HULT COFFEE HOUSE

Presented by MBAN Team 5

Mateo Franco

Karla Esmeralda Felipe Chavarria

Angel Lanto

Habib Mohamed Sultan Saleem

Phani Teja Venigalla

Elina Wallenborn

ABOUT US

Hells Kitchen
(Centre Manhattan)

Astoria (Queens)

Lower Manhattan

BUSINESS PROBLEM

Our goal is to provide the best service to be able to maximise revenue across our 3 locations.

Sales/location

Hell's Kitchen - 236,511 sales
Astoria - 232,243 sales
Lower Manhattan - 230,057 sales

- Sales Trends
- Product Performance
- Location Analysis
- Customer Purchase Behaviour
- Pricing Impact

~~PLANTING THE SEED~~

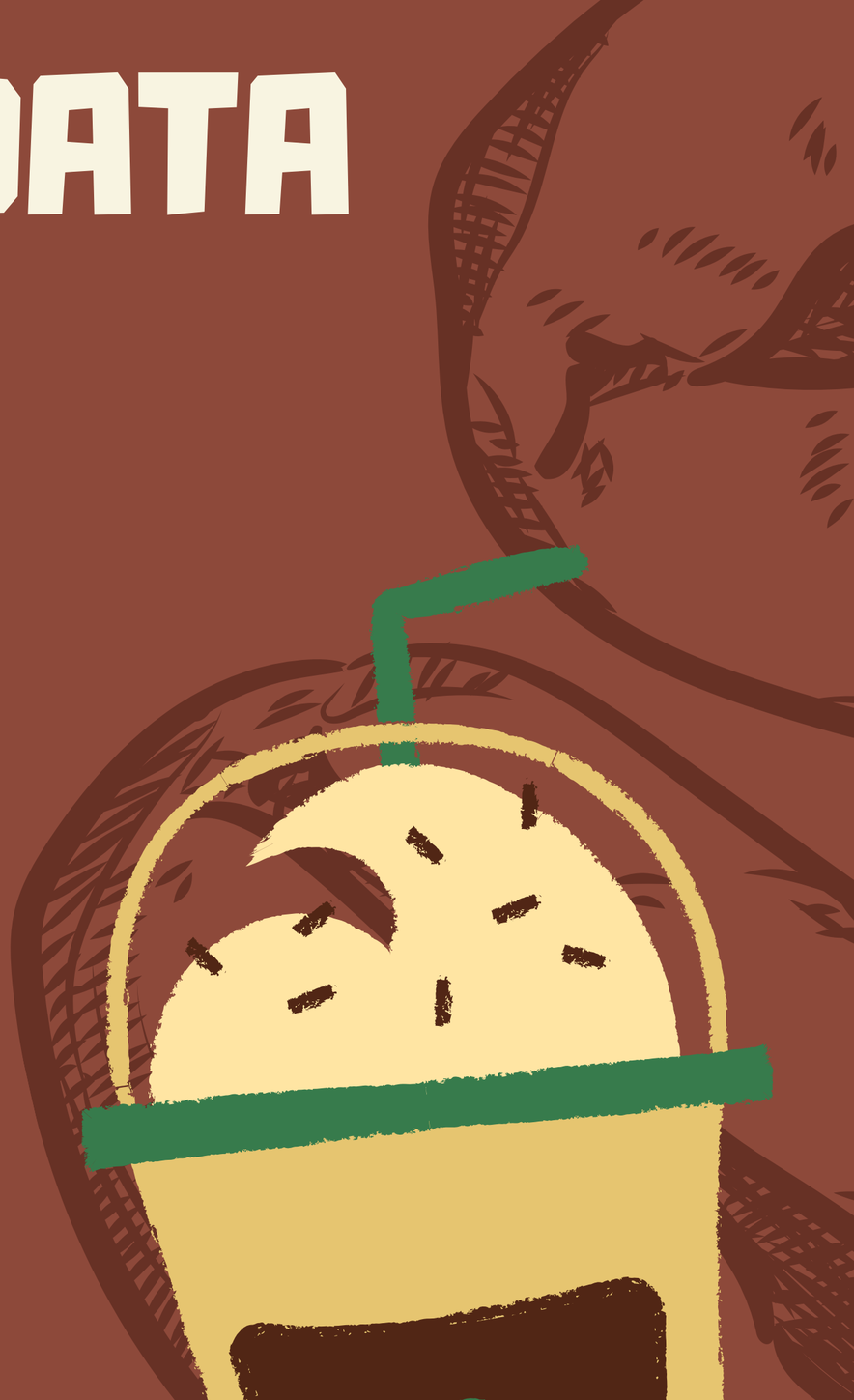
IMPORTING THE DATASET

```
IMPORT PANDAS AS PD
IMPORT NUMPY AS NP
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT SEABORN AS SNS

COFFEE_SHOP_SALES_DATA = PD.READ_CSV("/USERS/HABS/DOWNLOADS/PROJECT.CSV")
```

