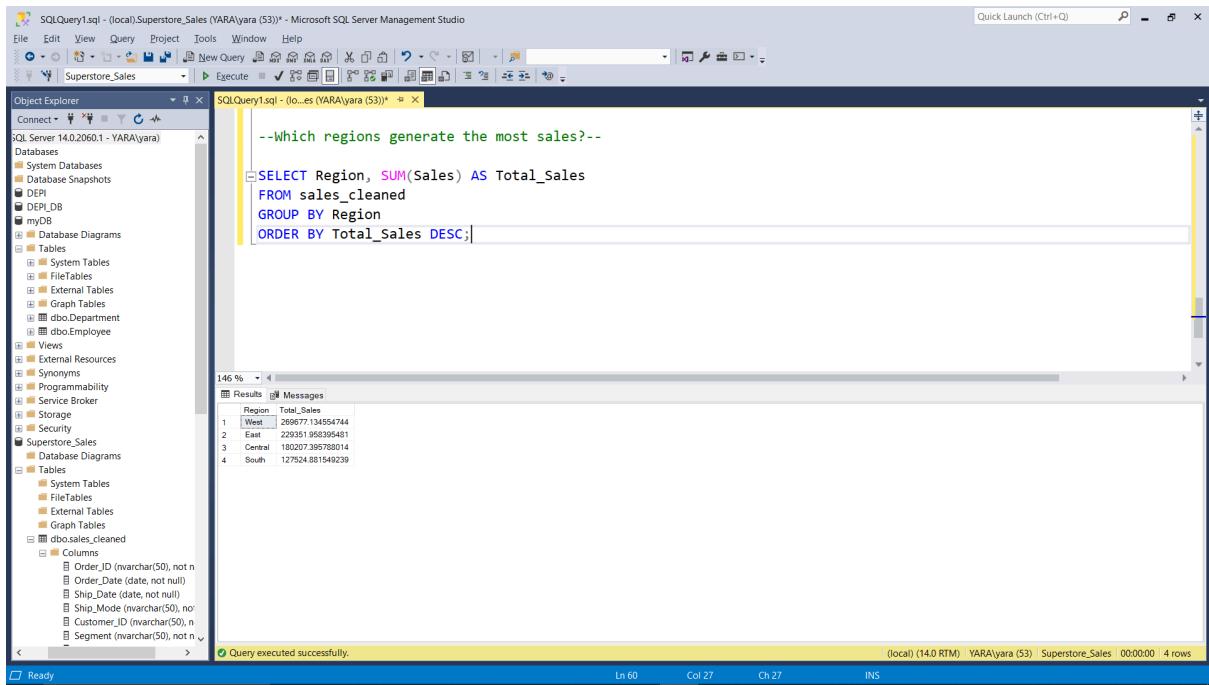
(local) (14.0 RTM) | LAPTOP-FHITGHHT\Topaz... | test | 00:00:00 | 3 rows

As we can see The "Consumer" segment generated the highest total sales, with a total of (1148060.52). The "Corporate" segment placed the most orders, with a total of 2953 orders. While the "Home Office" segment had the highest average sales per order, with an average of (243.4).

3. Regional Analysis:

11. Which regions generate the most sales?

To see which region in the states has the highest sales, the total sales in each region were summed and a code was created on SQL to help us answer this question and it was as follows:



```
--Which regions generate the most sales?--  
SELECT Region, SUM(Sales) AS Total_Sales  
FROM sales_cleaned  
GROUP BY Region  
ORDER BY Total_Sales DESC;
```

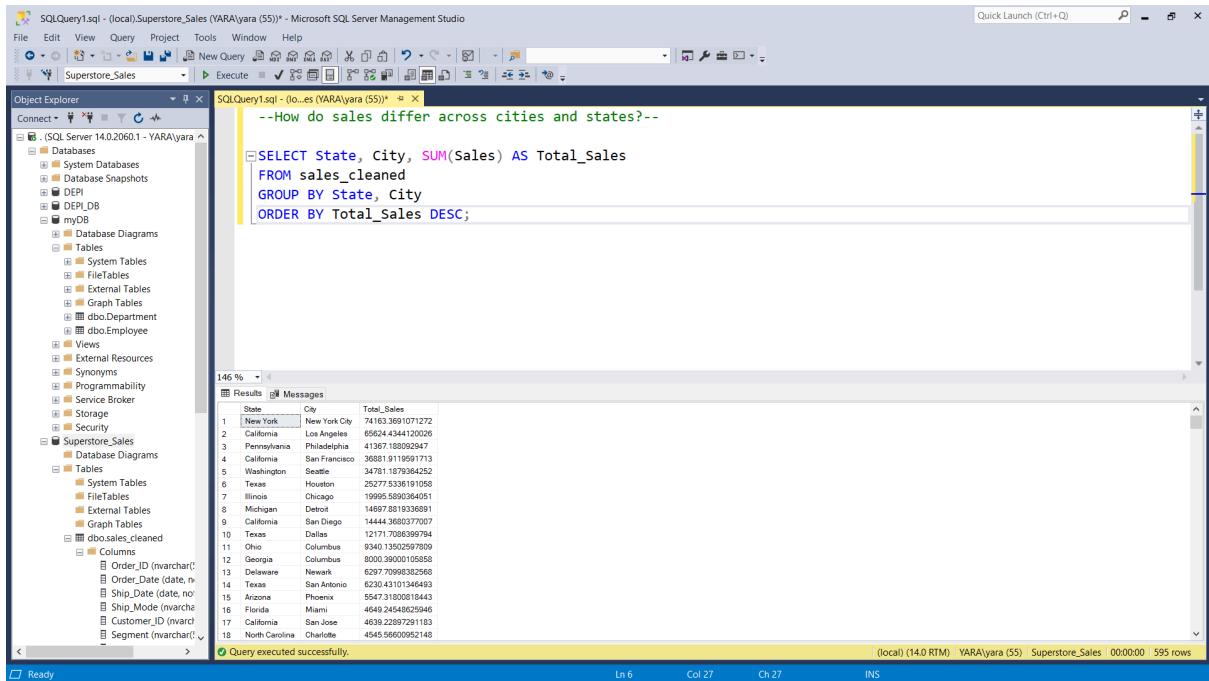
The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Superstore_Sales'. The 'Tables' node under 'dbo' contains 'sales_cleaned'. The 'Columns' node for 'sales_cleaned' lists various columns including Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, and Segment. The 'Results' tab in the center displays a table with four rows, showing the total sales for each region: West, East, Central, and South. The West region has the highest sales at 269677.13\$, followed by the East at 229351.96\$, the Central region at 180207.395\$, and the South region at 34781.19\$.

Region	Total_Sales
West	269677.134554744
East	229351.968395481
Central	180207.395788014
South	34781.191549239

As we can see the highest sales of (269677.13\$) were in the West region. Followed by the East with sales (229351.96\$). The Central region had the 3rd highest sales of (180207.395\$). Finally in fourth place was the South region with sales of (34781.19\$)

12. How do sales differ across Cities and States?

This question was answered using a code that shows the different Cities and States to see which states and cities should be focused on. The highest sales of (74163.37\$) were in New York City in the state of New York. while the lowest were in The city of Abilene in the state of Texas with sales being only (1.39\$).



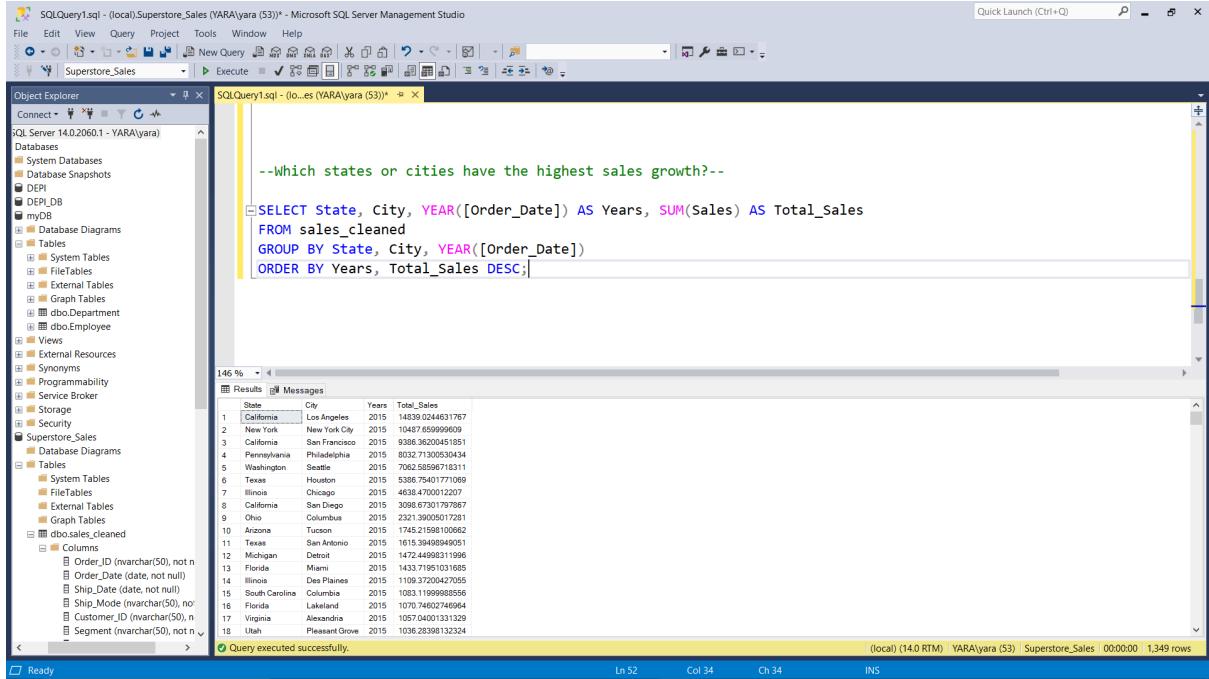
```
--How do sales differ across cities and states?--  
SELECT State, City, SUM(Sales) AS Total_Sales  
FROM sales_cleaned  
GROUP BY State, City  
ORDER BY Total_Sales DESC;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Superstore_Sales'. The 'Tables' node under 'dbo' contains 'sales_cleaned'. The 'Columns' node for 'sales_cleaned' lists various columns including Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, and Segment. The 'Results' tab in the center displays a table with 18 rows, showing the total sales for each city and state combination. The highest sales are in New York City at 74163.37\$, followed by Los Angeles at 65624.4344120026\$, and Philadelphia at 41362.188092947\$. The lowest sales are in Abilene, Texas at 1.39\$.

State	City	Total_Sales
New York	New York City	74163.3691071272
California	Los Angeles	65624.4344120026
Pennsylvania	Philadelphia	41362.188092947
California	San Francisco	36881.9119591713
Washington	Seattle	34781.1879364252
Texas	Houston	25277.5330507058
Texas	Abilene	1.39
Michigan	Detroit	14897.8019336951
California	San Diego	14442.3680277007
Texas	Dallas	12171.7085399794
Ohio	Columbus	9340.13502597809
Georgia	Columbus	8000.39000105858
Delaware	Newark	6297.7098382568
Texas	San Antonio	6230.43101346493
Arizona	Phoenix	5947.31800818443
Florida	Miami	4649.2454862594
California	San Jose	4639.22897291183
North Carolina	Charlotte	4545.56600952148

13. Which states or cities have the highest sales growth?

For this question we wanted to know which States and cities had the highest growth overtime, so a code was written to show the highest sales in every state and city in all the years available in the dataset. The results and code written were as follows:



The screenshot shows the Microsoft SQL Server Management Studio interface. The top menu bar includes File, Edit, View, Query, Project, Tools, Window, and Help. The title bar says "SQLQuery1.sql - (local)\Superstore_Sales [YARA\yara (53)*] - Microsoft SQL Server Management Studio". The Object Explorer on the left shows the database structure for "Superstore_Sales", including System Databases, System Tables, Tables, Views, and Synonyms. The "Tables" node is expanded, showing sales_cleaned, and its "Columns" node is also expanded, showing Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, and Segment. The "Script" button is highlighted in the toolbar. The main query window contains the following SQL code:

