Name: **N. Bharath**

Mail id: **bharath34.eee@gmail.com**

Contact Number: **8903676094**

Batch Code: **TN_DA_FNB01**

Project Title: **Grocery Store Sales Dataset**

Project Domain: **"Sales" in Grocery Store Chain Sales Dataset**

Mentor Name: **Kumaran M**

Submission Date: **08/10/2025**

Raw Dataset Link:
https://raw.githubusercontent.com/Bharathsarath/FINAL-PROJECT-ENTRI/refs/heads/main/grocery_chain_data.csv

Cleaned Dataset Link:
https://drive.google.com/file/d/1vXihLgOWy73_3AfQNESOMH0ZJiggOGFU/view?usp=sharing

Project Title:

Grocery Store Sales Dataset

Domain:

"Sales" in Grocery Store Chain Sales Dataset

Objective:

1. To analyse each customer purchased from various stores of a product dataset using exploratory data analysis (EDA).

2. To clean some store names, promotions, and loyalty tiers are missing.

3. To analyse Data Inconsistencies in the Quantity field.

4. Provide the Final Dataset with clean and meaningful Visualizations.

Outcome:

1. The dataset spans approximately 2 years of sales data (2023-2025) and provides insights into customer purchasing behavior, store performance, product popularity, and promotional effectiveness.

2. It contains 1,980 records from a multi-location grocery store with various features like customer id, store name, transaction date, etc,...

3. In these dataset we found missing values, date inconsistencies and some negative values. From this unclean dataset, the outcome is to provide clean and meaningful dataset by visualizing.

Dataset Information:

Source: Kaggle
Year / Timeline: August 2023 to August 2025

Dataset Description:

The Dataset contains 1,980 records from a multi-location grocery store chain, capturing detailed transaction data across various store locations, product categories, customer interactions, and promotional activities

Column Name DataType Descriptions

Customer ID---String----It tells about Unique customer identifier
Store Name----String-----Name of the Grocery store location
Transaction Date--Date---Date of transaction "i.e when"
Aisle---------String-----Product category
Product Name--String-----Name of the purchased product
Quantity------String-----Number of items purchased
Unit Price----Float------Price per individual item
Total Amount--Float------Total cost before discount
Discount Amount--Float---Total discount amount applied from total amount
Final Amount---Float------Its a Final amount after discount
Loyalty Points--Integer---Customer earned a Loyalty points

Type of Analysis:

Descriptive Analysis (summarizing the dataset)

1. Analysing the past records of each customer with the products and loyalty points earned.

Diagnostic Analysis (finding reasons behind patterns)

1. Why the customers are earning low loyalty points and to find what are the products they purchased from various stores.

2. Why purchase less products in various stores?

- Beyond this any applicable analysis, will be done in both "Predictive Analysis" and "Prescriptive Analysis"

Stages for DA Project:

Stage 1 – Problem Definition and Dataset Selection:

1. Dataset has uncleaned data with missing values, Date Inconsistencies and Type variations. From this the outcome will be a clean and meaningful dataset with visualizations.

2. Basically it is a grocery sales dataset with more than 1000 customers purchasing various products Freshfruits, Vegetables, Milk, etc,....

3. Import numpy as np, Import pandas as pd, Import matplotlib.pyplot as plt

4. Dataset description--- (1,980 rows, 11 columns and with 9 different grocery store branches)

5. Initial EDA (head, info, describe, shape, null checks)

Importing libraries and loading the dataset

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

Loading Dataset

```
df = pd.read_csv("/content/grocery_chain_data.csv")
df
```

```
df.info()
```

output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1980 entries, 0 to 1979
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customer_id       1980 non-null   int64
 1   store_name        1955 non-null   object
```

```
 2   transaction_date  1980 non-null   object
 3   aisle             1980 non-null   object
 4   product_name      1980 non-null   object
 5   quantity          1980 non-null   float64
 6   unit_price        1980 non-null   float64
 7   total_amount      1980 non-null   float64
 8   discount_amount   1980 non-null   float64
 9   final_amount      1980 non-null   float64
 10  loyalty_points    1980 non-null   int64
dtypes: float64(5), int64(2), object(4)
memory usage: 170.3+ KB
```

Initial EDA head():
df.head()

df.describe()

df.isnull()

EDA Shape():
df.shape
(1980, 11)

len(df)
output:
1980

df.columns.size
output:
11

df.dtypes
output:

	0
customer_id	int64
store_name	object
transaction_date	object
aisle	object
product_name	object

quantity	float64
unit_price	float64
total_amount	float64
discount_amount	float64
final_amount	float64
loyalty_points	int64

dtype: object

df.dtypes.value_counts()
Output:

	count
float64	5
object	4
int64	2

dtype: int64

print("Number of rows duplicated:", df.duplicated().sum())
output:

```
Number of rows duplicated: 0
```

null_counts = df.isnull().sum()
print(null_counts)
output:

```
customer_id           0
store_name           25
transaction_date      0
aisle                 0
product_name          0
quantity              0
unit_price            0
total_amount          0
discount_amount       0
final_amount          0
loyalty_points        0
dtype: int64
```

Stage 2 – Data Cleaning and Pre-processing:

1. Handle missing values (impute or drop)
2. Handle duplicates
3. Treat outliers if required
4. Check skewness and apply transformations
5. Convert data types if needed
6. Feature transformations (date parts, derived fields if required for analysis)

Before handling missing values:

print("\nMissing values per column:\n", df.isnull().sum())

output:

```
Missing values per column:
 customer_id           0
store_name           25
transaction_date      0
aisle                 0
product_name          0
quantity              0
unit_price            0
total_amount          0
discount_amount       0
final_amount          0
loyalty_points        0
dtype: int64
```

df["store_name"] = df["store_name"].fillna(df["store_name"].mode()[0])
df
output:

```
City Fresh Store
```

After handling missing values"
print("\nMissing values per column:\n", df.isnull().sum())
output:

```
Missing values per column:
 customer_id         0
```

```
store_name          0
transaction_date    0
aisle               0
product_name        0
quantity            0
unit_price          0
total_amount        0
discount_amount     0
final_amount        0
loyalty_points      0
dtype: int64
```

Output of Cleaned Dataset to get and To Download:

```
df.to_csv("final_grocery_sales.csv", index=False)

from google.colab import files
files.download("final_grocery_sales.csv")
```

```
df.dropna(inplace=True)
df

print("\nDuplicates before:", df.duplicated().sum())
df.drop_duplicates(inplace=True)
print("Duplicates after:", df.duplicated().sum())
```

output:

```
Duplicates before: 0
Duplicates after: 0
```

```
def detect_outliers(series):
    Q1, Q3 = series.quantile([0.25, 0.75])
    IQR = Q3 - Q1
    lower, upper = Q1 - 1.5*IQR, Q3 + 1.5*IQR
    return np.where(series < lower, lower, np.where(series > upper, upper, series))

    num_cols = ['quantity', 'unit_price', 'total_amount', 'discount_amount', 'final_amount',
'loyalty_points']
    for col in num_cols:
        df[col] = detect_outliers(df[col])
print(df)
```

For Skewness:

```
numeric_cols = df.select_dtypes(include=[np.number]).columns
print("\nSkewness of numeric columns:")
skew_values = df[numeric_cols].skew().sort_values(ascending=False)
print(skew_values)
```

output:

```
Skewness of numeric columns:
discount_amount    1.933832
final_amount       0.938330
total_amount       0.896945
quantity           0.024331
unit_price         0.011336
customer_id       -0.001840
loyalty_points    -0.076198
dtype: float64
```

Converting data types:

```
df['transaction_date'] = pd.to_datetime(df['transaction_date'], errors='coerce')
```

Category col to category:

```
cat_cols = ['store_name', 'aisle', 'product_name']
for col in cat_cols:
    df[col] = df[col].astype('category')
```

ID to String:

```
df['customer_id'] = df['customer_id'].astype(str)
df
```

Feature Transform:

```
df['year'] = df['transaction_date'].dt.year
df['month'] = df['transaction_date'].dt.month
df['day'] = df['transaction_date'].dt.day
df['weekday'] = df['transaction_date'].dt.day_name()
df['is_weekend'] = df['transaction_date'].dt.weekday >= 5
df['avg_basket_value'] = df.groupby('customer_id')['final_amount'].transform('mean')
df['discount_pct'] = (df['discount_amount'] / (df['total_amount'] + 1e-5)) * 100
df
```

Stage 3 – EDA and Visualizations:

1. Univariate Analysis → distribution of single variables (countplot, histogram, boxplot)

2. Bivariate Analysis → relation between two variables (scatterplot, barplot, correlation heatmap)

3. Multivariate Analysis → relation among 3+ variables (pairplot, grouped analysis, pivot tables, advanced plots)

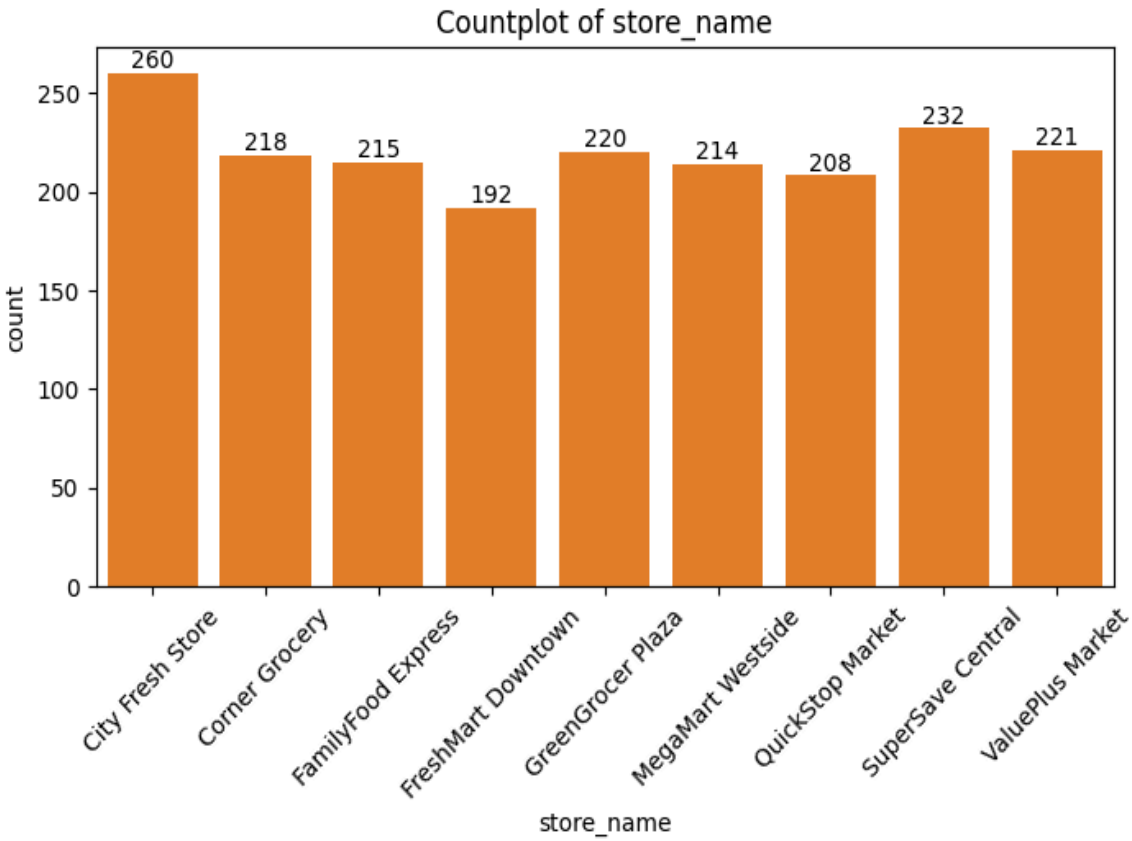
4. Interpretation MUST with every visualization

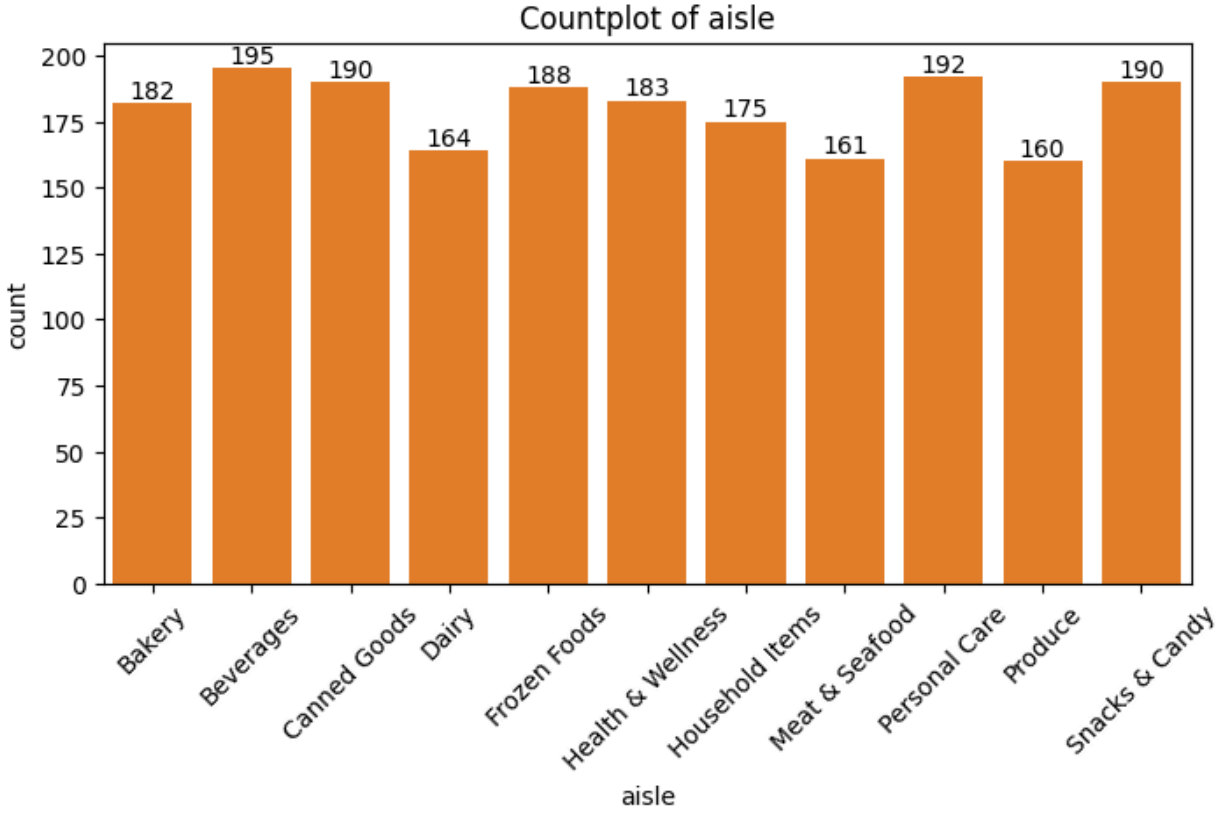
5. Focus on business story not just charts

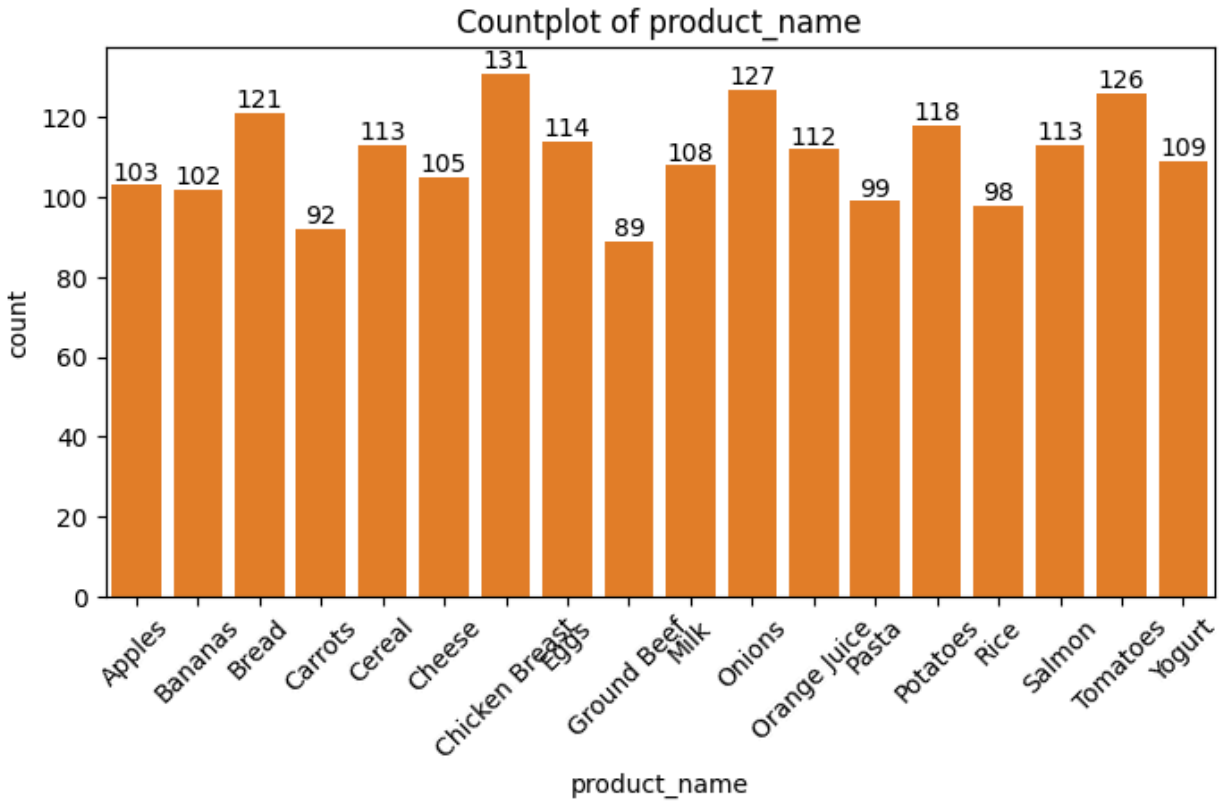
6. Each chart must be in separate cells.

Univariate Analysis:
Countplot → for categorical columns:

```
categorical_cols = ["store_name", "aisle", "product_name"]