TYPES OF COFFEE DATA

```
print(coffee_shop_sales_data.dtypes)
```

```
transaction_id        int64
transaction_date     object
transaction_time     object
store_id              int64
store_location       object
product_id            int64
transaction_qty       int64
unit_price          float64
Total_Bill          float64
product_category     object
product_type         object
product_detail       object
Size                 object
Month Name           object
Day Name             object
Hour                  int64
Month                 int64
Day of Week           int64
dtype: object
```



MISSING VALUES

Check for Missing Values

```
missing_values = coffee_shop_sales_data.isnull().sum()
print("Missing_values:\n", missing_values)
```

```
Missing_values:
 transaction_id           0
transaction_date          0
transaction_time          0
store_id                  0
store_location            0
product_id                0
transaction_qty           0
unit_price                0
Total_Bill                0
product_category          0
product_type              0
product_detail            0
Size                      0
Month Name                0
Day Name                  0
Hour_of_day               0
Month                     0
Day of Week               0
calculated_total_bill     0
dtype: int64
```

CONVERT DATA TYPE

convert 'transaction_date' & 'transaction_time' to datetime format

```
coffee_shop_sales_data['transaction_date'] = pd.to_datetime(coffee_shop_sales_data['transaction_date'], format='%d-%m-%Y')
coffee_shop_sales_data['transaction_time'] = pd.to_datetime(coffee_shop_sales_data['transaction_time'], format='%H:%M:%S')
```

convert 'Month Name' & 'Day Name' to categorical data type

```
coffee_shop_sales_data['Month Name'] = coffee_shop_sales_data['Month Name'].astype('category')
coffee_shop_sales_data['Day Name'] = coffee_shop_sales_data['Day Name'].astype('category')
coffee_shop_sales_data['Size'] = coffee_shop_sales_data['Size'].astype('category')
```

rename 'hour' column for clarity

```
coffee_shop_sales_data.rename(columns={'Hour': 'Hour_of_day'}, inplace=True)
```

FILTER & STANDARDIZE DATA

filter rows with values not within Range

```
coffee_shop_sales_data = coffee_shop_sales_data[coffee_shop_sales_data['Hour_of_day'].between(0, 23)]
coffee_shop_sales_data = coffee_shop_sales_data[coffee_shop_sales_data['Day of Week'].between(0, 6)]
coffee_shop_sales_data = coffee_shop_sales_data[coffee_shop_sales_data['Month'].between(1, 12)]
```

Standardize columns for consistancy

```
coffee_shop_sales_data['store_location'] = coffee_shop_sales_data['store_location'].str.lower()
coffee_shop_sales_data['product_category'] = coffee_shop_sales_data['product_category'].str.lower()
coffee_shop_sales_data['product_type'] = coffee_shop_sales_data['product_type'].str.lower()
coffee_shop_sales_data['product_detail'] = coffee_shop_sales_data['product_detail'].str.lower()
coffee_shop_sales_data['Size'] = coffee_shop_sales_data['Size'].str.lower()
coffee_shop_sales_data['Month Name'] = coffee_shop_sales_data['Month Name'].str.lower()
coffee_shop_sales_data['Day Name'] = coffee_shop_sales_data['Day Name'].str.lower()
```

FINDING OUTLIERS

```
Q1 = coffee_shop_sales_data['Total_Bill'].quantile(0.25)
Q3 = coffee_shop_sales_data['Total_Bill'].quantile(0.75)
IQR = Q3 - Q1

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

Creating a new column to flag outliers

```
coffee_shop_sales_data['Total_Bill_outlier'] = (coffee_shop_sales_data['Total_Bill'] < lower_bound) | \
                                               (coffee_shop_sales_data['Total_Bill'] > upper_bound)

print(coffee_shop_sales_data[['transaction_id', 'Total_Bill', 'Total_Bill_outlier']])
```

Result

```
        transaction_id  Total_Bill  Total_Bill_outlier
0               114301         3.0               False
1               115405         3.0               False
2               115478         3.0               False
3               116288         3.0               False
4               116714         3.0               False
...                ...         ...                 ...
149111          129465        17.0                True
149112          133523       360.0                True
149113          133674       360.0                True
149114          133744       360.0                True
149115          149043       360.0                True

[149116 rows x 3 columns]
```

COFFEE DATA

```
coffee_shop_sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149116 entries, 0 to 149115
Data columns (total 19 columns):
 #   Column              Non-Null Count   Dtype         
---  ------              --------------   -----         
 0   transaction_id      149116 non-null  int64         
 1   transaction_date    149116 non-null  datetime64[ns]
 2   transaction_time    149116 non-null  datetime64[ns]
 3   store_id            149116 non-null  int64         
 4   store_location      149116 non-null  object        
 5   product_id          149116 non-null  int64         
 6   transaction_qty     149116 non-null  int64         
 7   unit_price          149116 non-null  float64       
 8   Total_Bill          149116 non-null  float64       
 9   product_category    149116 non-null  object        
 10  product_type        149116 non-null  object        
 11  product_detail      149116 non-null  object        
 12  Size                149116 non-null  object        
 13  Month Name          149116 non-null  object        
 14  Day Name            149116 non-null  object        
 15  Hour_of_day         149116 non-null  int64         
 16  Month               149116 non-null  int64         
 17  Day of Week         149116 non-null  int64         
 18  Total_Bill_outlier  149116 non-null  bool          
dtypes: bool(1), datetime64[ns](2), float64(2), int64(7), object(7)
memory usage: 20.6+ MB
```

DELETING COLUMNS