```
--Which states or cities have the highest sales growth?--  
SELECT State, City, YEAR([Order_Date]) AS Years, SUM(Sales) AS Total_Sales  
FROM sales_cleaned  
GROUP BY State, City, YEAR([Order_Date])  
ORDER BY Years, Total_Sales DESC;
```

The results window shows a table with the following data:

State	City	Years	Total_Sales
California	Los Angeles	2015	14839.0244631767
New York	New York City	2015	10426.4800000000
California	San Francisco	2015	23182.2900000000
Pennsylvania	Philadelphia	2015	8032.713000536144
Washington	Seattle	2015	7062.535957175311
Texas	Houston	2015	5367.5401771069
Illinois	Chicago	2015	4638.7400122056
California	San Diego	2015	3098.67301797867
Ohio	Columbus	2015	2327.390005017281
Arizona	Tucson	2015	1745.21998100662
Texas	San Antonio	2015	1615.39498949051
Michigan	Detroit	2015	1416.89000000000
Florida	Miami	2015	1437.7191031695
Illinois	Des Plaines	2015	1109.7200427056
South Carolina	Columbia	2015	1053.1199889556
Florida	Lakeland	2015	1070.74602746964
Virginia	Alexandria	2015	1057.04001331329
Utah	Pleasant Grove	2015	1036.28398132324

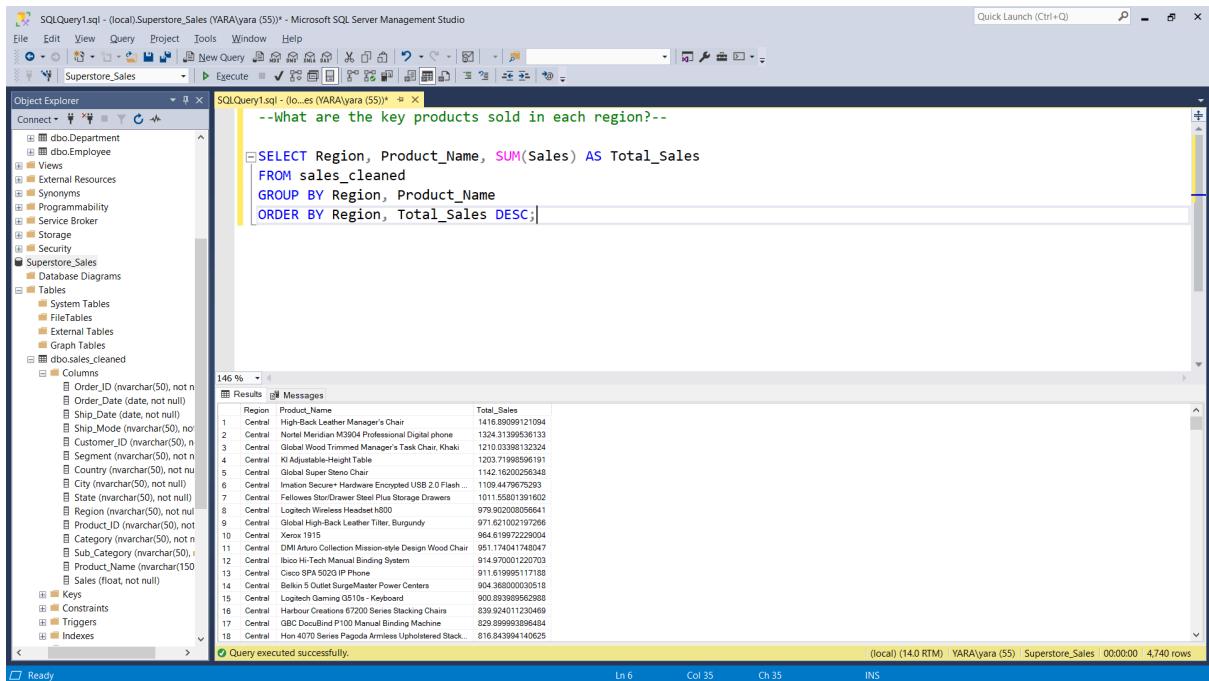
At the bottom of the results window, it says "Query executed successfully." and shows statistics: (local) (14.0 RTM) YARA\yara (53) Superstore_Sales 00:00:00 | 1,349 rows.

As we can see the results show the highest sales in every year grouped by every city and state. In 2015 Los Angeles was in the lead with total sales (14839.02\$). In 2016 New York City took the lead with total sales of (15216.41\$). It took the lead again in 2017 and further increased its sales to a total of (23182.29\$). Once again in 2018 New York City had the highest sales in 2018 that reached a total of (25277\$). Making New York City the region with the highest sales growth.

14. What are the key products sold in each region?

This question is used to find the highest selling product in each region. Starting with the Central Region the highest selling product is “High-Back Leather Manager's Chair” with total sales of (1416.89\$). The East Region had the highest sales of (1483.48\$) from the product “Situations Contoured Folding Chairs, 4/Set”. In the South Region the highest selling product was “Gould Plastics 18-Pocket Panel Bin, 34w x 5-1/4d x 20-1/2h” with sales of (1232.67\$). Finally the West Region's highest sales were (1777.05\$) with the product “Fellowes Super Stor/Drawer Files”.

The following shows the code written on SQL and the results:



```
--What are the key products sold in each region?--  
  
SELECT Region, Product_Name, SUM(Sales) AS Total_Sales  
FROM sales_cleaned  
GROUP BY Region, Product_Name  
ORDER BY Region, Total_Sales DESC;
```

Results

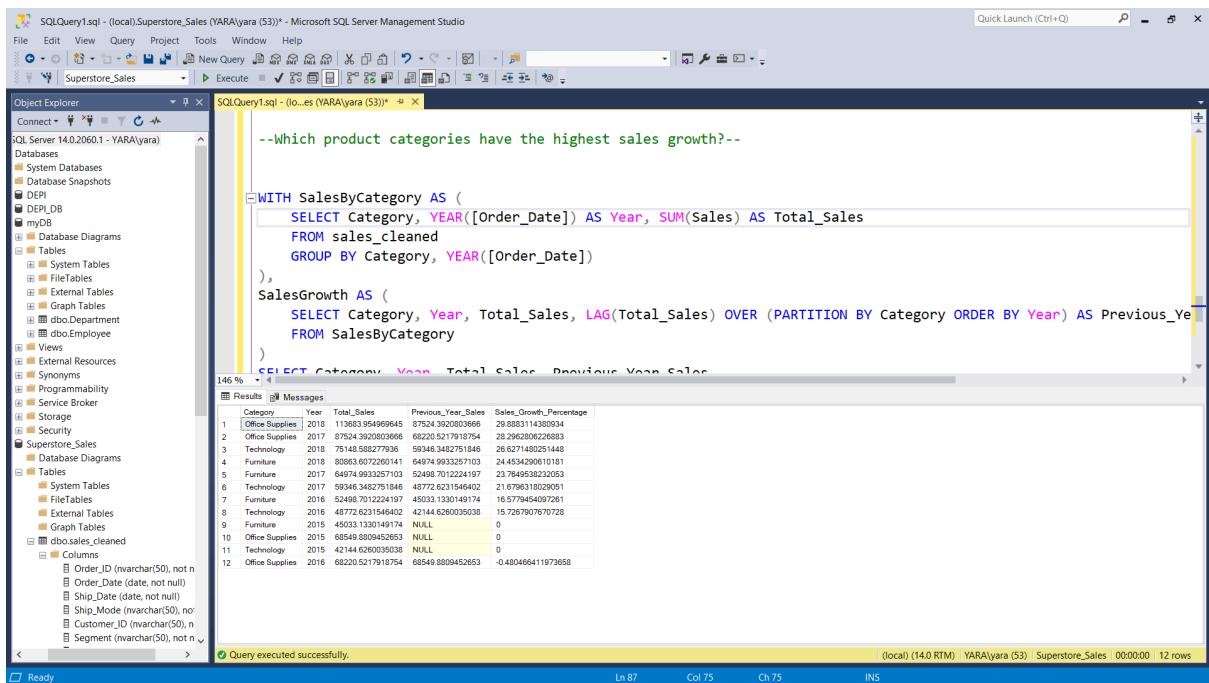
Region	Product_Name	Total_Sales
Central	High-Back Leather Manager's Chair	1416.89099121094
Central	Nordic Meridian K3904 Professional Digital phone	1324.31399536133
Central	Global Wood Trimmed Manager's Task Chair, Khaki	1210.03398132324
Central	KI Adjustable-Height Table	1203.71998596191
Central	Global Super Steno Chair	1142.16200256348
Central	Global 2.0 USB 3.0 Encrypted USB 2.0 Flash...	1101.55801391402
Central	Fellowes Star-Drawer Steel Plus Storage Drawers	679.90200569441
Central	Logitech Wireless Headset H800	971.621002197266
Central	Global High-Back Leather Tilter, Burgundy	964.61997229904
Central	Xerox 1915	951.174041748047
Central	DMI Arto Collection Mission-style Design Wood Chair	914.970001220703
Central	Cisco Hi-Tech Manual Binding System	911.61995117188
Central	Cisco SPA 5020 IP Phone	904.368000030518
Central	Belkin 5 Outlets SurgeMaster Power Centers	899.0240111230469
Central	Harbour Creations E7200 Series Stacking Chairs	839.9240111230469
Central	QBC Doubled P10 Manual Binding Machine	828.89993398484
Central	Hon 4070 Series Pagoda Armless Upholstered Stack...	816.843994140625

Query executed successfully. (local) (14.0 RTM) YARA(yara (55)) Superstore_Sales | 00:00:00 | 4,740 rows

4. Product Performance Analysis:

15. Which product categories have the highest sales growth?

For this question we needed to calculate the total sales of each product category over the years so a code was used to group the sales by the category and years. It was then followed by a code to calculate the sales growth and its percentage every year and for this case the LAG function was used to calculate the previous year's sales. Finally the percentage was calculated and then arranged in descending order.



```
--Which product categories have the highest sales growth?--  
  
WITH SalesByCategory AS (  
    SELECT Category, YEAR([Order_Date]) AS Year, SUM(Sales) AS Total_Sales  
    FROM sales_cleaned  
    GROUP BY Category, YEAR([Order_Date])  
)  
SalesGrowth AS (  
    SELECT Category, Year, Total_Sales, LAG(Total_Sales) OVER (PARTITION BY Category ORDER BY Year) AS Previous_Year_Sales  
    FROM SalesByCategory  
)  
SELECT Category, Year, Total_Sales, Previous_Year_Sales, (Total_Sales - Previous_Year_Sales) / Previous_Year_Sales AS Sales_Growth_Percentage
```

Results

Category	Year	Total_Sales	Previous_Year_Sales	Sales_Growth_Percentage
Office Supplies	2018	113683.954969645	87524.3920030666	29.6883114380934
Office Supplies	2017	87524.3920030666	68220.521791754	28.2962806226883
Technology	2018	75148.58277936	59348.34827517846	26.6271480251448
Furniture	2018	80893.6072260141	64974.993257192	24.4534296010181
Furniture	2017	64974.993257192	52409.701224197	23.764953220205
Technology	2017	52409.701224197	46772.651204042	12.000000000000001
Furniture	2016	46772.651204042	42144.6260303938	16.577454097281
Technology	2015	42144.6260303938	40303.130049174	4.857142857142857
Office Supplies	2015	40303.130049174	NULL	0
Technology	2015	68549.8809452653	NULL	0
Office Supplies	2016	68220.521791754	68549.8809452653	-0.480466411973658

Query executed successfully. (local) (14.0 RTM) YARA(yara (53)) Superstore_Sales | 00:00:00 | 12 rows

```

SQLQuery1.sql - (local)\Superstore_Sales (YARA\yara (53)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer
SQL Server 14.0.2006.1 - YARA\yara
Databases
System Databases
Database Snapshots
DEPI
DEPI_DB
myDB
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Department
dbo.Employee
News
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Superstore_Sales
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.sales_cleaned
Columns
Order_ID (nvarchar(50), not null)
Order_Date (date, not null)
Ship_Date (date, not null)
Ship_Mode (nvarchar(50), not null)
Customer_ID (nvarchar(50), not null)
Segment (nvarchar(50), not null)
Category (nvarchar(50), not null)
Sub_Category (nvarchar(50), not null)
Product_Name (nvarchar(50), not null)
Sales (float, not null)
Keys
Constraints
Triggers
Indexes
Statistics
Superstore_Sales
New Query Execute
SQLQuery1.sql (local)\Superstore_Sales (YARA\yara (53))*
SELECT Category, Year, Total_Sales, LAG(Total_Sales) OVER (PARTITION BY Category ORDER BY Year) AS Previous_Year_Sales
)
SalesGrowth AS (
    SELECT Category, Year, Total_Sales, Previous_Year_Sales
    FROM SalesByCategory
)
SELECT Category, Year, Total_Sales, Previous_Year_Sales,
CASE
    WHEN Previous_Year_Sales IS NULL THEN 0
    ELSE ((Total_Sales - Previous_Year_Sales) / Previous_Year_Sales) * 100
END AS Sales_Growth_Percentage
FROM SalesGrowth
ORDER BY Sales_Growth_Percentage DESC;

```

Category	Year	Total_Sales	Previous_Year_Sales	Sales_Growth_Percentage
Office Supplies	2018	113683.954969645	87524.3920803666	29.883114380934
Office Supplies	2017	87524.3920803666	68220.521791754	28.2962806226883
Technology	2018	75148.5827936	59348.3482751848	26.6271480251448
Furniture	2018	80813.60722641	53474.993575194	24.4534290610181
Furniture	2017	53474.993575194	52470.1940527194	23.76495322053
Books	2017	59348.3482751848	48773.631544602	21.065270000000002
Furniture	2016	63498.7012234197	48623.132049174	16.5779454097281
Technology	2016	48772.621546402	42144.6260035038	15.7267907670728
Furniture	2015	49033.133049174	NULL	0
Office Supplies	2015	68549.8809452653	NULL	0
Technology	2015	42144.6260035038	NULL	0
Office Supplies	2016	68220.521791754	68549.8809452653	-0.480466411973658

As we can see in the results the year 2015 can't be considered as there's no previous year's data. However we can see that in 2018 the office supplies had the highest growth rate of (29.9%). While the office supplies had a negative growth in rate in 2016 of (-0.84%) putting it last place even though it has higher total sales than the other categories.

16. Which products have the highest return rates or lowest sales performance?

Product with highest return: KI Adjustable - height table

Product with lowest sales performance: Eureka Disposable Bags

