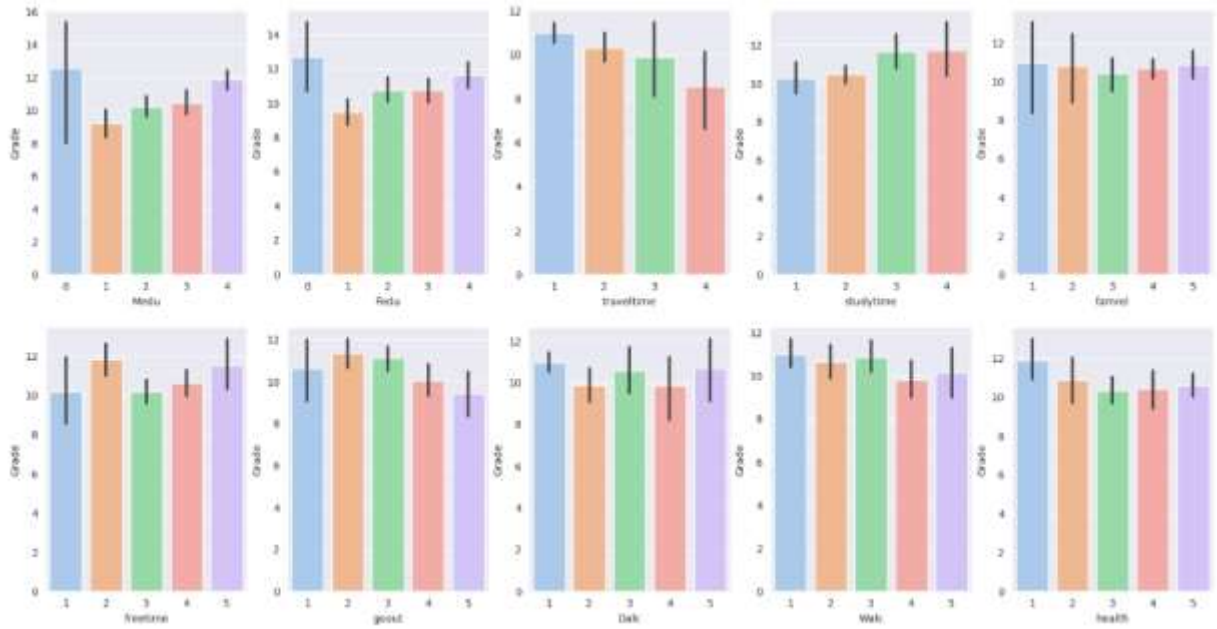
[22]:

```
nrows, ncols = 2, 5
i, j = 0, 0
fig, axes = plt.subplots(nrows, ncols, figsize=(20, 10))
for col in num_cat_cols:
    if col == 'failures':
        continue
    ax = axes[i//ncols][j%ncols]
    sns.barplot(data=data, x=col, y="Grade", ax=ax);
    i += 1
    j += 1
plt.savefig("barplot numerical categories")
```



PROJECT :- 2

E-commerce Sales Analysis

Declaration : -

I declare that this E-commerce Sales Analysis is my original work and has been completed to the best of my abilities. All data used in this analysis has been sourced from reliable and authorized channels, and any external references or sources have been duly cited. The findings, visualizations, and recommendations presented in this report are based on the data analysis conducted and are aimed at providing actionable insights to enhance e-commerce sales performance.

I affirm that the techniques and methodologies applied in this analysis adhere to standard data analysis practices and ethical guidelines. This analysis has been conducted with integrity and objectivity, ensuring that the conclusions drawn are unbiased and supported by the data.

Any resemblance to existing works is purely coincidental, and any errors or inaccuracies in the analysis are unintentional. I take full responsibility for the content and outcomes of this analysis and am committed to making any necessary corrections if discrepancies are identified.

By submitting this analysis, I agree to abide by the principles of academic and professional integrity, and I understand the importance of maintaining high standards of honesty and transparency in all analytical endeavors.

PROBLEM STATEMENT : -

E-commerce Sales Analysis

Work with a dataset of e-commerce transactions including product details, prices,and purchase timestamps.

Analyze sales trends over time and identify peak selling periods.

Explore the distribution of product prices and customer spending habits.

Segment customers based on purchasing behavior or demographic information.Generate insights to optimize product offerings and marketing strategies.

Introduction

In the digital age, e-commerce has revolutionized the way businesses operate, offering unprecedented access to global markets and a wealth of data about consumer behavior. This transformation has created immense opportunities for businesses to leverage data analytics to gain insights into sales performance, optimize operations, and drive growth. E-commerce sales analysis involves the systematic examination of sales data to understand trends, patterns, and correlations that can inform strategic decision-making.