```
coffee_shop_sales_data.drop(columns=['Total_Bill_outlier'], inplace=True)
```

```
coffee_shop_sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149116 entries, 0 to 149115
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   transaction_id    149116 non-null  int64         
 1   transaction_date  149116 non-null  datetime64[ns]
 2   transaction_time  149116 non-null  datetime64[ns]
 3   store_id          149116 non-null  int64         
 4   store_location    149116 non-null  object        
 5   product_id        149116 non-null  int64         
 6   transaction_qty   149116 non-null  int64         
 7   unit_price        149116 non-null  float64       
 8   Total_Bill        149116 non-null  float64       
 9   product_category  149116 non-null  object        
 10  product_type      149116 non-null  object        
 11  product_detail    149116 non-null  object        
 12  Size              149116 non-null  object        
 13  Month Name        149116 non-null  object        
 14  Day Name          149116 non-null  object        
 15  Hour_of_day       149116 non-null  int64         
 16  Month             149116 non-null  int64         
 17  Day of Week       149116 non-null  int64         
dtypes: datetime64[ns](2), float64(2), int64(7), object(7)
memory usage: 20.5+ MB
```

~~FROM BEAN TO CUP~~

FROM BRAIN TO CODE

STATISTICS

```python
# Display summary statistics
summary_stats = coffee_shop_sales_data.describe()
print("Summary Statistics:")
print(summary_stats)
```

```
Summary Statistics:
       transaction_id       store_id     product_id  transaction_qty  \
count   149116.000000  149116.000000  149116.000000    149116.000000
mean     74737.371872       5.342063      47.918607         1.438276
std      43153.600016       2.074241      17.930020         0.542509
min          1.000000       3.000000       1.000000         1.000000
25%      37335.750000       3.000000      33.000000         1.000000
50%      74727.500000       5.000000      47.000000         1.000000
75%     112094.250000       8.000000      60.000000         2.000000
max     149456.000000       8.000000      87.000000         8.000000

          unit_price     Total_Bill    Hour_of_day          Month  \
count  149116.000000  149116.000000  149116.000000  149116.000000
mean        3.382219       4.686367      11.735790       3.988881
std         2.658723       4.227099       3.764662       1.673091
min         0.800000       0.800000       6.000000       1.000000
25%         2.500000       3.000000       9.000000       3.000000
50%         3.000000       3.750000      11.000000       4.000000
75%         3.750000       6.000000      15.000000       5.000000
max        45.000000     360.000000      20.000000       6.000000

         Day of Week  calculated_total_bill
count  149116.000000          149116.000000
mean        2.982336               4.686367
std         1.996650               4.227099
min         0.000000               0.800000
25%         1.000000               3.000000
50%         3.000000               3.750000
75%         5.000000               6.000000
max         6.000000             360.000000
```

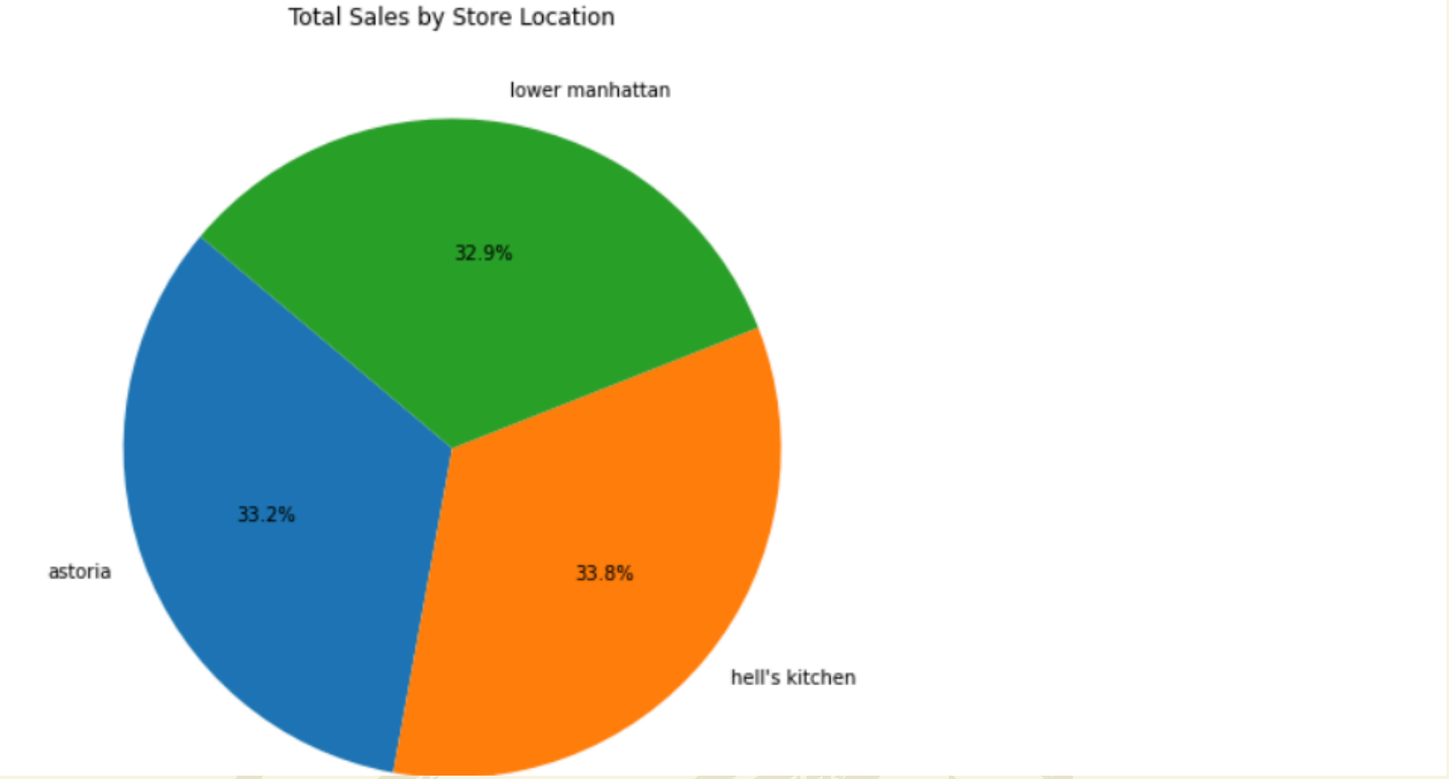
LEADING STORE LOCATION BY SALES VOLUME

