Import Some Import Library

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

```
#Load data
df=pd.read_excel(r"C:\Users\pratik rao\OneDrive\Desktop\Coffee Sales Project in Excel\coffee_shop_sales.xlsx")
```

```
#view data
df.head()
```

	transaction_id	transaction_date	transaction_time	store_id	store_location	product_id	transaction_qty	unit_price	product_category	product_type	product_detail	Size	Total_bill	Month Name	
0	114301	2023-06-01	11:33:29	3	Astoria	45	1	3.0	Tea	Brewed herbal tea	Peppermint	Large	3	June	Thur
1	115405	2023-06-02	11:18:24	3	Astoria	45	1	3.0	Tea	Brewed herbal tea	Peppermint	Large	3	June	F
2	115478	2023-06-02	12:02:45	3	Astoria	45	1	3.0	Tea	Brewed herbal tea	Peppermint	Large	3	June	F
3	116288	2023-06-02	19:39:47	3	Astoria	45	1	3.0	Tea	Brewed herbal tea	Peppermint	Large	3	June	F
4	116714	2023-06-03	12:24:57	3	Astoria	45	1	3.0	Tea	Brewed herbal tea	Peppermint	Large	3	June	Satu

```
#Get the information about the Data
df.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  object
 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   product_category  149116 non-null  object
 9   product_type      149116 non-null  object
 10  product_detail    149116 non-null  object
 11  Size              149116 non-null  object
 12  Total_bill        149116 non-null  int64
 13  Month Name        149116 non-null  object
 14  Day Name          149116 non-null  object
```

```
 15  Hour              149116 non-null  int64
 16  Day of Week       149116 non-null  int64
 17  Month             149116 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(8), object(8)
memory usage: 20.5+ MB
```

```
#describe table
df.describe()
```

	transaction_id	store_id	product_id	transaction_qty	unit_price	Total_bill	Hour	Day of Week	Month
count	149116.000000	149116.000000	149116.000000	149116.000000	149116.000000	149116.0	149116.000000	149116.00000	149116.000000
mean	74737.371872	5.342063	47.918607	1.438276	3.382219	3.0	11.735790	2.99202	3.988881
std	43153.600016	2.074241	17.930020	0.542509	2.658723	0.0	3.764662	1.99028	1.673091
min	1.000000	3.000000	1.000000	1.000000	0.800000	3.0	6.000000	0.00000	1.000000
25%	37335.750000	3.000000	33.000000	1.000000	2.500000	3.0	9.000000	1.00000	3.000000
50%	74727.500000	5.000000	47.000000	1.000000	3.000000	3.0	11.000000	3.00000	4.000000
75%	112094.250000	8.000000	60.000000	2.000000	3.750000	3.0	15.000000	5.00000	5.000000
max	149456.000000	8.000000	87.000000	8.000000	45.000000	3.0	20.000000	6.00000	6.000000

```
#view the all columns name in this table
df.columns
```

```
Index(['transaction_id', 'transaction_date', 'transaction_time', 'store_id',
       'store_location', 'product_id', 'transaction_qty', 'unit_price',
       'product_category', 'product_type', 'product_detail', 'Size',
       'Total_bill', 'Month Name', 'Day Name', 'Hour', 'Day of Week', 'Month'],
      dtype='object')
```

```
#view the shape of table
df.shape
```

```
(149116, 18)
```

```
#find the record where transaction_id=116714
df.loc[df.transaction_id==116714]
```

	transaction_id	transaction_date	transaction_time	store_id	store_location	product_id	transaction_qty	unit_price	product_category	product_type	product_detail	Size	Total_bill	Month Name	
4	116714	2023-06-03	12:24:57	3	Astoria	45	1	3.0	Tea	Brewed herbal tea	Peppermint	Large	3	June	Satu

```
#Filter the data based on columns
df.iloc[10:21,:6]
```