Importance of E-commerce Sales Analysis

Analyzing e-commerce sales data is crucial for several reasons:

1. Understanding Customer Behavior: By examining purchasing patterns, businesses can gain insights into customer preferences, seasonal trends, and buying cycles. This information helps in tailoring marketing strategies to meet customer needs more effectively.

2. Optimizing Inventory Management: Sales analysis helps in predicting demand, reducing overstock and stockouts, and ensuring that popular products are always available. This leads to better inventory turnover and reduced holding costs.

3. Enhancing Marketing Effectiveness: By identifying which marketing campaigns and channels drive the most sales, businesses can allocate their marketing budget more efficiently, targeting the right audience with the right message at the right time.

4. Improving Customer Retention: Analyzing sales data can help identify patterns that lead to repeat purchases and customer loyalty. Businesses can then implement strategies to enhance the customer experience and retain valuable customers.

5. Driving Strategic Decisions: Comprehensive sales analysis provides the insights needed to make informed decisions about product development, pricing strategies, market expansion, and more.

Key Components of E-commerce Sales Analysis

1. Data Collection and Preparation: The first step in any sales analysis is to gather relevant data. This typically includes sales transactions, product information, customer demographics, and marketing data. Ensuring data accuracy and completeness is essential for reliable analysis.

2. Descriptive Statistics: Basic statistical measures, such as mean, median, and standard deviation, provide an overview of the data and help identify outliers and trends. This initial analysis sets the stage for more detailed exploration.

3. Data Visualization: Visual tools like bar charts, histograms, scatter plots, and time series graphs are invaluable for understanding data at a glance. They make it easier to spot patterns, trends, and anomalies that might not be immediately apparent in raw data.

4. Trend Analysis: Examining sales trends over time helps businesses understand seasonal variations, growth patterns, and the impact of external factors like holidays and economic changes. This analysis is crucial for forecasting future sales and planning inventory and marketing strategies.

5. Correlation Analysis: Investigating relationships between different variables, such as advertising spend and sales, or customer demographics and product preferences, can reveal insights into what drives sales and how different factors interact.

6. Predictive Analytics: Advanced techniques, such as machine learning models, can predict future sales trends based on historical data. These predictions help businesses make proactive decisions to capitalize on upcoming opportunities and mitigate potential risks.

Steps in E-commerce Sales Analysis

1. Load and Explore the Dataset

The first step is to load the dataset into a data analysis tool such as Python or R. Exploring the dataset involves understanding its structure, identifying key variables, and checking for missing or incorrect data.

2. Data Cleaning and Preparation

Data cleaning involves handling missing values, correcting errors, and formatting the data for analysis. This step ensures that the analysis is based on accurate and reliable data.

3. Descriptive Statistics

Calculating basic statistics provides a summary of the data, including measures of central tendency (mean, median) and dispersion (standard deviation). This summary helps in understanding the overall distribution and variability of the sales data.

4. Data Visualization

Visualizing the data using charts and graphs makes it easier to understand complex relationships and trends. Common visualizations include:

- **Histograms**: Show the distribution of sales data.
- **Bar Charts**: Compare sales across different categories, such as product types or regions.
- **Scatter Plots**: Explore relationships between two variables, such as price and sales volume.
- **Time Series Plots**: Analyze sales trends over time.

5. Trend Analysis

Analyzing trends involves examining how sales change over time, identifying seasonal patterns, and understanding the impact of external events. This analysis helps in forecasting future sales and planning for peak periods.

6. Correlation Analysis

Correlation analysis examines the relationships between different variables to identify factors that influence sales. For example, it might reveal that higher advertising spend is associated with increased sales, or that certain customer demographics are more likely to purchase specific products.

7. Predictive Analytics

Using historical data to build predictive models can help forecast future sales trends. Techniques such as regression analysis, time series forecasting, and machine learning algorithms can provide valuable predictions that inform strategic planning.

Recommendations for Improving Sales Performance

Based on the analysis, businesses can develop targeted strategies to improve sales performance. Recommendations might include:

- Optimizing Marketing Campaigns: Focus on the most effective marketing channels and tailor campaigns to target high-value customer segments.
- Enhancing Product Offerings: Identify top-performing products and expand the range of similar items. Address underperforming products by improving or discontinuing them.
- Improving Customer Experience: Use insights from sales data to enhance the customer journey, from personalized recommendations to streamlined checkout processes.
- Adjusting Pricing Strategies: Analyze price elasticity to find the optimal pricing strategy that maximizes revenue without deterring customers.
- Planning Inventory Management: Use demand forecasts to optimize inventory levels, ensuring availability of popular products while minimizing holding costs.

