SUPERMARKET

(python Analysis report)

INTRODUCTION:

In today's competitive retail landscape, understanding customer behavior and sales trends is crucial for supermarkets to thrive. By analyzing data sets containing various transactional details such as invoice ID, date, time, branch, city, customer type, gender, product line, unit price, quantity, tax, total, payment method, cost of goods sold (COGS), gross margin percentage, gross income, and ratings, valuable insights can be gained. This analysis aims to delve into the provided dataset to uncover patterns, preferences, and opportunities for improvement within the supermarket's operations.

WHY ANALYSIS IS CONDUCTED:

Customer Segmentation: By categorizing customers based on demographics and purchase behavior, the supermarket can tailor marketing strategies and product offerings to specific segments, maximizing customer satisfaction and loyalty.

Product Performance Evaluation: Analyzing sales data allows the identification of top-performing and underperforming product lines. Insights gained can inform decisions regarding inventory management, pricing strategies, and product promotions.

Sales Trends and Seasonality: Understanding fluctuations in sales volume over time and across different branches helps in forecasting demand, optimizing inventory levels, and scheduling staffing accordingly.

Profitability Assessment: Examining gross margin percentage, gross income, and cost of goods sold provides insights into the supermarket's profitability. This analysis aids in cost control measures and pricing adjustments to enhance overall profitability.

Customer Satisfaction Analysis: Ratings provided by customers offer valuable feedback on their shopping experience. By analyzing ratings alongside other transactional data, the supermarket can identify areas for improvement in service quality and customer satisfaction.

Payment Method Preference: Studying payment methods preferred by customers can inform decisions related to payment processing systems and partnerships with financial institutions.

City and Branch Comparison: Comparing performance across different cities and branches helps in identifying geographic variations in consumer behavior and market dynamics, guiding strategic expansion plans and resource allocation.

```python
import numpy as np
import pandas as pd
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
import scipy as sp
import datetime
%matplotlib inline

# I am using numpy for numerical computations, pandas for data manipulation and analysis, seaborn for statistical data visualization, matplotlib for creating plots and graphs, scipy for scientific computing, and datetime for handling date and time data. These libraries collectively provide a comprehensive toolkit for exploring and analyzing the sample superstore dataset efficiently.
```

```python
df = pd.read_csv('supermarket')
df

#The code imports the Pandas library and reads a CSV file named 'supermarket' into a DataFrame. This facilitates data manipulation and analysis for the provided supermarket dataset.
```

Invoice ID	Date	Time	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Payment	cogs	gross margin percentage	gross income	Rating	
0	750-67-8428	01-05-2019	13:08	A	Pune	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	Ewallet	522.83	4.761905	26.1415	9.1
1	226-31-3081	03-08-2019	10:29	C	out of pune	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	Cash	76.40	4.761905	3.8200	9.6
2	631-41-3108	03-03-2019	13:23	A	Pune	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	Credit card	324.31	4.761905	16.2155	7.4
3	123-19-1176	1/27/2019	20:33	A	Pune	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	Ewallet	465.76	4.761905	23.2880	8.4
4	373-73-7910	02-08-2019	10:37	A	Pune	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	Ewallet	604.17	4.761905	30.2085	5.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	233-67-5758	1/29/2019	13:46	C	out of pune	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	Ewallet	40.35	4.761905	2.0175	6.2
996	303-96-2227	03-02-2019	17:16	B	pimpri chinchwad	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	Ewallet	973.80	4.761905	48.6900	4.4
997	727-02-1313	02-09-2019	13:22	A	Pune	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	Cash	31.84	4.761905	1.5920	7.7

df.columns

df.columns retrieves the column names of a DataFrame in Python pandas library, commonly used to access and manipulate data in tabular form, such as the Super Store dataset.

```
Index(['Invoice ID', 'Date', 'Time', 'Branch', 'City', 'Customer type',
       'Gender', 'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total',
       'Payment', 'cogs', 'gross margin percentage', 'gross income', 'Rating'],
      dtype='object')
```

`df.describe()`

#it provides summary statistics such as count, mean, standard deviation, minimum, maximum, and quartile values for each numerical column in the DataFrame df.

Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating		
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1000.000000	1000.00000	
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379369	6.97270	
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.708825	1.71858	
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508500	4.00000	
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924875	5.50000	
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088000	7.00000	
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445250	8.50000	
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650000	10.00000	

df.head() #let's a look at the first few rows

Invoice ID	Date	Time	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Payment	cogs	gross margin percentage	gross income	Rating	
0	750-67-8428	01-05-2019	13:08	A	Pune	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	Ewallet	522.83	4.761905	26.1415	9.1
1	226-31-3081	03-08-2019	10:29	C	out of pune	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	Cash	76.40	4.761905	3.8200	9.6
2	631-41-3108	03-03-2019	13:23	A	Pune	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	Credit card	324.31	4.761905	16.2155	7.4
3	123-19-1176	1/27/2019	20:33	A	Pune	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	Ewallet	465.76	4.761905	23.2880	8.4
4	373-73-7910	02-08-2019	10:37	A	Pune	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	Ewallet	604.17	4.761905	30.2085	5.3

df.tail() #let's a look at the last few rows

Invoice ID	Date	Time	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Payment	cogs	gross margin percentage	gross income	Rating	
995	233-67-5758	1/29/2019	13:46	C	out of pune	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	Ewallet	40.35	4.761905	2.0175	6.2
996	303-96-2227	03-02-2019	17:16	B	pimpri chinchwad	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	Ewallet	973.80	4.761905	48.6900	4.4
997	727-02-1313	02-09-2019	13:22	A	Pune	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	Cash	31.84	4.761905	1.5920	7.7
998	347-56-2442	2/22/2019	15:33	A	Pune	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	Cash	65.82	4.761905	3.2910	4.1
999	849-09-3807	2/18/2019	13:28	A	Pune	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	Cash	618.38	4.761905	30.9190	6.6

df.info

#is a method used to display a concise summary of a DataFrame, including its index dtype and column dtypes, non-null values, and memory usage.

```
<bound method DataFrame.info of      Invoice ID        Date   Time Branch              City  
Customer type  \
0    750-67-8428  01-05-2019  13:08      A              Pune        Member
1    226-31-3081  03-08-2019  10:29      C       out of pune        Normal
2    631-41-3108  03-03-2019  13:23      A              Pune        Normal
3    123-19-1176   1/27/2019  20:33      A              Pune        Member
4    373-73-7910  02-08-2019  10:37      A              Pune        Normal
..           ...         ...    ...    ...               ...           ...
995  233-67-5758   1/29/2019  13:46      C       out of pune        Normal
996  303-96-2227  03-02-2019  17:16      B  pimpri chinchwad        Normal
997  727-02-1313  02-09-2019  13:22      A              Pune        Member
998  347-56-2442   2/22/2019  15:33      A              Pune        Normal
999  849-09-3807   2/18/2019  13:28      A              Pune        Member

     Gender            Product line  Unit price  Quantity   Tax 5%      Total  \
0    Female       Health and beauty       74.69         7  26.1415   548.9715
1    Female  Electronic accessories       15.28         5   3.8200    80.2200
2      Male      Home and lifestyle       46.33         7  16.2155   340.5255
3      Male       Health and beauty       58.22         8  23.2880   489.0480
4      Male       Sports and travel       86.31         7  30.2085   634.3785
..      ...                     ...         ...       ...      ...        ...
995    Male       Health and beauty       40.35         1   2.0175    42.3675
996  Female      Home and lifestyle       97.38        10  48.6900  1022.4900
997    Male      Food and beverages       31.84         1   1.5920    33.4320
998    Male      Home and lifestyle       65.82         1   3.2910    69.1110
999  Female     Fashion accessories       88.34         7  30.9190   649.2990

        Payment    cogs  gross margin percentage  gross income  Rating
0       Ewallet  522.83                 4.761905       26.1415     9.1
```

```
1          Cash   76.40                  4.761905        3.8200      9.6
2    Credit card  324.31                 4.761905       16.2155      7.4
3        Ewallet  465.76                 4.761905       23.2880      8.4
        Ewallet  604.17                 4.761905       30.2085      5.3
```