for col in categorical_cols:
    if col in df.columns:
        plt.figure(figsize=(8,4))
        cp = sns.countplot(data=df, x=col)
        cp.bar_label(cp.containers[0])
        sns.countplot(data=df, x=col)
        plt.title(f"Countplot of {col}")
        plt.xticks(rotation=45)
        plt.show()
```


Countplot of store_name
count
store_name
260
218
215
192
220
214
208
232
221
City Fresh Store
Corner Grocery
FamilyFood Express
FreshMart Downtown
GreenGrocer Plaza
MegaMart Westside
QuickStop Market
SuperSave Central
ValuePlus Market


Countplot of aisle
count
aisle
182
195
190
164
188
183
175
161
192
160
190
Bakery
Beverages
Canned Goods
Dairy
Frozen Foods
Health & Wellness
Household Items
Meat & Seafood
Personal Care
Produce
Snacks & Candy

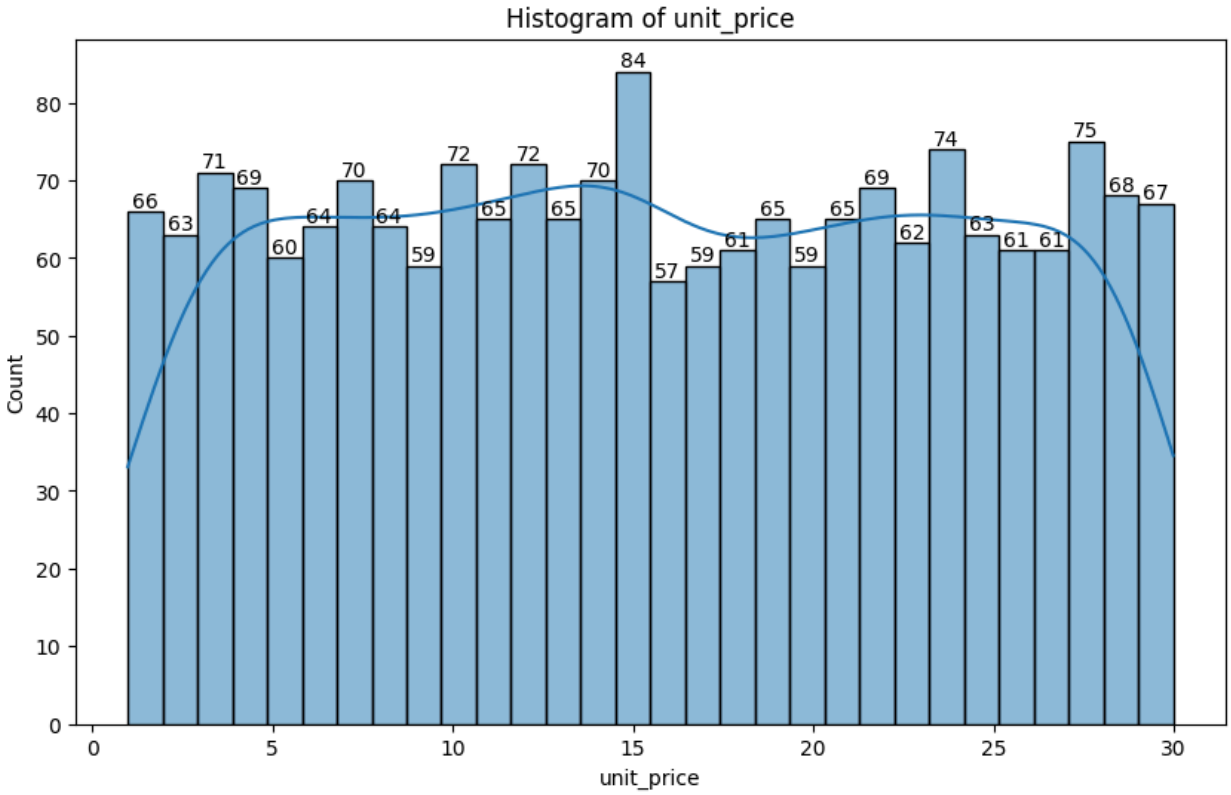


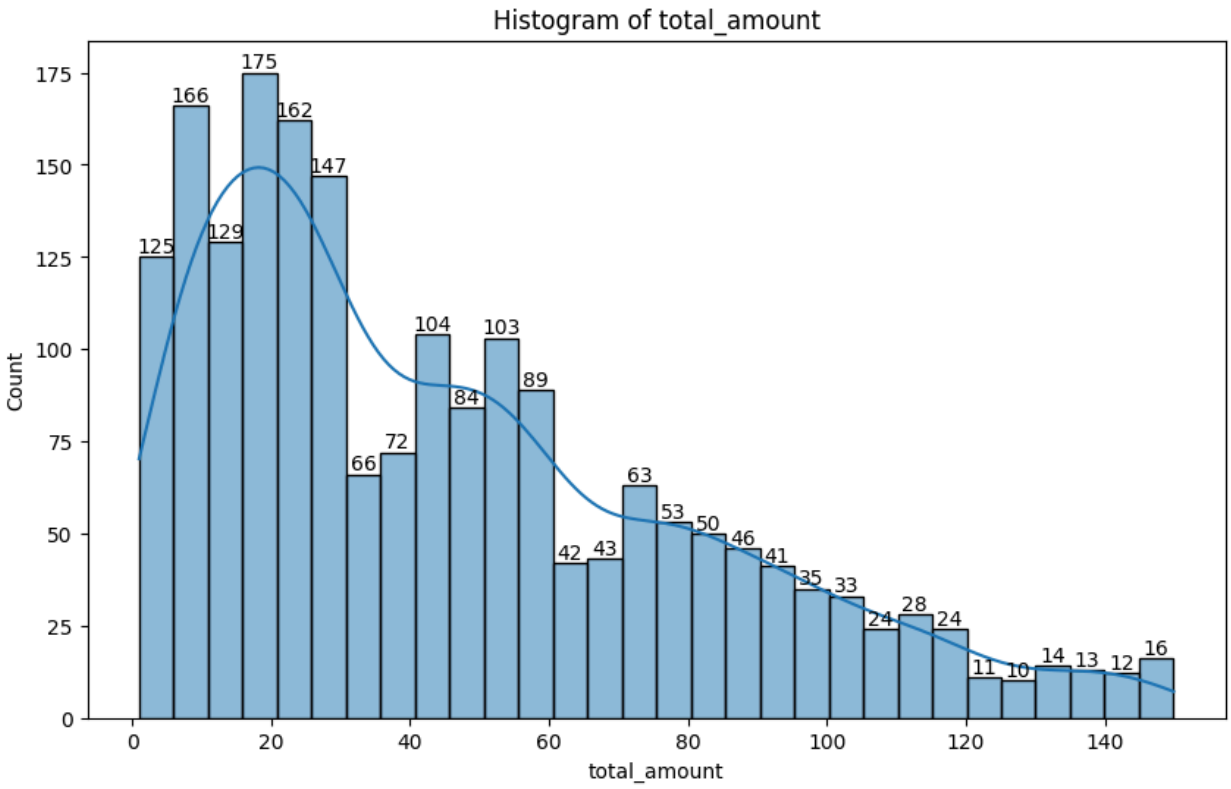
****Interpretation of Countplot****

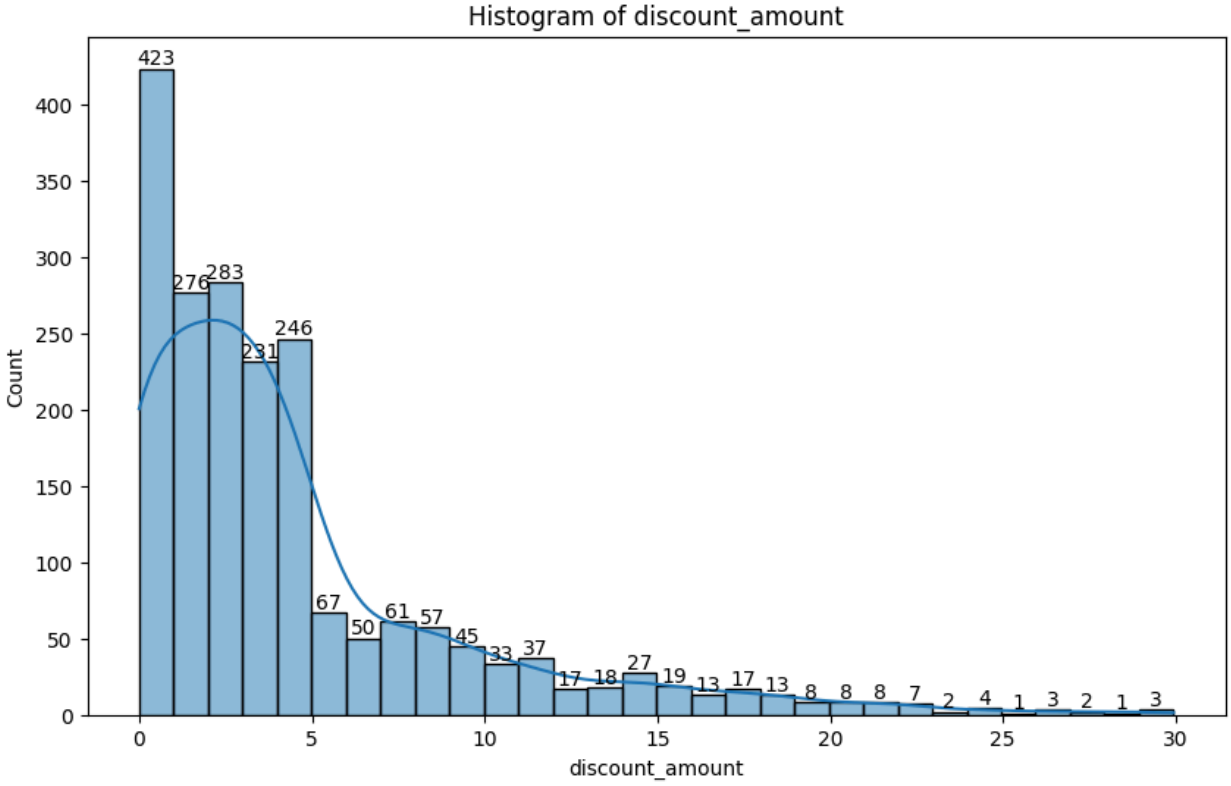
1. In Countplot mostly we use "seaborn". In this seaborn, for getting results "plt.show()" of matplotlib is of most benefit.
2. By visualizing countplot "barplot" is helpful, "y-axis" automatically will produce a label as "count" and "x-axis" is what we give in "sns.countplot(x=col)".
3. Countplot is most commonly used for categorical columns like "store_name, aisle, product_name".
4. Here, I have used a variable name called "categorical_cols" and provided all the categorical column names.
5. Using "for loop" I have combined all the categorical columns. In each categorical column we found how many counts are performed.
6. "figsize=(8,4)" this is the first step for all the plots (Globally) were to make clear and provide readability to increase or decrease the size of each plot.
7. "cp.bar_label(cp.containers[0])" is used for how many counts of each column are performed.
8. In "x-axis" categorical names called "store_name, aisle, product_name" are used.
9. In "y-axis" automatically "counts" are used.
10. The title has been given as the "Countplot of each column" name.

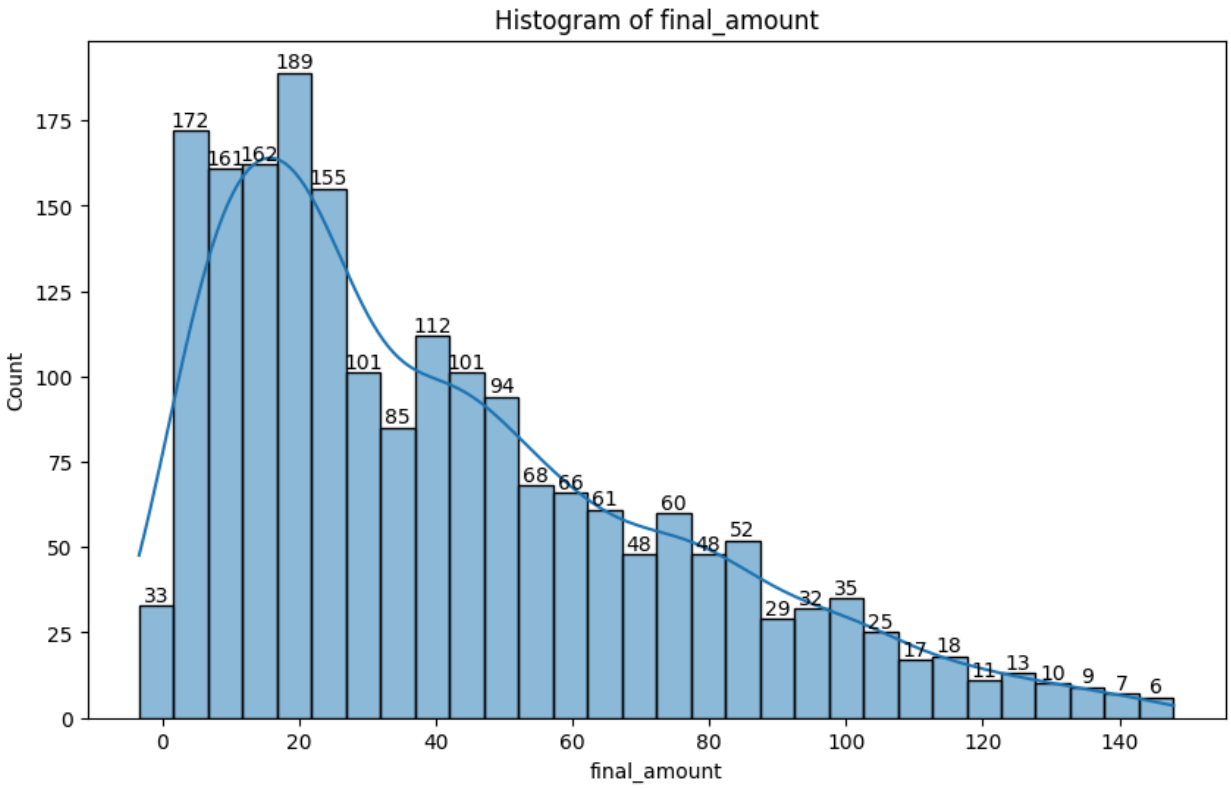
Histogram → for numeric variables:

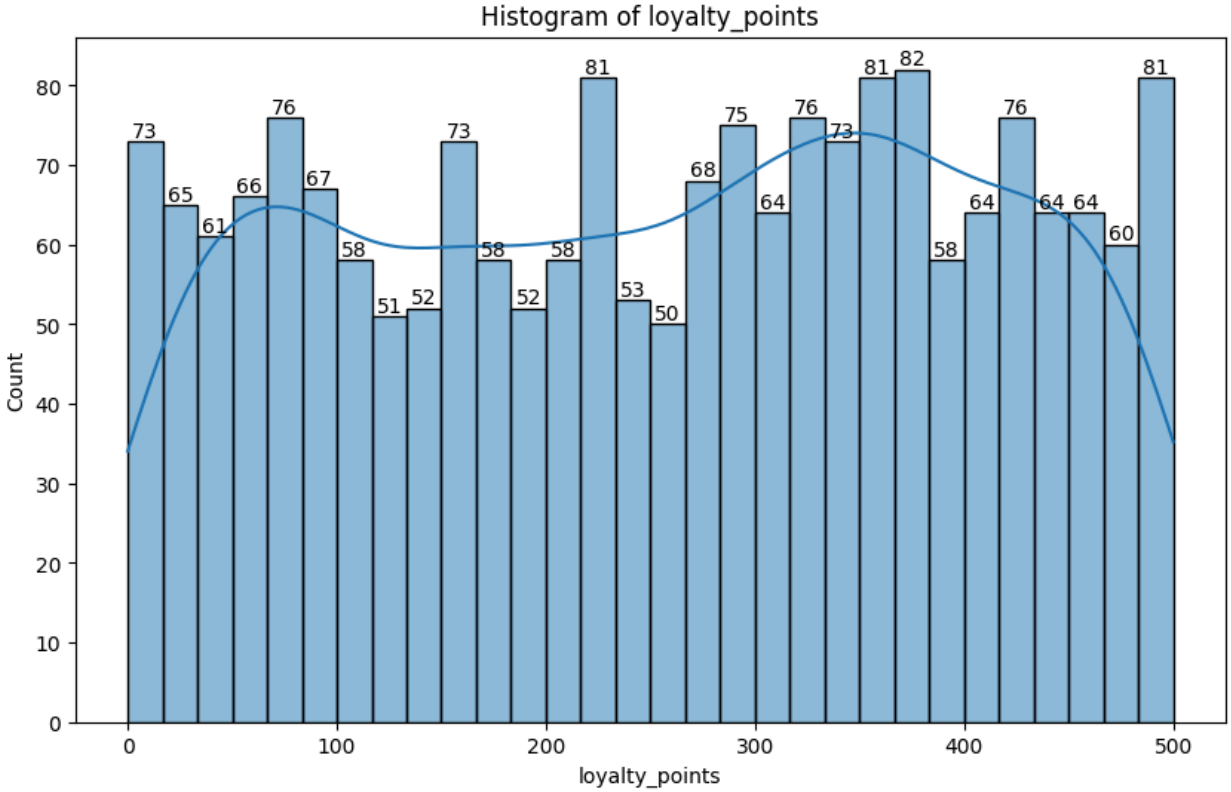
```
numeric_cols = ["unit_price", "total_amount",
                "discount_amount", "final_amount", "loyalty_points"]

