Super-mart Grocery Sales - Retail Analytics Dataset

BY

KRITHEESHWAR.S

S.no	CONTENTS	Pg.No
1	**Introduction**	3
2	**Objectives**	3
3	**Data-set Description**	4
4	**Methodology**	5
5	**Problem statement**	6
6	**Analysis**	6
7	**Future Work**	24
8	**Conclusion**	24

Introduction

The goal of this project is to analyze the Super mart Grocery Sales data-set to gain insights into sales performance, customer behavior, and inventory management in Tamil Nadu. By leveraging data analytics techniques, the project aims to provide actionable recommendations to enhance business strategies and improve overall operational efficiency.The data collected from and among surveyed sample among people whom bought groceries.

Objectives

1.Sales Performance Analysis:

- Evaluate overall sales performance across different time periods (daily, weekly, monthly, yearly).
- Identify top-performing products and categories.
- Analyze sales trends and seasonal patterns.
- Determine the contribution of different regions and stores to overall sales.

2.Customer Behavior Analysis

- Segment customers based on purchasing behavior.
- Analyze customer demographics and their impact on sales.
- Identify customer preferences and buying patterns.
- Predict customer churn and develop retention strategies.

3.Promotion and Discount Analysis

- Evaluate the effectiveness of promotional campaigns and discounts.
- Determine the impact of promotions on sales and profit margins.
- Identify the best times and products for promotions.

4.Operational Efficiency:

- Analyze checkout and billing processes to reduce wait times.
- Optimize supply chain and logistics operations.
- Improve shelf space allocation based on product performance.
- Various test analysis

Data-set Description

The Super-mart Grocery Sales data-set contains detailed sales records from various stores over a specified period. The data-set includes the following key attributes:

- **Date:** The date of the transaction.
- **Region:** Geographical location of the store.
- **Order ID:** Unique identifier for each product.
- **Product Category:** Category to which the product belongs.
- **Quantity Sold:** Number of units sold.
- **Sales Amount:** Total sales amount for the transaction.
- **Customer Name:** Name of each customer.
- **Discount**: Offers given to each product
- **Profit**: Profit gained for each unit

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	Profit	State
0	OD1	Harish	Oil & Masala	Masalas	Vellore	11/8/2017	North	1254	0.12	401.28	Tamil Nadu
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11/8/2017	South	749	0.18	149.80	Tamil Nadu
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	6/12/2017	West	2360	0.21	165.20	Tamil Nadu
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10/11/2016	South	896	0.25	89.60	Tamil Nadu
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10/11/2016	South	2355	0.26	918.45	Tamil Nadu

Methodology

1.Data Collection and Processioning:

- Import the data-set and clean the data by handling missing values, duplicates, and outliers.
- Perform data transformation and normalization as needed.

2.Exploratory Data Analysis (EDA):

- Conduct EDA to understand the data distribution and relationships between variables.
- Visualize sales trends, customer segments, and inventory metrics using charts and graphs.

3.Statistical Analysis and Modeling:

- Use statistical methods to identify significant factors affecting sales performance.
- Develop predictive models for sales forecasting, customer segmentation, and inventory optimization.

4.Data Visualization:

- Create interactive dashboards to present key findings and insights.
- Use visualization tools like Tableau, Power BI, or Python libraries (Matplotlib, Seaborn) for effective data presentation.

Problem statement

The business problem statement sets the stage for a comprehensive analysis and provides a clear direction for the project.Optimizing Sales Performance and Customer Experience at Super-mart Grocery

Super-mart Grocery, with multiple stores across various regions, faces several challenges in maintaining competitive sales performance and delivering an exceptional customer experience. Despite having a diverse product portfolio and a broad customer base, the company struggles with understanding the intricate patterns of sales, customer preferences, and inventory dynamics.

Analysis

- First import libraries and read file from the stored data location i.e import numpy import pandas as pd,df = pd.read_csv('Grocery Sales.csv')
- df.info() - To check the values data types and the counts(data frame)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Order ID       9994 non-null   object
 1   Customer Name  9994 non-null   object
 2   Category       9994 non-null   object
 3   Sub Category   9994 non-null   object
 4   City           9994 non-null   object
 5   Order Date     9994 non-null   object
 6   Region         9994 non-null   object
 7   Sales          9994 non-null   int64
 8   Discount       9994 non-null   float64
 9   Profit         9994 non-null   float64
 10  State          9994 non-null   object
dtypes: float64(2), int64(1), object(8)
memory usage: 859.0+ KB
```

- df.isna().sum() - this function used to check the null values in the data frame

```
Order ID        0
Customer Name   0
Category        0
```

```
Sub Category    0
City            0
Order Date      0
Region          0
Sales           0
Discount        0
Profit          0
State           0
dtype: int64
```

- df['Category'].value_counts(normalize = False) - normalize=False: This argument specifies that the method should return the raw counts of unique values. If normalize - True were used, it would return the relative frequencies instead of raw counts.