	transaction_id	transaction_date	transaction_time	store_id	store_location	product_id
10	118428	2023-06-04	17:53:30	3	Astoria	45
11	118913	2023-06-05	12:01:03	3	Astoria	45
12	119196	2023-06-05	14:19:59	3	Astoria	45
13	119240	2023-06-05	14:42:16	3	Astoria	45
14	119351	2023-06-05	15:51:35	3	Astoria	45
15	119444	2023-06-05	16:52:40	3	Astoria	45
16	119692	2023-06-05	19:51:09	3	Astoria	45
17	121279	2023-06-07	10:16:40	3	Astoria	45
18	121416	2023-06-07	10:59:19	3	Astoria	45
19	121493	2023-06-07	12:01:18	3	Astoria	45
20	121656	2023-06-07	15:05:47	3	Astoria	45

```
# Chick null Value
df.isna().sum()
```

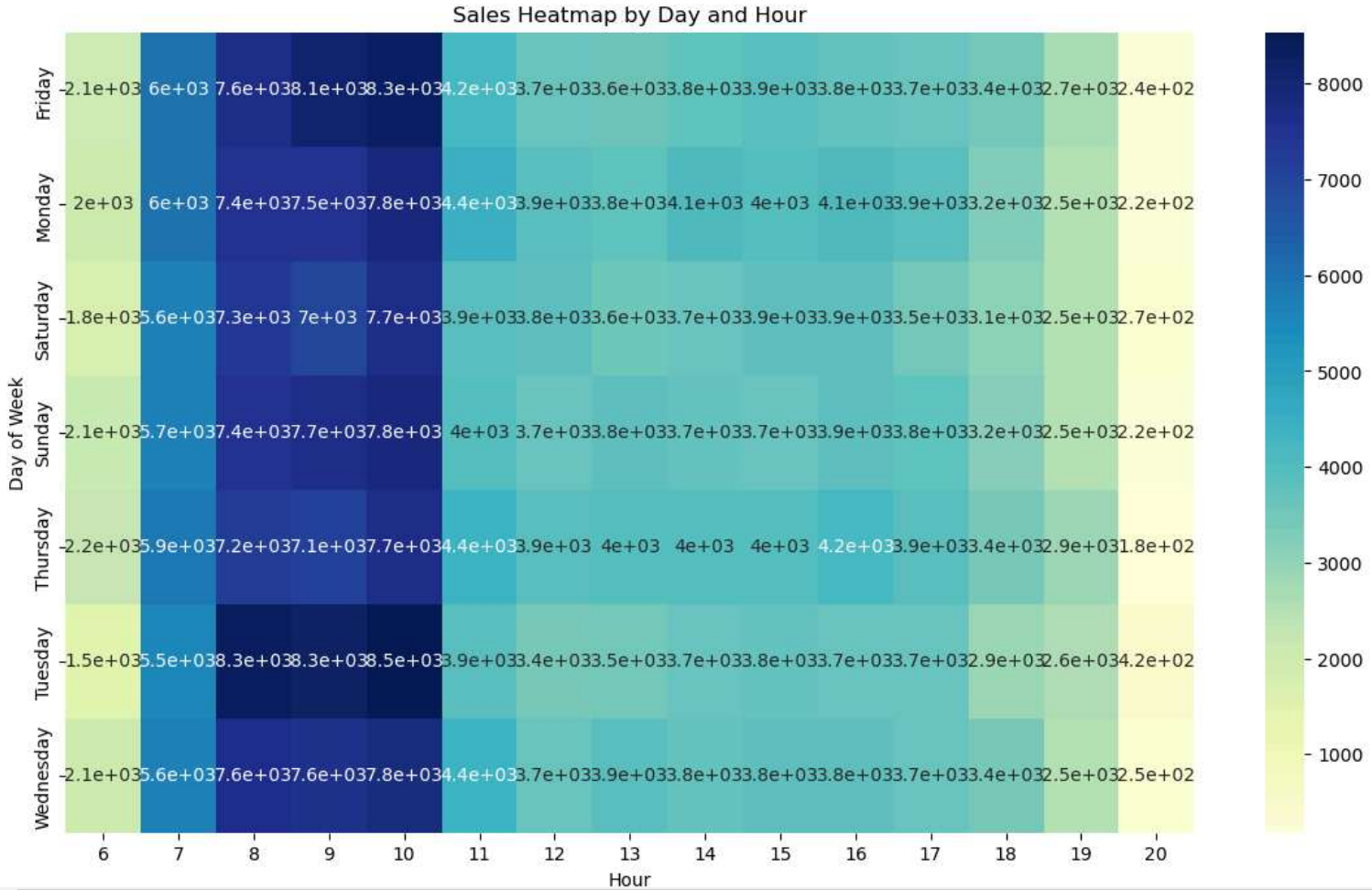
```
transaction_id      0
transaction_date    0
transaction_time    0
store_id            0
store_location      0
product_id          0
transaction_qty     0
unit_price          0
product_category    0
product_type        0
product_detail      0
Size                0
Total_bill          0
Month Name          0
Day Name            0
Hour                0
Day of Week         0
Month               0
dtype: int64
```

```
#view the data
print(df.info())
print(df.head())
```