```

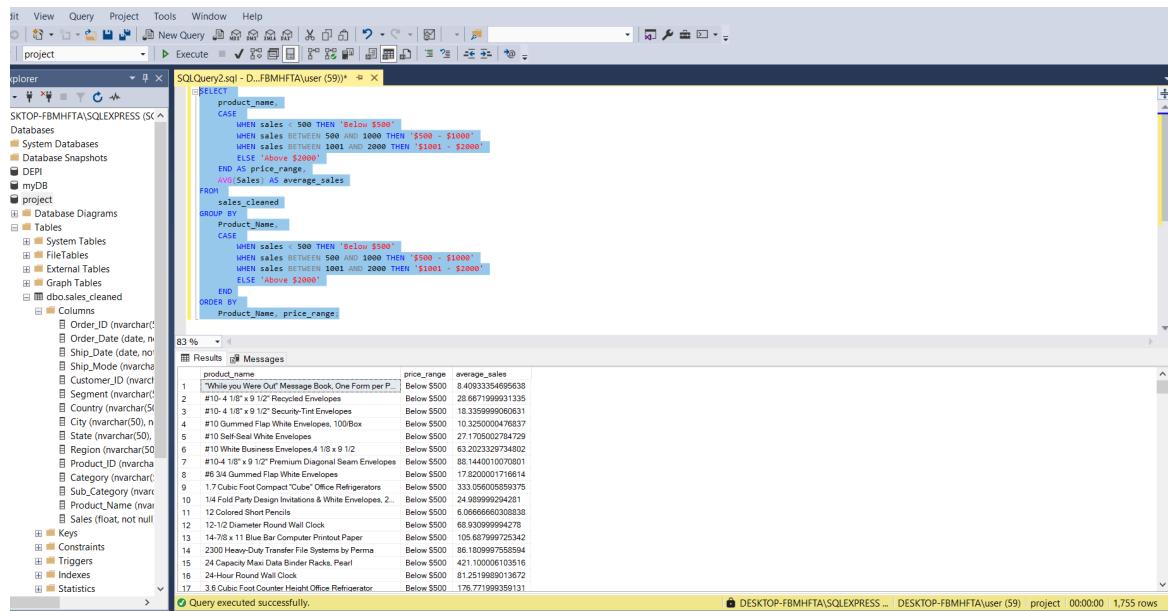
File Edit View Query Project Tools Window Help
Object Explorer
SQLQuery2.sql - D...\FBMHFTA\user (59) - Microsoft SQL Server Management Studio
Connect
DESKTOP-FBMHFTA\SQLEXPRESS (59)
Databases
System Databases
Database Snapshots
DEPI
DEPI_DB
myDB
project
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.sales_cleaned
Columns
Order_ID (nvarchar(50), not null)
Order_Date (date, not null)
Ship_Date (date, not null)
Ship_Mode (nvarchar(50), not null)
Customer_ID (nvarchar(50), not null)
Segment (nvarchar(50), not null)
Category (nvarchar(50), not null)
Sub_Category (nvarchar(50), not null)
Product_Name (nvarchar(50), not null)
Sales (float, not null)
Keys
Constraints
Triggers
Indexes
Statistics
Superstore_Sales
New Query Execute
SQLQuery2.sql - D...\FBMHFTA\user (59) -
SELECT Product_Name, sum(sales) as Total_Sales
From sales_cleaned
group by Product_Name
order by Total_Sales desc

```

Product_Name	Total_Sales
KI Adjustable-Height Table	3950.78104782104
Global Wood Trimmed Manager's Task Chair, Khaki	3621.00400161743
Sitkations Contoured Folding Chairs, 4/Set	2959.86603164673
Global High-Back Leather Tilt, Burgundy	2841.06900024414
Notre Damean M3004 Professional Digital phone	2802.61804199219
Ibico Hi-Tech Manual Binding System	2653.159000490234
Fellowes Officeware Wire Shelving	2513.22997169569
Executive Pad	2414.15980497554
Leggett & Platt G510+ - Keyboard	2365.7310619771
Space Solutions HD Industrial Steel Shelving	2345.38003100586
Cerica Double Wide Media Storage Towers in Natural	2299.83198547363
Gould Plastic 18-Pocket Panel Bin, 34w x 5-1/4d x 22d	2299.75003014697
Lean Shelf Collection Coffee Table, End Table...	2298.11399841309
GBC DocuBind P100 Manual Binding Machine	2290.52395626983
SAFCO Bolless Steel Shelving	2272.80001831055
Geemarc AmpliPOWER60	2227.199893185
DMI Arturo Collection Mission-style Design Wood Ch...	2204.308021748

17. How does product price impact sales?

Average sales per product price range. First we indicated the price ranges and calculated the average sales for each product in said ranges.



```

SELECT
    product_name,
    CASE
        WHEN Sales < 500 THEN 'Below $500'
        WHEN Sales BETWEEN 500 AND 1000 THEN '$500 - $1000'
        WHEN Sales BETWEEN 1001 AND 2000 THEN '$1001 - $2000'
        ELSE 'Above $2000'
    END AS price_range,
    AVG(Sales) AS average_sales
FROM sales_cleaned
GROUP BY
    product_name,
    CASE
        WHEN Sales < 500 THEN 'Below $500'
        WHEN Sales BETWEEN 500 AND 1000 THEN '$500 - $1000'
        WHEN Sales BETWEEN 1001 AND 2000 THEN '$1001 - $2000'
        ELSE 'Above $2000'
    END
ORDER BY
    product_name, price_range

```

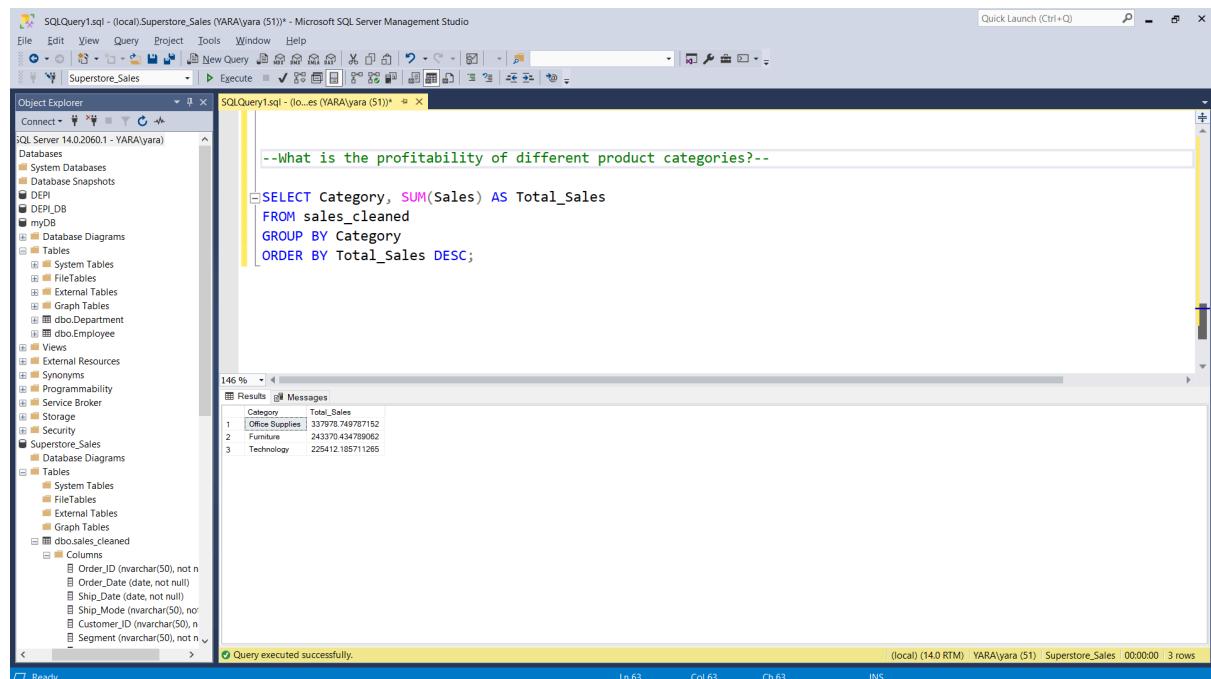
Results (17 rows)

product_name	price_range	average_sales
'While you Were Out' Message Book, One Form per Page	Below \$500	8,409,354,469,638
10-4 1/8" x 9 1/2" Recycled Envelopes	Below \$500	28,667,199,931,335
#10-4 1/8" x 9 1/2" Security-Tint Envelopes	Below \$500	18,335,999,060,631
#10 Gummied Flap White Envelopes, 100Box	Below \$500	10,325,000,476,837
#10 Self-Seal White Envelopes	Below \$500	27,170,902,784,729
#10 White Business Envelopes,4 1/8 x 9 1/2	Below \$500	63,202,329,734,802
#10-4 1/8" x 9 1/2" Premium Diagonal Seam Envelopes	Below \$500	14,440,010,070,001
85-1000 100% Recycled Flag White Envelopes	Below \$500	17,800,001,716,141
1-7 Cube-Folded 24" x 36" Cube Office Refrigerators	Below \$500	333,059,999,357,375
14 Fold-Party Design Invitations & White Envelopes, 25	Below \$500	24,989,999,294,281
12 Colored Short Pencils	Below \$500	8,066,660,020,838
12-1/2-Diameter Round Wall Clock	Below \$500	68,930,999,942,78
14-7/8" x 11 Blue Bar Computer Printout Paper	Below \$500	105,687,999,725,342
2300 Heavy-Duty Transfer File Systems by Perma	Below \$500	86,180,999,755,594
24 Capacity Max Data Binder Racks, Pearl	Below \$500	421,100,980,010,516
24-Hour Round Wall Clock	Below \$500	81,251,980,013,872
3.6 Cubic Foot Counter Height Office Refrigerator	Below \$500	176,771,999,359,131

Query executed successfully.

18. What is the profitability of different product categories?

There is no column for costs in the table, so it's not possible to calculate profit which is why we'll use the sales instead to determine which product category has the highest sales.



```

--What is the profitability of different product categories?--

SELECT Category, SUM(Sales) AS Total_Sales
FROM sales_cleaned
GROUP BY Category
ORDER BY Total_Sales DESC;

```

Results (3 rows)

Category	Total_Sales
Office Supplies	337978.49787152
Furniture	243370.43478902
Technology	225412.185711265

Query executed successfully.

As we can see, the Office supplies category has the highest total sales of (337978.75\$) followed by Furniture and Technology with sales respectively (243370.43\$) and (225412.19\$).

5. Logistics and Operational Analysis:

19. What is the average delivery time from order to shipping for each shipping mode?

Calculated the time difference between order date and shipping date using the built in function DATEDIFF.

Object Explorer

File Edit View Query Project Tools Window Help

project

Object Explorer

Connect to...

DESKTOP-FBMHFTA\SQLEXPRESS (SC^)

- Databases
 - System Databases
 - Database Snapshots
- DEPI
- myDB
- project
 - Database Diagrams
- Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
- Columns
- dbo.sales_cleaned
 - Order_ID (nvarchar(50))
 - Order_Date (date, not null)
 - Ship_Date (date, not null)
 - Ship_Mode (nvarchar(10))
 - Customer_ID (nvarchar(13))
 - Segment (nvarchar(45))
 - Country (nvarchar(50))
 - City (nvarchar(50), not null)
 - State (nvarchar(50), not null)
 - Region (nvarchar(50))
 - Product_ID (nvarchar(13))
 - Category (nvarchar(45))
 - Sub_Category (nvarchar(45))
 - Product_Name (nvarchar(255))
 - Sales (float, not null)
 - Quantity (float, not null)
 - Discount (float, not null)
 - UnitPrice (float, not null)
 - UnitPriceDiscount (float, not null)
 - UnitPrice * Quantity AS Total
 - UnitPrice * Quantity * (1 - UnitPriceDiscount) AS TotalNet
- Keys
- Constraints
- Triggers
- Indexes
- Statistics

SQLQuery2.sql - D...DESKTOP-FBMHFTA\SQLEXPRESS (SC^) (user 59)*

```
SELECT
    order_id,
    order_date,
    ship_date,
    DATEDIFF(DAY, order_date, ship_date) AS days_to_ship
FROM
    sales_cleaned;
```

100 %

Results

order_id	order_date	ship_date	days_to_ship
CA-2017-152156	2017-11-08	2017-11-11	3
CA-2017-138862	2017-06-12	2017-06-16	4
US-2016-10986	2016-10-11	2016-10-18	7
CA-2015-115812	2015-06-09	2016-06-14	5
CA-2015-115812	2015-06-09	2015-06-14	5
CA-2015-115812	2015-06-09	2015-06-14	5
CA-2015-115812	2015-06-09	2015-06-14	5
CA-2018-14412	2018-04-11	2018-04-20	5
CA-2017-137330	2017-12-09	2017-12-13	4
US-2016-11983	2016-11-22	2016-11-28	4
CA-2015-167164	2015-05-13	2015-05-16	2
CA-2015-143320	2015-08-27	2015-09-01	5
CA-2015-143320	2015-08-27	2015-09-01	5
CA-2015-143320	2015-08-27	2015-09-01	5
CA-2017-137330	2017-12-09	2017-12-13	4
CA-2017-137330	2017-12-09	2017-12-13	4

Query executed successfully.

20. Which shipping methods are the most used by customers, and how do they impact delivery times?

Shipping mode frequency and its impact on delivery times by calculating the number of orders for each shipping mode and the average delivery time in days for each delivery mode.

```
SELECT
    ship_mode,
    COUNT(order_id) AS frequency,
    AVG(DATEDIFF(DAY, order_date, Ship_Date)) AS average_delivery_time
FROM
    sales_cleaned
GROUP BY
    ship_mode
ORDER BY
    frequency DESC;
```

ship_mode	frequency	average_delivery_time
Standard Class	9108	5
Second Class	1871	3
First Class	1325	2
Same Day	473	0

21. Are there any trends between shipping time and specific regions or customer segments?

SQLQuery1.sql - (local).master (DESKTOP-6R31JB\\$Amira talaat (55))* - Microsoft SQL Server Management Studio

```

SELECT
    region, segment,
    AVG(DATEDIFF(day, "order date", "ship date")) AS average_delivery_time
FROM
    sales_cleaned
GROUP BY
    region, segment
ORDER BY
    average_delivery_time DESC;

```

Activate Windows

	region	segment	average_delivery_time
1	East	Corporate	4
2	Central	Consumer	4
3	West	Corporate	4
4	Central	Corporate	4
5	South	Corporate	4
6	Central	Home Office	4
7	East	Consumer	3
8	South	Home Office	3
9	West	Home Office	3
10	West	Consumer	3
11	South	Consumer	3

Query executed successfully.

22. What is the correlation between shipping mode and customer satisfaction (using repeat orders)?

SQLQuery1.sql - (local).master (DESKTOP-6R31JB\\$Amira talaat (55))* - Microsoft SQL Server Management Studio

```

SELECT
    "Ship_Mode",
    COUNT(DISTINCT "Customer ID") AS Unique_Customers,
    COUNT("Order ID") AS Repeat_Orders
FROM
    sales_cleaned
GROUP BY
    "Ship_Mode"
ORDER BY
    Repeat_Orders DESC;

```

Activate Windows

	Ship Mode	Unique_Customers	Repeat_Orders
1	Standard Class	769	5186
2	Second Class	525	1671
3	First Class	474	1325
4	Same Day	206	473

Query executed successfully.

6. Time-Based Analysis:

23. What are the peak sales periods (days, months, or seasons)?

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'SQL Server 14.0.2060.1 - YARA\yara'. The 'Superstore_Sales' database is selected. The 'Tables' node under 'Superstore_Sales' is expanded, showing 'Sales' and 'Sales_Cleaned'. The 'Sales_Cleaned' table is selected in the 'Tables' list. The 'Script' button is highlighted. The 'Script' pane contains a T-SQL query:

```
--What are the peak sales periods (days, months, or seasons)?--  
  
SELECT MONTH([Order_Date]) AS Sales_Month, SUM(Sales) AS Total_Sales  
FROM sales_cleaned  
WHERE sales IS NOT NULL  
GROUP BY MONTH([Order_Date])  
ORDER BY Total_Sales desc;
```

The 'Results' pane shows the output of the query as a table:

Sales_Month	Total_Sales
11	117716.813904762
12	116090.202714205
9	105905.40448347
10	65907.002605255
7	59325.481074948
8	59331.910623223
3	59314.851788723
5	57079.8263260126
6	55716.3024857044
4	52625.668126407
1	31277.8029924637
2	27269.7343201637

The status bar at the bottom indicates 'Query executed successfully.' and shows the session details: (local) (14.0 RTM) YARA\yara (62) Superstore_Sales 00:00:00 | 12 rows.

24. How do sales change over holidays or promotional periods?

As we can see in the previous results the months of (9, 10, 11 and 12) had the highest sales and these months coincide with the christmas holiday showing how the holidays can have a large effect in increasing the customers willingness to buy products and the store should make preparations for the holiday seasons by increasing their products.

Task 3:

Week 3: Forecasting Questions Phase

- **Tasks:**

- Determine a set of forecasting questions and answer them using the trends found in the given dataset.

- Tools: Python (scikit-learn, pandas, Matplotlib).

- **Deliverables:**

- Visualization plots answering forecasting questions.

For our data set the following questions will be used to give us insight into the trends we have in the data and help the store predict the future sales to be prepared with appropriate inventory that can match the demand in the holiday seasons as well as in normal months.

Forecasting Questions:

1. What will the sales be in the next month/quarter/year for each product category?
 - Forecasting sales trends helps plan for inventory, marketing, and resource allocation.