print(df.info()) #summery of the dataset: 1000 rows, 17 columns, no null values

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Date                     1000 non-null   object 
 2   Time                     1000 non-null   object 
 3   Branch                   1000 non-null   object 
 4   City                     1000 non-null   object 
 5   Customer type            1000 non-null   object 
 6   Gender                   1000 non-null   object 
 7   Product line             1000 non-null   object 
 8   Unit price               1000 non-null   float64
 9   Quantity                 1000 non-null   int64  
 10  Tax 5%                   1000 non-null   float64
 11  Total                    1000 non-null   float64
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
None
```

df['Product line'].unique()

#This would print out an array of unique values present in the 'Product line' column of your DataFrame df.

```
array(['Health and beauty', 'Electronic accessories',
       'Home and lifestyle', 'Sports and travel', 'Food and beverages',
       'Fashion accessories'], dtype=object)
```

branch_counts = df.groupby(['City', 'Branch']).size()
print(branch_counts)

#df by 'City' and 'Branch', and then counting the occurrences of each combination. The resulting object branch_counts likely contains the count of occurrences for each unique combination of 'City' and 'Branch'.

```
City                  Branch
Pune                  A         340
out of pune           C         328
pimpri chinchwad      B         332
dtype: int64
```

df.groupby("Product line").Total.sum().sort_values(ascending=False).head(5)

#DataFrame 'df' by 'Product line', summing the 'Total' values for each group, and then sort them in descending order. Finally, it extracts the top five results.

```
Product line
Food and beverages        56144.8440
Sports and travel         55122.8265
Electronic accessories    54337.5315
Fashion accessories       54305.8950
Home and lifestyle        53861.9130
Name: Total, dtype: float64
```

df_ratings = df.select_dtypes(include=['int', 'float']) # Select columns containing integers or floats (assumed to represent ratings)
print(df_ratings.head())

#It then prints the first few rows of this new DataFrame. This operation is likely conducted to focus on columns relevant to ratings for further analysis.

```
   Unit price  Quantity   Tax 5%     Total    cogs  gross margin percentage  \
0       74.69         7  26.1415  548.9715  522.83                 4.761905
1       15.28         5   3.8200   80.2200   76.40                 4.761905
2       46.33         7  16.2155  340.5255  324.31                 4.761905
3       58.22         8  23.2880  489.0480  465.76                 4.761905
4       86.31         7  30.2085  634.3785  604.17                 4.761905

   gross income  Rating
0       26.1415     9.1
1        3.8200     9.6
2       16.2155     7.4
3       23.2880     8.4
4       30.2085     5.3
```

df.pivot_table(index=["Invoice ID","Product line"],values=["gross income"])

#pivoting on the 'Invoice ID' and 'Product line' columns. It calculates the mean (by default) of the 'gross income' values for each combination of 'Invoice ID' and 'Product line'. This pivot table summarizes the average gross income for each product line within each invoice.

		gross income
Invoice ID	Product line	
101-17-6199	Food and beverages	16.0265
101-81-4070	Health and beauty	6.2820
102-06-2002	Sports and travel	6.3125
102-77-2261	Health and beauty	22.8585
105-10-6182	Fashion accessories	2.1480

gross income		
Invoice ID	Product line	
...	...	...
894-41-5205	Food and beverages	17.2720
895-03-6665	Fashion accessories	16.4295
895-66-0685	Food and beverages	2.7120
896-34-0956	Fashion accessories	1.0660
898-04-2717	Fashion accessories	34.3800

1000 rows × 1 columns

