Introduction

Coffee shops generate more than just great coffee; they generate a wealth of data. In this analysis, we explore the daily revenue of a coffee shop, uncovering hidden insight and building a prediction model to understand what drives revenue.

Table of Contents

In [5]:
```
# Import required libraries and suppress warnings
import warnings
warnings.filterwarnings('ignore')

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

# Set default aesthetics for seaborn
sns.set(style='whitegrid', palette='muted', color_codes=True)
```

Data Import and Setup

In this section, we load the coffee shop daily revenue data. The dataset includes several features that describe customer behavior and operational factors,which we will later use for exploratory analysis and predictiong modeling.

In [10]:
```
# Load the dataset
df = pd.read_csv(r"C:\Users\chitt\Downloads\coffee_shop_revenue.csv")
df
```

Out[10]:

	Number_of_Customers_Per_Day	Average_Order_Value	Operating_Hours_Per_Day	N
0	152	6.74	14	
1	485	4.50	12	
2	398	9.09	6	
3	320	8.48	17	
4	156	7.44	17	
...	...	...	...	
1995	372	6.41	11	
1996	105	3.01	11	
1997	89	5.28	16	
1998	403	9.41	7	
1999	89	6.88	13	

2000 rows × 7 columns

In [14]:
```
df.shape
```

Out[14]:
```
(2000, 7)
```

In [16]:
```
df.head()
```

Out[16]:

	Number_of_Customers_Per_Day	Average_Order_Value	Operating_Hours_Per_Day	Num
0	152	6.74	14	
1	485	4.50	12	
2	398	9.09	6	
3	320	8.48	17	
4	156	7.44	17	

In [18]:
```
df.tail()
```

Out[18]:

	Number_of_Customers_Per_Day	Average_Order_Value	Operating_Hours_Per_Day	N
1995	372	6.41	11	
1996	105	3.01	11	
1997	89	5.28	16	
1998	403	9.41	7	
1999	89	6.88	13	

In [20]:

```
df.info
```

Out[20]:

```
<bound method DataFrame.info of       Number_of_Customers_Per_Day  Average_Orde
r_Value  \
0                             152                 6.74
1                             485                 4.50
2                             398                 9.09
3                             320                 8.48
4                             156                 7.44
...                           ...                  ...
1995                          372                 6.41
1996                          105                 3.01
1997                           89                 5.28
1998                          403                 9.41
1999                           89                 6.88

      Operating_Hours_Per_Day  Number_of_Employees  Marketing_Spend_Per_Day  \
0                          14                    4                   106.62
1                          12                    8                    57.83
2                           6                    6                    91.76
3                          17                    4                   462.63
4                          17                    2                   412.52
...                       ...                  ...                      ...
1995                       11                    4                   466.11
1996                       11                    7                    12.62
1997                       16                    9                   376.64
1998                        7                   12                   452.49
1999                       13                   14                    78.46

      Location_Foot_Traffic  Daily_Revenue
0                        97        1547.81
1                       744        2084.68
2                       636        3118.39
3                       770        2912.20
4                       232        1663.42
...                     ...            ...
1995                    913        2816.85
1996                    235         337.97
1997                    310         951.34
1998                    577        4266.21
1999                    322         914.24

[2000 rows x 7 columns]>
```

In [22]:

```
df.dtypes
```

Out[22]:
```
Number_of_Customers_Per_Day      int64
Average_Order_Value            float64
Operating_Hours_Per_Day          int64
Number_of_Employees              int64
Marketing_Spend_Per_Day        float64
Location_Foot_Traffic            int64
Daily_Revenue                  float64
dtype: object
```

Exploratory Data Analysis

We start our exploratory analysis by reviewing summary statistics and visualizing the distribution of various features. This includes histograms, box plots, and pair plots to examine relationships between different features and the target variable, Daily_Revenue.