```
df_cleaned['Order Date'] = pd.to_datetime(df_cleaned['Order Date'],
errors='coerce')

monthly_sales = df_cleaned.groupby([pd.Grouper(key='Order Date',
freq='M'), 'Category'])['Sales'].sum().reset_index()

monthly_sales_pivot = monthly_sales.pivot(index='Order Date',
columns='Category', values='Sales')

monthly_sales_pivot.plot(figsize=(10, 6), title='Monthly Sales by
Category')

plt.ylabel('Total Sales')

plt.show()
```



```
from sklearn.model_selection import train_test_split
```

```

from sklearn.linear_model import LinearRegression

furniture_sales =
monthly_sales_pivot['Furniture'].dropna().reset_index()

furniture_sales['Month_Num'] = np.arange(len(furniture_sales))

X = furniture_sales[['Month_Num']]

y = furniture_sales['Furniture']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

plt.figure(figsize=(10, 6))

plt.plot(furniture_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(furniture_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('Furniture Sales Forecast')

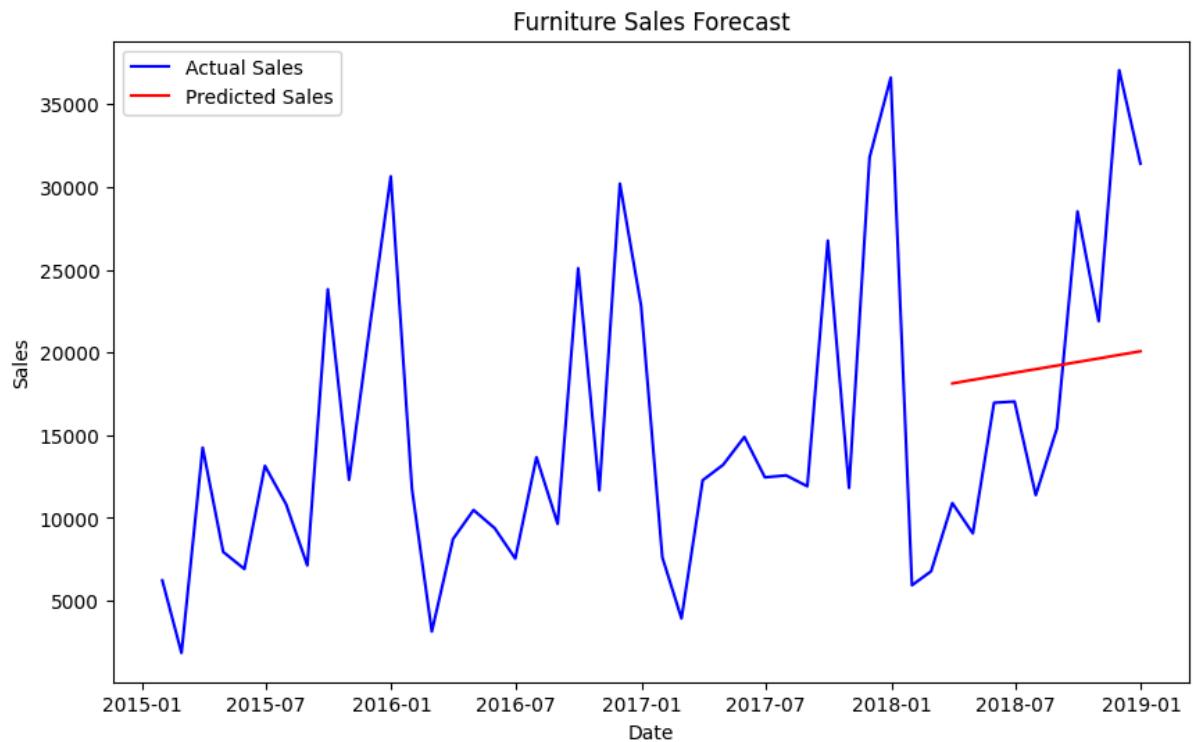
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

future_months = np.arange(len(furniture_sales), len(furniture_sales) +
12).reshape(-1, 1)

future_sales_pred = model.predict(future_months)

future_dates = pd.date_range(furniture_sales['Order Date'].max(),
periods=12, freq='ME')

plt.plot(future_dates, future_sales_pred, label='Forecasted Sales',
color='green')

plt.title('Furniture Sales Forecast for 2019')

plt.xlabel('Date')

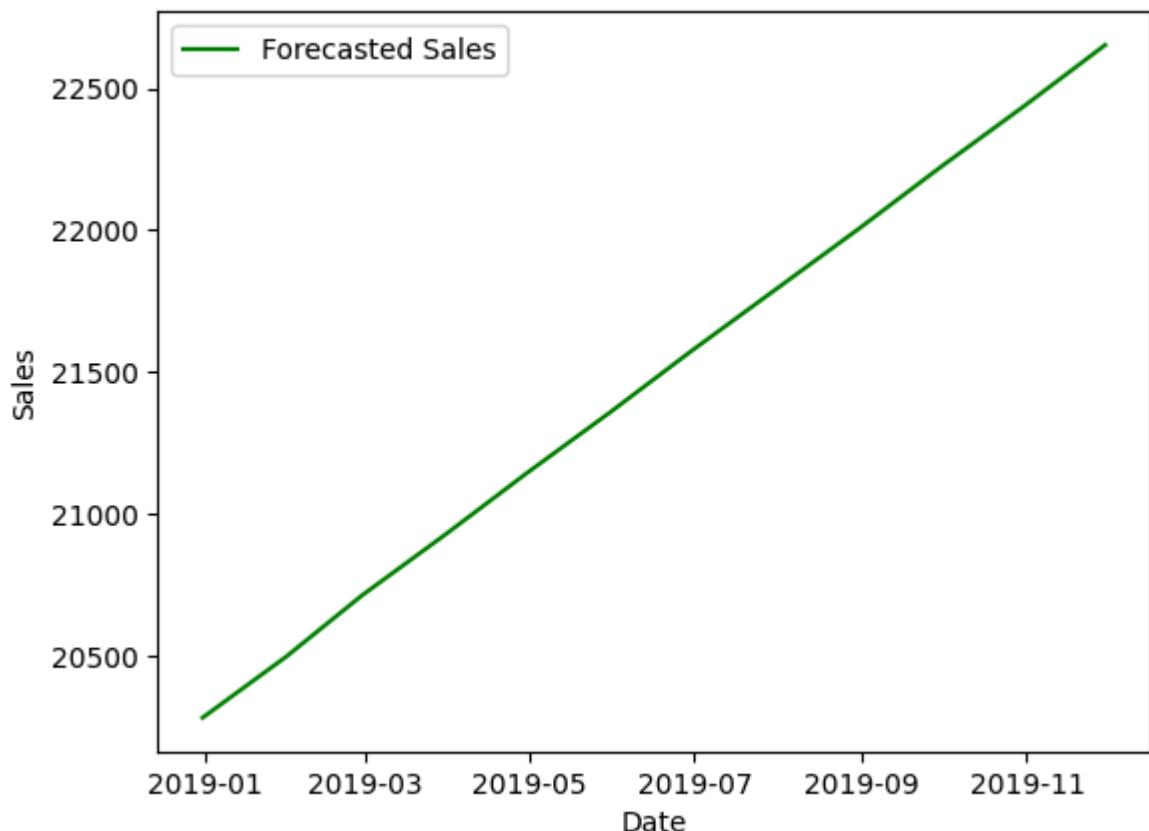
plt.ylabel('Sales')

plt.legend()

plt.show()

```

Furniture Sales Forecast for 2019



```
office_supplies_sales = monthly_sales_pivot['Office
Supplies'].dropna().reset_index()

office_supplies_sales['Month_Num'] =
np.arange(len(office_supplies_sales))

X = office_supplies_sales[['Month_Num']]

y = office_supplies_sales['Office Supplies']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()
```

```

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

plt.figure(figsize=(10, 6))

plt.plot(office_supplies_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(office_supplies_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('Office Supplies Sales Forecast')

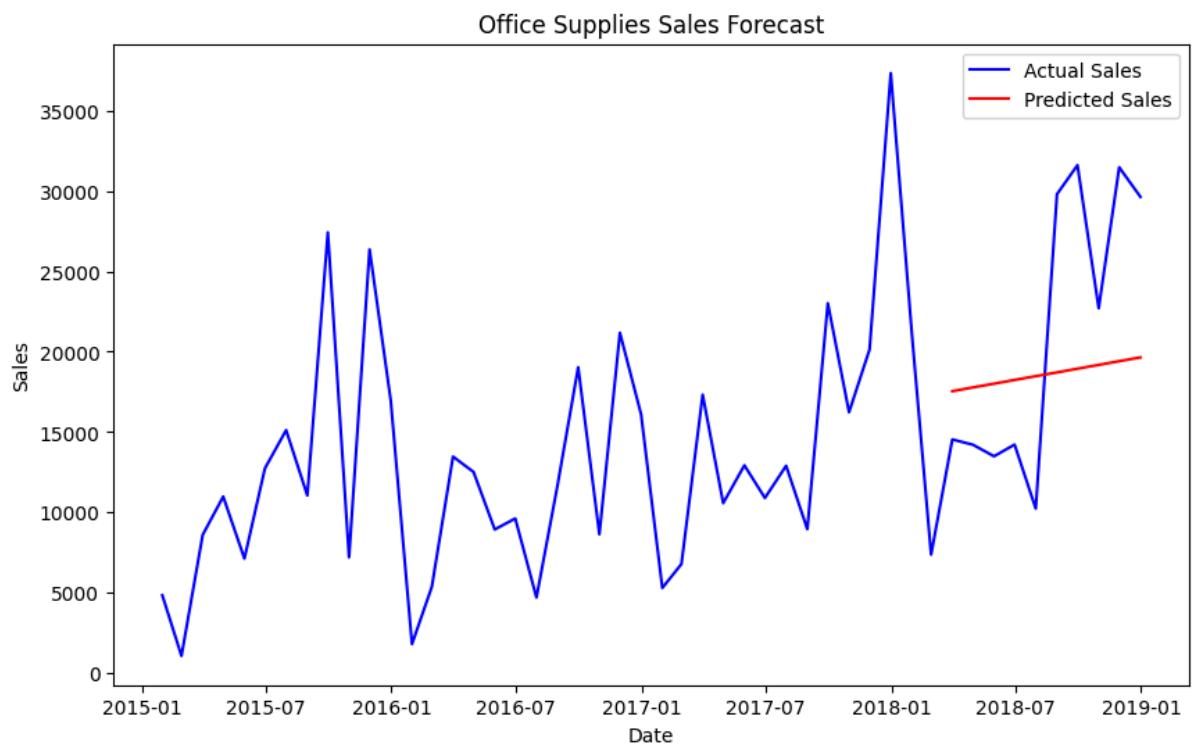
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

future_months = np.arange(len(office_supplies_sales),
len(office_supplies_sales) + 12).reshape(-1, 1)

future_sales_pred = model.predict(future_months)

future_dates = pd.date_range(office_supplies_sales['Order Date'].max(),
periods=12, freq='M')

plt.plot(future_dates, future_sales_pred, label='Forecasted Sales',
color='green')

plt.title('Office Supplies Sales Forecast for 2019')