```
df.pivot_table(index=["Invoice ID"],
               values=["Unit price", "Quantity", "Tax 5%"],
               aggfunc=[np.sum],
               fill_value=0, margins=True)
```

#indexing it by 'Invoice ID' and calculating the sum of 'Unit price', 'Quantity', and 'Tax 5%' for each invoice. It fills missing values with zeros, includes row and column margins, and aggregates the values using the sum function. This pivot table provides a comprehensive summary of the total unit price, quantity, and tax for each invoice, along with overall totals in the margins.

	sum		
	Quantity	Tax 5%	Unit price
Invoice ID			
101-17-6199	7	16.0265	45.79
101-81-4070	2	6.2820	62.82
102-06-2002	5	6.3125	25.25
102-77-2261	7	22.8585	65.31
105-10-6182	2	2.1480	21.48
...	...	...	...
895-03-6665	9	16.4295	36.51
895-66-0685	3	2.7120	18.08
896-34-0956	1	1.0660	21.32
898-04-2717	9	34.3800	76.40
All	5510	15379.3690	55672.13

1001 rows × 3 columns

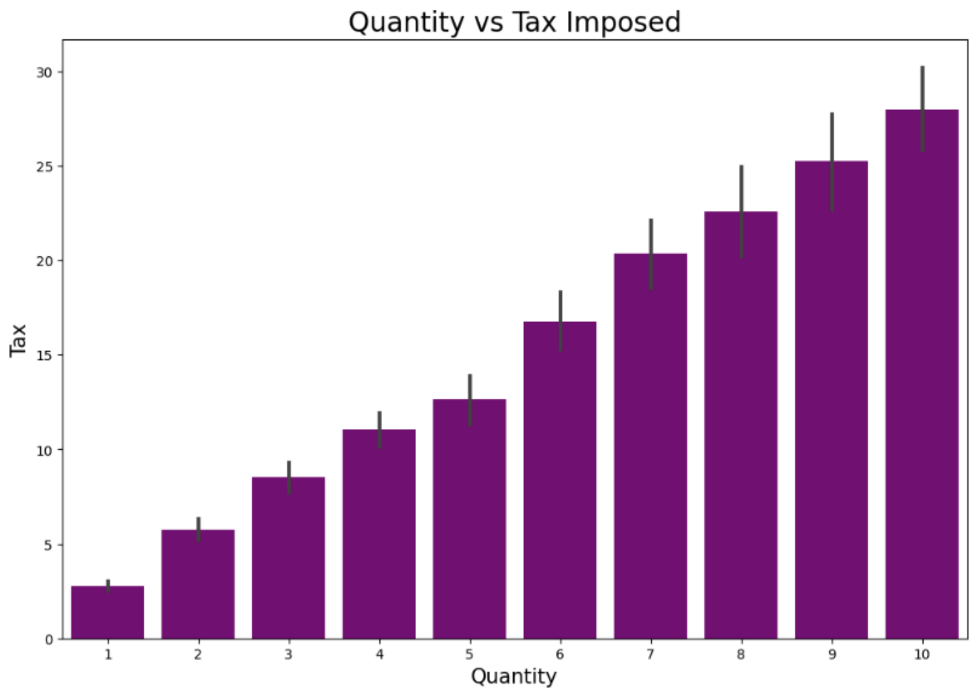
```python
n_by_category = df.groupby("Product line")["Branch"].count()
n_by_category.head(10)
```

#grouping by the 'Product line' column and counting the occurrences of 'Branch' within each group. It then displays the count of branches for the first ten product lines. This summary provides insight into the distribution of branches across different product lines.