Conclusion

E-commerce sales analysis is a powerful tool that enables businesses to leverage data for strategic decision-making. By understanding customer behavior, optimizing operations, and predicting future trends, businesses can enhance their competitive edge and drive growth in the ever-evolving digital marketplace. Through careful data collection, thorough analysis, and actionable insights, e-commerce businesses can unlock their full potential and achieve sustained success.

What is Sales Analysis

Sales analysis is mining your data to evaluate the performance of your sales team against its goals. It provides insights about the top performing and underperforming products/services, the problems in selling and market opportunities, sales forecasting, and sales activities that generate revenue.

Regular sales data analysis provides an understanding of the products that your customers are buying and helps you dissect why they are behaving in a certain way. You can also find patterns in your lead conversions and drop offs. All of these aspects enable you to optimize your sales process.

Importance of Sales Analysis

- Make data-driven decisions instead of relying on gut instinct
- Find most profitable customers
- Get awareness on the market trends
- Expand your market reach

Here, I'll conduct a sales analysis on the E-Commerce dataset.

Data Collection: -

In [1]:

```
import numpy as np
import pandas as pd
import calendar

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

In [2]:

```
order = pd.read_csv("../input/ecommerce-data/List of Orders.csv")
order.head()
```

Out[2]:

	Order ID	Order Date	CustomerName	State	City
0	B-25601	01-04-2018	Bharat	Gujarat	Ahmedabad
1	B-25602	01-04-2018	Pearl	Maharashtra	Pune
2	B-25603	03-04-2018	Jahan	Madhya Pradesh	Bhopal
3	B-25604	03-04-2018	Divsha	Rajasthan	Jaipur
4	B-25605	05-04-2018	Kasheen	West Bengal	Kolkata

In [3]:
```
details = pd.read_csv("../input/ecommerce-data/Order Details.csv")
details.head()
```

Out[3]:

	Order ID	Amount	Profit	Quantity	Category	Sub-Category
0	B-25601	1275.0	-1148.0	7	Furniture	Bookcases
1	B-25601	66.0	-12.0	5	Clothing	Stole
2	B-25601	8.0	-2.0	3	Clothing	Hankerchief
3	B-25601	80.0	-56.0	4	Electronics	Electronic Games
4	B-25602	168.0	-111.0	2	Electronics	Phones

In [4]:
```
target = pd.read_csv("../input/ecommerce-data/Sales target.csv")
target.head()
```

Out[4]:

	Month of Order Date	Category	Target
0	Apr-18	Furniture	10400.0
1	May-18	Furniture	10500.0
2	Jun-18	Furniture	10600.0
3	Jul-18	Furniture	10800.0
4	Aug-18	Furniture	10900.0

Data Pre-processing

Data Cleaning

In this dataset, the data cleaning process will consists of:

1. Changing the variables to appropriate Data types
2. Removing Null Values

In [5]:
```
# Cleaning the order dataset
order.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 560 entries, 0 to 559
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Order ID      500 non-null    object
 1   Order Date    500 non-null    object
 2   CustomerName  500 non-null    object
 3   State         500 non-null    object
 4   City          500 non-null    object
dtypes: object(5)
memory usage: 22.0+ KB
```

In [6]:
```
# Changing the Order Date variable to datetime data type
order['Order Date'] = order['Order Date'].astype('datetime64[ns]')
```

In [7]:
```
# Checking null values
order.isnull().sum()
```

Out[7]:
```
Order ID        60
Order Date      60
CustomerName    60
State           60
City            60
dtype: int64
```

In [8]:
```
# Droping Null Values
order = order.dropna()
order.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 500 entries, 0 to 499
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   Order ID      500 non-null    object        
 1   Order Date    500 non-null    datetime64[ns]
 2   CustomerName  500 non-null    object        
 3   State         500 non-null    object        
 4   City          500 non-null    object        
dtypes: datetime64[ns](1), object(4)
memory usage: 23.4+ KB
```

In [9]:

```
# Cleaning the detail dataset
details.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Order ID      1500 non-null   object 
 1   Amount        1500 non-null   float64
 2   Profit        1500 non-null   float64
 3   Quantity      1500 non-null   int64  
 4   Category      1500 non-null   object 
 5   Sub-Category  1500 non-null   object 
dtypes: float64(2), int64(1), object(3)
memory usage: 70.4+ KB
```

In [14]:
```
# Cleanded Details data
details.head()
```

Out[14]:

	Order ID	Amount	Profit	Quantity	Category	Sub-Category
0	B-25601	1275.0	-1148.0	7	Furniture	Bookcases
1	B-25601	66.0	-12.0	5	Clothing	Stole
2	B-25601	8.0	-2.0	3	Clothing	Hankerchief
3	B-25601	80.0	-56.0	4	Electronics	Electronic Games
4	B-25602	168.0	-111.0	2	Electronics	Phones

In [15]:
```
# Cleaned Order Data
order.head()
```

Out[15]:

	Order ID	Order Date	CustomerName	State	City
0	B-25601	2018-01-04	Bharat	Gujarat	Ahmedabad
1	B-25602	2018-01-04	Pearl	Maharashtra	Pune
2	B-25603	2018-03-04	Jahan	Madhya Pradesh	Bhopal
3	B-25604	2018-03-04	Divsha	Rajasthan	Jaipur
4	B-25605	2018-05-04	Kasheen	West Bengal	Kolkata

In [16]:

```
# Cleaned Target Dataset
target.head()
```

Out[16]:

	Month of Order Date	Category	Target
0	Apr-18	Furniture	10400.0
1	May-18	Furniture	10500.0
2	Jun-18	Furniture	10600.0
3	Jul-18	Furniture	10800.0
4	Aug-18	Furniture	10900.0

Making a new dataframe containing the Amount, Profit and Quantity of the different orders. Then joining it with the Order datasets by taking Order ID as the Primary Key.

In [17]:

```
profits = details.groupby('Order ID').sum().reset_index()
profits.head()
```

Out[17]:

	Order ID	Amount	Profit	Quantity
0	B-25601	1429.0	-1218.0	19
1	B-25602	3889.0	975.0	22
2	B-25603	2025.0	-180.0	25
3	B-25604	222.0	22.0	11
4	B-25605	75.0	0.0	7

Sales Trend Analysis

Trend analysis is to find patterns in data, such as ups & downs. A “trend” is an upwards or downwards shift in a data set over time. In retail, this analysis of past trends in sales or revenue; allows to predict the future market. This analysis useful for budgeting and forecasting. Total sales of any business on a trend line may obtain some significant information.

In [19]:

```
df['Year'] = pd.DatetimeIndex(df['Order Date']).year
df['Month_Number'] = pd.DatetimeIndex(df['Order Date']).month
df['Month'] = df['Month_Number'].apply(lambda x: calendar.month_abbr[x])

year_month = df.groupby(['Year', 'Month','Month_Number']).sum().sort_values(['Year','Month_Num
ber'])
year_month
```

Out[19]:

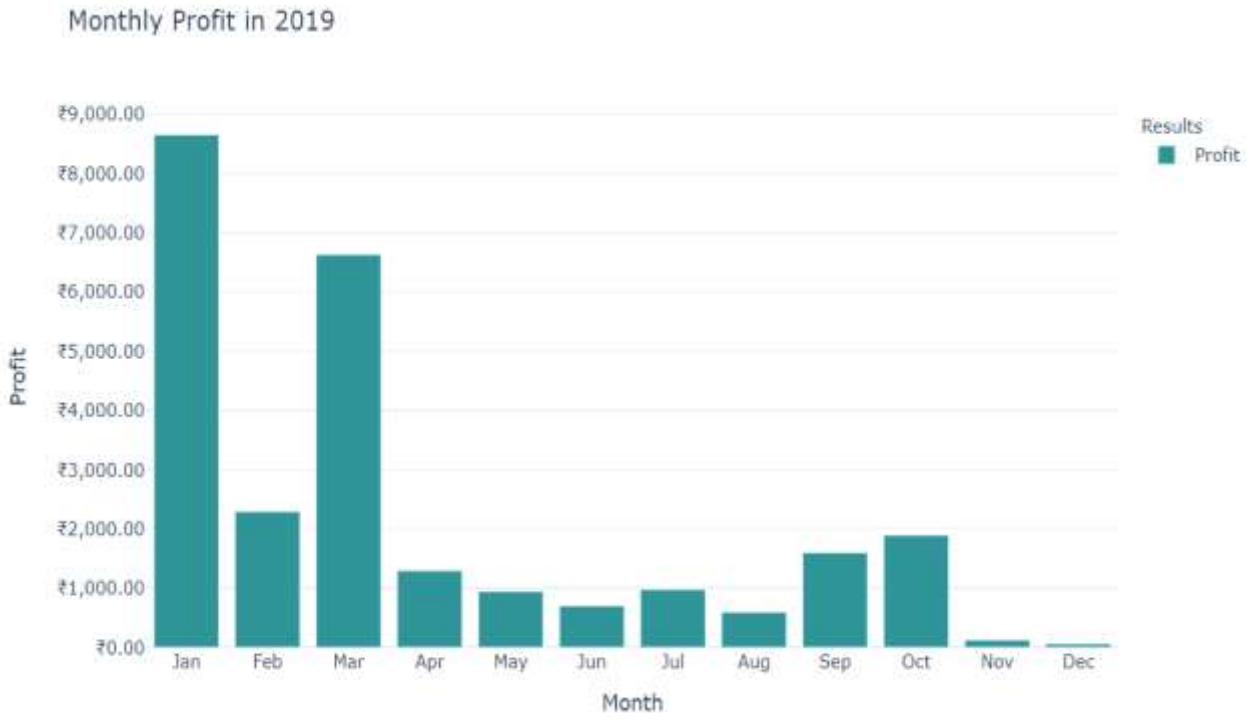
			Amount	Profit	Quantity
Year	Month	Month_Number			
2018	Jan	1	18035.0	-3296.0	203
	Feb	2	6566.0	685.0	58
	Mar	3	7434.0	669.0	144
	Apr	4	26170.0	-1043.0	337
	May	5	20422.0	-891.0	306
	Jun	6	17406.0	-3759.0	353
	Jul	7	15682.0	-2065.0	239
	Aug	8	45269.0	-1059.0	601
	Sep	9	20210.0	-3509.0	310
	Oct	10	32758.0	5979.0	414
	Nov	11	38858.0	4955.0	433
	Dec	12	23892.0	1535.0	209
2019	Jan	1	50448.0	8655.0	640
	Feb	2	15894.0	2291.0	253
	Mar	3	39700.0	6633.0	485
	Apr	4	11079.0	1295.0	106

[20]:

```
year_month = year_month.reset_index()
year_month["Color"] = np.where(year_month["Profit"]<0, 'Loss', 'Profit')
year_month_2018 = year_month[year_month['Year']==2018]
fig = px.bar(year_month_2018, x='Month_Number', y='Profit', color='Color',
             title="Monthly Profit in 2018",
             labels=dict(Month_Number="Month", Profit="Profit", Color="Results"),
             color_discrete_map={
                 'Loss': '#EC2049',
                 'Profit': '#2F9599'},
             hover_data=["Month", "Profit"],
             template='plotly_white')

fig.update_layout(yaxis_tickprefix = '₹', yaxis_tickformat = ',.2f')

fig.update_layout(
    xaxis = dict(
        tickvals = [1, 2, 3, 4, 5, 6, 7,8 ,9, 10, 11, 12],
        ticktext = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
    )
)
fig.show()
```

[21]:

```
year_month_2019 = year_month[year_month['Year']==2019]
fig = px.bar(year_month_2019, x='Month_Number', y='Profit', color='Color',
             title="Monthly Profit in 2019",
             labels=dict(Month_Number="Month", Profit="Profit", Color="Results"),
             color_discrete_map={
                 'Loss': '#EC2049',
                 'Profit': '#2F9599'},
             hover_data=["Month", "Profit"],
             template='plotly_white')

fig.update_layout(yaxis_tickprefix = '₹', yaxis_tickformat = ',.2f')