for col in numeric_cols:
    if col in df.columns:
        plt.figure(figsize=(10,6))
        cp = sns.histplot(df[col], kde=True, bins=30)
        cp.bar_label(cp.containers[0])
        plt.title(f"Histogram of {col}")
        plt.show()
```




Histogram of total_amount
Count
total_amount
0
25
50
75
100
125
150
175
0
20
40
60
80
100
120
140
125
166
129
175
162
147
66
72
104
84
103
89
42
43
63
53
50
46
41
35
33
24
28
24
11
10
14
13
12
16


Histogram of discount_amount
Count
discount_amount
0
50
100
150
200
250
300
350
400
0
5
10
15
20
25
30
423
276
283
231
246
67
50
61
57
45
33
37
17
18
27
19
13
17
13
8
8
8
7
2
4
1
3
2
1
3


Histogram of final_amount
Count
final_amount
0
25
50
75
100
125
150
175
0
20
40
60
80
100
120
140
33
172
161
162
189
155
101
85
112
101
94
68
66
61
48
60
48
52
29
32
35
25
17
18
11
13
10
9
7
6

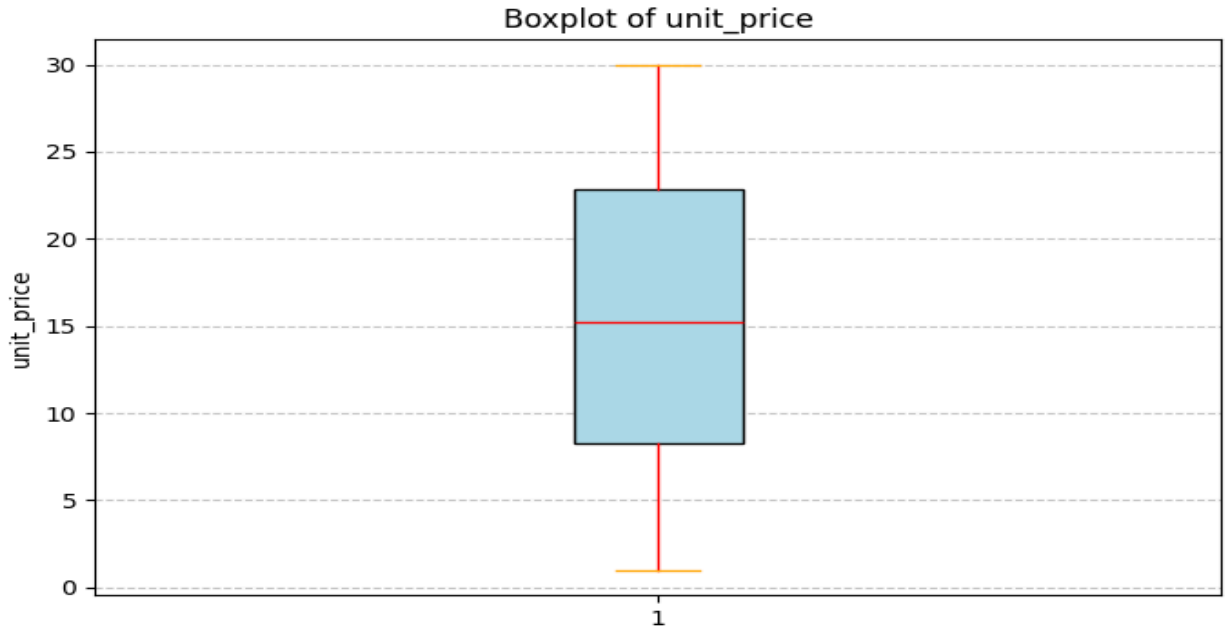

Histogram of loyalty_points
Count
loyalty_points
0
10
20
30
40
50
60
70
80
0
100
200
300
400
500
73
65
61
66
76
67
58
51
52
73
58
52
58
81
53
50
68
75
64
76
73
81
82
58
64
76
64
64
60
81

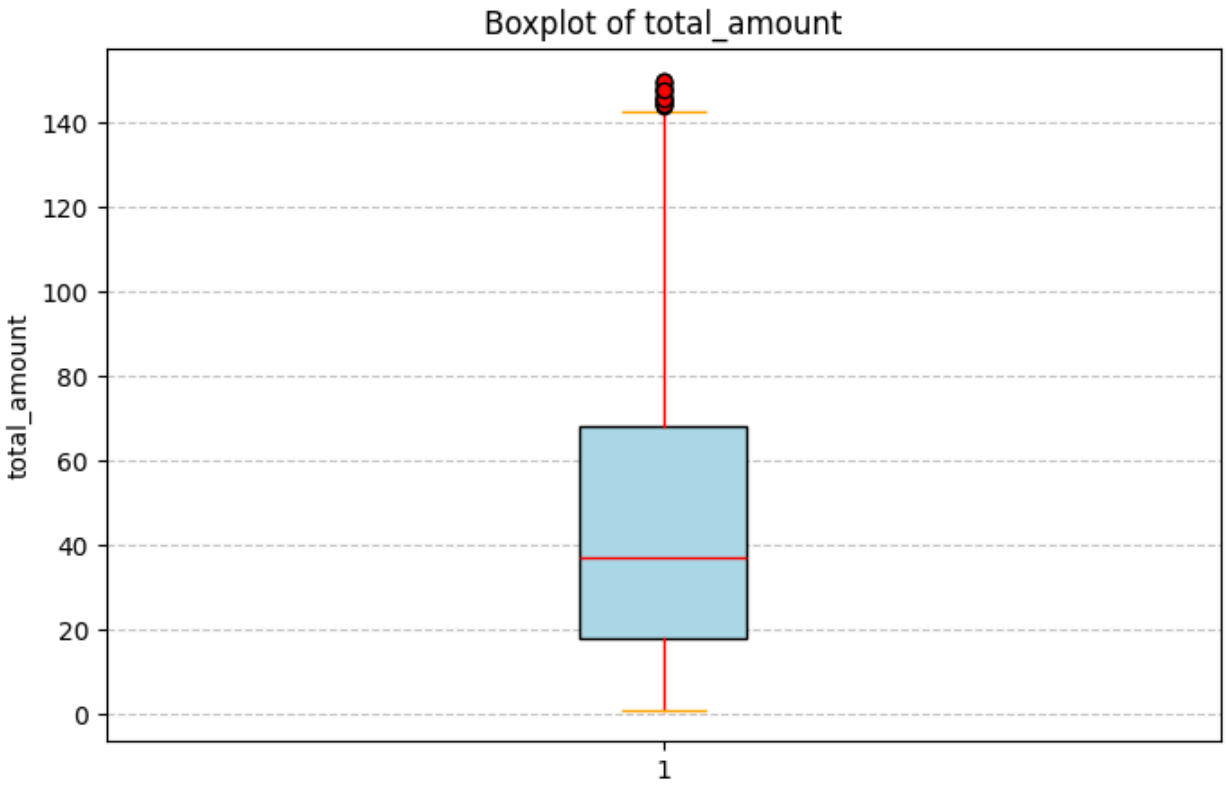
****Interpretation of Histogram****

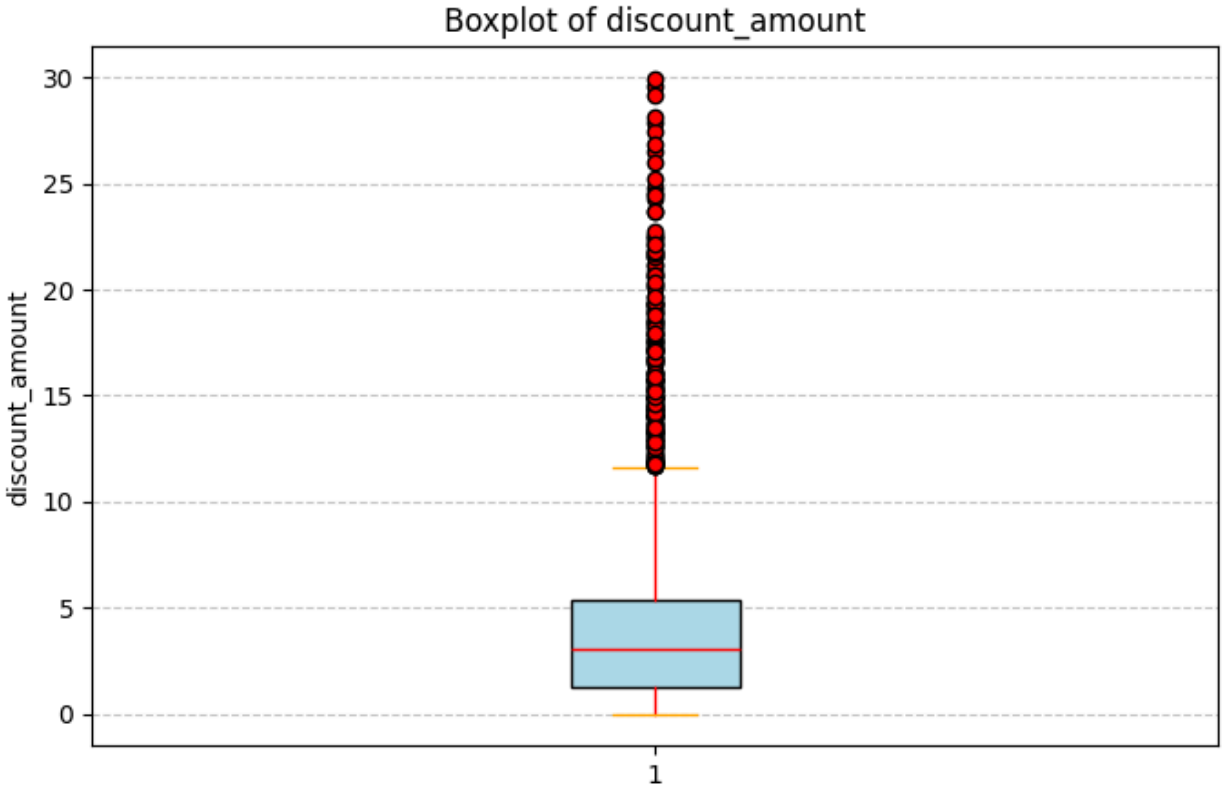
1. "kde" is used for the trending analysis which we can predict in future as well.
2. Numerical columns are used for the "histogram". Columns are "unit_price", "total_amount", "discount_amount", "final_amount", "loyalty_points".
3. All the numerical columns are updated in the "X-axis" and "y-axis" are automatically updated as "counts".
4. "cp.bar_label(cp.containers[0])" is used for how many counts of each column are performed.
5. Histograms are basically performed with the "ranges".

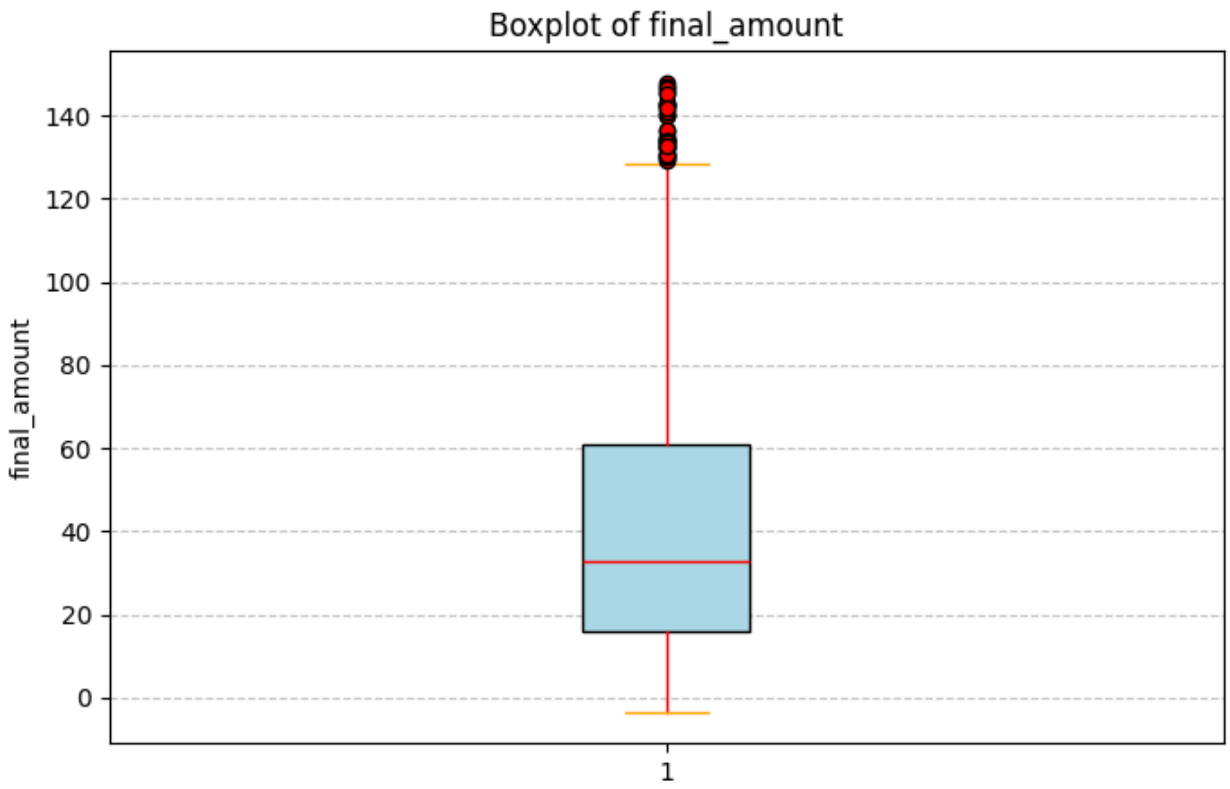
Boxplot → for numeric variables:

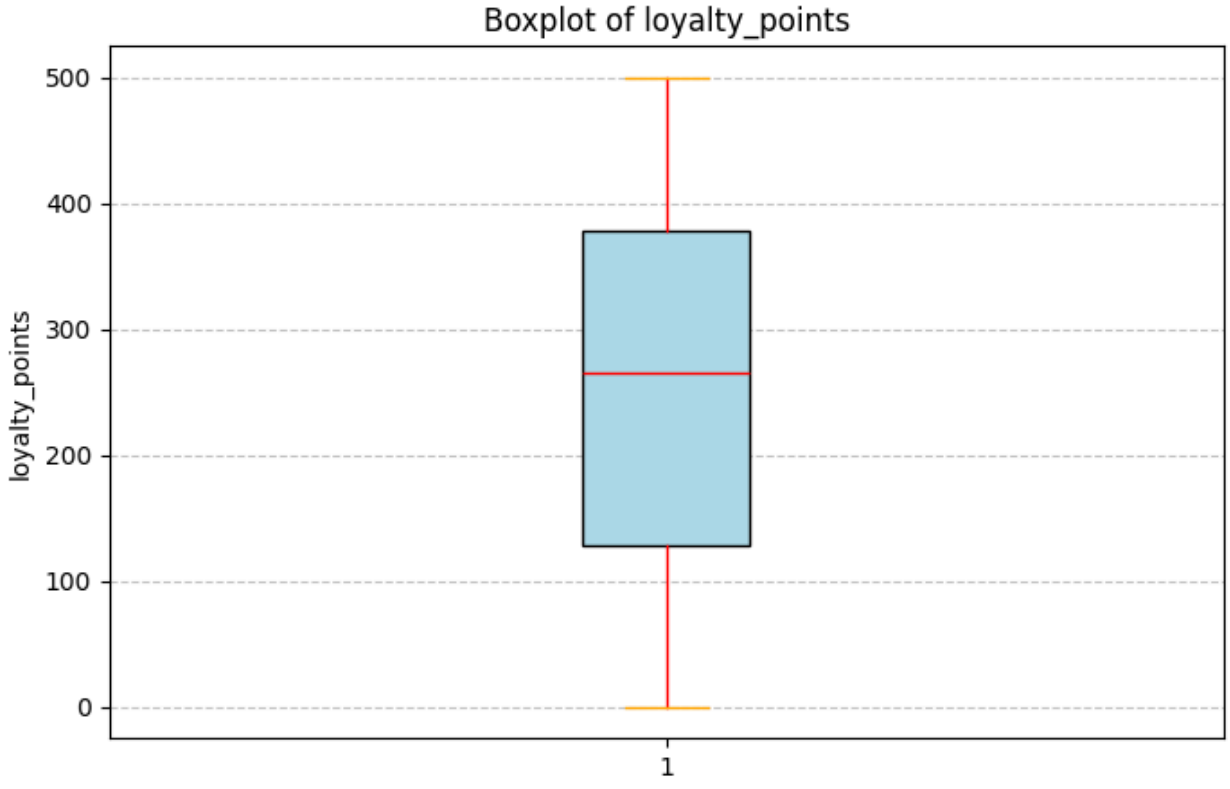
```
for col in numeric_cols:
    if col in df.columns:
        plt.figure(figsize=(8,5))
        plt.boxplot(x=df[col], patch_artist=True, boxprops=dict(facecolor='lightblue',
color='black'),
        medianprops=dict(color='red'), whiskerprops=dict(color='red'),
capprops=dict(color='orange'),
        flierprops=dict(marker='o', markerfacecolor='red'))
        plt.title(f"Boxplot of {col}")
        plt.ylabel(col)
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.show()
```




Boxplot of total_amount
total_amount
0
20
40
60
80
100
120
140
1


Boxplot of discount_amount
discount_amount
0
5
10
15
20
25
30
1


Boxplot of final_amount
final_amount
140
120
100
80
60
40
20
0
1

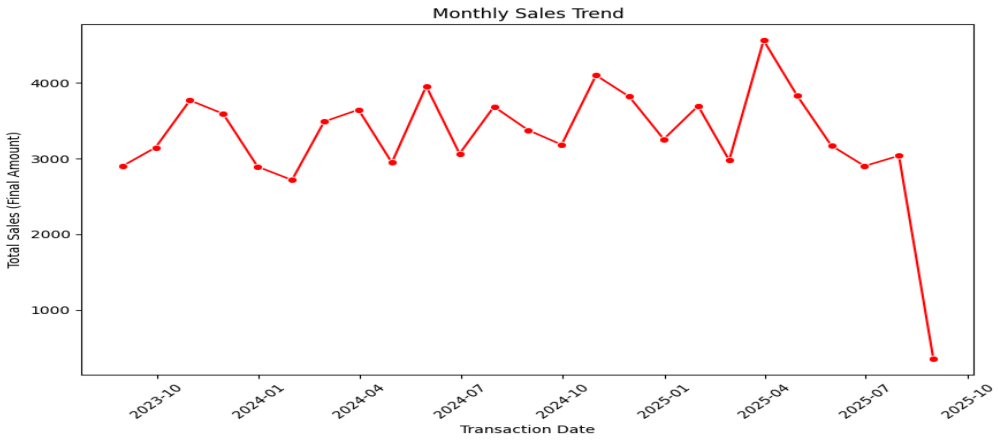

Boxplot of loyalty_points
loyalty_points
500
400
300
200
100
0
1

****Interpretation of Boxplot****

1. Boxplot is better for the Numerical columns. Shows the best result of outliers in numerical columns.
2. A box plot is a statistical graph which describes what is the Minimum, Maximum and Median (Q1, Q2, Q3).
3. The median is mostly called "Quartile 1 (25th percentile), Quartile 2 (50th percentile) and Quartile 3 (75th percentile)".
4. "o" are the outliers of the boxplot which is named "flierprops".
5. "patch_artist=True" is used for making color inside the box name called "boxprops".
6. "medianprops" is also separated with different colors in boxplot.
7. "whiskerprops" is called the lines flow from minimum to maximum in boxplot.
8. "capprops" is used for, where the values are at minimum and maximum point marked.

Lineplot:

```
monthly_sales = df.resample("ME",
on="transaction_date")["final_amount"].sum().reset_index()

plt.figure(figsize=(10,6))
sns.lineplot(data=monthly_sales, x="transaction_date", y="final_amount", marker="o",
color='red')
plt.title("Monthly Sales Trend")
plt.xlabel("Transaction Date")
plt.ylabel("Total Sales (Final Amount)")
plt.xticks(rotation=45)
plt.show()
```



****Interpretation of Lineplot****

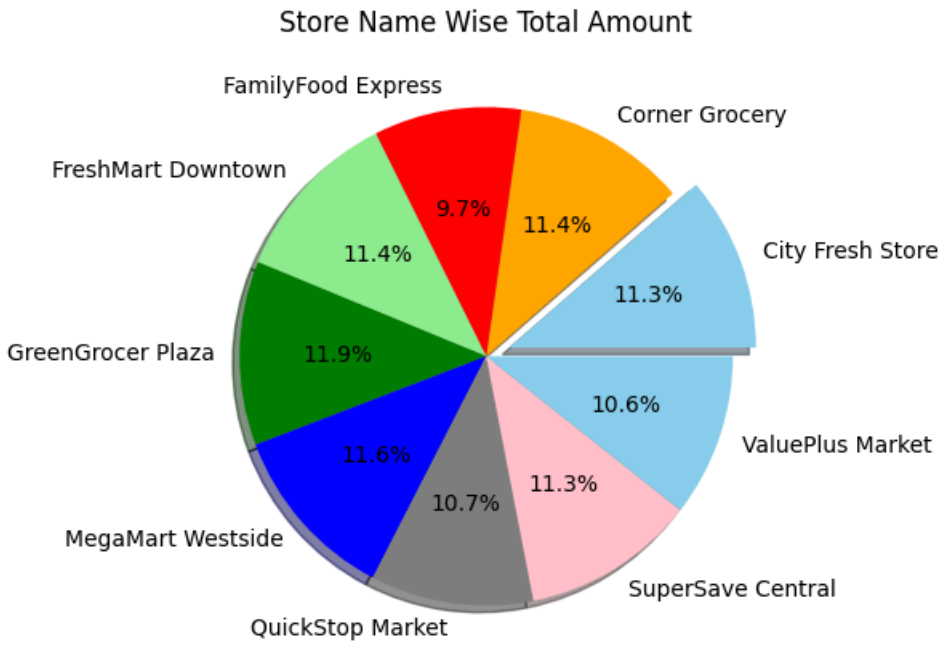
1. Line charts are mostly used for trend analysis with the help of date/time.
2. Trend analysis done by "transaction_date and final_amount".
3. "df.resample" is used for resampling the transaction_date with the "Monthly, Weekly and Yearly".
4. "seaborn" is used to provide trending analysis.

Pie chart grouped for the "store_name":

```
store_name_wise = df.groupby("store_name")['total_amount'].mean().reset_index()
store_name_wise
```

Pie chart

```
plt.figure(figsize=(10,5))
plt.pie(store_name_wise['total_amount'],
        labels = store_name_wise['store_name'],
        colors = ['skyblue', 'orange', 'red', 'lightgreen', 'green', 'blue', 'grey', 'pink'],
        autopct = '%1.1f%%',
        explode = [0.1,0,0,0,0,0,0,0],
        shadow=True)
plt.title("Store Name Wise Total Amount")
plt.show()
```

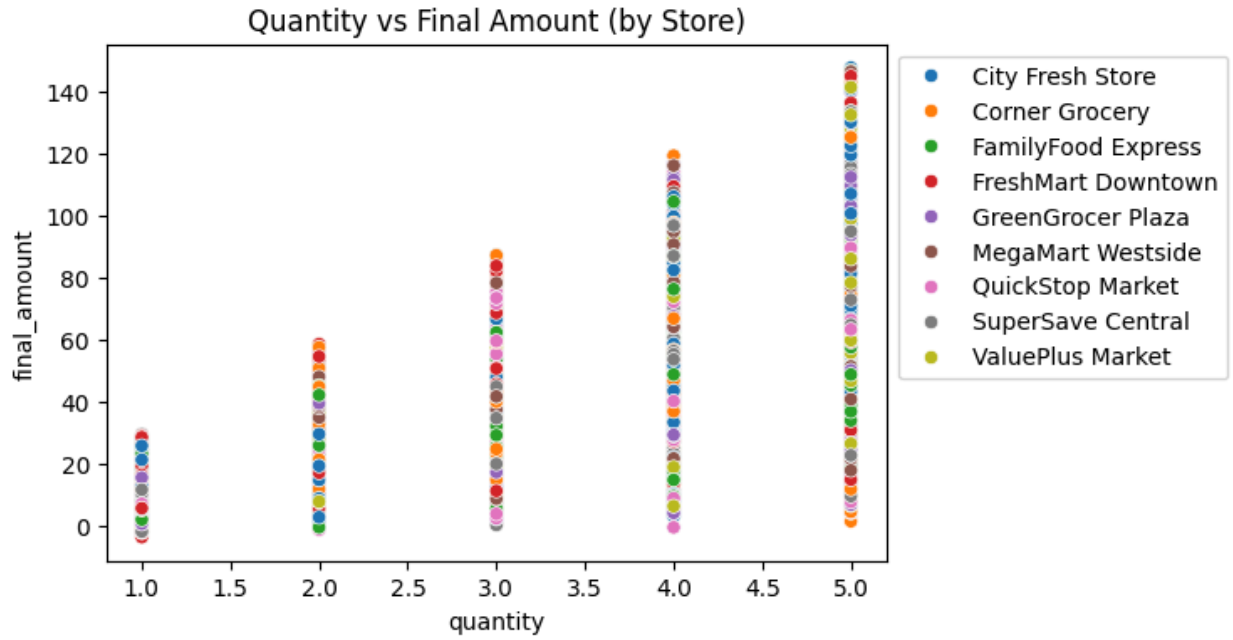


****Interpretation of Pie Chart****

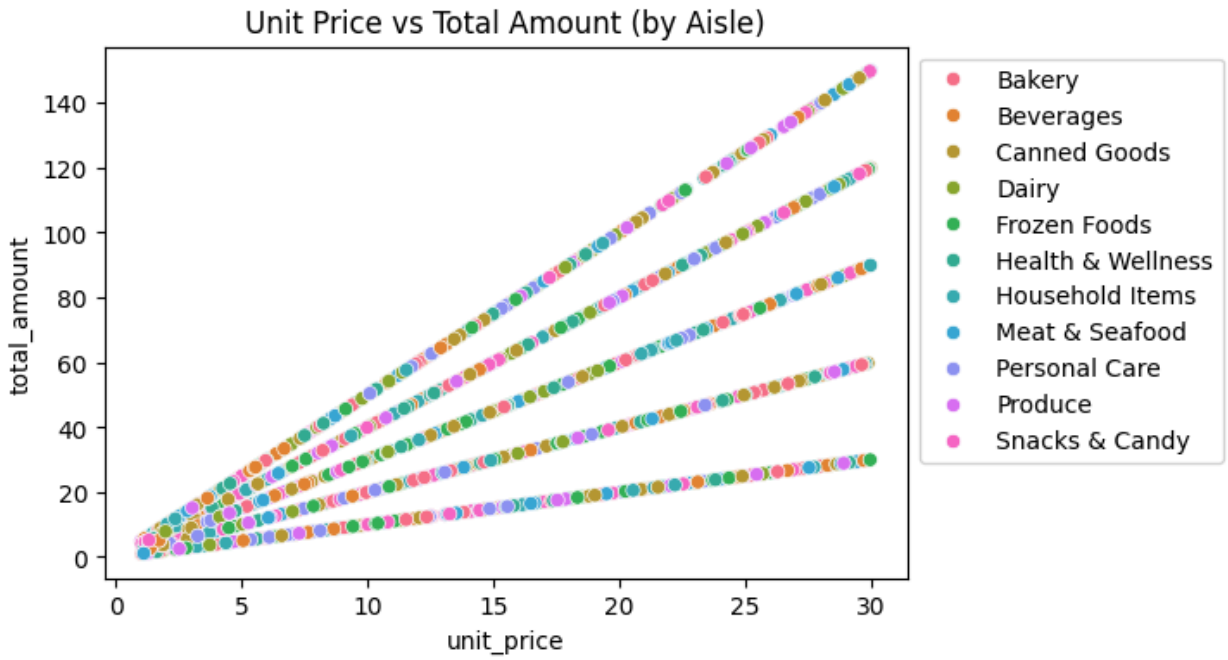
1. Pie chart is used for a circular graph where the whole pie represents 100%.
2. Each slice of the "store_name" describes a part of the whole chart.
3. Easy to visualize who has the bigger slice in the stores.
4. "explode" is used to separate a particular slice from other slices.
5. "shadow" is provided for the outer look of slices.
6. "autopct = '%1.1f%%'" is used to find how many percentages of each slice are present.

Bivariate Analysis:
Scatterplots (Numerical vs Numerical):