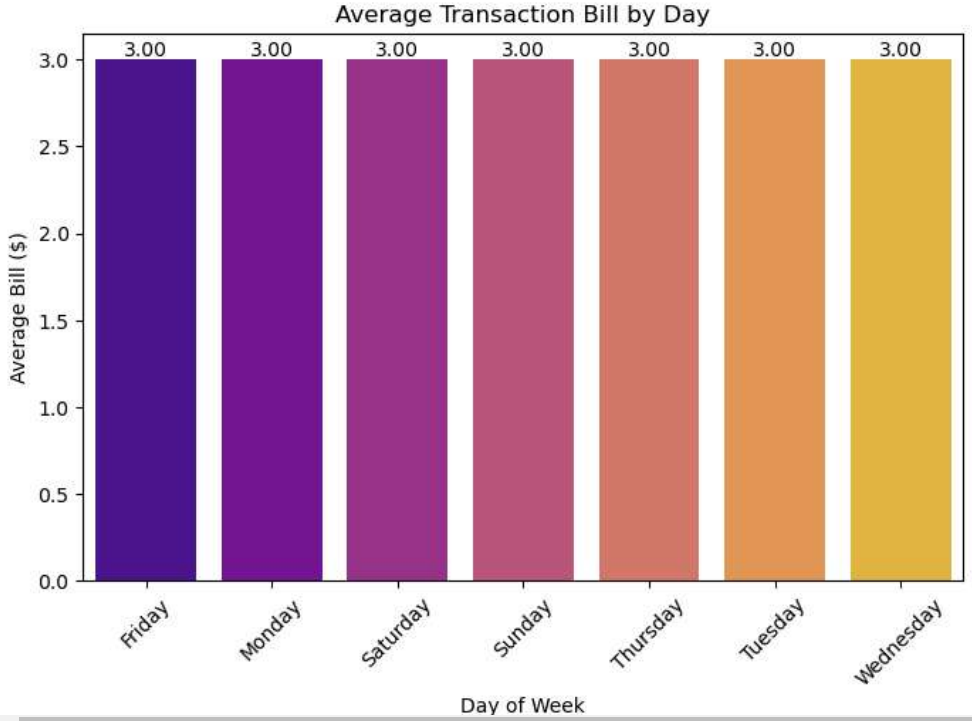
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149116 entries, 0 to 149115
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   transaction_id    149116 non-null  int64
 1   transaction_date  149116 non-null  datetime64[ns]
 2   transaction_time  149116 non-null  object
 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   product_category  149116 non-null  object
 9   product_type      149116 non-null  object
 10  product_detail    149116 non-null  object
 11  Size              149116 non-null  object
 12  Total_bill        149116 non-null  int64
 13  Month Name        149116 non-null  object
 14  Day Name          149116 non-null  object
 15  Hour              149116 non-null  int64
 16  Day of Week       149116 non-null  int64
 17  Month             149116 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(8), object(8)
memory usage: 20.5+ MB
None
   transaction_id transaction_date transaction_time  store_id store_location  \
0          114301       2023-06-01         11:33:29         3        Astoria
1          115405       2023-06-02         11:18:24         3        Astoria
2          115478       2023-06-02         12:02:45         3        Astoria
3          116288       2023-06-02         19:39:47         3        Astoria
4          116714       2023-06-03         12:24:57         3        Astoria