fig.update_layout(
    xaxis = dict(
        tickvals = [1, 2, 3, 4, 5, 6, 7,8 ,9, 10, 11, 12],
        ticktext = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
    )
)
fig.show()
```

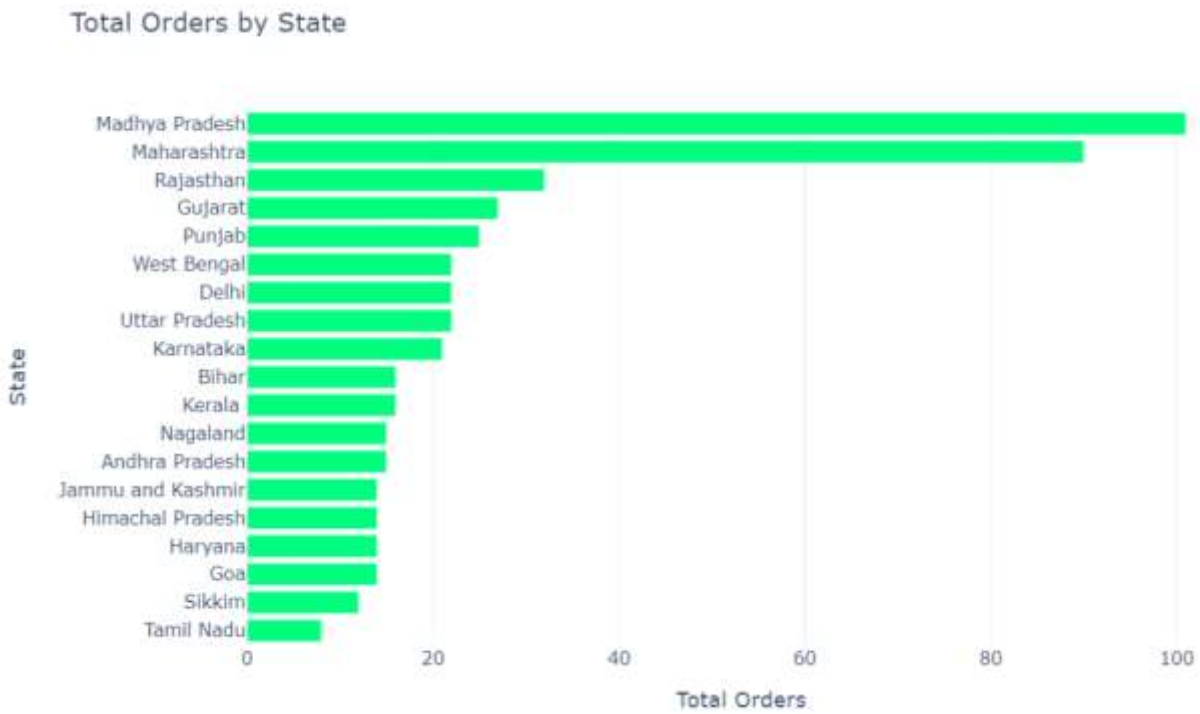


Customer Demographic Analysis

Customer demographics are categories of consumer populations that are relevant to a business' purposes, such as marketing and product design. The term also refers to the study of such categories in a business context.

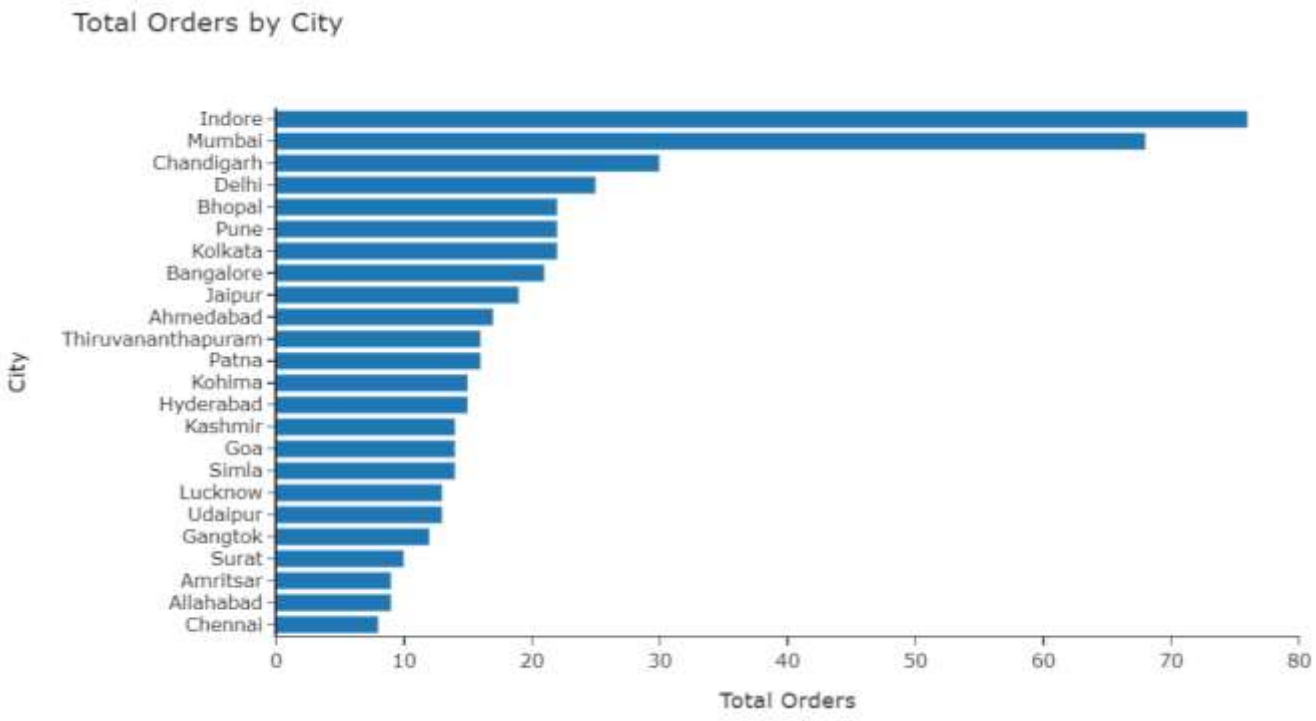
[28]:

```
fig = px.bar(orders_by_state, y='State', x='Total Orders',
             title="Total Orders by State",
             color_discrete_sequence=["springgreen"],
             template='plotly_white')