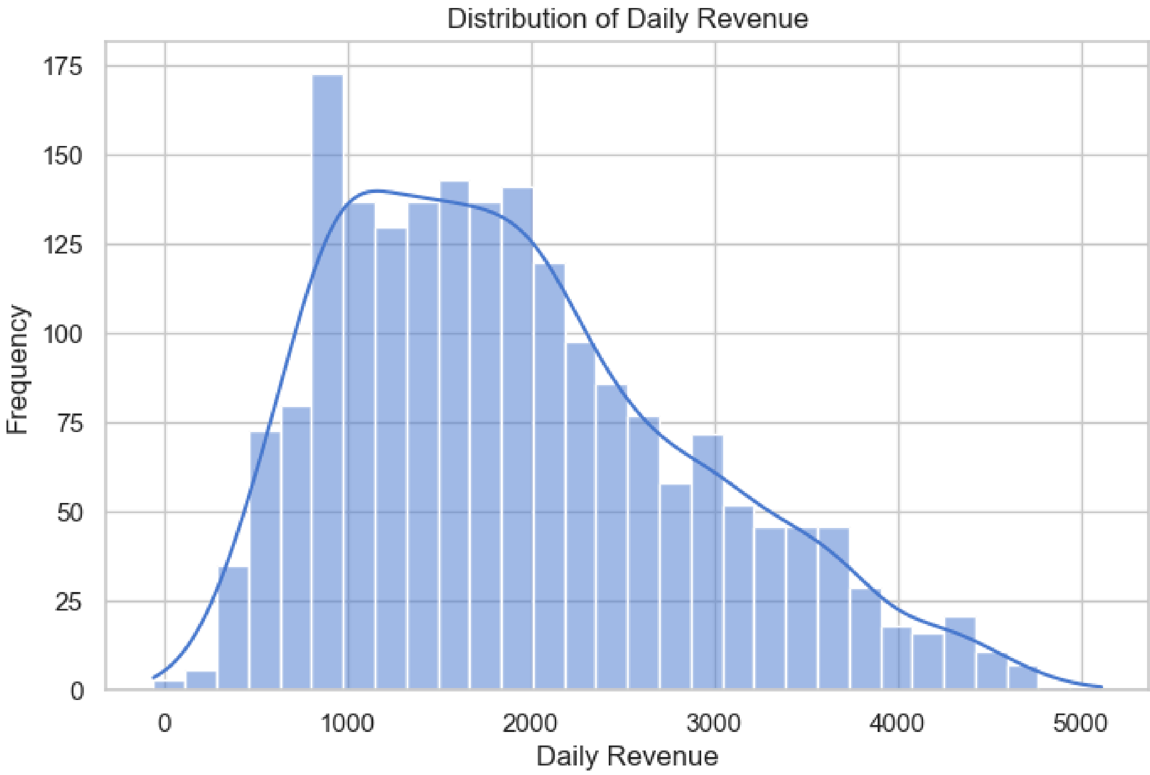
Since we have more than four numeric columns, we will also generate a correlation heatmap to understand the interrelationships between numeric features.