```
Category
Snacks             1514
Eggs, Meat & Fish  1490
Fruits & Veggies   1418
Bakery             1413
Beverages          1400
Food Grains        1398
Oil & Masala       1361
Name: count, dtype: int64
```

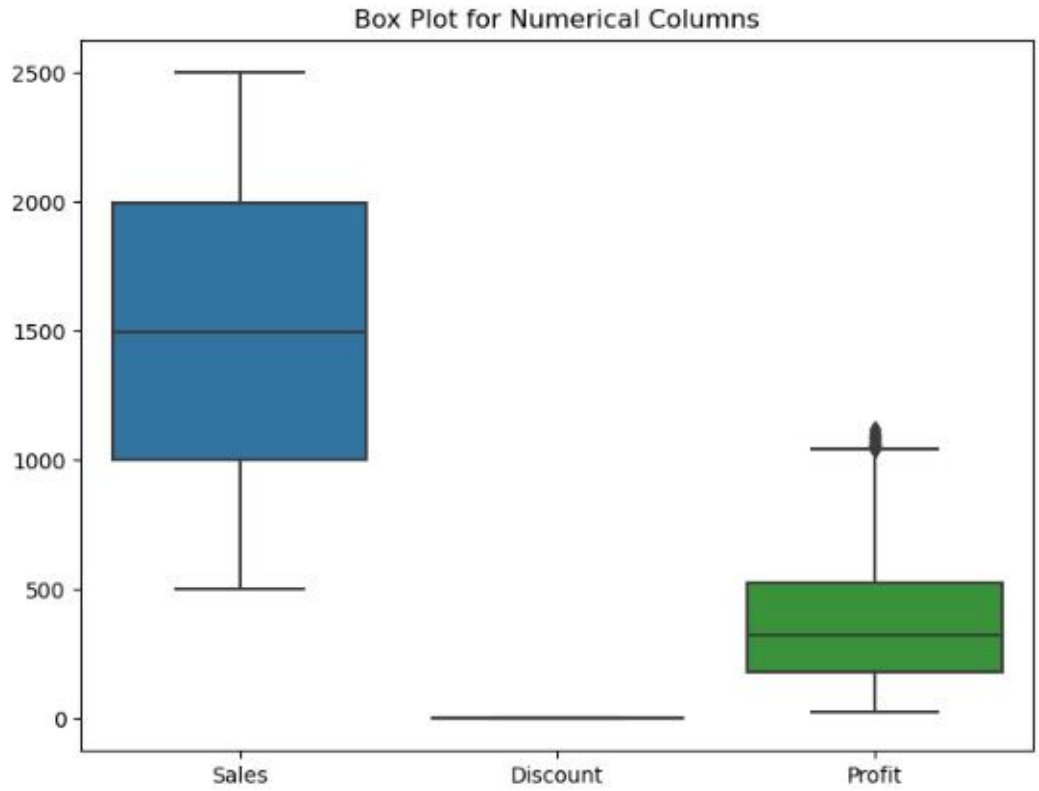
- print(df[['Category','Sub Category','City']].value_counts().to_markdown()) - datas separated into different category

('Beverages', 'Soft Drinks', 'Perambalur')	44
('Beverages', 'Health Drinks', 'Ramanadhapuram')	40
('Beverages', 'Health Drinks', 'Chennai')	39
('Beverages', 'Soft Drinks', 'Tirunelveli')	38
('Beverages', 'Health Drinks', 'Ooty')	38
('Beverages', 'Soft Drinks', 'Madurai')	38
('Beverages', 'Health Drinks', 'Viluppuram')	37
('Beverages', 'Health Drinks', 'Tenkasi')	36
('Beverages', 'Health Drinks', 'Vellore')	35
('Beverages', 'Health Drinks', 'Tirunelveli')	33
('Beverages', 'Health Drinks', 'Salem')	33
('Beverages', 'Soft Drinks', 'Theni')	33

- df.describe() - to find the minimum and maximum values of the datas

	Sales	Discount	Profit
count	9994.000000	9994.000000	9994.000000
mean	1496.596158	0.226817	374.937082
std	577.559036	0.074636	239.932881
min	500.000000	0.100000	25.250000
25%	1000.000000	0.160000	180.022500
50%	1498.000000	0.230000	320.780000
75%	1994.750000	0.290000	525.627500
max	2500.000000	0.350000	1120.950000

- plt.figure(figsize=(8, 6))
- sns.boxplot(data=numrical_columns)
- plt.title('Box Plot for Numerical Columns')



Handling Outlier

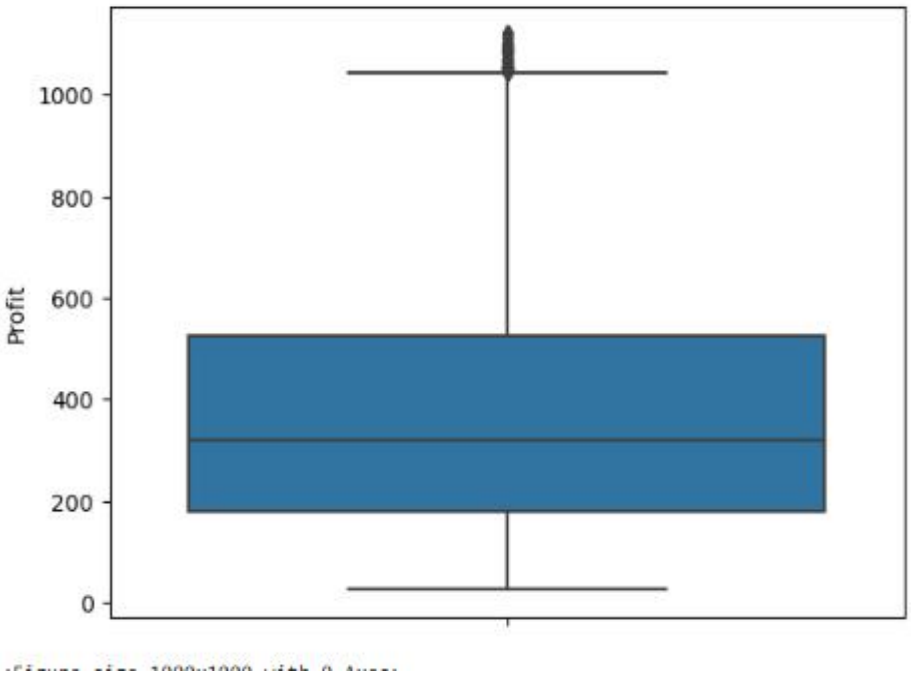
- An observation far away from other data points
- Profit Column has outliers so we need to handle it.

- visualize Profit columnsns.

boxplot(data=df,y='Profit')

plt.show()

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



- 75th percentile

seventy_fifth=df['Profit'].quantile(0.75)

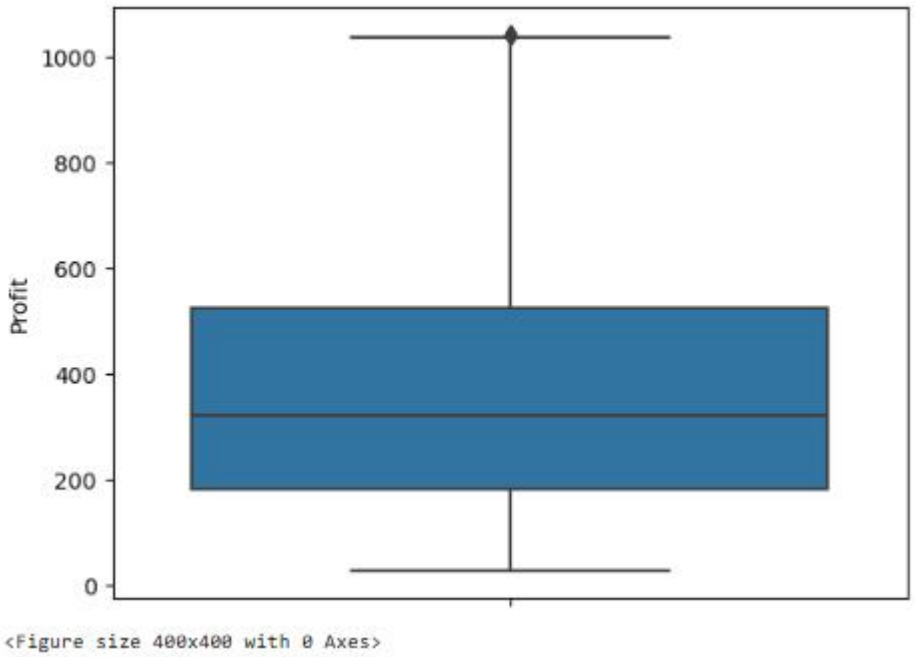
25th percentile

twenty_fifth=df['Profit'].quantile(0.25)

Inter quartile range