```
plt.figure(figsize=(6,4))
sns.scatterplot(data=df, x="quantity", y="final_amount", hue="store_name")
plt.legend(bbox_to_anchor=(1, 1))
plt.title("Quantity vs Final Amount (by Store)")
plt.show()
```




```
plt.figure(figsize=(6,4))
sns.scatterplot(data=df, x="unit_price", y="total_amount", hue="aisle")
plt.legend(bbox_to_anchor=(1, 1))
plt.title("Unit Price vs Total Amount (by Aisle)")
plt.show()
```

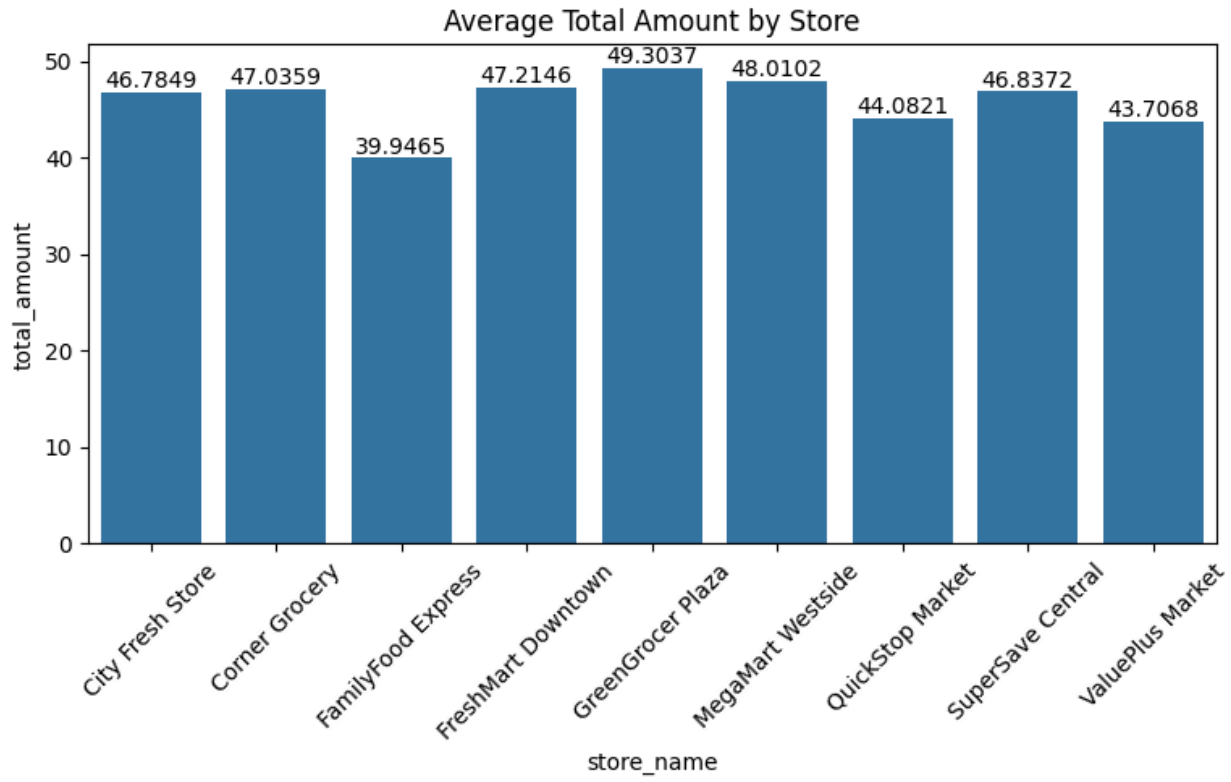


****Interpretation of Scatterplot****

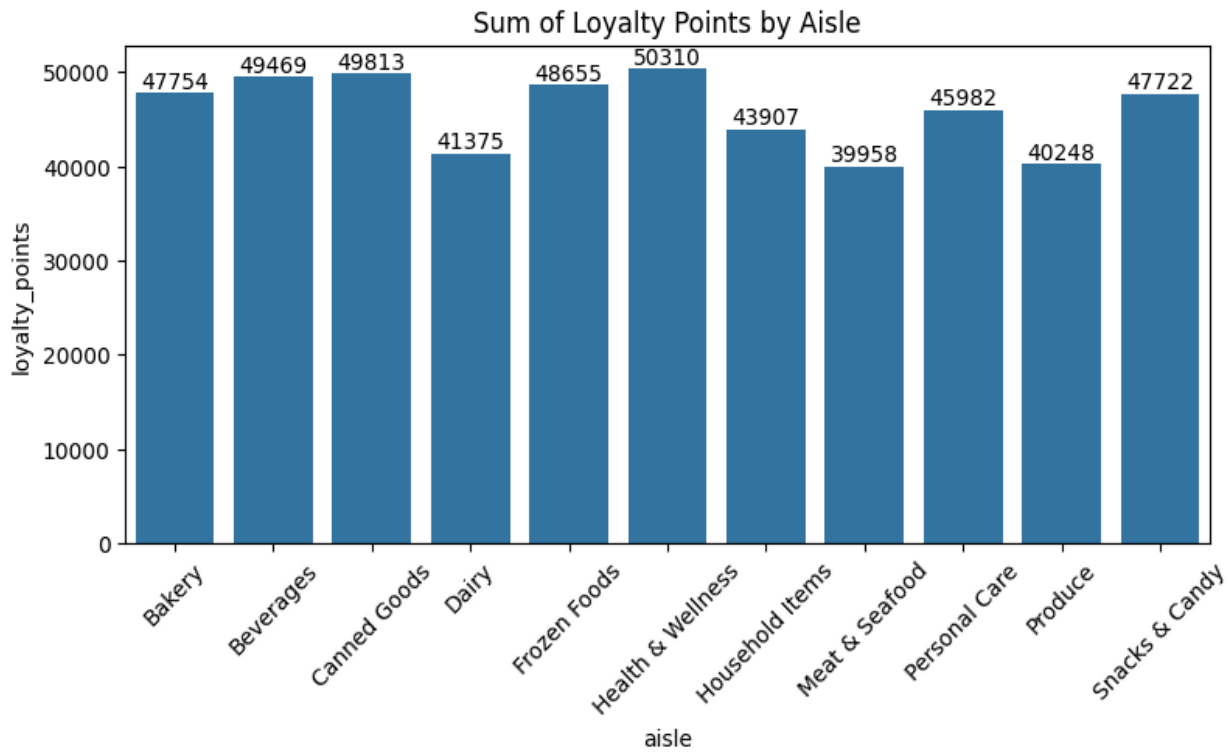
1. In scatterplot, Numerical columns are used. Here we describe "quantity and final_amount" and "unit_price and total_amount".
2. Two variable charts of scatterplot are visualized for clarity.
3. Higher quantities usually increase final_amount, but some stores show discounts that lower totals.
4. Each and every unit_price are equally contributed with the total_amount.
5. Discounts influence spending behavior differently across stores, meaning promotions are not equally effective everywhere.
6. "plt.legend" is used for the meaningful visualization which is kept alone on the "right side".
7. "x-axis" updated as "quantity and unit_price" and "y-axis" updated as "final_amount and total_amount".
8. In quantity vs final_amount, it shows who is buying more products and increases the bill.
9. In unit_price vs total_amount, it checks the effectiveness of expensive products on bills.

Barplots (Categorical vs Numerical):

```
plt.figure(figsize=(9,4))
bp = sns.barplot(data=df, x="store_name", y="total_amount", estimator="mean",
errorbar=None)

for i in bp.containers:
    bp.bar_label(i)

plt.title("Average Total Amount by Store")
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(9,4))
ap = sns.barplot(data=df, x="aisle", y="loyalty_points", estimator="sum", errorbar=None)

for i in ap.containers:
    ap.bar_label(i)

plt.title("Sum of Loyalty Points by Aisle")
plt.xticks(rotation=45)
plt.show()
```

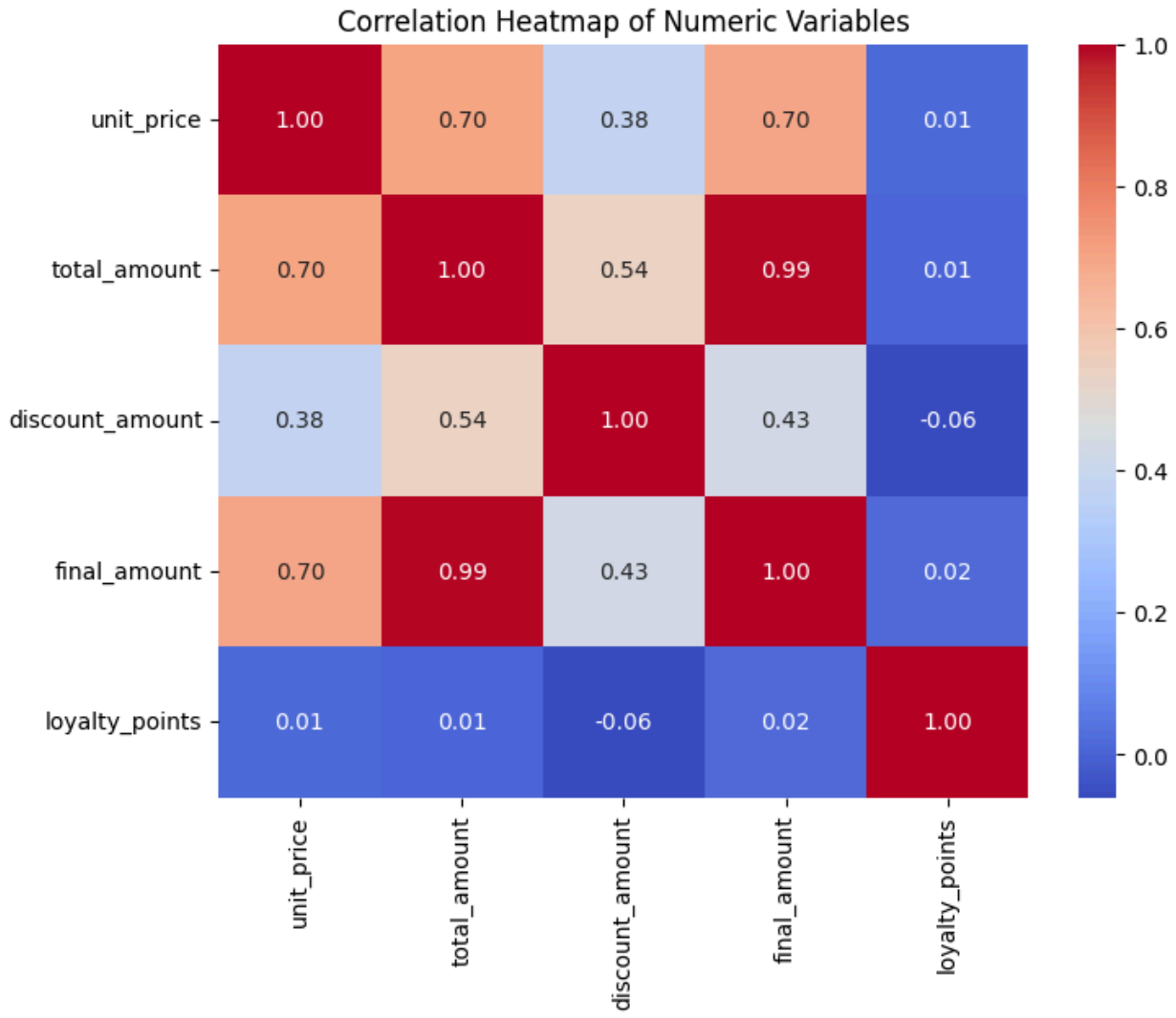


****Interpretation of Barplot****

1. It probably provides a bar chart, while giving both "x-axis" and "y-axis".
2. This chart summarizes automatically a "mean or sum" value by giving an "estimator" parameter.
3. By using Barplot we can use both "Categorical" and "Numerical" columns.
4. Here, I have taken two bar charts like the categorical column as "store_name and aisle" and Numerical column as "total_amount and loyalty_points".
5. "For loop" is used for the "containers" where values are displayed on the top of each bar.
6. Errorbar (instead of "ci"---Confidence Interval) is used to disable the lines inside each bar.
7. In store_name vs total_amount, it shows the result of which store earns more per transaction?
8. In aisle vs loyalty_points, which product category rewards more points?

Correlation Heatmap (Numeric Columns):