```
Product line
Electronic accessories    170
Fashion accessories       178
Food and beverages        174
Health and beauty         152
Home and lifestyle        160
Sports and travel         166
Name: Branch, dtype: int64
```

```python
plt.figure(figsize=(12, 8))
ax = sns.barplot(x="Quantity", y="Tax 5%", data=df, color='Purple')
ax.set_title("Quantity vs Tax Imposed", fontsize=20)
ax.set_xlabel("Quantity", fontsize=15)
ax.set_ylabel("Tax", fontsize=15)
plt.show()
```

The code utilizes Seaborn and Matplotlib libraries to create a bar plot comparing the relationship between 'Quantity' and 'Tax 5%' from the DataFrame df. Each bar represents the tax imposed corresponding to different quantities. The bars are colored purple for visualization. The plot is titled "Quantity vs Tax Imposed", with appropriately labeled axes. This visualization aims to illustrate how the tax varies with different quantities of products sold.

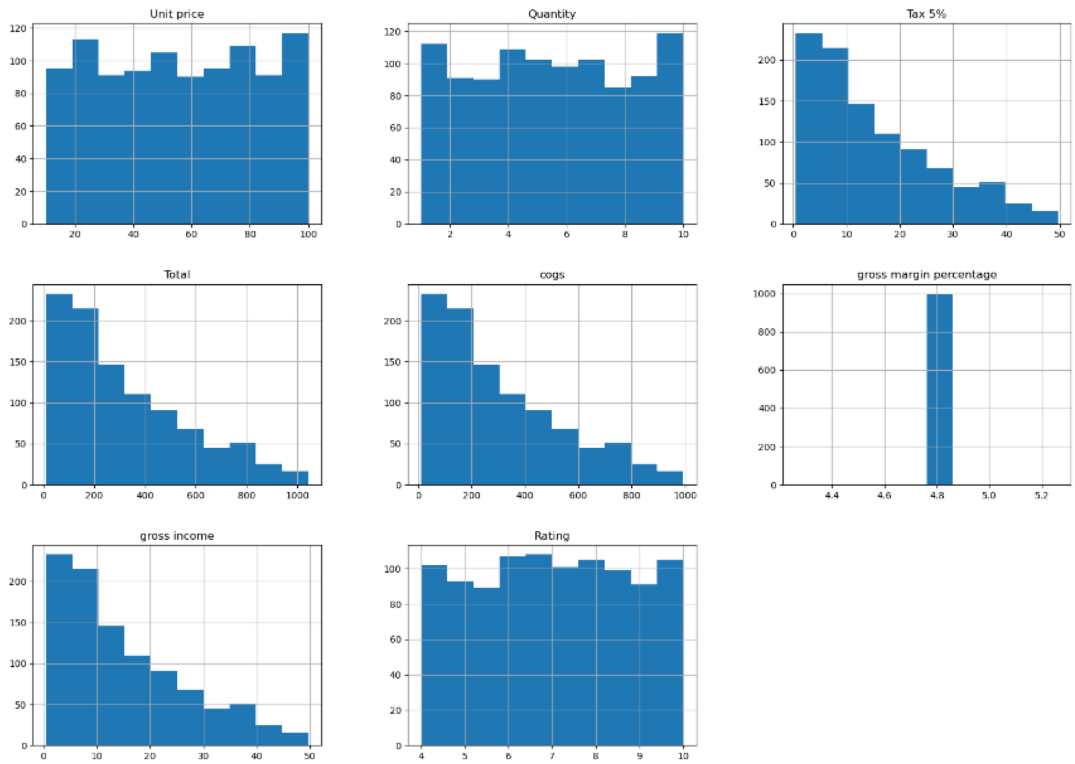