```
#highest sales and its location
location_sales = coffee_shop_sales_data.groupby('store_location')['Total_Bill'].sum().sort_values(ascending = False)
highest_sales_location = location_sales.idxmax()
highest_sales_amount = location_sales.max()
#displaying the result
print("Total Sales by Location:")
print(location_sales)
print(f"\nLocation with the Highest Sales: {highest_sales_location}")
print(f"Highest Sales Amount: {highest_sales_amount}")
```

```
Total Sales by Location:
store_location
hell's kitchen     236511.17
astoria            232243.91
lower manhattan    230057.25
Name: Total_Bill, dtype: float64

Location with the Highest Sales: hell's kitchen
Highest Sales Amount: 236511.17
```

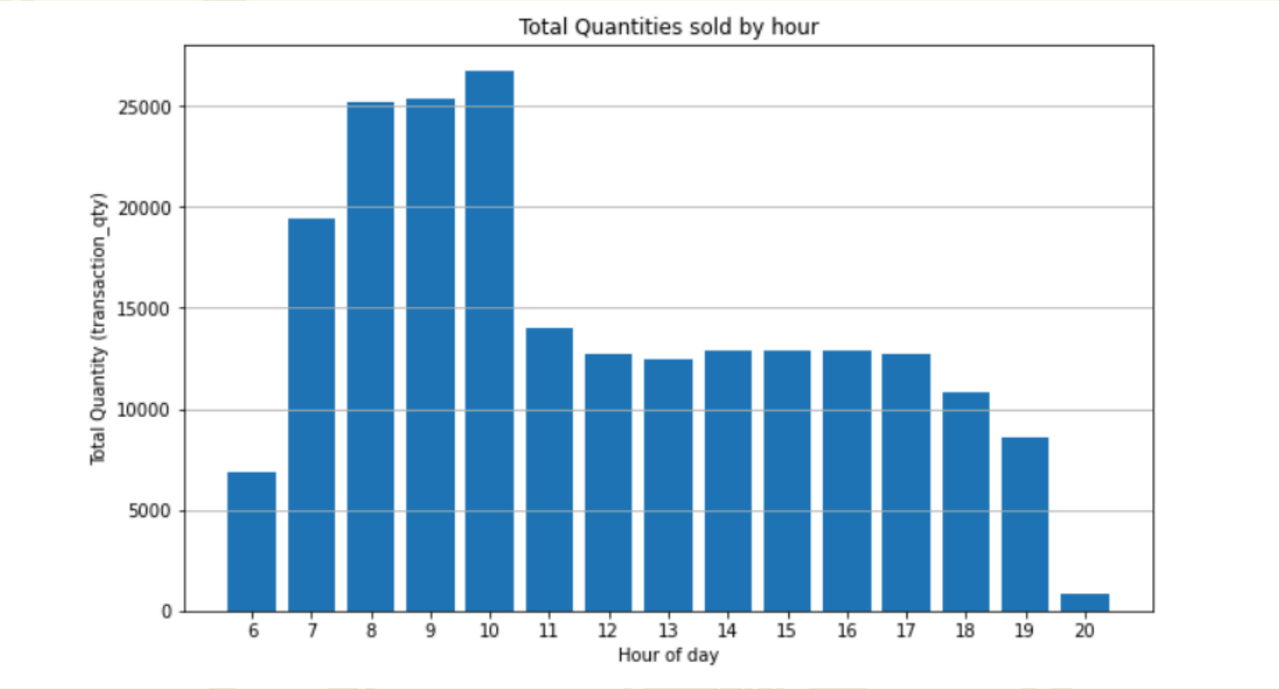
PIE CHART