# Disabling Zoom
fig.layout.xaxis.fixedrange = True
fig.layout.yaxis.fixedrange = True

fig.show()
```



[29]:

```
#
orders_by_city = order.groupby(['City']).size().reset_index(name='Total Orders').sort_values(['Total Order

fig = px.bar(orders_by_city, y='City', x='Total Orders',
             title='Total Orders by City',
             template='simple_white')

fig.layout.yaxis.tickmode='linear'
# Disabling Zoom
fig.layout.xaxis.fixedrange = True
fig.layout.yaxis.fixedrange = True

fig.show()
```

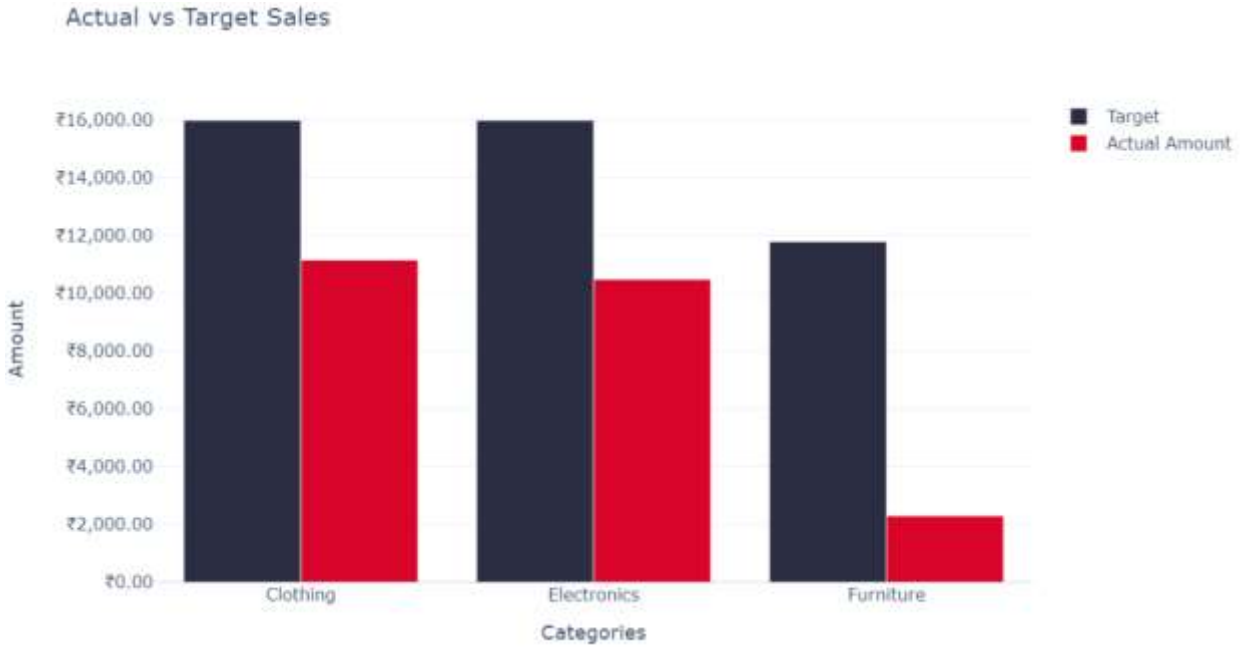


The state with the highest quantity sold Madhya Pradesh, followed by Maharastra and Rajasthan. There is a biggest gap between the quantity sold in Maharastra an Rajasthan with a difference of 58 units. While in case of Cities, it is Indore and Mumbai by a very wide margin. Chennai, Allahabad and Amritsar have the lowest quantity sold with less than 10 units sold.

Sales Target

A sales target is a goal set for a salesperson or sales department measured in revenue or units sold for a specific time.

[30]:
```
target_category = target.groupby('Category').max().reset_index()
details_category = details.groupby('Category').sum().reset_index()

target_category['Actual_Amount'] = details_category['Profit']

fig = go.Figure(data=[
    go.Bar(name='Target', x=target_category['Category'], y=target_category['Target'],
           marker_color='#2b2d42'),
    go.Bar(name='Actual Amount', x=target_category['Category'], y=target_category['Actual_Amount'],
           marker_color='#d90429')
])

fig.update_layout(title_text='Actual vs Target Sales',
                  template='plotly_white')

fig.update_xaxes(title_text='Categories')
fig.update_yaxes(title_text='Amount')

fig.update_layout(yaxis_tickprefix = '₹', yaxis_tickformat = ',.2f')