   product_id  transaction_qty  unit_price product_category  \
0          45                1         3.0              Tea
1          45                1         3.0              Tea
2          45                1         3.0              Tea
3          45                1         3.0              Tea
4          45                1         3.0              Tea

         product_type product_detail   Size  Total_bill Month Name  Day Name  \
0  Brewed herbal tea     Peppermint  Large           3       June  Thursday
1  Brewed herbal tea     Peppermint  Large           3       June    Friday
2  Brewed herbal tea     Peppermint  Large           3       June    Friday
3  Brewed herbal tea     Peppermint  Large           3       June    Friday
4  Brewed herbal tea     Peppermint  Large           3       June  Saturday

   Hour  Day of Week  Month
0    11            4      6
1    11            5      6
2    12            5      6
3    19            5      6
4    12            6      6
```

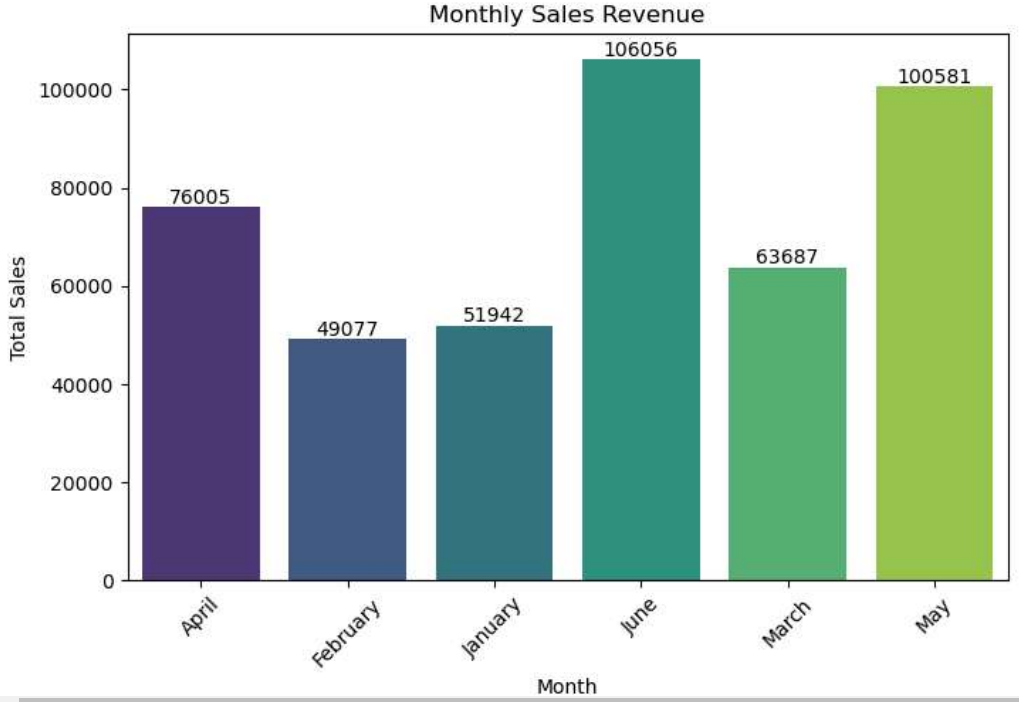
```
# Temporal Sales Analysis: Sales by Day and Hour
sales_by_day_hour = df.groupby(['Day Name', 'Hour'])['Total_bill'].sum().reset_index()
plt.figure(figsize=(14, 8))