```
sns.scatterplot(x='Rating', y='gross income', data=df)
plt.xlabel('Rating')
plt.ylabel('Gross Income')
plt.title('Scatter Plot of Rating vs. Gross Income')
plt.show()
```

code utilizes Seaborn and Matplotlib libraries to create a scatter plot showing the relationship between 'Rating' and 'gross income' from the DataFrame df. Each point on the plot represents a data entry, with 'Rating' on the x-axis and 'Gross Income' on the y-axis. The plot is titled "Scatter Plot of Rating vs. Gross Income", and both axes are appropriately labeled. This visualization aims to explore any potential correlation or patterns between the rating given and the gross income generated.

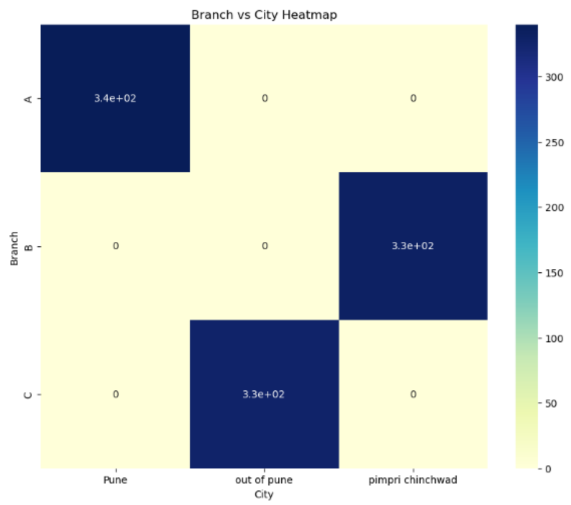

```
df.hist(figsize=(20, 14))
plt.show()
```

#arranging them in a grid layout. The figsize parameter specifies the size of the resulting figure. After generating the histograms, it displays them using Matplotlib. This visualization provides a quick overview of the distribution of values within each numerical feature in the dataset, facilitating insights into their ranges and distributions.



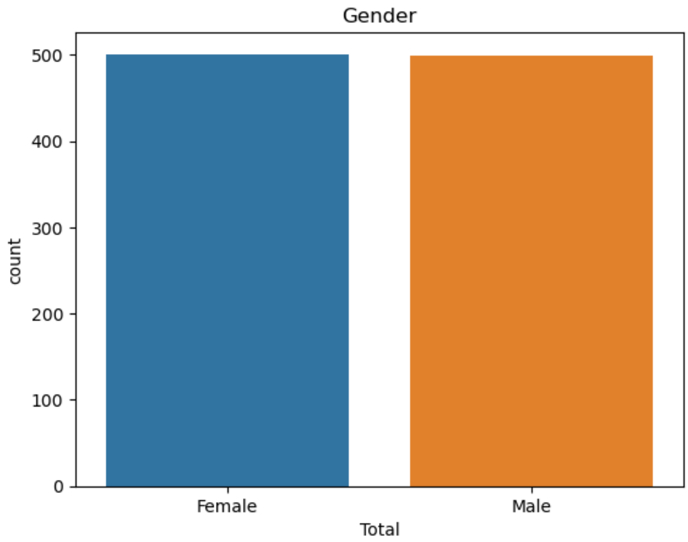