In [25]:
```python
# Quick summary statistics
print(df.describe())
```

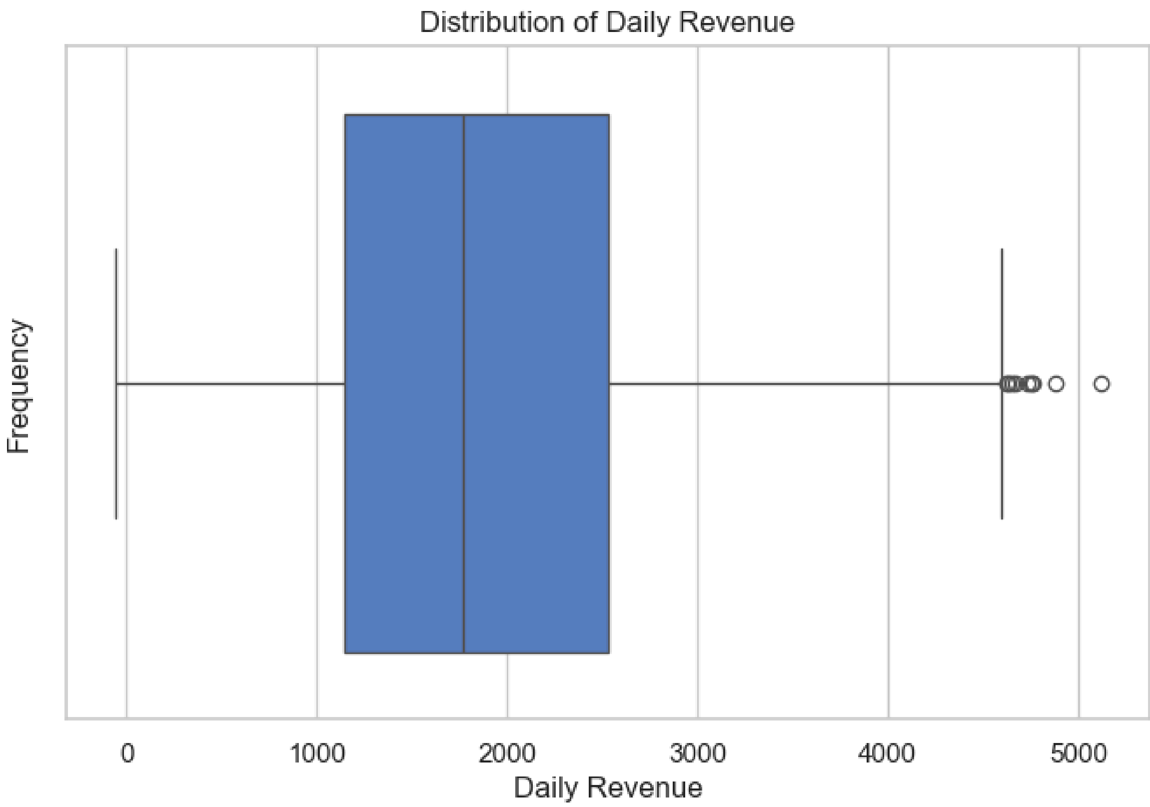
```
       Number_of_Customers_Per_Day  Average_Order_Value  \
count                  2000.000000          2000.000000
mean                    274.296000             6.261215
std                     129.441933             2.175832
min                      50.000000             2.500000
25%                     164.000000             4.410000
50%                     275.000000             6.300000
75%                     386.000000             8.120000
max                     499.000000            10.000000

       Operating_Hours_Per_Day  Number_of_Employees  Marketing_Spend_Per_Day  \
count              2000.000000          2000.000000              2000.000000
mean                 11.667000             7.947000               252.614160
std                   3.438608             3.742218               141.136004
min                   6.000000             2.000000                10.120000
25%                   9.000000             5.000000               130.125000
50%                  12.000000             8.000000               250.995000
75%                  15.000000            11.000000               375.352500
max                  17.000000            14.000000               499.740000

       Location_Foot_Traffic  Daily_Revenue
count            2000.000000    2000.000000
mean              534.893500    1917.325940
std               271.662295     976.202746
min                50.000000     -58.950000
25%               302.000000    1140.085000
50%               540.000000    1770.775000
75%               767.000000    2530.455000
max               999.000000    5114.600000
```

In [27]:
```python
# Distribution of the target variable: Daily_Revenue
plt.figure(figsize=(8,5))
sns.histplot(df['Daily_Revenue'], kde=True, bins=30)
plt.title('Distribution of Daily Revenue')
plt.xlabel('Daily Revenue')
```