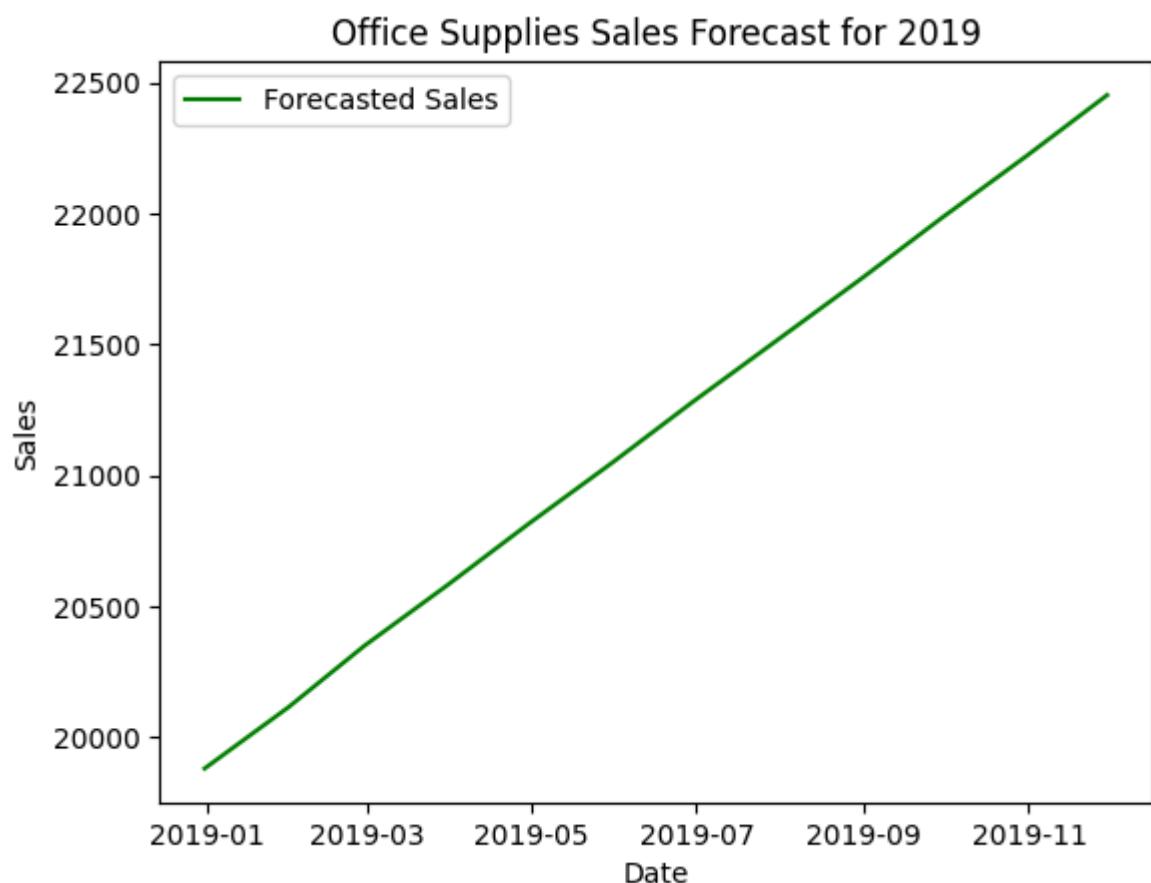
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

Technology_sales =
monthly_sales_pivot['Technology'].dropna().reset_index()

```

```

Technology_sales['Month_Num'] = np.arange(len(Technology_sales))

X = Technology_sales[['Month_Num']]

y = Technology_sales['Technology']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

plt.figure(figsize=(10, 6))

plt.plot(Technology_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(Technology_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('Technology Supplies Sales Forecast')

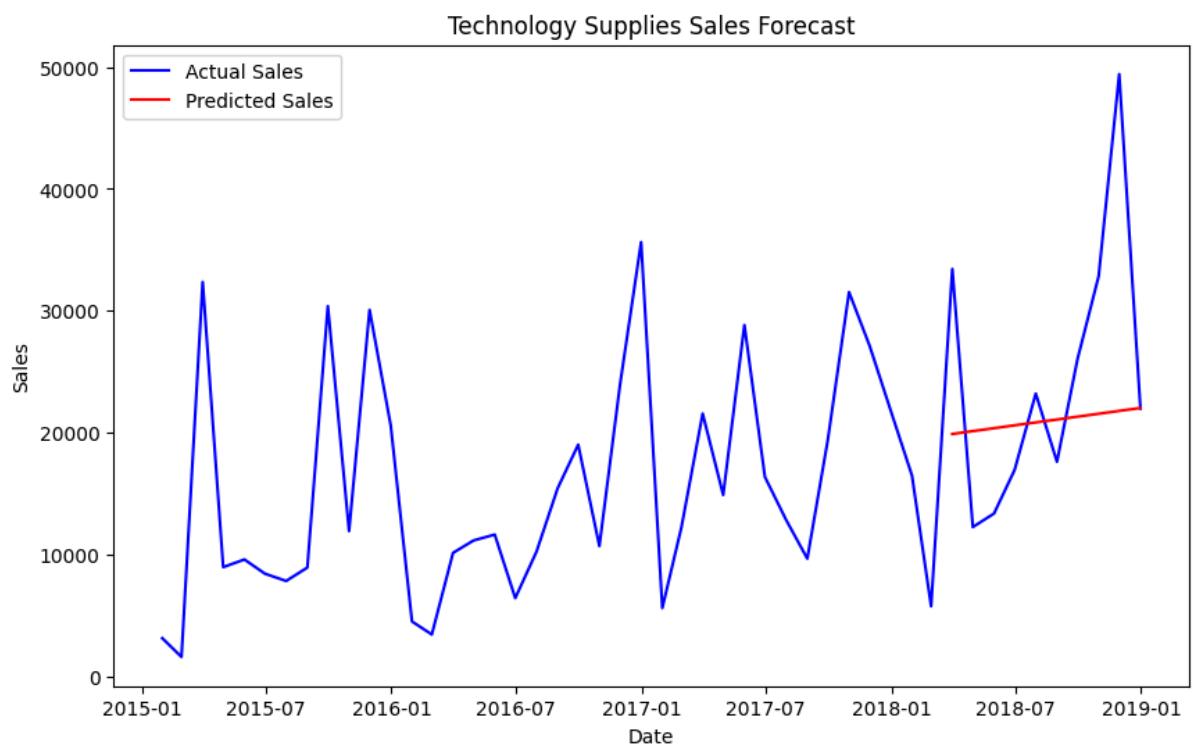
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

future_months = np.arange(len(Technology_sales), len(Technology_sales)
+ 12).reshape(-1, 1)

future_sales_pred = model.predict(future_months)

future_dates = pd.date_range(Technology_sales['Order Date'].max(),
periods=12, freq='ME')

plt.plot(future_dates, future_sales_pred, label='Forecasted Sales',
color='green')

plt.title('Office Supplies Sales Forecast for 2019')

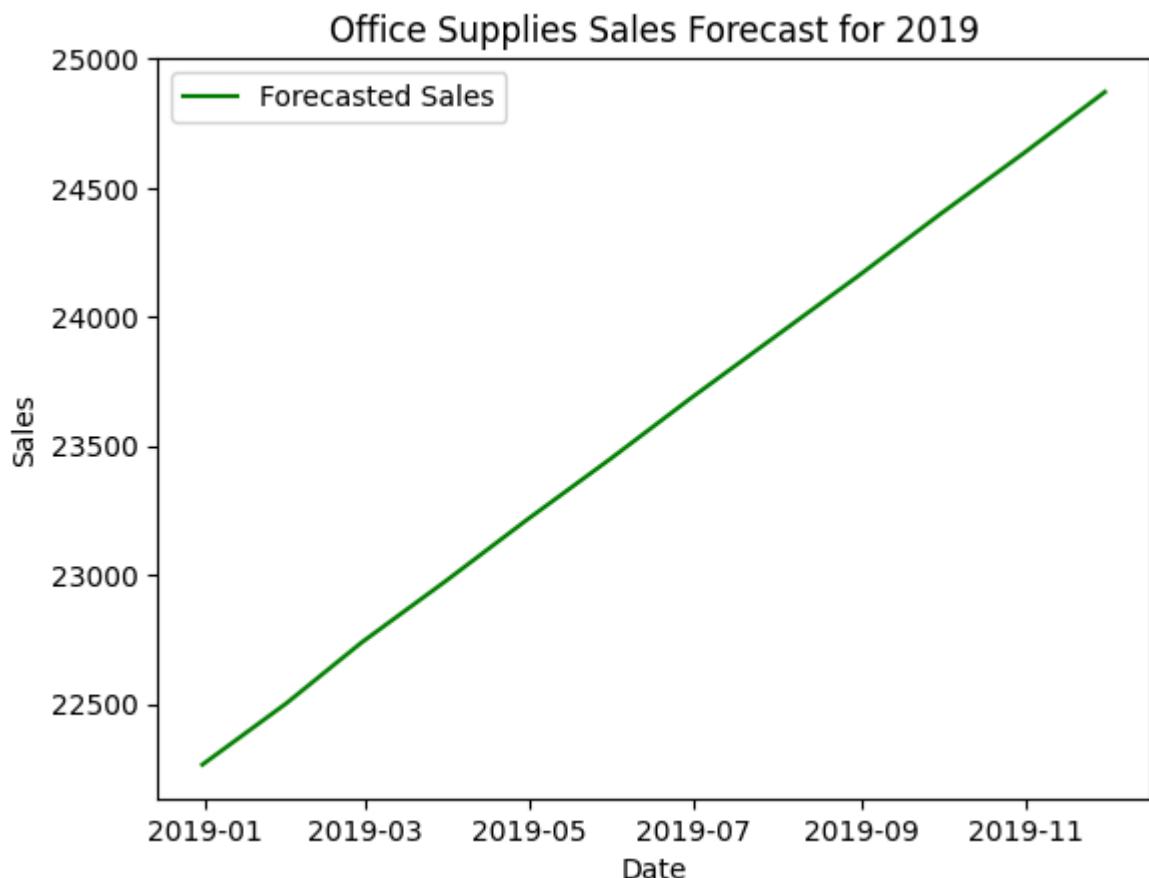
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



2. Which regions are expected to see the most growth in sales?
 - Regional sales forecasting can help target high-potential areas and optimize resource distribution.

```

monthly_sales_region = df.groupby([pd.Grouper(key='Order Date',
freq='M'), 'Region']).sum()['Sales'].reset_index()

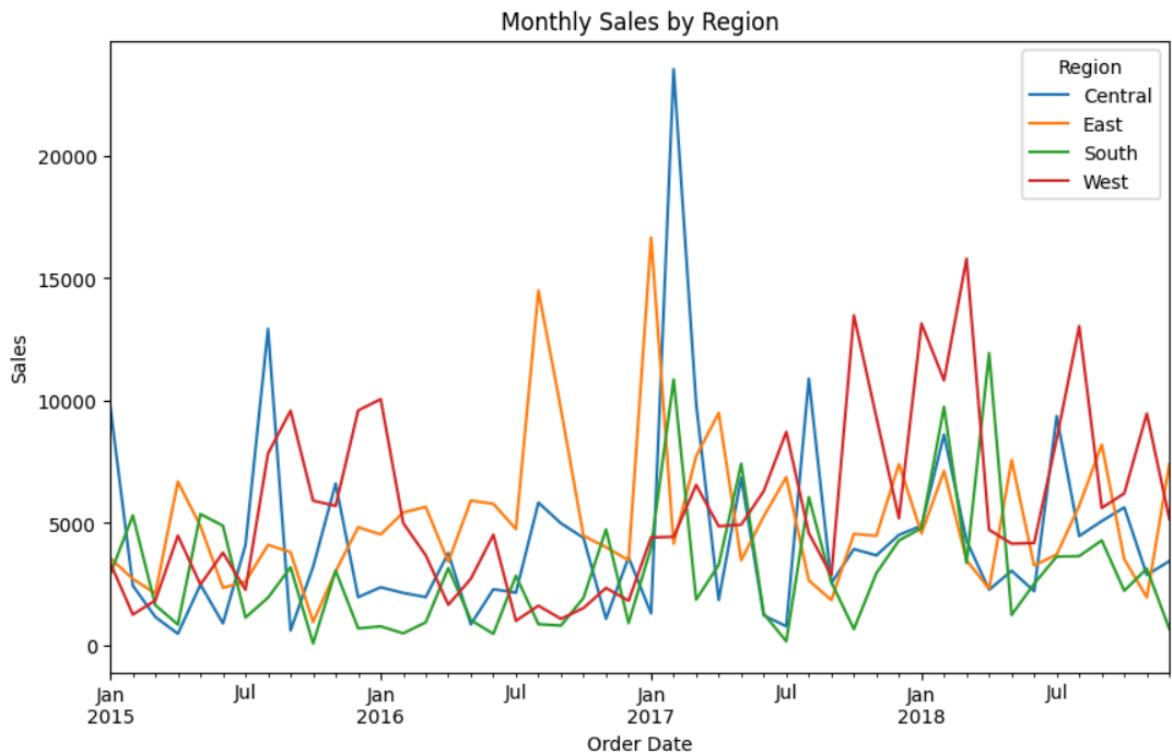
monthly_sales_region_pivot = monthly_sales_region.pivot(index='Order
Date', columns='Region', values='Sales')

monthly_sales_region_pivot.plot(figsize=(10, 6), title='Monthly Sales
by Region')

plt.ylabel('Sales')

plt.show()

```



```

west_sales = monthly_sales_region_pivot['West'].dropna().reset_index()

west_sales['Month_Num'] = range(len(west_sales))

X = west_sales[['Month_Num']]

y = west_sales['West']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

plt.plot(west_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(west_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