```
plt.figure(figsize=(8,6))
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of Numeric Variables")
plt.show()
```

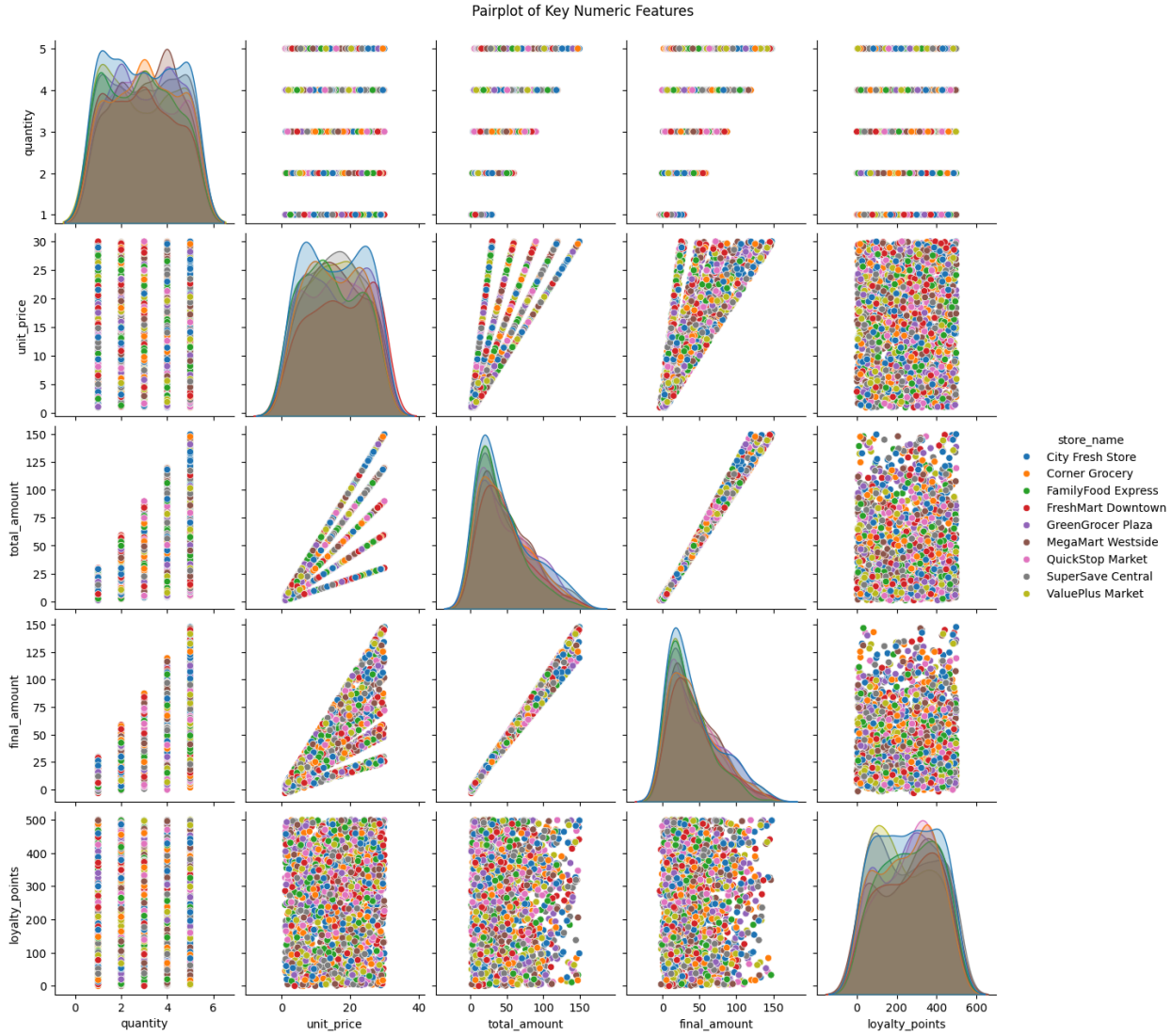


****Interpretation of Heatmap****

1. For statistics, the most commonly used map is heatmap.
2. Heatmap works only in Numerical columns.
3. Finding the correlations "(Relationship)" between two numerical columns.
4. Output ranges will be present between "-1, 0, 1". "-1" explains the Negative linear relationships, "+1" explains the positive linear relationships and "0" explains "No" relationships.
5. This Heatmap shows how numeric columns are related.
6. From this, we can understand that (for eg., "total_amount and final_amount" are highly correlated).

MultiVariate--Three Variables:
Pairplot (multiple numeric columns together):

```
sns.pairplot(data=df, vars=df[["quantity", "unit_price", "total_amount", "final_amount",
"loyalty_points"]], hue='store_name')
plt.suptitle("Pairplot of Key Numeric Features", y=1.02)
plt.show()
```

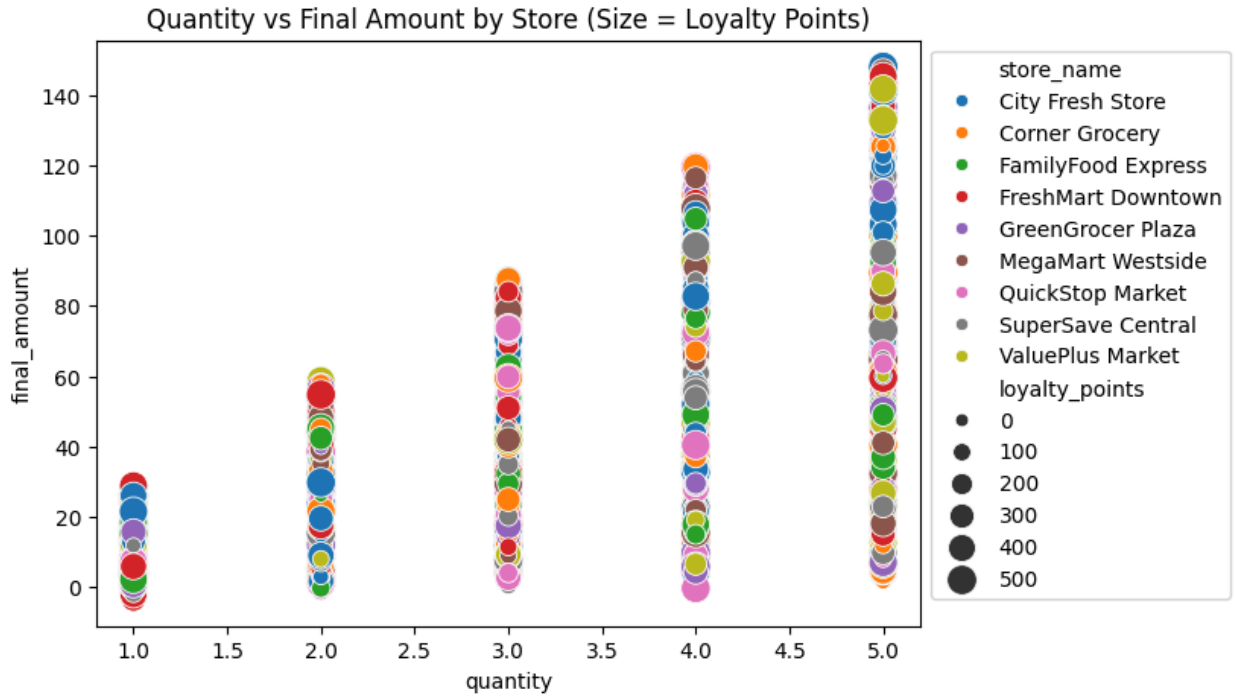


****Interpretation of Pairplot****

1. Pairplot works under multiple numeric columns together.
2. It provides the number columns of "quantity", "unit_price", "total_amount", "final_amount", "loyalty_points".
3. For fast and quick analysis, we can use a "pairplot" chart.
4. legends are used on the right side of the pairplot by using "hue".

Scatterplot with Hue (3 variables):

```
plt.figure(figsize=(7,5))
sns.scatterplot(data=df, x="quantity", y="final_amount", hue="store_name",
size="loyalty_points", sizes=(40,200))
plt.legend(bbox_to_anchor=(1, 1))
plt.title("Quantity vs Final Amount by Store (Size = Loyalty Points)")
plt.show()
```

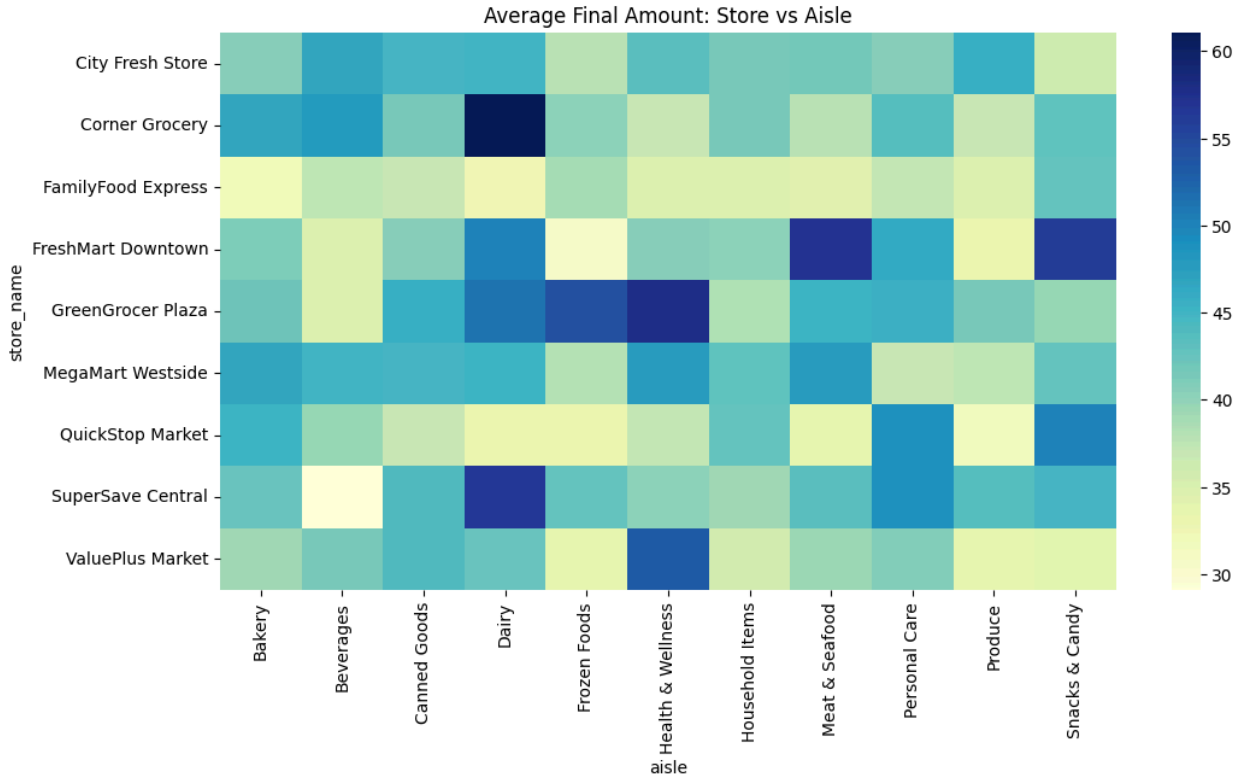


****Interpretation of Scatterplot****

1. In scatterplot, there is a relationship of three variables. Here we describe "quantity and final_amount" by "store_name".
2. Three variable charts of scatterplot are visualized for clarity.
3. "plt.legend" is used for the meaningful visualization which is kept alone on the "right side".
4. "x-axis" updated as "quantity" and "y-axis" updated as "final_amount and total_amount".
5. For Visualizing, higher quantities usually increase final_amount, but some stores show discounts that lower totals.
6. Discounts influence spending behavior differently across stores, meaning promotions are not equally effective everywhere.

Grouped Analysis (Pivot Table):