```
profit_IQR=seventy_fifth-twenty_fifth

print(profit_IQR)
```

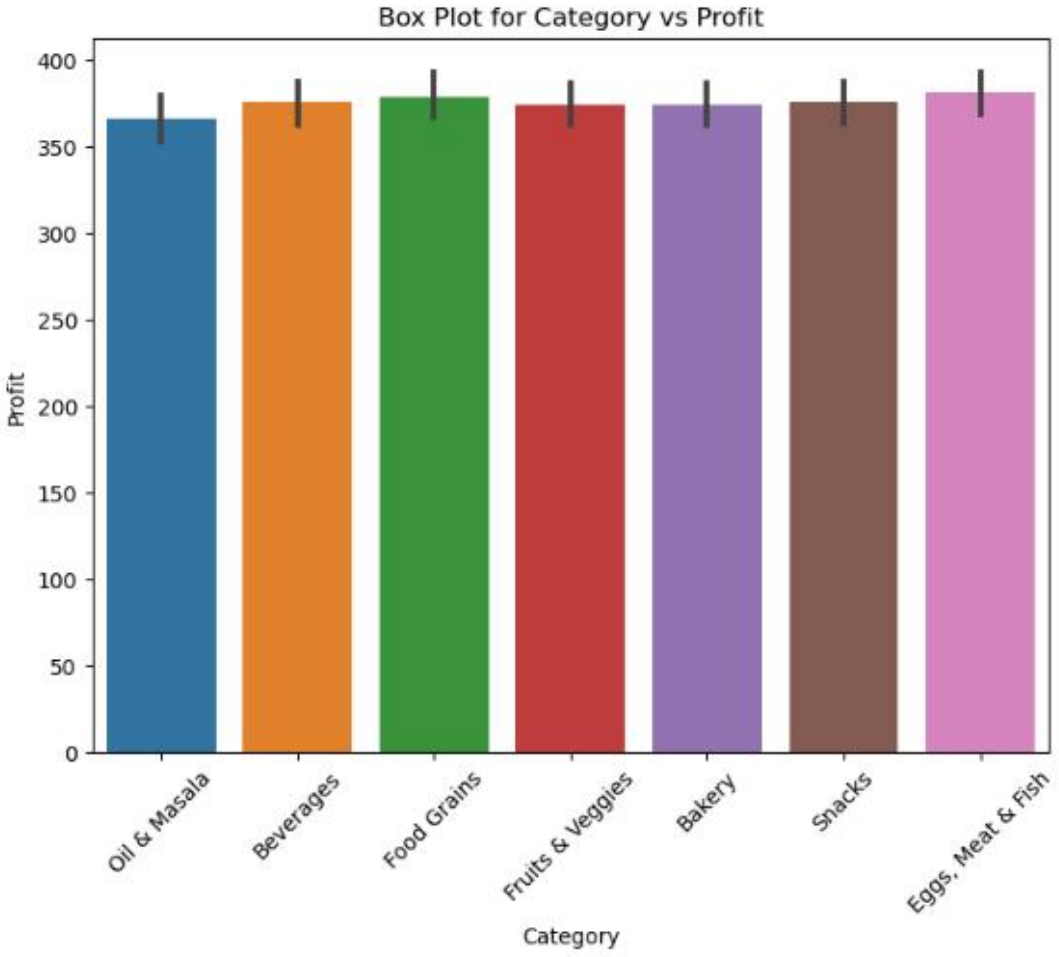
- ```
  # boxplot for profit after removing outliers

  sns.boxplot(data=df1,y='Profit')

  plt.figure(figsize=(4,4))

  plt.show()
  ```




```
<Figure size 400x400 with 0 Axes>
```

- ```
  plt.figure(figsize=(8, 6))