```

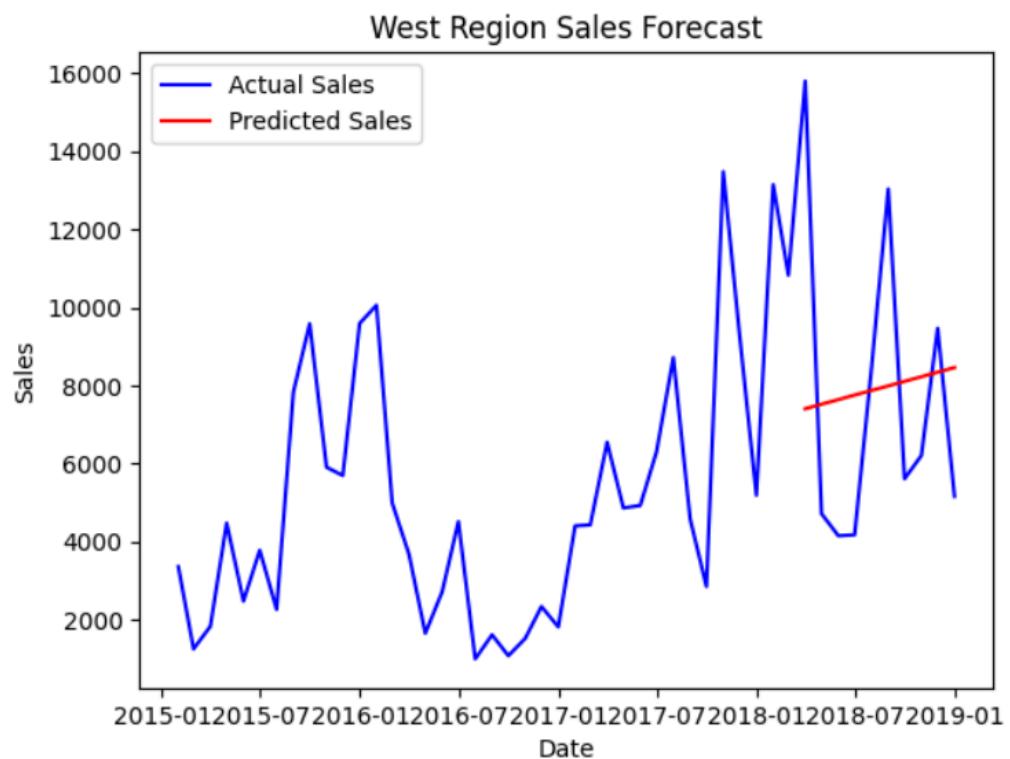
```
plt.title('West Region Sales Forecast')

plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()
```



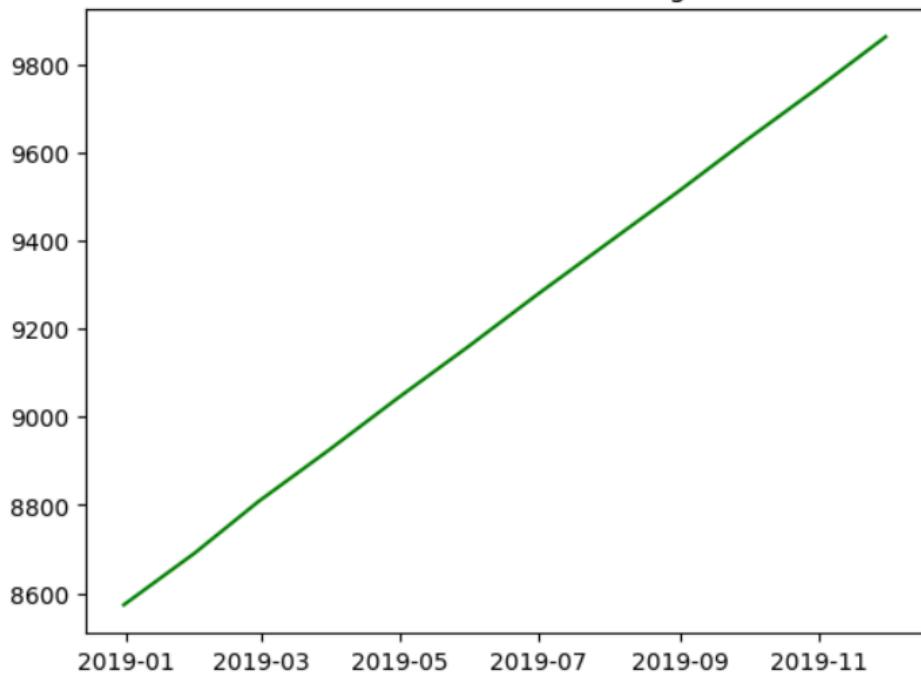
```
future_sales_pred = model.predict(future_months)

plt.plot(future_dates, future_sales_pred, label='Future Forecast for
West Region', color='green')

plt.title('Future Forecast for West Region')

plt.show()
```

Future Forecast for West Region



```
south_sales =  
monthly_sales_region_pivot['South'].dropna().reset_index()  
  
south_sales['Month_Num'] = range(len(south_sales))  
  
x = south_sales[['Month_Num']]  
  
y = south_sales['South']  
  
  
x_train, x_test, y_train, y_test = train_test_split(x, y,  
test_size=0.2, shuffle=False)  
  
  
model = LinearRegression()  
  
model.fit(x_train, y_train)
```

```
y_pred = model.predict(x_test)  
  
plt.plot(south_sales['Order Date'], y, label='Actual Sales',  
color='blue')
```

```

plt.plot(south_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('South Region Sales Forecast')

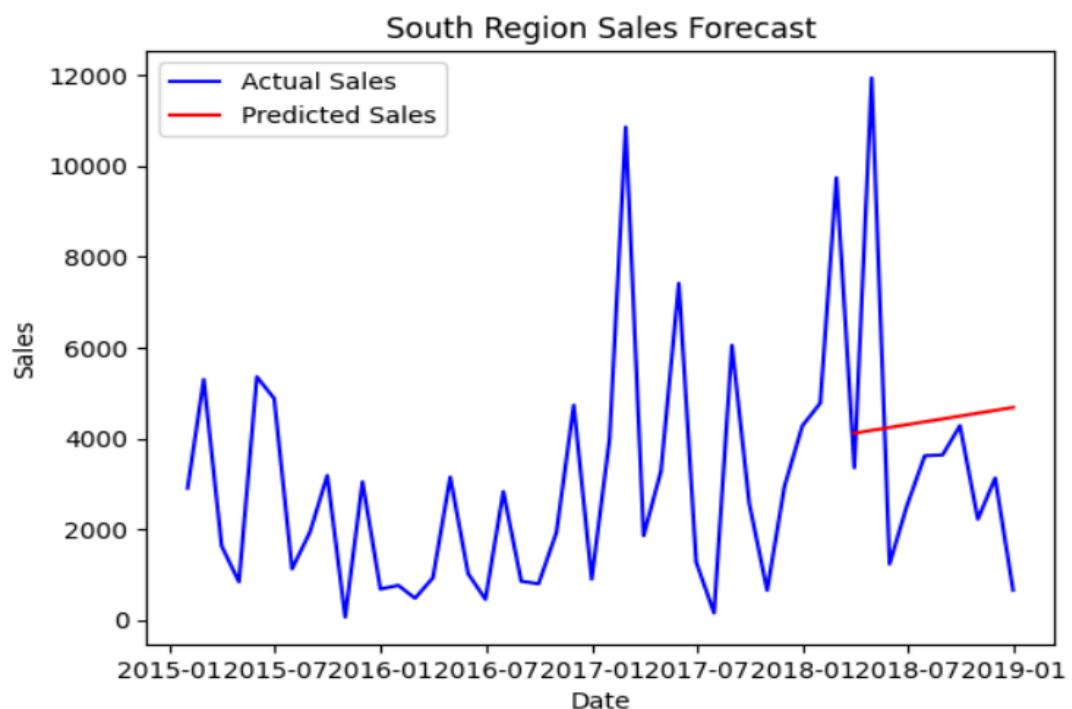
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

future_sales_pred = model.predict(future_months)

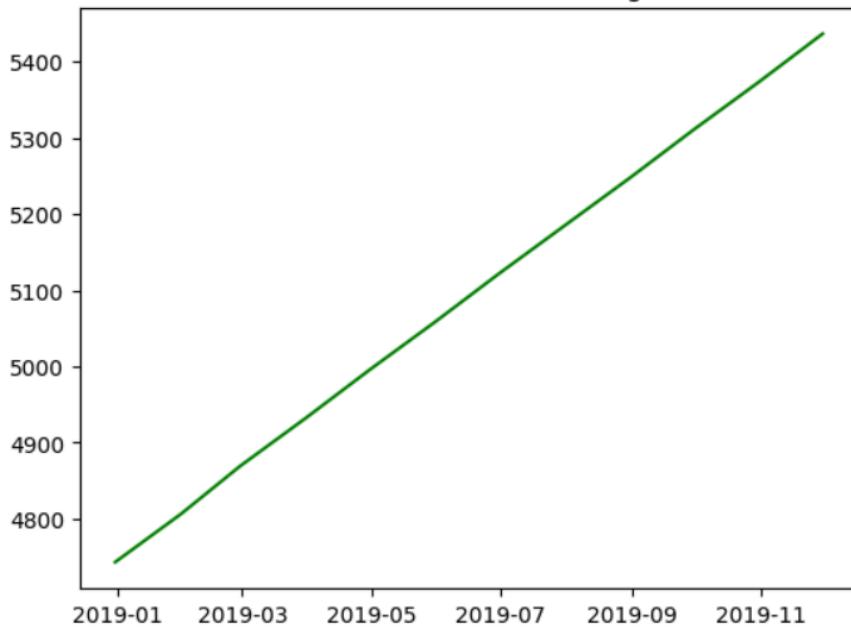
plt.plot(future_dates, future_sales_pred, label='Future Forecast for
West Region', color='green')

plt.title('Future Forecast for South Region')

plt.show()

```

Future Forecast for South Region



```
East_sales = monthly_sales_region_pivot['East'].dropna().reset_index()

East_sales['Month_Num'] = range(len(East_sales))

X = East_sales[['Month_Num']]

y = East_sales['East']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)

plt.plot(East_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(East_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

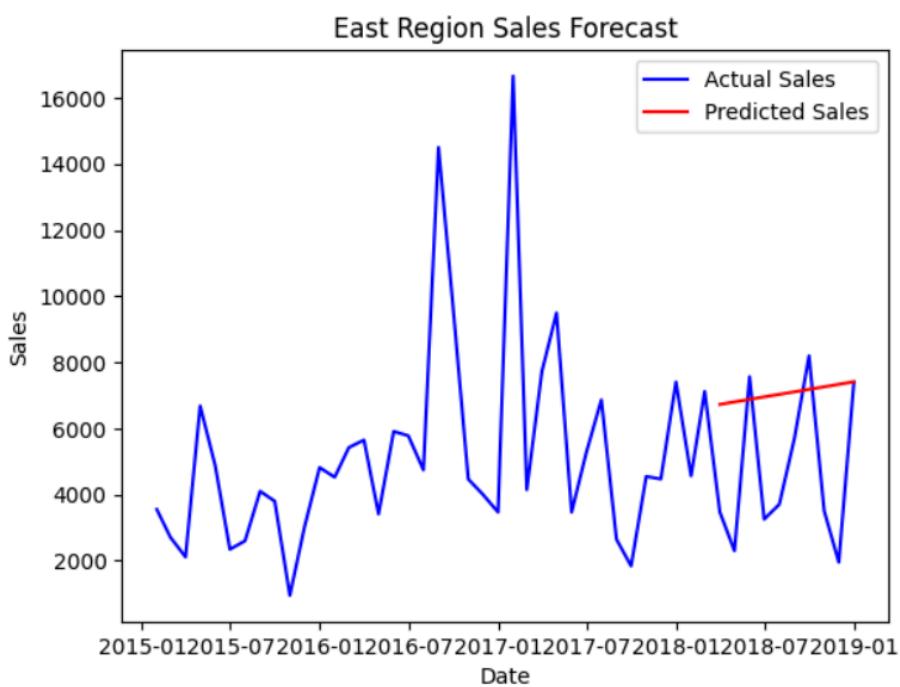
plt.title('East Region Sales Forecast')
```

```
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()
```

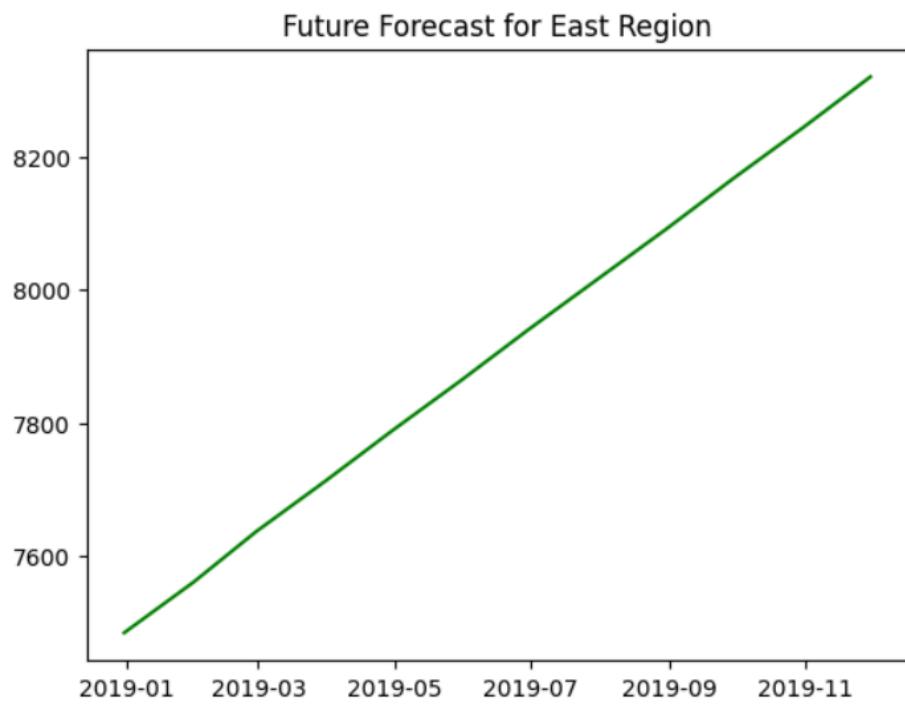


```
future_sales_pred = model.predict(future_months)

plt.plot(future_dates, future_sales_pred, label='Future Forecast for
East Region', color='green')

plt.title('Future Forecast for East Region')

plt.show()
```



```

central_sales =
monthly_sales_region_pivot['South'].dropna().reset_index()

central_sales['Month_Num'] = range(len(central_sales))

X = central_sales[['Month_Num']]

y = central_sales['South']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(X_train, y_train)

```

```

y_pred = model.predict(X_test)

plt.plot(central_sales['Order Date'], y, label='Actual Sales',
color='blue')