```
pivot = df.pivot_table(values="final_amount",
                       index="store_name",
                       columns="aisle",
                       aggfunc="mean").fillna(0)
plt.figure(figsize=(12,6))
sns.heatmap(pivot, cmap="YlGnBu", annot=False)
plt.title("Average Final Amount: Store vs Aisle")
plt.show()
```



****Interpretation of Pivot Table (Grouped Analysis)****

Why Heatmap?

Because it converts the pivot table into a colored grid.

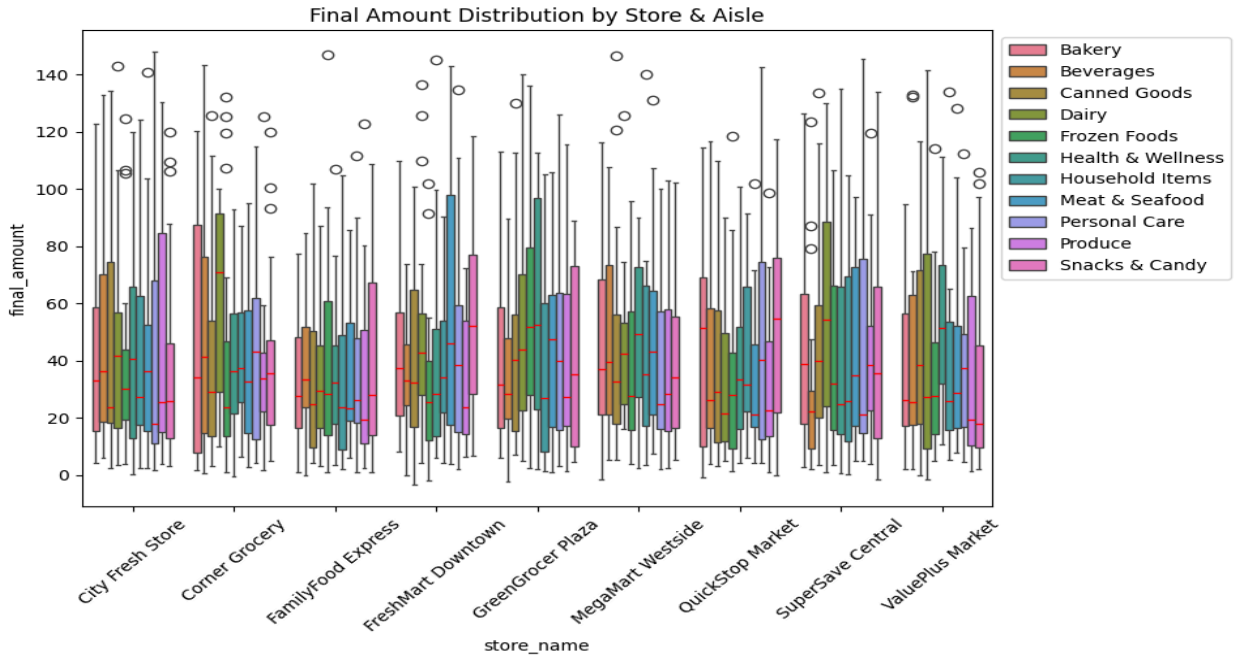
* Darker color = higher average spend.
* Lighter color = lower average spend.

1. This Pivot table compares two categorical columns and one numeric column. For e.g. we can say "average of final_amount across store_name and aisle".

2. Pivot table reshapes the data into a table format where rows and columns are based on categories, and values are aggregated.
3. I want to analyse the final_amount after the discount. Rows will present the "store_names" and Columns will present the "aisle".
4. "aggfunc="mean", it shows the average value of "final_amount" per store_name and aisle.
   e.g. "GreenGrocer Plaza" average spend in "Fresh Produce" aisle"
5. ".fillna(0)", it replaces "NaN" with "0" if any store doesn't have sales in a particular aisle.
6. "cmap='YlGnBu'", it is a color scheme of "Yellow, Green, Blue".
7. By default numbers will be present inside the heatmap. Hence, if we use "annot=False", numbers will not be shown in heatmap.

Advanced plot:
Boxplot (Categorical + Numeric + Hue):

```
plt.figure(figsize=(9,6))
sns.boxplot(data=df, x="store_name", y="final_amount", hue="aisle",
medianprops=dict(color='red'))
plt.legend(bbox_to_anchor=(1, 1))
plt.title("Final Amount Distribution by Store & Aisle")
plt.xticks(rotation=45)
plt.show()
```



****Interpretation of Boxplot****

1. Boxplot is used with both Categorical and Numerical columns with the "hue".
2. "o" are the outliers of the boxplot chart.
3. "x-axis" are from categorical columns and "y-axis" are from Numerical columns.
4. "legends" are used on the right side of the chart.
5. "medianprops" is used for highlighting the middle point of a boxplot with the color "red".
6. It is mainly used for the distribution of sales "final_amount" across multiple categories called "store_name and aisle".

Stage 4 – Documentation, Insights and Presentation:

1. Show a dashboard aligning all the charts in powerbi by connecting python in PowerBI

2. Summarize findings in plain English (what do the patterns mean)

3. Highlight 3–5 key insights (trends, anomalies, correlations)

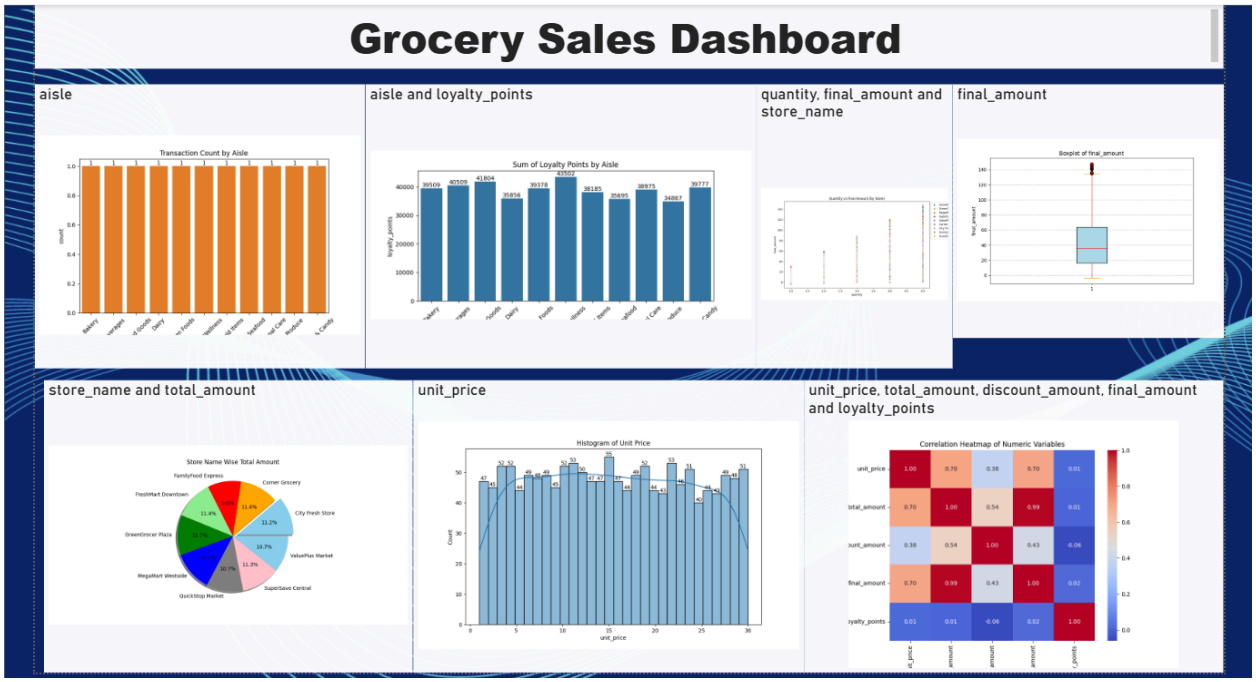
4. Provide recommendations for business or decision-making

5. Explain the FINAL STORY WITH THE DASHBOARD

6. Create a PDF pasting your DASHBOARD, explaining all the above mentioned concepts and submit it

your stage 4 pdf Link

'''
https://drive.google.com/file/d/1sIGbM89uzrjGQa8FTSXStgPbAoe_6cqD/view?usp=sharing
'''



Dashboard Explanation:

- From the above Grocery Sales Dashboard, I have made a few charts to explore what are the major differences in each column.
- Almost all Dataset has uncleaned data like Missing Values, Inconsistencies, Negative Values and Type variations.
- Probably most of the data is clean and few rows are not in proper values. Hence, they are updated.
- Now, I have made all the changes in the dataset and they are clean and ready for visualization.
- In this Dashboard, few charts are explored with meaningful visualizations like “Count Plot, Pie Chart, Histogram Plot, Heatmap, Bar Plot, Scatter Plot and Box Plot”.

Count Plot:

- It Shows which product aisles (Bakery, Beverages, Dairy, Snacks, etc.) have the highest number of transactions.
- Countplot used for categorical columns to count how many numbers are available in each column.

Pie Chart:

- Pie chart is used for circular graphs where it presents 100% results with the Categorical and Numerical columns.
- Each slice represents the whole chart with clear and neat visuals. Easy to visualize who has the bigger slice in this chart.
- Visualizes sales contribution of each store. One store (ValuePlus Market) has a much larger share compared to others.

Histogram Plot:

- Distribution of product prices, showing pricing strategy (mostly clustered around affordable ranges).
- Numerical columns are mostly used. Histogram is basically performed with the “Ranges”.

Heatmap:

- Displays relationships among numeric variables “unit_price, total_amount, discount_amount, final_amount, and loyalty_points”.
- Unit_price and final_amount are strongly correlated → higher priced items lead to higher revenue.
- Loyalty_points have very little correlation with sales values, meaning loyalty rewards may not be driving purchases effectively.
- Finding the correlations "(Relationship)" between two numerical columns.

Bar Plot:

- It probably provides a bar chart, while giving both "x-axis" and "y-axis".
- By using Barplot we can use both Categorical columns and Numerical columns.

Scatter Plot:

- Compares quantities purchased and sales revenue across different stores.
- Scatterplot is visualized for clarity. In quantity vs final_amount, it shows who is buying more products and increases the bill.
- Higher quantities usually increase final_amount, but some stores show discounts that lower totals.

Box Plot:

- A box plot is a statistical graph which describes what is the Minimum, Maximum and Median
- Highlights spending patterns, median purchase value, and outliers “customers who spend much more than average”.

Key Insights from the Dashboard:

From Aisles:

- High transactions in Beverages, Snacks, and Dairy → focus on promotions in these categories.

From Store Performance:

- One store (ValuePlus Market) contributes ~30% of sales → dependency risk. Need to strengthen other stores to balance sales.

From Customer Distribution:

- Boxplot shows the majority of purchases are small, but a few high-value customers exist → potential for VIP customer programs.

Pricing Strategy:

- Most products fall within a middle price range; outliers exist. Could optimize price tiers.

In Loyalty Points Impact:

- Weak correlation with revenue → current loyalty program may not be effective in boosting sales.

Business Recommendations (Future Strategy):

- Invest in marketing or product mix adjustments in stores contributing less revenue.
- Identify and target high-value customers (from boxplot outliers) with personalized offers.

- Cross-sell items from popular aisles (e.g., Beverages + Snacks) to increase basket size.
- Redefine rewards to actually influence purchase behavior (since the current program has weak impact).
- Analyze unit price sensitivity; introduce premium and economy product ranges to capture more customer segments.
- Add transaction_date-based line charts to monitor seasonal trends (festivals, weekends).

Final Story:

- This dashboard analyzes grocery sales across multiple stores, focusing on transactions, pricing, loyalty programs, and customer purchasing behavior.
- It describes Sales Distribution and Store Performance which contributes the highest share of sales. In the product category, transactions are evenly distributed across aisles like **Bakery, Beverages, Snacks, and Dairy**, which are consistently popular.
- Products are mostly priced in a moderate range, which supports affordability and high transaction volume.

Future Enhancement:

1. Time-Series Analysis----We can Add monthly/weekly/daily trends of sales and loyalty points.
2. Customer Segmentation----Apply RFM (Recency, Frequency, Monetary) analysis to design targeted promotions.

- Group customers by spending patterns (high-value, average, discount-seekers).
- Personalize loyalty rewards instead of flat-point schemes.

3. Store & Regional Comparison---Expand dataset to include location/region details of stores.
4. Advanced Visualizations---Add drill-down dashboards: Store → Aisle → Product-level analysis.

Finally:

By integrating these future enhancements, the grocery dataset project can move from descriptive analytics (what happened) to predictive (what will happen) and prescriptive (what should be done) analytics. This makes the dashboard not only a reporting tool but also a decision-making system for business leaders.

Conclusion:

This grocery dataset reveals that a single store dominates sales, certain aisles drive frequent purchases, and loyalty programs aren't strongly influencing revenue. By optimizing store strategies, revisiting loyalty design, and leveraging customer insights, the business can improve sales distribution, retain high-value customers, and drive overall profitability.