```
# Highest sales by location data for pie chart
location_sales = coffee_shop_sales_data.groupby('store_location')['Total_Bill'].sum()

# Plotting the pie chart for sales by location
plt.figure(figsize=(8, 8))
plt.pie(location_sales, labels=location_sales.index, autopct='%1.1f%%', startangle=140)
plt.title('Total Sales by Store Location')
plt.show()
```



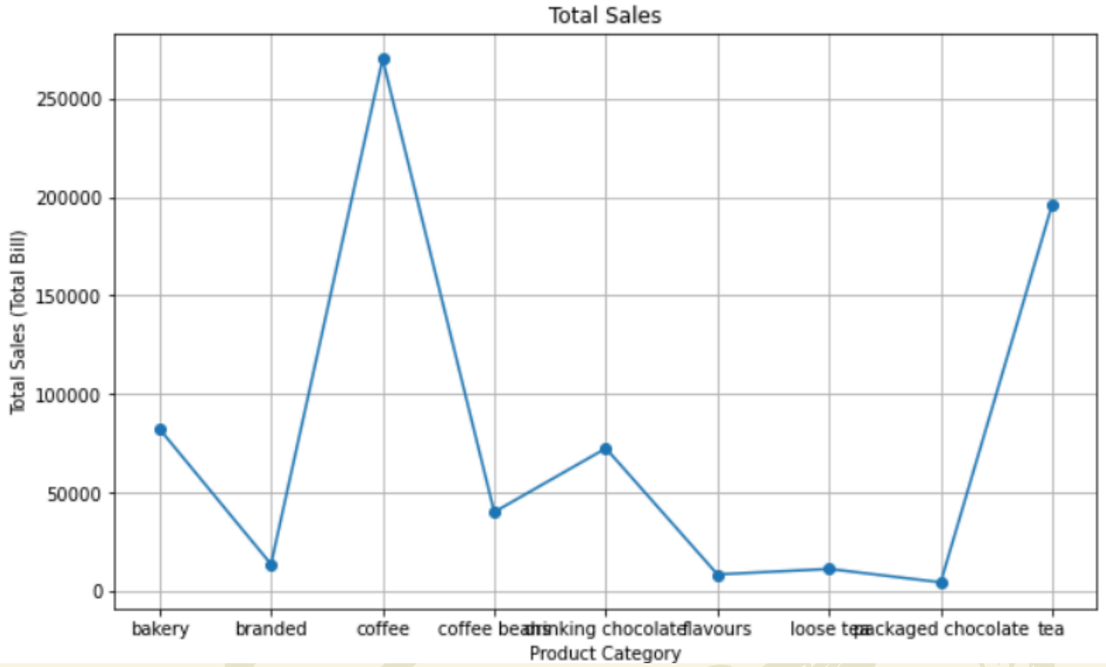
PEAK SALES HOURS

```
#quantities sold by hour
sales_trend_by_day = coffee_shop_sales_data.groupby('Hour_of_day')['transaction_qty'].sum().sort_values(ascending = False)

# Total quantities sold by hour
sales_trend_by_day = coffee_shop_sales_data.groupby('Hour_of_day')['transaction_qty'].sum().sort_index()

# Plotting the bar chart for quantities sold by month
plt.figure(figsize=(10, 6))
plt.bar(sales_trend_by_day.index, sales_trend_by_day.values)
plt.title('Total Quantities sold by hour')
plt.xlabel('Hour of day')
plt.ylabel('Total Quantity (transaction_qty)')
plt.xticks(sales_trend_by_day.index)  # Ensure all hours are labeled on the x-axis
plt.grid(axis='y')
plt.show()
```



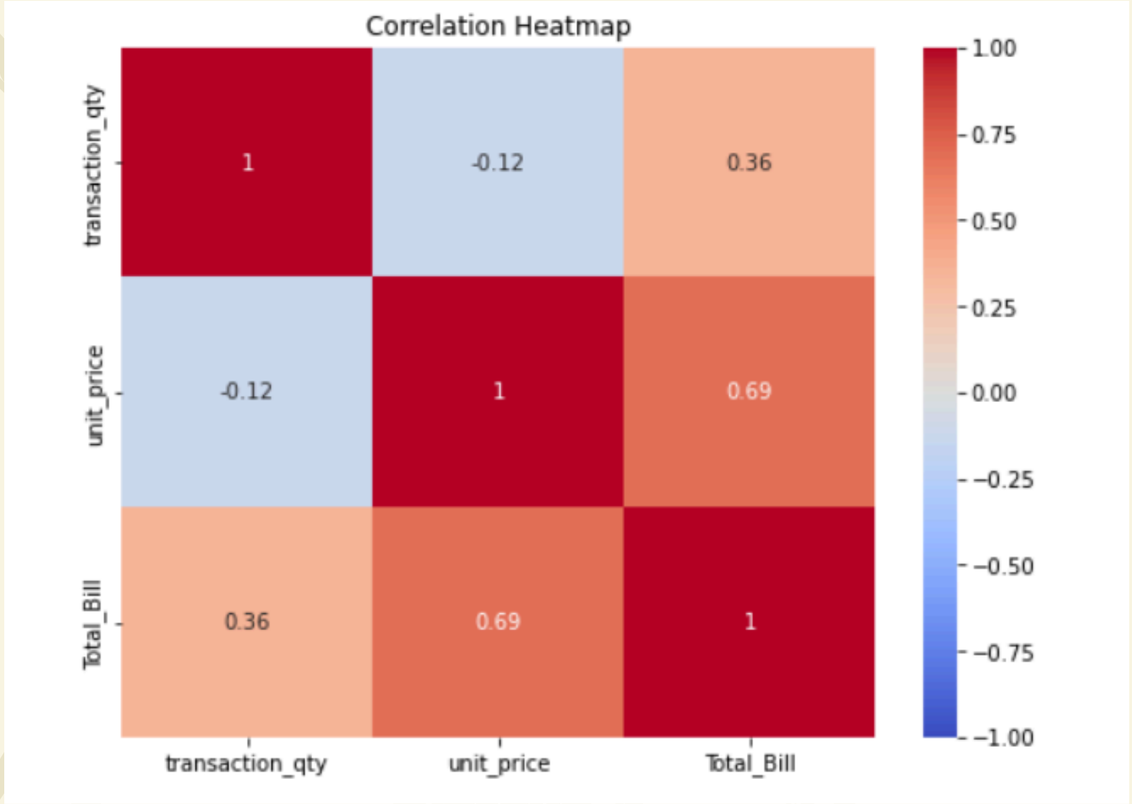
TOP-SELLING PRODUCT CATEGORIES