```
plt.figure(figsize=(10, 8))
heatmap_data = df.groupby(['Branch', 'City']).size().unstack(fill_value=0)
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu')
plt.title('Branch vs City Heatmap')
plt.show()
```

#The code creates a heatmap depicting the distribution of branches across different cities. Each cell represents the count of branches in a particular city-branch combination. The heatmap is annotated for clarity and uses the 'YlGnBu' colormap. It provides a visual summary of branch distribution.



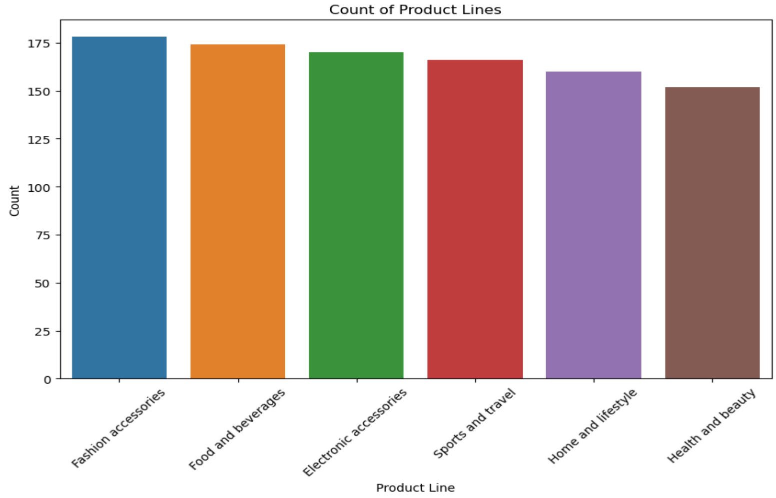
```
sns.countplot(data=df, x='Gender')
plt.title('Gender')
plt.xlabel('Total')
plt.show()
```

#Each bar represents the count of occurrences of each gender category. It's titled "Gender", with the x-axis representing the total count of each gender category. This visualization offers a straightforward summary of gender distribution within the dataset.



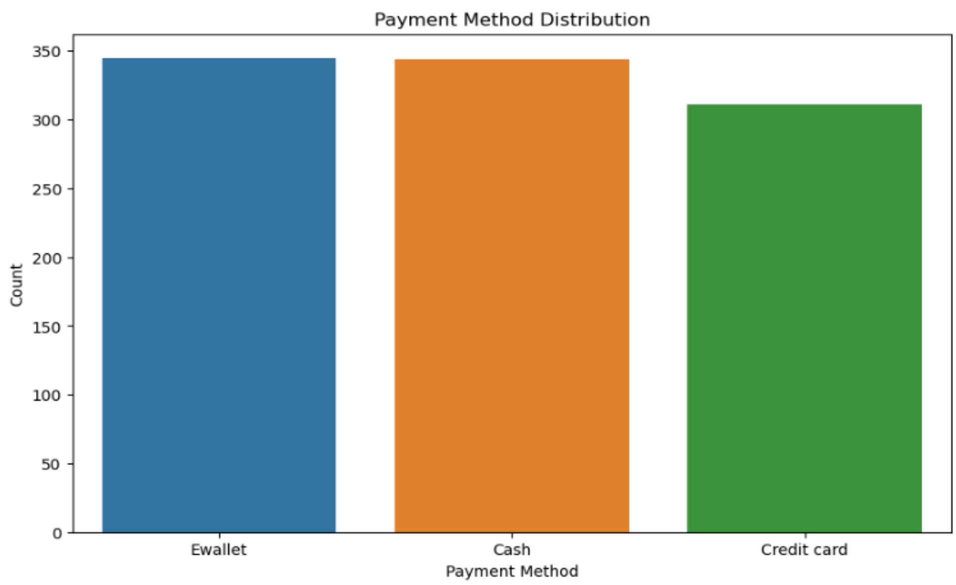

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Product line', order=df['Product line'].value_counts().index)
plt.title('Count of Product Lines')
plt.xlabel('Product Line')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

#It orders the product lines based on their frequency. The plot is titled "Count of Product Lines", with appropriately labeled axes. The x-axis represents the product line categories, while the y-axis indicates the count of occurrences. Rotating the x-axis labels by 45 degrees enhances readability. This visualization provides a clear summary of the distribution of product lines within the dataset.



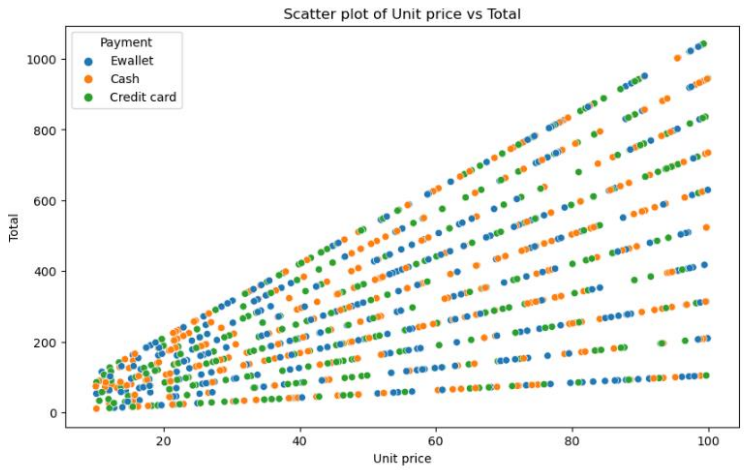

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Payment')
plt.title('Payment Method Distribution')
plt.xlabel('Payment Method')
plt.ylabel('Count')
plt.show()
```