      #sns.boxplot(data=df,x='Category',y='Profit', orient='v', fliersize=3, width=0.5)

      sns.barplot(data=df,x='Category',y='Profit')

      plt.title('Box Plot for Category vs Profit')

      plt.xticks(rotation=45)
  ```

plt.show()



Inference

- This bar plot represents profit and category
- categories such as oil & masala,beverages,food grains,fruits & veggies ,bakery ,snacks,eggs,meat&fish
- In plot the most bought category was eggs, meat,&fish and above the food grains.

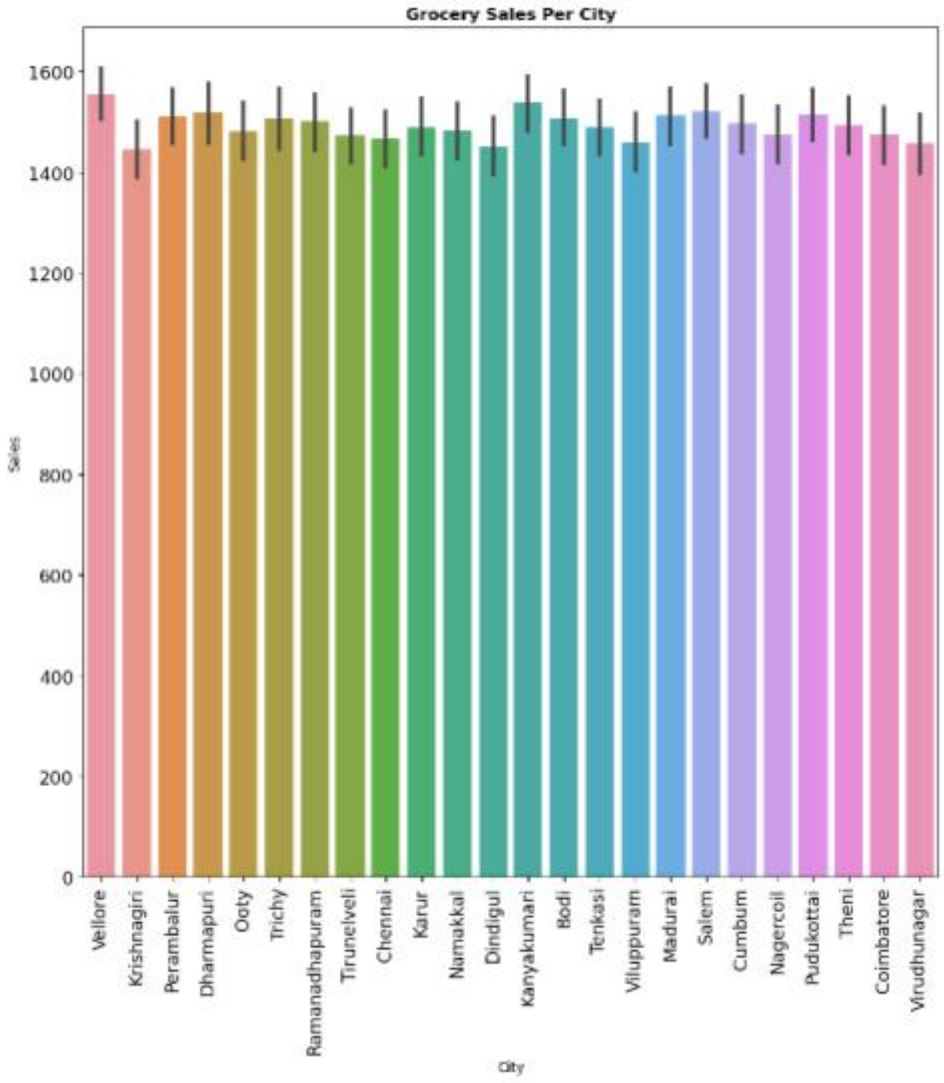
- plt.figure(figsize=(11,11))

  sns.barplot(x=df1['City'],y=df1['Sales'])

  plt.xticks(rotation=90,fontsize=13)

  plt.yticks(fontsize=13)

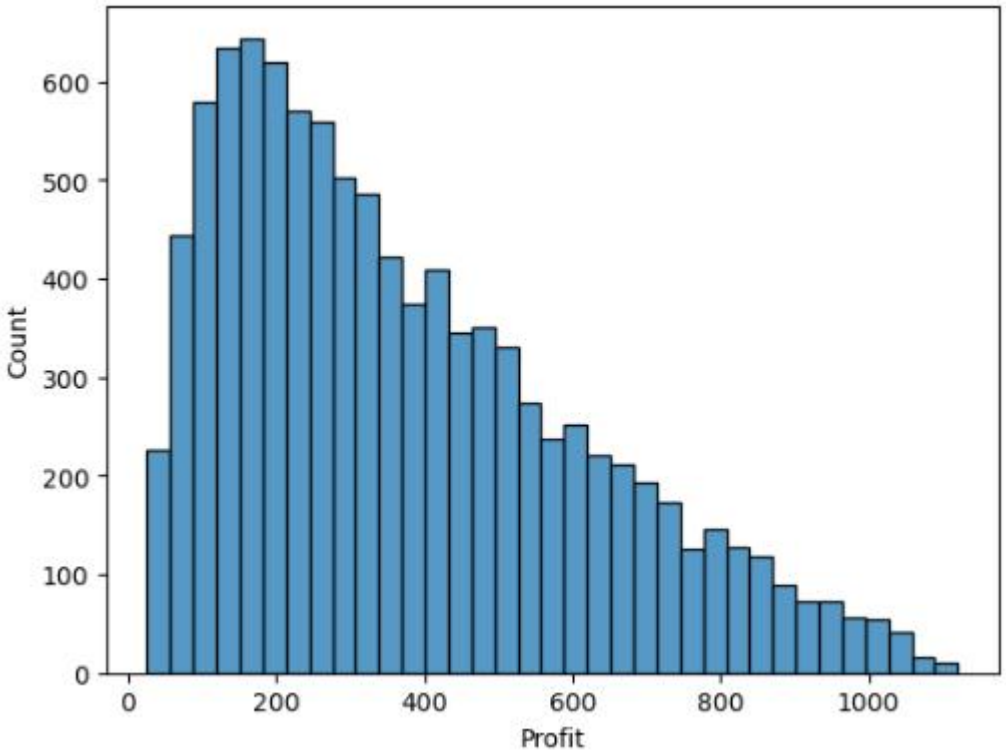
```
plt.title('Grocery Sales Per City',fontweight='bold')

plt.show()
```



Inference¶

- In this type of bar plot based on different cities and their sales
- plot shows that every city has a their sales margin above 1400

- sns.histplot(df['Profit'])



Inference

- The height of the bars indicates how frequently values within each bin occurs,showing the distribution's
- In this sudden rise of profit when it reaches to the count of 600 and above
- gradual decrese of profit after it crosses 1000

- calculate average sales for each sub_category per region