```
# Total Product sales
product_category_sales = coffee_shop_sales_data.groupby('product_category')['Total_Bill'].sum().sort_index()
# Plotting the line chart for total monthly bill
plt.figure(figsize=(10, 6))
plt.plot(product_category_sales.index, product_category_sales.values, marker='o')
plt.title('Total Sales')
plt.xlabel('Product Category')
plt.ylabel('Total Sales (Total Bill)')
plt.xticks(product_category_sales.index)  # Ensure all product categories are labeled on the x-axis
plt.grid(True)
plt.show()
```



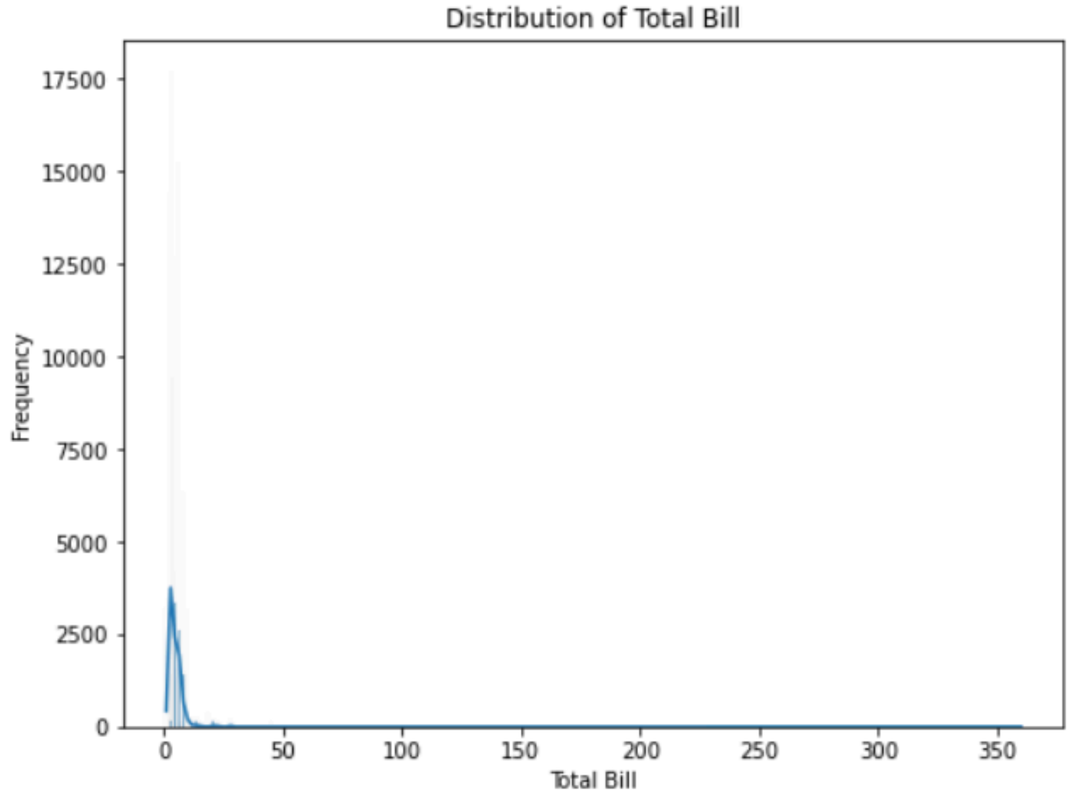
CORRELATION HEATMAP: TRANSACTION QUANTITY, TOTAL BILL & UNIT PRICE