Each bar represents the count of occurrences for a particular payment method. The plot is titled "Payment Method Distribution", with appropriately labeled axes. This visualization offers a clear summary of the frequency of different payment methods used in the dataset.



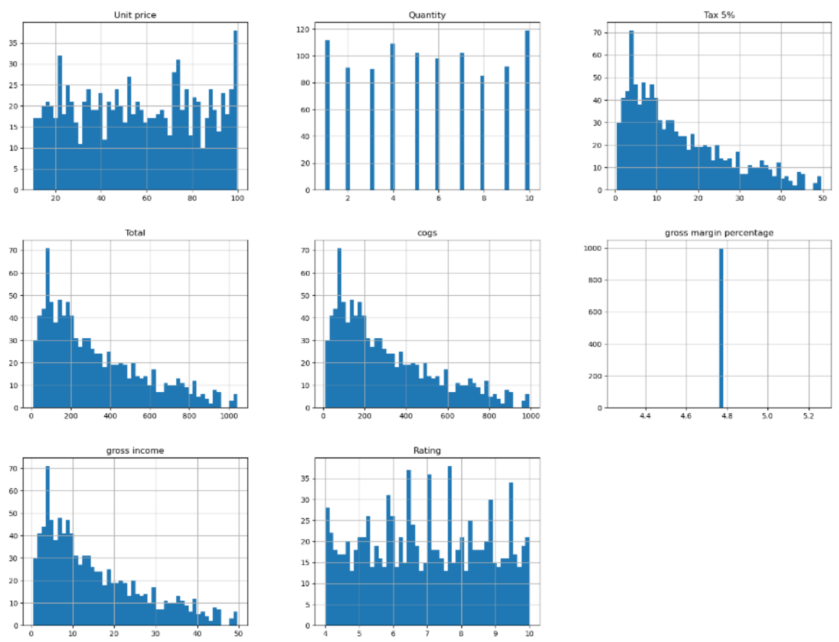
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Unit price', y='Total', hue='Payment')
plt.title('Scatter plot of Unit price vs Total')
plt.show()
```

#Each point on the plot represents an observation, with 'Unit price' on the x-axis and 'Total' on the y-axis. Additionally, the points are differentiated by color based on the 'Payment' method. The plot is titled "Scatter plot of Unit price vs Total". This visualization aims to explore the correlation between unit price and total sales while also considering different payment




```
df.hist(bins=50 ,figsize=(20,15))
plt.show()
```

The code generates histograms for each numerical column in the DataFrame df, with bins set to 50 for higher granularity. The histograms are displayed in a grid layout with a specified figure size of 20 by 15 inches using Matplotlib. This visualization provides a comprehensive overview of the distribution of values within each numerical feature, facilitating insights into their ranges and distributions with finer binning.



sns.pairplot(df)

#Each scatterplot in the pairplot represents the relationship between two variables, while histograms along the diagonal show the distribution of individual variables. This visualization is useful for identifying potential correlations and patterns between different features in the dataset.