fig.layout.xaxis.fixedrange = True
fig.layout.yaxis.fixedrange = True

fig.show()
```



Customer Segmentation via Cluster Analysis

Cluster analysis uses mathematical models to discover groups of similar customers based on the smallest variations among customers within each group.

Cluster Analysis

Cluster analysis is the use of a mathematical model to discover groups of similar customers based on finding the smallest variations among customers within each group. The goal of cluster analysis in marketing is to accurately segment customers in order to achieve more effective customer marketing via personalization. A common cluster analysis method is a mathematical algorithm known as k-means cluster analysis, sometimes referred to as scientific segmentation. The clusters that result assist in better customer modeling and predictive analytics, and are also are used to target customers with offers and incentives personalized to their wants, needs and preferences.

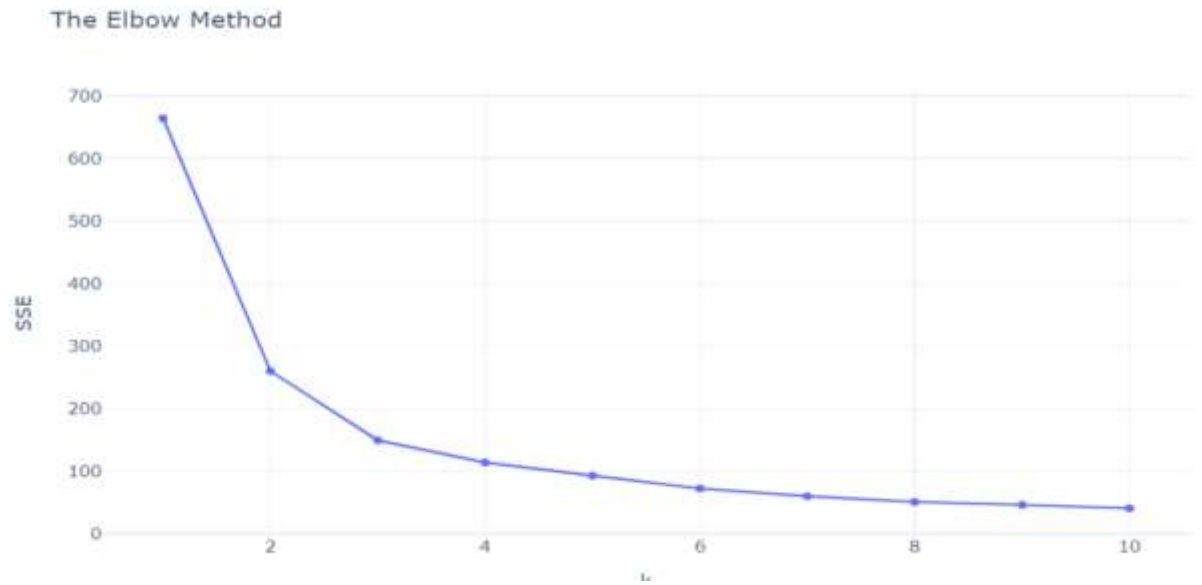
[31]:
```
customer_seg = df.groupby('CustomerName').sum().reset_index()
customer_seg = customer_seg[['CustomerName', 'Amount', 'Quantity']]
customer_seg.head()
```

[31]:

	CustomerName	Amount	Quantity
0	Aakanksha	74.0	8
1	Aarushi	4701.0	49
2	Aashna	1931.0	32
3	Aastha	3276.0	28
4	Aayush	556.0	18

[32]:

```
# Standardizing
customer_seg2 = customer_seg[['Amount', 'Quantity']]
scaler = StandardScaler()
scaler.fit(customer_seg2)

customers_normalized = scaler.transform(customer_seg2)
customers_normalized

# Elbow Method to find best number of clusters
sse = {}
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(customers_normalized)
    sse[k] = kmeans.inertia_ # SSE to closest cluster centroid

# Plotting SSE
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=list(sse.keys()),
    y=list(sse.values()),
    connectgaps=True # override default to connect the gaps
))
```



[33]:

```
# KMeans
model = KMeans(n_clusters=3)
model.fit(customers_normalized)
customer_seg['Cluster'] = model.labels_ + 1
customer_seg['Cluster'] = customer_seg['Cluster'].astype('category')
customer_seg.head()
```

[33]:

	CustomerName	Amount	Quantity	Cluster
0	Aakanksha	74.0	8	1
1	Aarushi	4701.0	49	2
2	Aashna	1931.0	32	3
3	Aastha	3276.0	28	3
4	Aayush	556.0	18	1

[35]:

```
fig = px.scatter(customer_seg, x="Quantity", y="Amount",
                 color="Cluster",
                 template='plotly_white',
                 title="Amount vs Quantity - Customer Segmentation")
fig.layout.xaxis.fixedrange = True
fig.layout.yaxis.fixedrange = True

fig.show()
```