sns.heatmap(sales_by_day_hour.pivot('Day Name', 'Hour', 'Total_bill'), cmap='YlGnBu', annot=True)
plt.title('Sales Heatmap by Day and Hour')
plt.xlabel('Hour')
plt.ylabel('Day of Week')
plt.show()
```



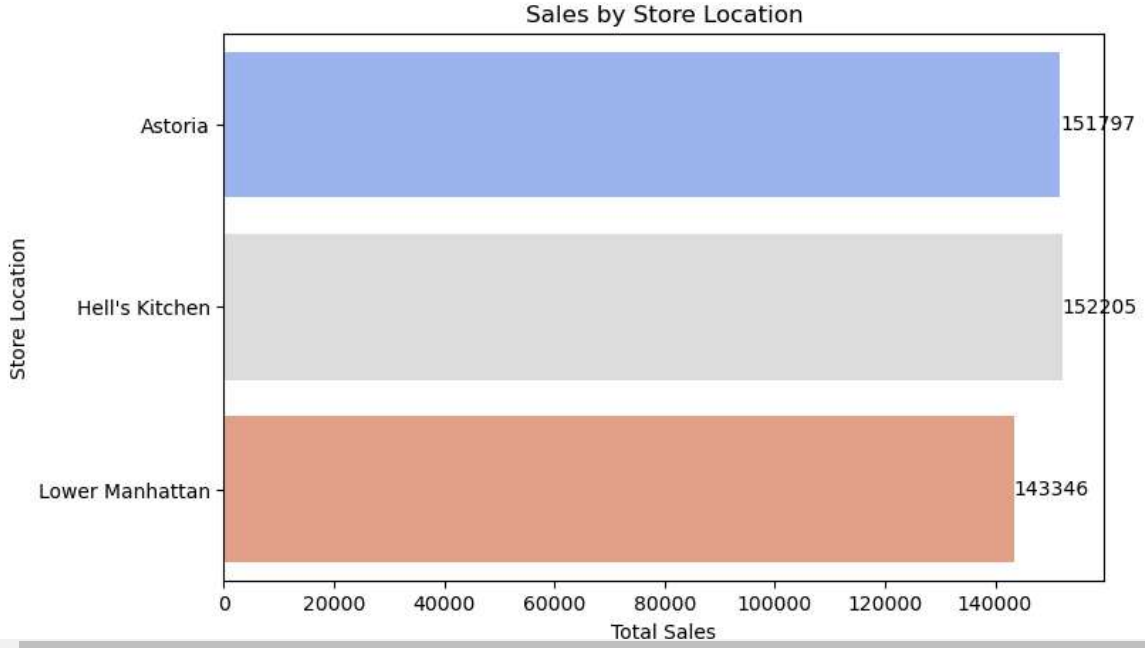

```
# Average Price Per Transaction by Day with Data Labels
avg_price_per_day = df.groupby('Day Name')['Total_bill'].mean().reset_index()
plt.figure(figsize=(8, 5))
ax = sns.barplot(x='Day Name', y='Total_bill', data=avg_price_per_day, palette='plasma')

# Adding data labels
for i in ax.containers:
    ax.bar_label(i, fmt='%.2f', label_type='edge')  # '%.2f' for two decimal places

plt.title('Average Transaction Bill by Day')
plt.xlabel('Day of Week')
plt.ylabel('Average Bill ($)')
plt.xticks(rotation=45)
plt.show()
```




```
# Monthly Revenue Analysis with Data Labels
monthly_sales = df.groupby('Month Name')['Total_bill'].sum().reset_index()
plt.figure(figsize=(8, 5))
ax = sns.barplot(x='Month Name', y='Total_bill', data=monthly_sales, palette='viridis')
# Adding data labels
for i in ax.containers:
    ax.bar_label(i, fmt='%.0f', label_type='edge')  # '%.0f' for integer labels, use '%.2f' for decimals

plt.title('Monthly Sales Revenue')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```