```
Sub_Sales_Region=pd.pivot_table(df1,columns='Region',index=['Sub Category'],values='Sales',aggfunc=np.mean,  fill_value=0)
Sub_Sales_Region.loc[Sub_Sales_Region.mean(axis=1).sort_values(ascending=False).index]
```

Region	Central	East	North	South	West
Sub Category					
Masalas	1599.883117	1440.891026	1254.0	1494.366197	1526.611465
Mutton	1618.259740	1507.632479	0.0	1508.323944	1574.496124
Eggs	1507.743902	1550.215517	0.0	1605.927273	1447.392000
Dals & Pulses	1511.646341	1494.977273	0.0	1565.322034	1532.575221
Fish	1508.535714	1559.392857	0.0	1578.351852	1445.743590
Soft Drinks	1461.867816	1499.095238	0.0	1554.272727	1547.029412
Atta & Flour	1441.633803	1579.322917	0.0	1567.483333	1466.838710
Cakes	1452.688073	1461.723214	0.0	1568.205128	1564.940397
Spices	1475.838384	1523.194030	0.0	1547.339623	1483.132075
Organic Staples	1486.535714	1534.553719	0.0	1506.566038	1478.342105
Rice	1528.986111	1477.431818	0.0	1432.084746	1565.126126
Fresh Fruits	1523.277778	1537.928571	0.0	1512.280702	1412.418033
Chicken	1600.436782	1403.058252	0.0	1498.134615	1480.673077

- Top 5 sub_category sales per region

```
df2=Sub_Sales_Region.nlargest(5,'Central')
df2.style.bar(color='#FBEEAC')
```

Region	Central	East	North	South	West
Sub Category					
Mutton	1618.259740	1507.632479	0.000000	1508.323944	1574.496124
Chicken	1600.436782	1403.058252	0.000000	1498.134615	1480.673077
Masalas	1599.883117	1440.891026	1254.000000	1494.366197	1526.611465
Chocolates	1541.488000	1430.680851	0.000000	1451.412500	1447.344371
Rice	1528.986111	1477.431818	0.000000	1432.084746	1565.126126

- calculate average category profit per city

```
Cat_Profit=pd.pivot_table(df1,columns=['Category'],index=['City'],values='Profit',aggfunc=np.mean)
Cat_Profit.loc[Cat_Profit.mean(axis=1).sort_values(ascending=False).index]
```

Category City	Bakery	Beverages	Eggs, Meat & Fish	Food Grains	Fruits & Veggies	Oil & Masala	Snacks
Vellore	409.358475	422.389219	410.237727	407.048571	401.418571	347.632667	400.459701
Karur	434.574516	389.250588	368.935658	396.556250	381.916769	345.583846	415.772931
Bodi	371.980645	354.523548	379.301429	464.354848	401.483684	328.867273	428.158596
Perambalur	371.121562	368.325352	404.575714	360.960962	362.911905	443.821667	400.641739
Pudukottai	378.090862	362.444828	366.222264	411.477500	382.137536	368.776765	378.496613
Kanyakumari	359.107500	387.179020	385.450274	374.008793	402.372203	366.725823	349.676351
Dharmapuri	361.895965	334.541591	382.007818	394.165091	364.539057	364.378246	415.188148
Trichy	333.736591	347.825849	386.154576	360.591509	397.198654	407.742927	380.242353
Cumbum	396.266182	364.809298	384.029155	373.760175	359.355490	371.092833	350.564375
Ramanadhapuram	363.754286	385.710492	395.261014	367.553621	361.976863	342.818679	381.762698
Madurai	333.533269	399.268429	411.893636	325.657797	387.844419	392.375172	344.486102
Salem	371.380145	325.816167	358.287059	411.334717	356.803191	364.629403	403.310308
Ooty	338.357091	344.084677	381.468065	409.244697	408.203061	356.472321	348.309434
Chennai	352.054706	391.285000	408.367424	360.239123	315.598103	396.855238	360.422941
Nagercoil	370.915424	356.265208	378.510833	360.934565	386.456029	354.750408	358.297037

- Top 5 category profit sorted according to Bakery Profit's

```
new=Cat_Profit.nlargest(5,'Bakery')
new.style.bar(color='skyblue')
```

Category City	Bakery	Beverages	Eggs, Meat & Fish	Food Grains	Fruits & Veggies	Oil & Masala	Snacks
Karur	434.574516	389.250588	368.935658	396.556250	381.916769	345.583846	415.772931
Vellore	409.358475	422.389219	410.237727	407.048571	401.418571	347.632667	400.459701
Viluppuram	399.205714	351.566875	343.279111	349.088226	346.264800	391.592037	338.341563
Cumbum	396.266182	364.809298	384.029155	373.760175	359.355490	371.092833	350.564375
Namakkal	390.601935	401.524727	418.452069	368.927167	342.561724	302.601277	284.363387

- creating function that calculate max value (sales or profit) for each row to know in which region or city was the max sales or profit for category or sub_category

```
def highlight_max(s):
    is_max = s == s.max()
    return ['background-color: yellow' if v else '' for v in is_max]