```
# CORRELATION HEATMAP
PLT.FIGURE(FIGSIZE=(8, 6))
SNS.HEATMAP(COFFEE_SHOP_SALES_DATA[['TRANSACTION_QTY', 'UNIT_PRICE', 'TOTAL_BILL']].CORR(), ANNOT=TRUE,
CMAP='COOLWARM', VMIN=-1, VMAX=1)
PLT.TITLE('CORRELATION HEATMAP')
PLT.SHOW()
```



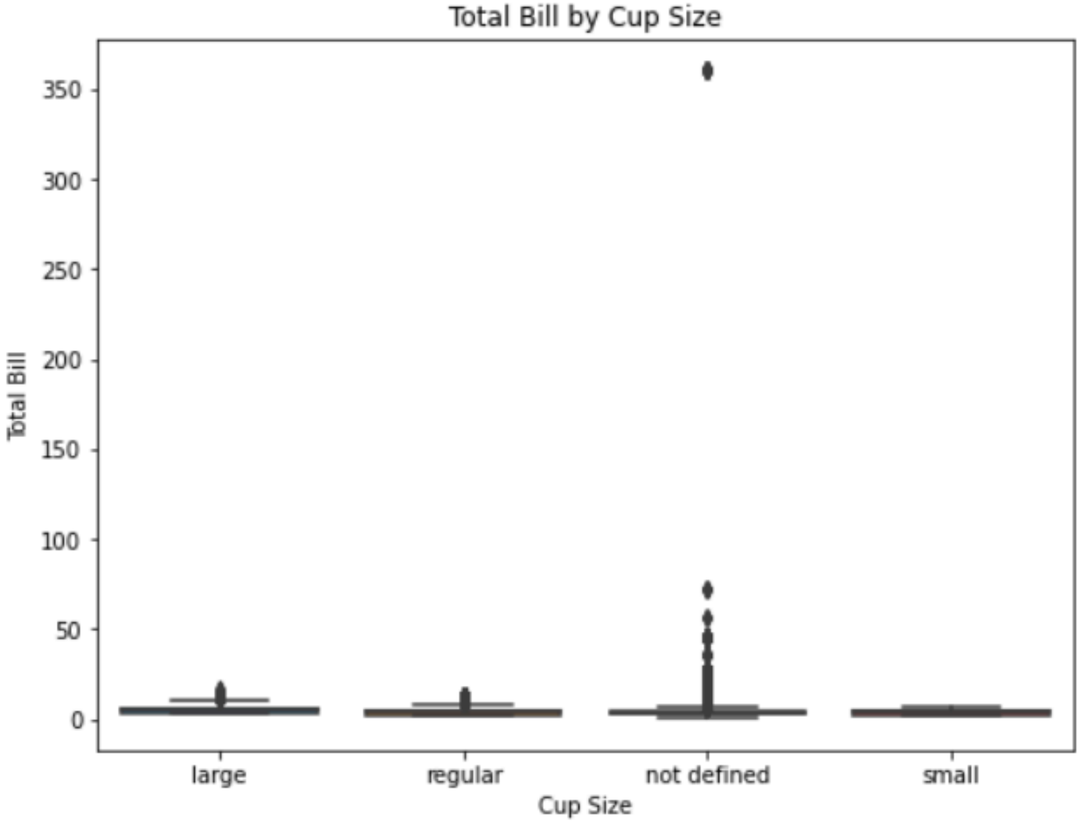
ANALYZING THE SPREAD OF TOTAL BILL AMOUNTS

```
# Distribution plot of Total Bill
plt.figure(figsize=(8, 6))
sns.histplot(coffee_shop_sales_data['Total_Bill'], kde=True)
plt.title('Distribution of Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Frequency')
plt.show()
```



DISTRIBUTION OF TOTAL BILL BY PRODUCT SIZE