```

```

plt.plot(central_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('East Region Sales Forecast')

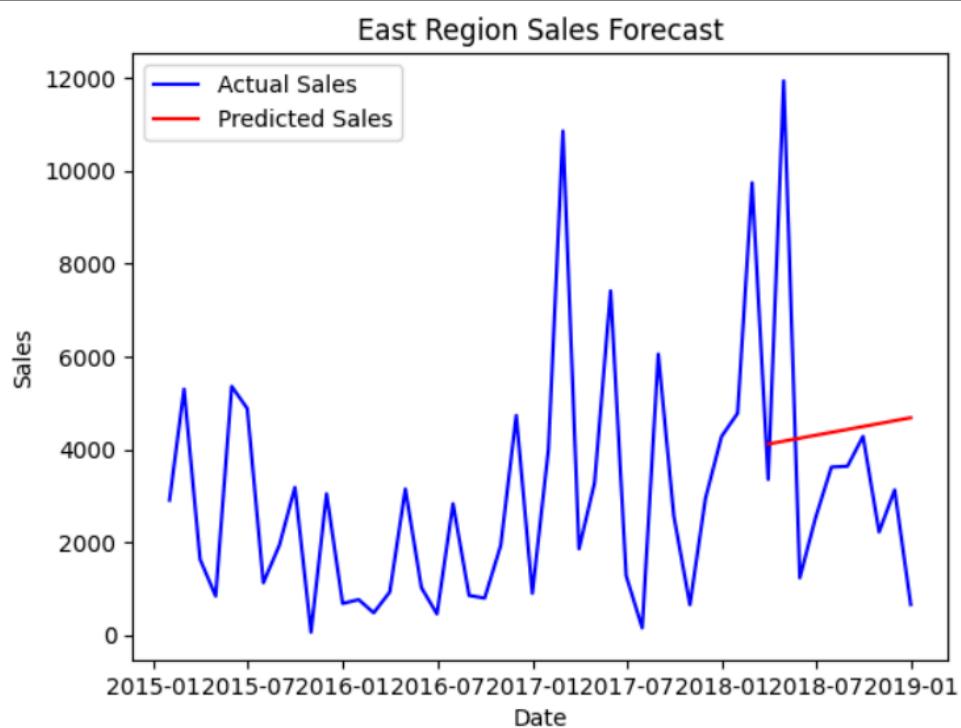
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

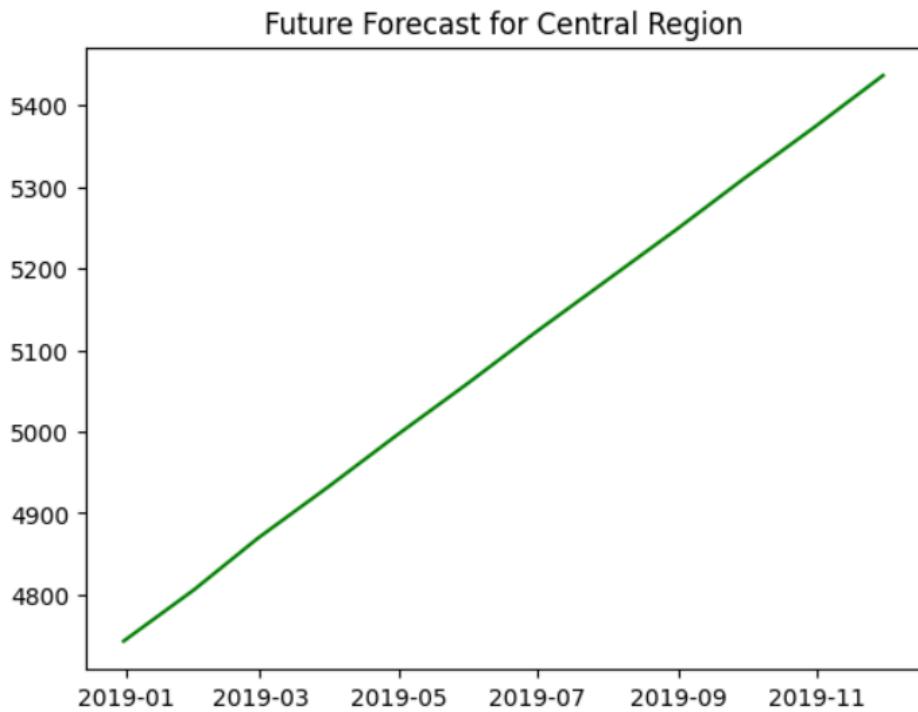
future_sales_pred = model.predict(future_months)

plt.plot(future_dates, future_sales_pred, label='Future Forecast for
Central Region', color='green')

plt.title('Future Forecast for Central Region')

plt.show()

```



- 3. How will the demand for different product categories evolve over time?**
- Predicting demand changes by category will help with supply chain and inventory management.

```

for category in monthly_sales_pivot.columns:

    category_sales =
monthly_sales_pivot[category].dropna().reset_index()

    category_sales['Month_Num'] = range(len(category_sales))

    X = category_sales[['Month_Num']]

    y = category_sales[category]

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

    model = LinearRegression()

```

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

plt.figure(figsize=(10, 6))

plt.plot(category_sales['Order Date'], y, label=f'Actual {category} Sales', color='blue')

plt.plot(category_sales['Order Date'][len(X_train):], y_pred, label=f'Predicted {category} Sales', color='red')

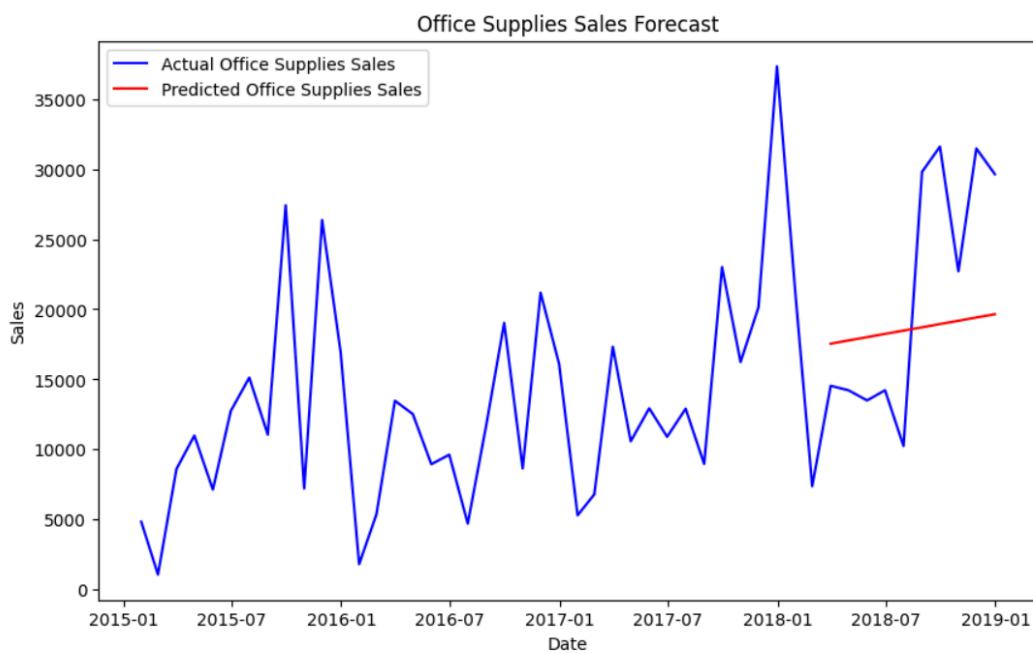
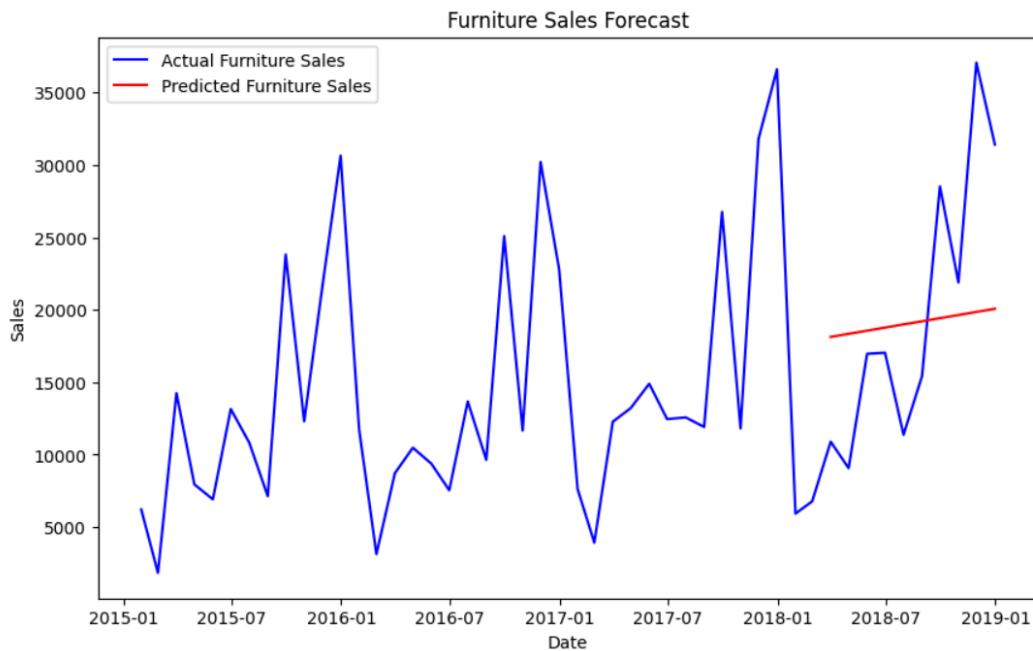
plt.title(f'{category} Sales Forecast')

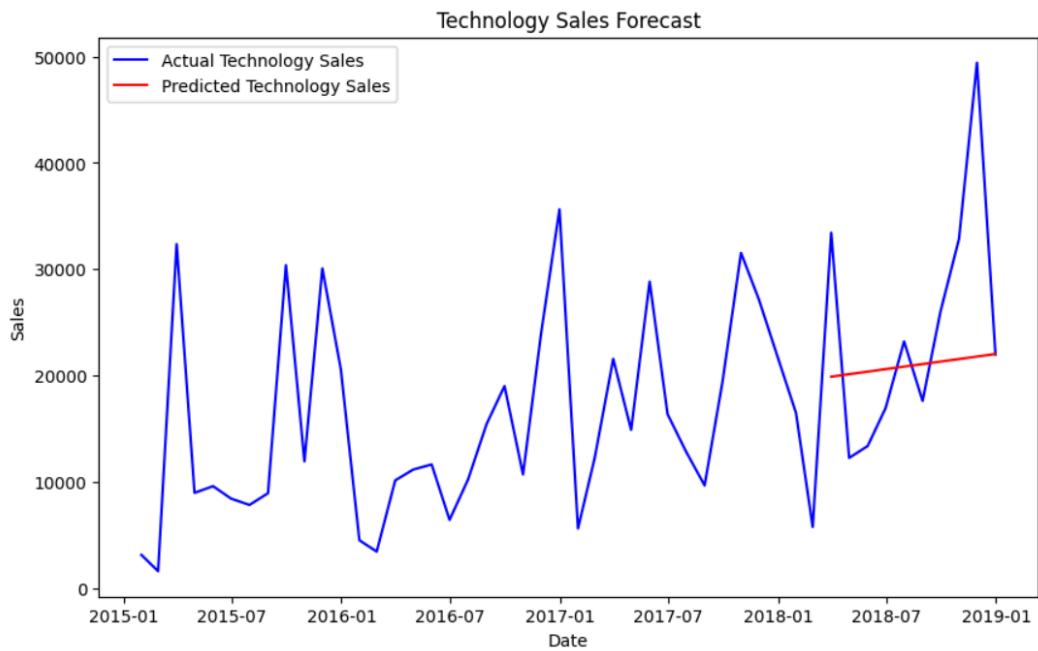
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()
```





4. What are the expected peak sales periods in the upcoming months?

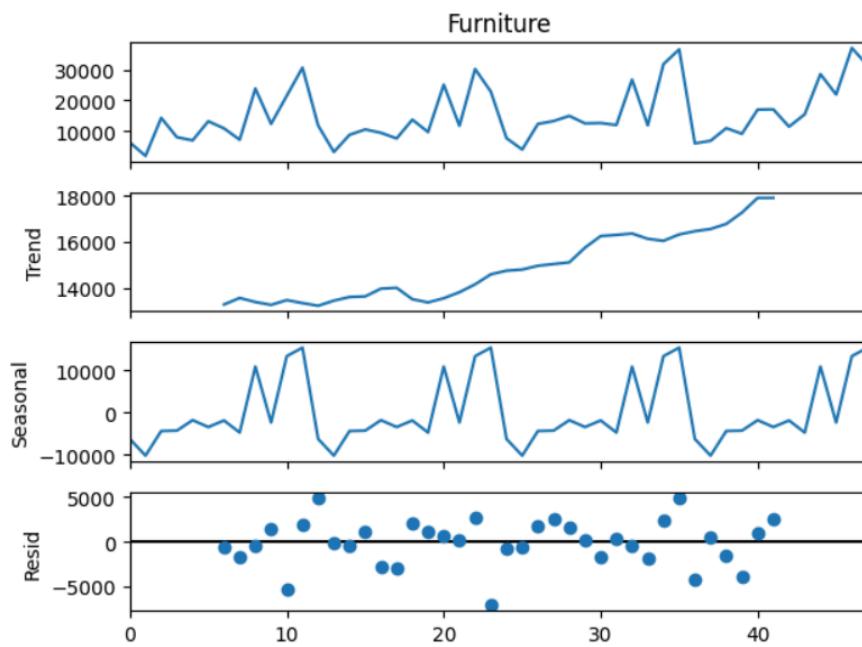
- Knowing peak sales periods can assist in inventory planning and promotional strategies.