# Apply the function to each row of the pivot table
styled_table = Sub_Sales_Region.style.apply(highlight_max, axis=1)
```

styled_table

Region Sub Category	Central	East	North	South	West
Atta & Flour	1441.633803	1579.322917	0.000000	1567.483333	1466.838710
Biscuits	1446.123596	1550.531250	0.000000	1446.952941	1471.461039
Breads & Buns	1434.266055	1468.026846	0.000000	1495.776316	1494.381818
Cakes	1452.688073	1461.723214	0.000000	1568.205128	1564.940397
Chicken	1600.436782	1403.058252	0.000000	1498.134615	1480.673077
Chocolates	1541.488000	1430.680851	0.000000	1451.412500	1447.344371
Cookies	1500.256881	1408.372549	0.000000	1474.500000	1509.615385
Dals & Pulses	1511.646341	1494.977273	0.000000	1565.322034	1532.575221
Edible Oil & Ghee	1475.739837	1518.959677	0.000000	1407.792208	1487.849206
Eggs	1507.743902	1550.215517	0.000000	1605.927273	1447.392000
Fish	1508.535714	1559.392857	0.000000	1578.351852	1445.743590
Fresh Fruits	1523.277778	1537.928571	0.000000	1512.280702	1412.418033
Fresh Vegetables	1470.387755	1542.322917	0.000000	1480.259259	1441.142857
Health Drinks	1448.874251	1494.687179	0.000000	1484.581197	1430.476987

- creating function that calculate min value (sales or profit) for each row to know in which region or city was the min sales or profit for category or sub_category

```
def highlight_min(m):
    is_min = m == m.min()
    return ['background-color: red' if v else '' for v in is_min]

# Apply the function to each row of the pivot table
min_table = Cat_Profit.style.apply(highlight_min, axis=1)

min_table
```

Category City	Bakery	Beverages	Eggs, Meat & Fish	Food Grains	Fruits & Veggies	Oil & Masala	Snacks
Bodi	371.980645	354.523548	379.301429	464.354848	401.483684	328.867273	428.158596
Chennai	352.054706	391.285000	408.367424	360.239123	315.598103	396.855238	360.422941
Coimbatore	334.444769	417.836885	327.853788	366.425000	334.756479	363.964211	394.734828
Cumbum	396.266182	364.809298	384.029155	373.760175	359.355490	371.092833	350.564375
Dharmapuri	361.895965	334.541591	382.007818	394.165091	364.539057	364.378246	415.188148
Dindigul	362.088594	353.530000	361.348864	358.814286	365.616290	391.408333	359.209265
Kanyakumari	359.107500	387.179020	385.450274	374.008793	402.372203	366.725823	349.676351
Karur	434.574516	389.250588	368.935658	396.556250	381.916769	345.583846	415.772931
Krishnagiri	385.338429	359.971132	338.479437	394.252063	351.306552	362.923000	348.655000
Madurai	333.533269	399.268429	411.893636	325.657797	387.844419	392.375172	344.486102
Nagercoil	370.915424	356.265208	378.510833	360.934565	386.456029	354.750408	358.297037
Namakkal	390.601935	401.524727	418.452069	368.927167	342.561724	302.601277	284.363387
Ooty	338.357091	344.084677	381.468065	409.244697	408.203061	356.472321	348.309434
Perambalur	371.121562	368.325352	404.575714	360.960962	362.911905	443.821667	400.641739
Pudukottai	378.090862	362.444828	366.222264	411.477500	382.137536	368.776765	378.496613

- filtering data

df1.loc[(df1['Customer Name']=='Amrish')&(df1['Category']=='Bakery')&(df1['Sales']<1000)].sort_values(by='Sales',ascending=False)

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	Profit	State
2085	OD2086	Amrish	Bakery	Biscuits	Tirunelveli	3/13/2018	East	968	0.12	174.24	Tamil Nadu
9909	OD9910	Amrish	Bakery	Breads & Buns	Viluppuram	7/21/2018	East	920	0.14	220.80	Tamil Nadu
1435	OD1436	Amrish	Bakery	Breads & Buns	Salem	7/31/2017	Central	918	0.35	91.80	Tamil Nadu
4899	OD4900	Amrish	Bakery	Biscuits	Theni	11/15/2015	West	868	0.21	373.24	Tamil Nadu
275	OD276	Amrish	Bakery	Cakes	Pudukottai	9/16/2018	West	744	0.11	66.96	Tamil Nadu
8751	OD8752	Amrish	Bakery	Biscuits	Dindigul	4/6/2015	East	701	0.29	126.18	Tamil Nadu
9076	OD9077	Amrish	Bakery	Breads & Buns	Namakkal	8/11/2017	Central	667	0.22	166.75	Tamil Nadu
5140	OD5141	Amrish	Bakery	Cakes	Ramanadhapuram	10/31/2015	West	659	0.15	230.65	Tamil Nadu
2007	OD2008	Amrish	Bakery	Cakes	Viluppuram	2/8/2016	Central	626	0.10	175.28	Tamil Nadu
4645	OD4646	Amrish	Bakery	Biscuits	Namakkal	5/20/2018	West	610	0.28	207.40	Tamil Nadu

- Who are the customers with the highest and least amount of purchases?

- Group the DataFrame by customer name and aggregate the sales and profit values

super_group=df1.groupby('Customer Name').agg({'Sales':'sum','Profit':'sum'})

- Compute a performance metric based on sales and profit

super_group['Performance']=super_group['Sales']+super_group['Profit']

- Sort the customers by their performance metric in descending order

super_group = super_group.sort_values(by='Performance', ascending=False)

- Print the names of the top 5 customers

```
print("The top 5 customers are:")