```
# Boxplot for Total Bill by Size
plt.figure(figsize=(8, 6))
sns.boxplot(data=coffee_shop_sales_data, x='Size', y='Total_Bill')
plt.title('Total Bill by Cup Size')
plt.xlabel('Cup Size')
plt.ylabel('Total Bill')
plt.show()
```



MORE THAN JUST AN DRINK ANALYSIS

RESEARCH QUESTION: DOES THE LOCATION OF A STORE SIGNIFICANTLY INFLUENCE THE TOTAL BILL OF CUSTOMERS?

Independent variable:
Store Location

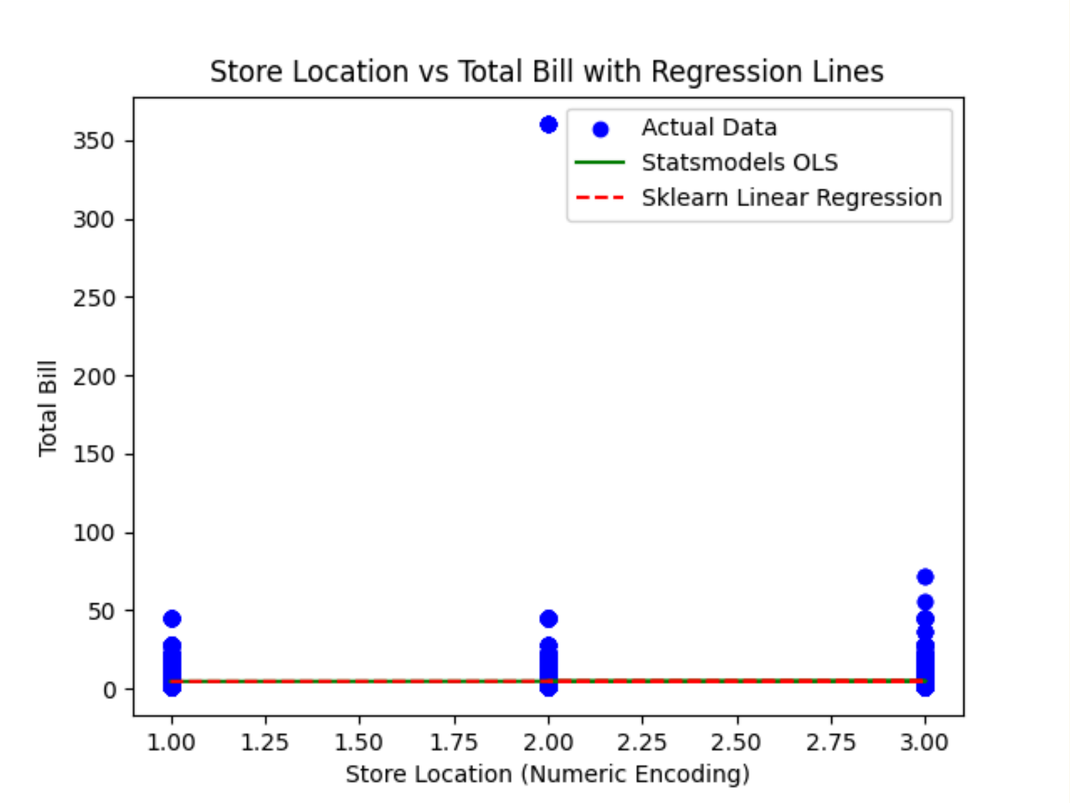
Dependent variable:
Total Bill

Hypothesis:
Ho: The store location has an impact on the total amount spent by customers.

Ha: The store location does not have an impact on the total amount spent by customers.

```
Statsmodels OLS Regression Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:             Total_Bill   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     69.09
Date:                Tue, 12 Nov 2024   Prob (F-statistic):           9.50e-17
Time:                        14:59:17   Log-Likelihood:            -4.2650e+05
No. Observations:              149116   AIC:                         8.530e+05
Df Residuals:                  149114   BIC:                         8.530e+05
Df Model:                           1
Covariance Type:            nonrobust
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
store_location  0.1120      0.013      8.312      0.000       0.086       0.138
==============================================================================
Omnibus:                   440358.933   Durbin-Watson:                   0.310
Prob(Omnibus):                  0.000   Jarque-Bera (JB):      70298492849.781
Skew:                          41.770   Prob(JB):                        0.00
Kurtosis:                    3365.657   Cond. No.                        6.73
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Mean Squared Error (sklearn): 18.105116562529723
```

SCATTER PLOT



BOX PLOT

RECOMENDATIONS

Sales Performance

- Schedule the most qualified baristas during peak hours.
- Set cross-training during off-peak hours.
- Schedule part-time staff to work during off-peak hours.
- Set combos for hot beverage and pastries with a discount during low-month sales.

1

2

3

4

RECOMENDATIONS

Product

- Product bundles with popular products.
- Visibles spots for lower-consumed products.

1

2

RECOMMENDATIONS

Location

- Plan events as workshops, gatherings for tasting new products
- 'Visit our 3 locations and we will be giving 20% of discount on a regular coffee
- Sell striking merchandise, only on this location.

1

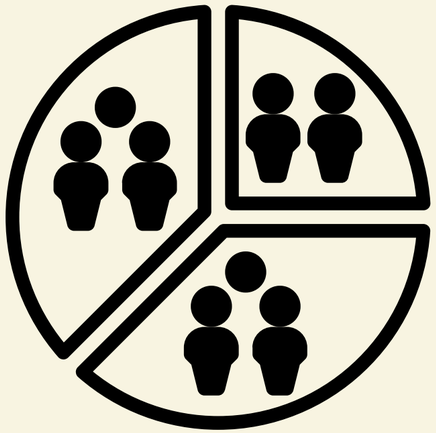
2

3

RECOMMENDATIONS

On the other hand, for the advance modeling :

- Add more variables
- Improve data quality
- Segment the analysis



CONCLUSION

Key insights from our analysis:

- Customer spending patterns, peak sales hours, and top-selling product categories.

By identifying correlations in transaction data and evaluating sales volume by location, we gained an understanding of factors that influence our revenue.

Recommended targeted actions:

- More optimized staffing, product bundling, and community engagement initiatives.

These are tailored to maximize customer satisfaction and profitability.

Refining our data model with additional variables will enhance the precision of our insights and drive more effective business decisions in the future.


TIME FOR A COFFEE!