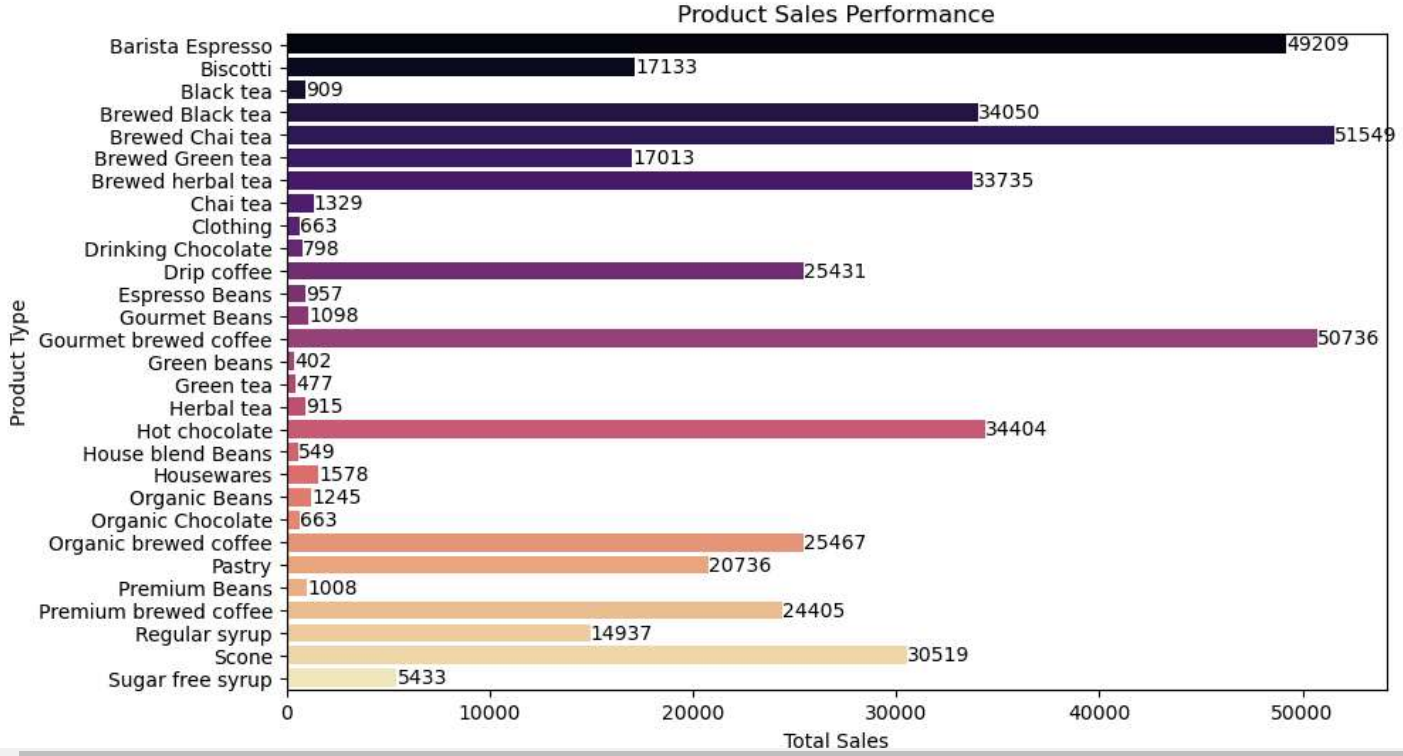

```
# Location-based Sales Comparison with Data Labels
location_sales = df.groupby('store_location')['Total_bill'].sum().reset_index()
plt.figure(figsize=(8,5))
ax = sns.barplot(x='Total_bill', y='store_location', data=location_sales, palette='coolwarm')

# Adding data labels
for i in ax.containers:
    ax.bar_label(i, fmt='%.0f', label_type='edge')  # '%.0f' for integer labels

plt.title('Sales by Store Location')
plt.xlabel('Total Sales')
plt.ylabel('Store Location')
plt.show()
```