print(super_group.index[:5].tolist())
```

- Print the names of the last 3 customers

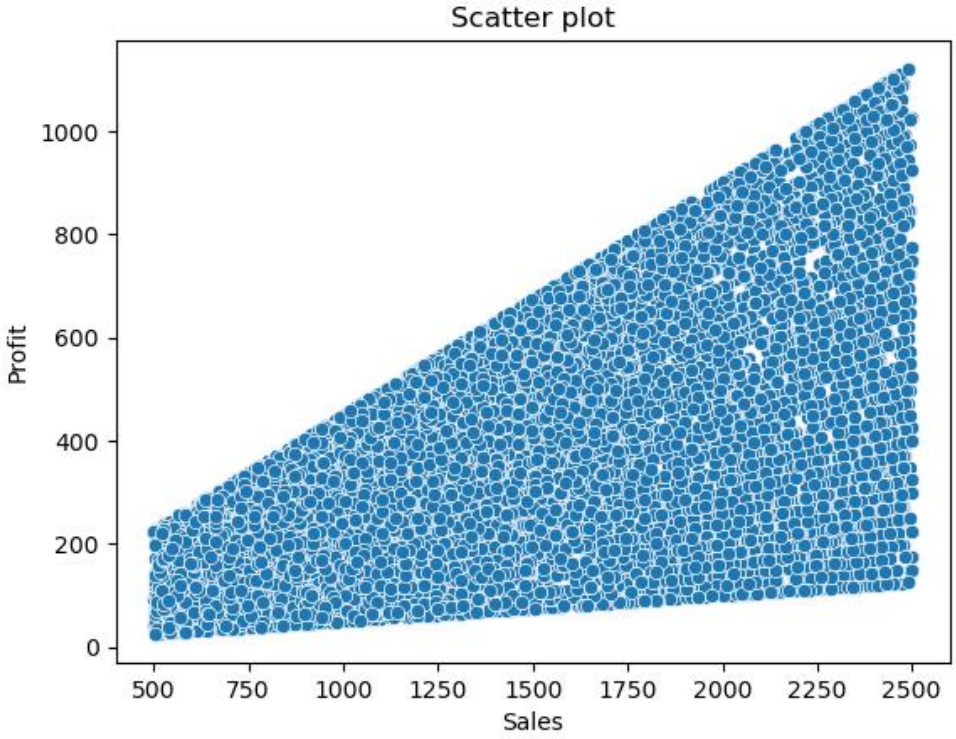
```
print("The last 3 customers are:")

print(super_group.tail(3).index.tolist())
```

Output:The top 5 customers are:
['Krithika', 'Amrish', 'Vidya', 'Verma', 'Arutra']
The last 3 customers are:
['Sudha', 'Kumar', 'Hafiz']

- Scatter plot: import seaborn as sns

```
sns.scatterplot( x='Sales', y= 'Profit', data = df)