```
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(furniture_sales['Furniture'],
model='additive', period=12)

decomposition.plot()

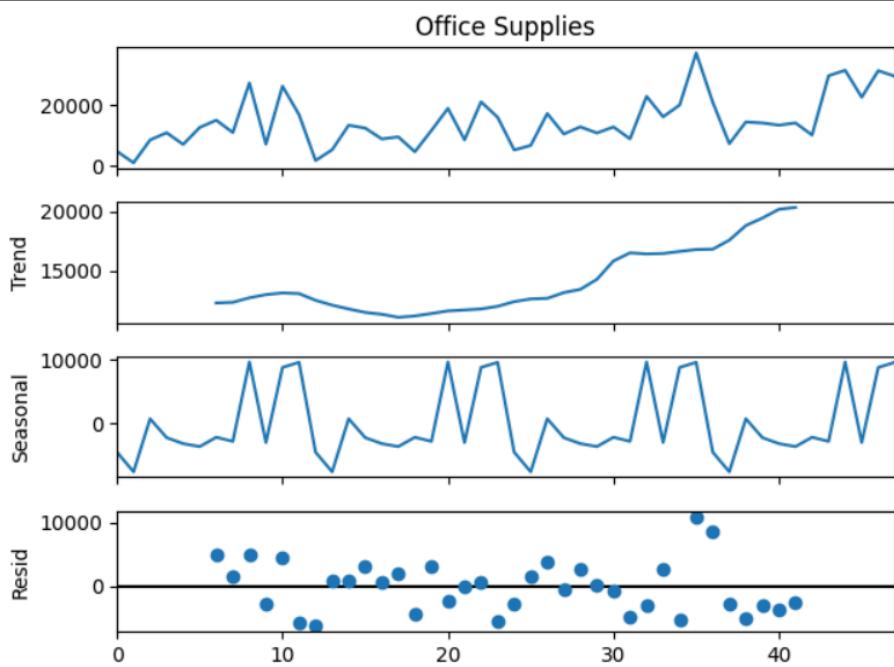
plt.show()
```



```
decomposition = seasonal_decompose(office_supplies_sales['Office
Supplies'], model='additive', period=12)

decomposition.plot()

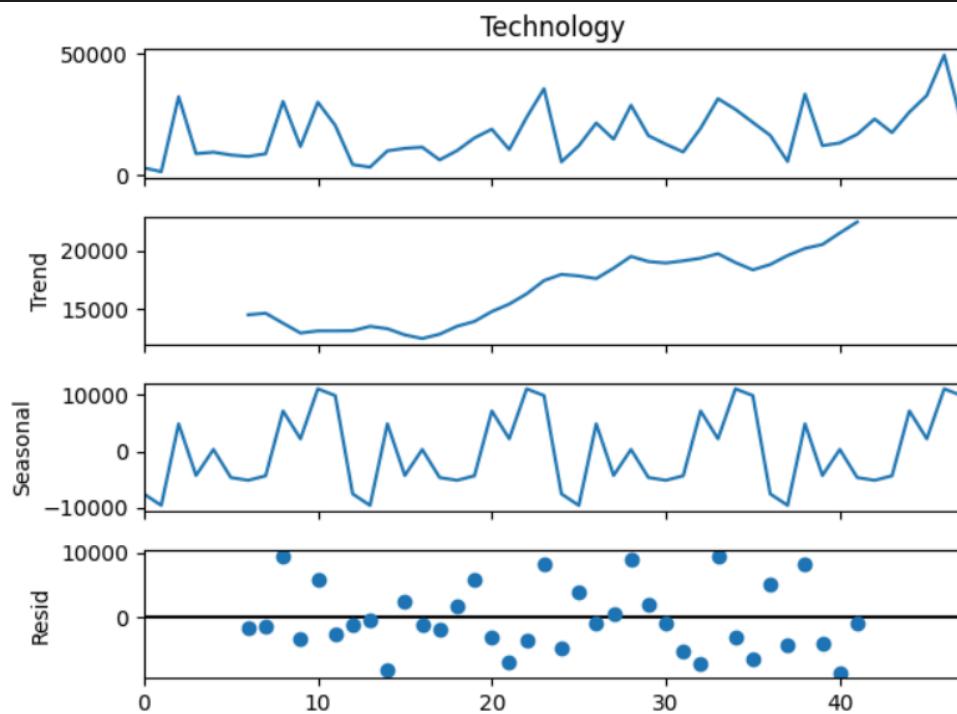
plt.show()
```



```
decomposition = seasonal_decompose(Technology_sales['Technology'],
model='additive', period=12)

decomposition.plot()
```

```
plt.show()
```



5. What will be the expected revenue for key customer segments (e.g., Consumer, Corporate, Home Office)?

- Segment-based forecasting helps customize marketing campaigns for specific customer groups.

```
monthly_sales_segment = df.groupby([pd.Grouper(key='Order Date', freq='M'), 'Segment']).sum()['Sales'].reset_index()

monthly_sales_segment_pivot = monthly_sales_segment.pivot(index='Order Date', columns='Segment', values='Sales')

monthly_sales_segment_pivot.plot(figsize=(10, 6), title='Monthly Sales by Customer Segment')

plt.ylabel('Sales')

plt.show()
```



```

consumer_sales =
monthly_sales_segment_pivot['Consumer'].dropna().reset_index()

consumer_sales['Month_Num'] = np.arange(len(consumer_sales))

X = consumer_sales[['Month_Num']]

y = consumer_sales['Consumer']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

plt.figure(figsize=(10, 6))

plt.plot(consumer_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(consumer_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('Consumer Segment Sales Forecast')

```

```

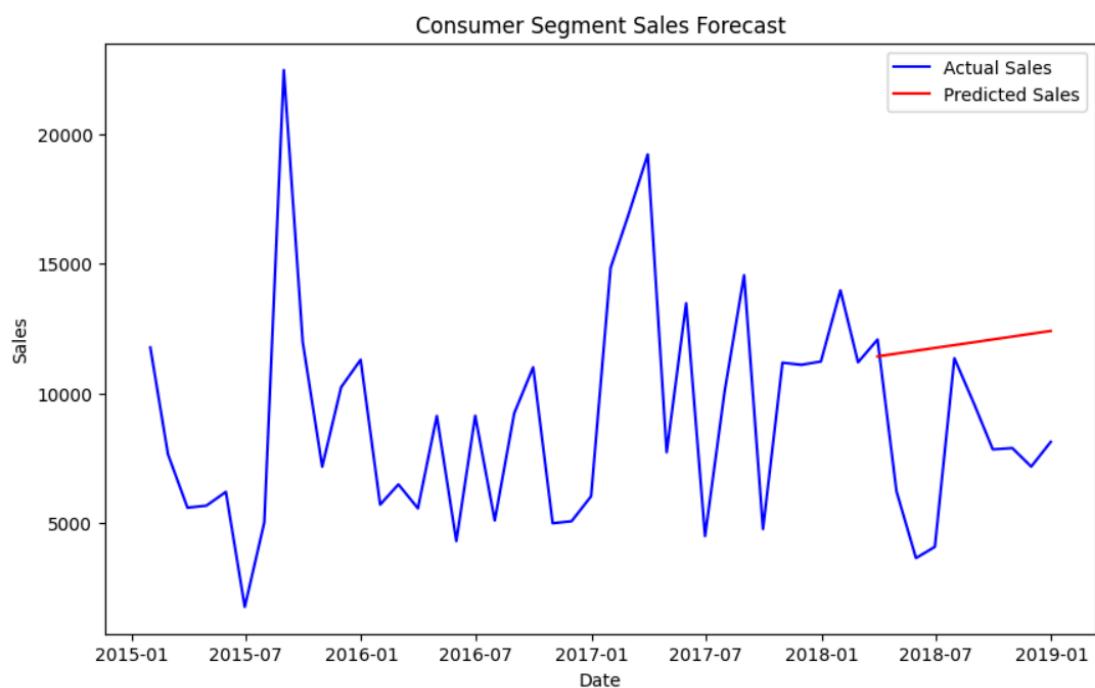
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

corporate_sales =
monthly_sales_segment_pivot['Corporate'].dropna().reset_index()

corporate_sales['Month_Num'] = np.arange(len(corporate_sales))

X = corporate_sales[['Month_Num']]

y = corporate_sales['Corporate']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

```

```

plt.figure(figsize=(10, 6))

plt.plot(corporate_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(corporate_sales['Order Date'][len(X_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('Corporate Segment Sales Forecast')

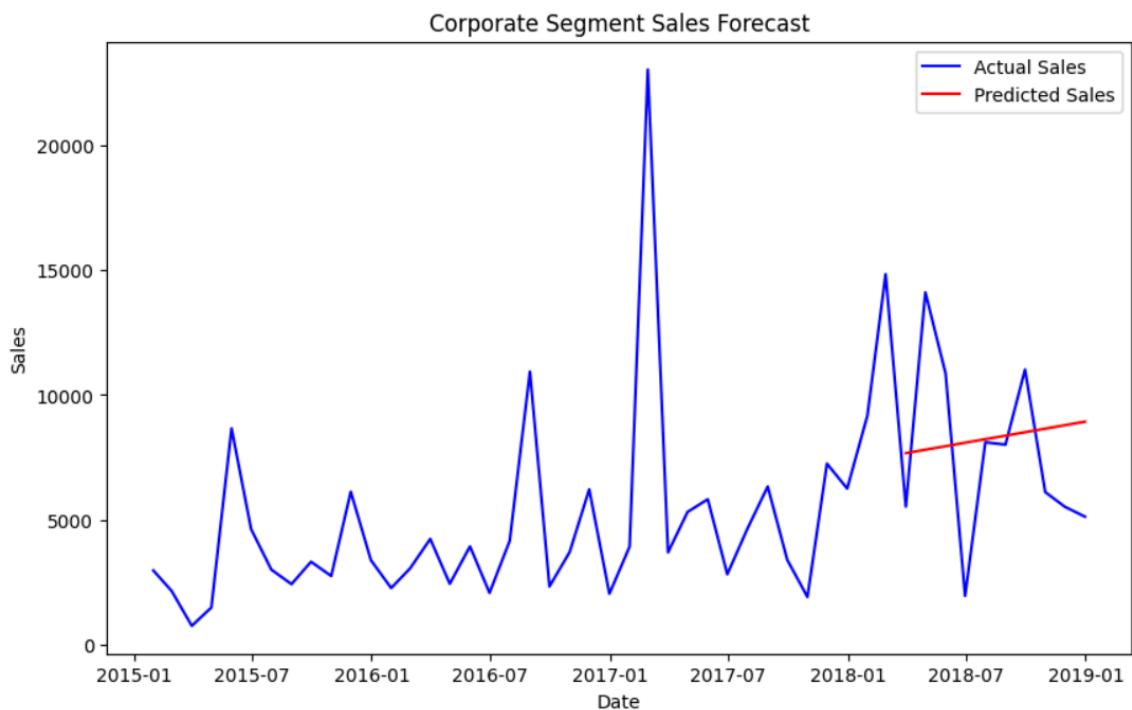
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



```

home_office_sales = monthly_sales_segment_pivot['Home
Office'].dropna().reset_index()

home_office_sales['Month_Num'] = np.arange(len(home_office_sales))

X = home_office_sales[['Month_Num']]

y = home_office_sales['Home Office']

```

```

x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

model = LinearRegression()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

plt.figure(figsize=(10, 6))

plt.plot(home_office_sales['Order Date'], y, label='Actual Sales',
color='blue')

plt.plot(home_office_sales['Order Date'][len(x_train):], y_pred,
label='Predicted Sales', color='red')

plt.title('Home Office Segment Sales Forecast')

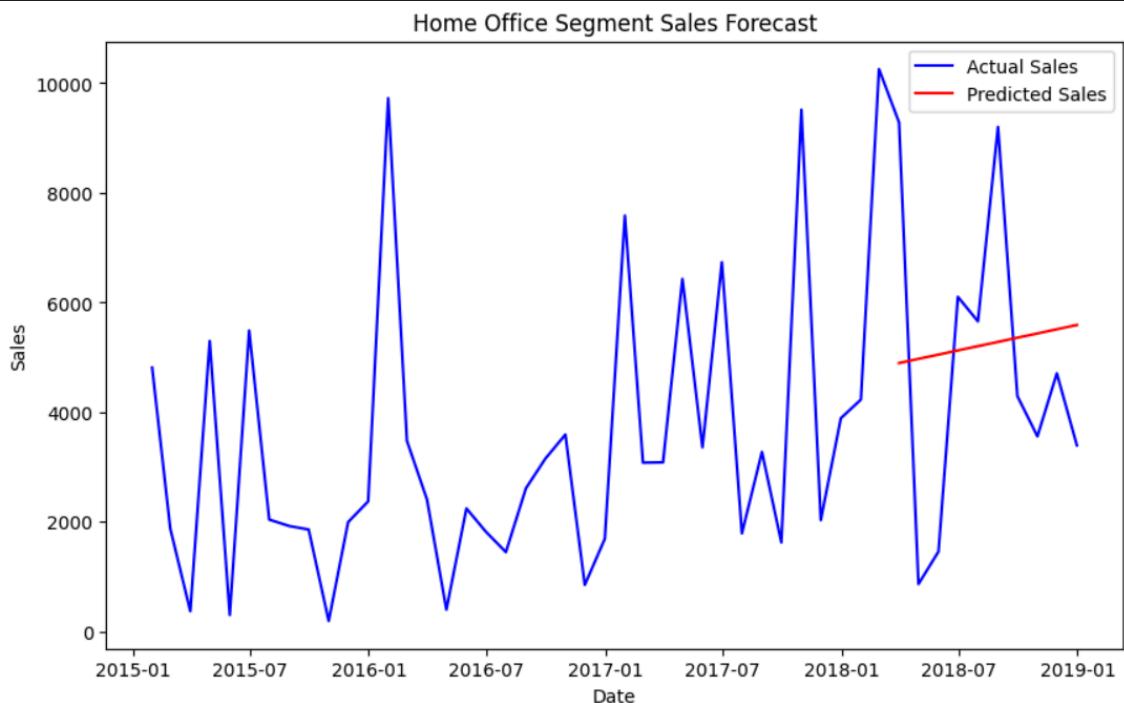
plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

```



Task 4

Week 4: Visualization Dashboard and Final Presentation

- **Tasks:**

- Build a Visualization Dashboard: Build a Tableau visualization dashboard that visualizes the answers to all answered questions.
- Final Presentation: Prepare a report and presentation summarizing the project work, including data analysis, model development, and deployment.
- Tools: SQL, Python (pandas, Matplotlib), Tableau.

- **Deliverables:**

- Visualization dashboard.

Final report and presentation.