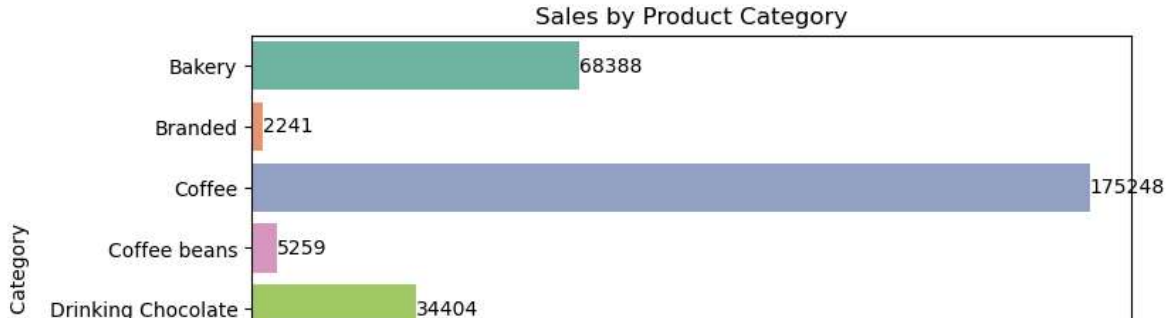

```
# Customer Spending Habits
average_order_value = df['Total_bill'].mean()
print(f'Average Order Value: ${average_order_value:.2f}')
```

```
Average Order Value: $3.00
```

```
# Product Performance Analysis with Data Labels
product_sales = df.groupby('product_type')['Total_bill'].sum().reset_index()
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='Total_bill', y='product_type', data=product_sales, palette='magma')

# Adding data labels
for i in ax.containers:
    ax.bar_label(i, fmt='%.0f', label_type='edge')  # '%.0f' for integer labels

plt.title('Product Sales Performance')
plt.xlabel('Total Sales')
plt.ylabel('Product Type')
plt.show()
```




```
# Category-wise Sales Distribution with Data Labels
category_sales = df.groupby('product_category')['Total_bill'].sum().reset_index()
plt.figure(figsize=(8, 5))
ax = sns.barplot(x='Total_bill', y='product_category', data=category_sales, palette='Set2')

# Adding data labels
for i in ax.containers:
    ax.bar_label(i, fmt='%.0f', label_type='edge')  # '%.0f' for integer labels

plt.title('Sales by Product Category')
plt.xlabel('Total Sales')
plt.ylabel('Product Category')
plt.show()
```




```
# Specific Analysis Tasks
# Total Sales for Brewed Herbal Tea
herbal_tea_sales = df[df['product_type'] == 'Brewed herbal tea']['Total_bill'].sum()
print(f'Total Sales for Brewed Herbal Tea: ${herbal_tea_sales:.2f}')
```

```
Total Sales for Brewed Herbal Tea: $33735.00
```

```
# Count of Transactions Per Store
transactions_per_store = df['store_location'].value_counts()
print(transactions_per_store)
```

```
Hell's Kitchen     50735
Astoria            50599
Lower Manhattan    47782
Name: store_location, dtype: int64
```

```
# Transactions Above $5
high_value_transactions = df[df['Total_bill'] > 5]
print(f'Transactions above $5: {len(high_value_transactions)}')
```

```
Transactions above $5: 0
```

Report

Coffee Shop Sales Analysis Report

1. Introduction

This report presents an analysis of coffee shop sales data to provide actionable insights for optimizing sales performance. The analysis covers temporal sales trends, monthly revenue, location-based comparisons, customer spending habits, product performance, and category-wise sales distribution.

2. Temporal Sales Analysis

Insight: Sales vary significantly by day and hour, with peak hours observed during mornings and early evenings. This suggests the need for optimized staff scheduling during peak times.

3. Monthly Revenue Analysis