plt.title("Scatter plot")
```

plt.show()



Inference

- this plot represents that when the sales are high profit increases

selected_col = ['Sales','Profit']

df_selcted = df[selected_col]

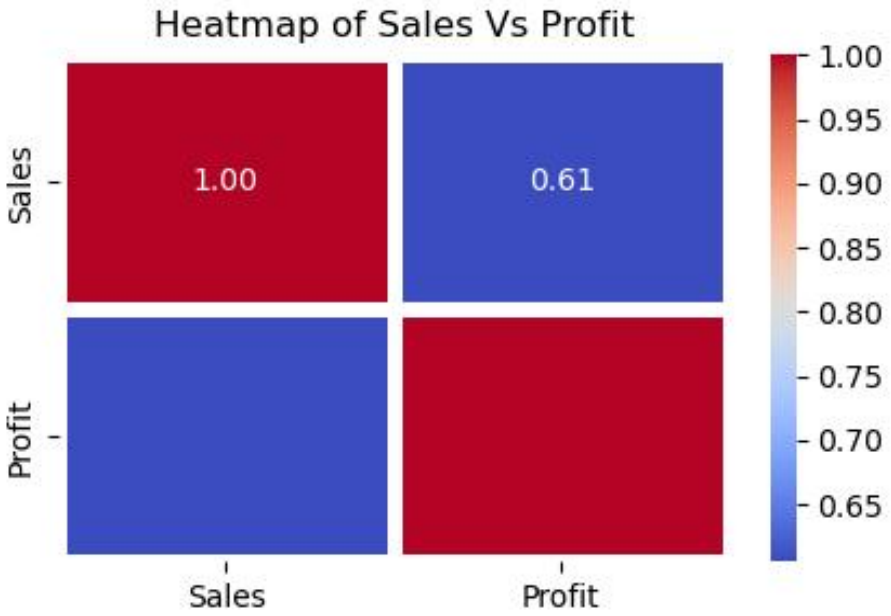
correlation = df_selcted.corr()

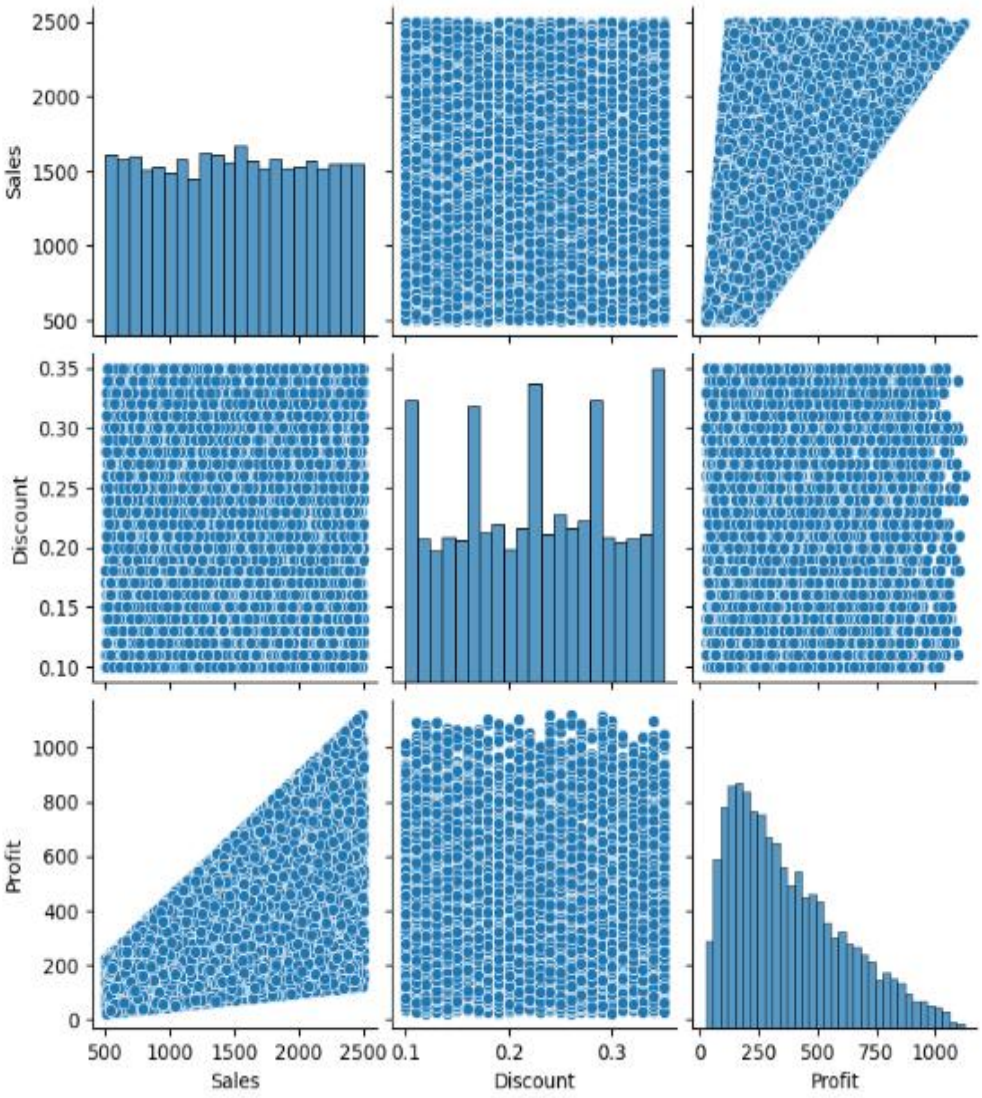
plt.figure(figsize=(5, 3))

sns.heatmap(correlation, annot = True , cmap = 'coolwarm' , linewidth = 5, fmt = '.2f')

plt.title('Heatmap of Sales Vs Profit')

plt.show()


Heatmap of Sales Vs Profit
Sales
Profit
1.00
0.61
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
Sales
Profit


Sales
Discount
Profit
2500
2000
1500
1000
500
0.35
0.30
0.25
0.20
0.15
0.10
1000
800
600
400
200
0
500 1000 1500 2000 2500
Sales
0.1 0.2 0.3
Discount
0 250 500 750 1000
Profit

Inference

- It is a combination of numerical categories
- In this pair plot i used profit,sales and discount as x and y axis
- profit vs sales,profit vs discount,profit vs profit,discount vs sales,discount vs discount,discount vs profit,sales vs sales,sales vs discount,sales vs profit.

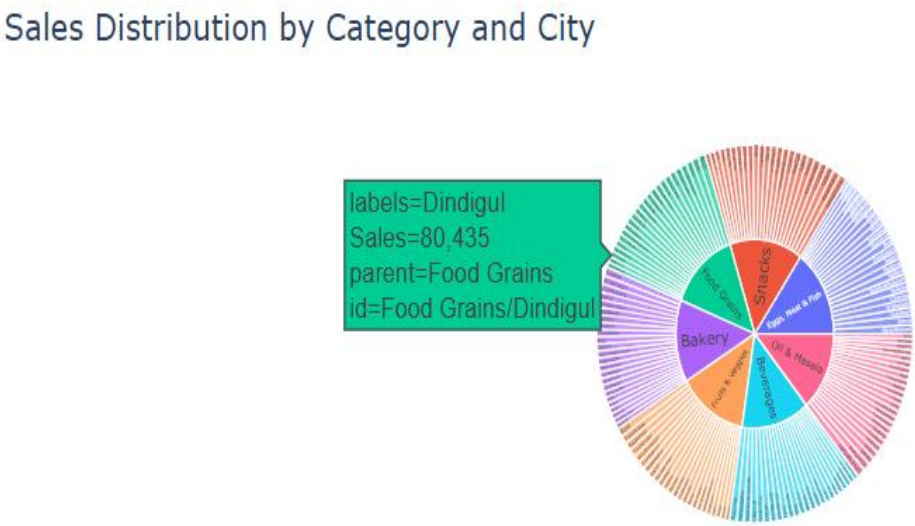
1. if we alternatively take profit and sales the values suddenly increases.

What is Plotly Express?

Plotly Express is a part of the Plotly library, which is widely used for creating interactive plots and visualizations in Python. Plotly Express provides a simple syntax for generating common types of visualizations such as line charts, bar charts, scatter plots, histograms, and more.

The line import plotly.express as px is used to import the Plotly Express module from the Plotly library in Python. Plotly Express is a high-level interface for creating interactive and complex visualizations easily and concisely. By importing it as px, you can access its functions using this shorthand.

```
import plotly.express as px

fig = px.sunburst(df, path=['Category', 'City'], values='Sales', title='Sales Distribution by Category and City')

fig.show()
```



Interpreting the Sunburst Chart

- **Inner Circle (Category)**: Represents the top level of the hierarchy (e.g., 'Fruit', 'Vegetable', 'Dairy').
- **Outer Circle (City)**: Represents the next level of the hierarchy, showing the distribution within each category.
- **Segment Size**: The size of each segment corresponds to the value specified in the values parameter (e.g., 'Sales').

import warnings

warnings.filterwarnings('ignore')

- It is used to handle warnings
- Also filter warnings functions defines what to interpreter must do and in this program we say to ignore the warnings

Future Work

While this project provided substantial insights, there are additional areas for further exploration:

1. **Advanced Predictive Modeling:** Develop more sophisticated predictive models to forecast future sales and customer behavior with greater accuracy.
2. **Real-time Analytics:** Implement real-time data analytics to monitor sales and inventory continuously, allowing for more agile decision-making.
3. **Expanded Data Integration:** Integrate additional data sources, such as social media sentiment and competitor analysis, to gain a more comprehensive understanding of market dynamics and customer preferences.

Conclusion

This retail analytics project has successfully leveraged the Supermart Grocery Sales dataset to deliver actionable insights that can drive strategic decisions and operational improvements. By focusing on sales performance, customer behavior, inventory management, and promotional effectiveness, we have provided a robust foundation for enhancing Supermart Grocery's business performance and achieving sustainable growth in a competitive retail market.This projects have the following outcomes

- Data cleaning
- Remove outliers
- Describe plots
- Handle Hypothesis tests