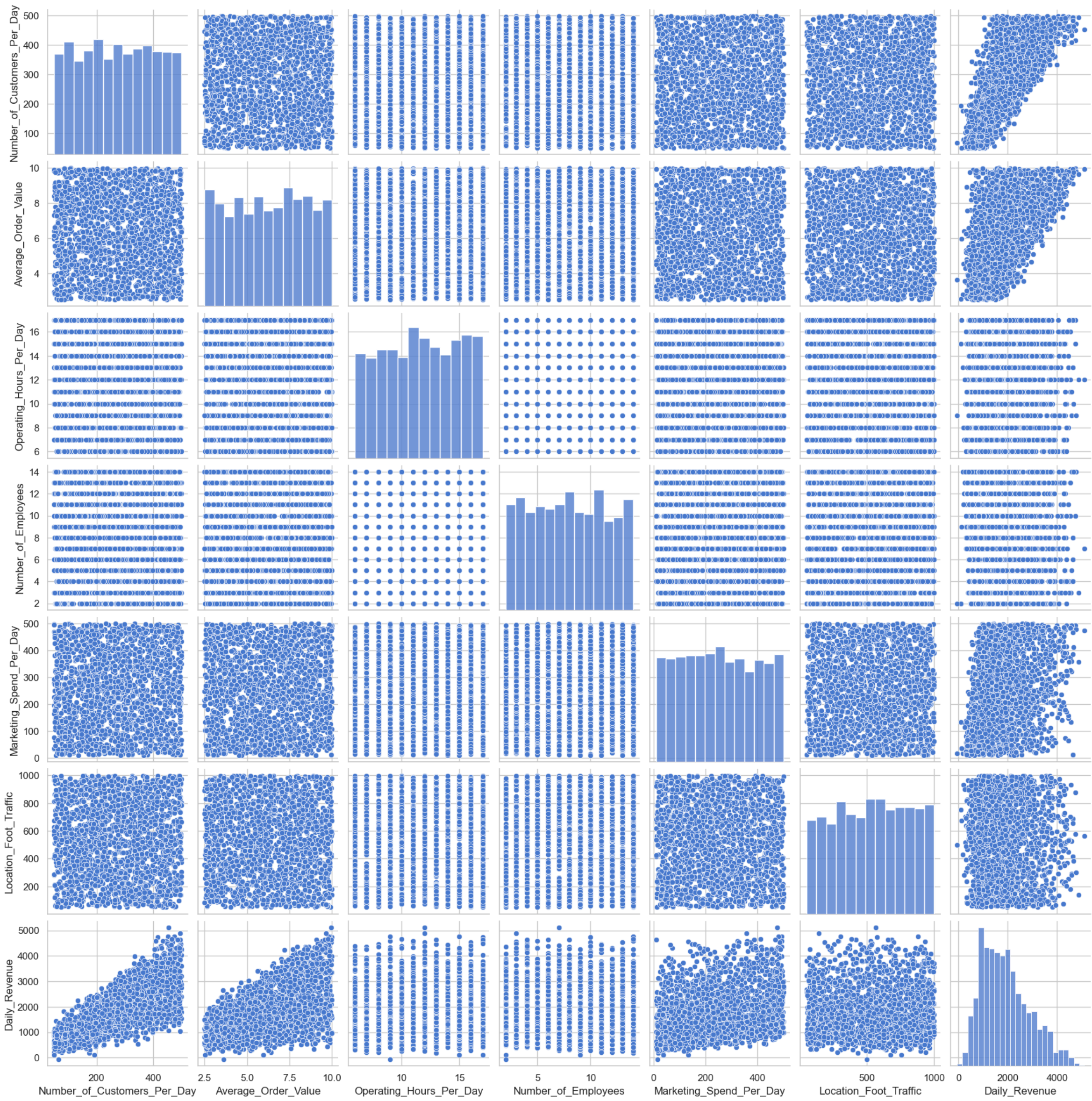
```
plt.ylabel('Frequency')
plt.show()
```



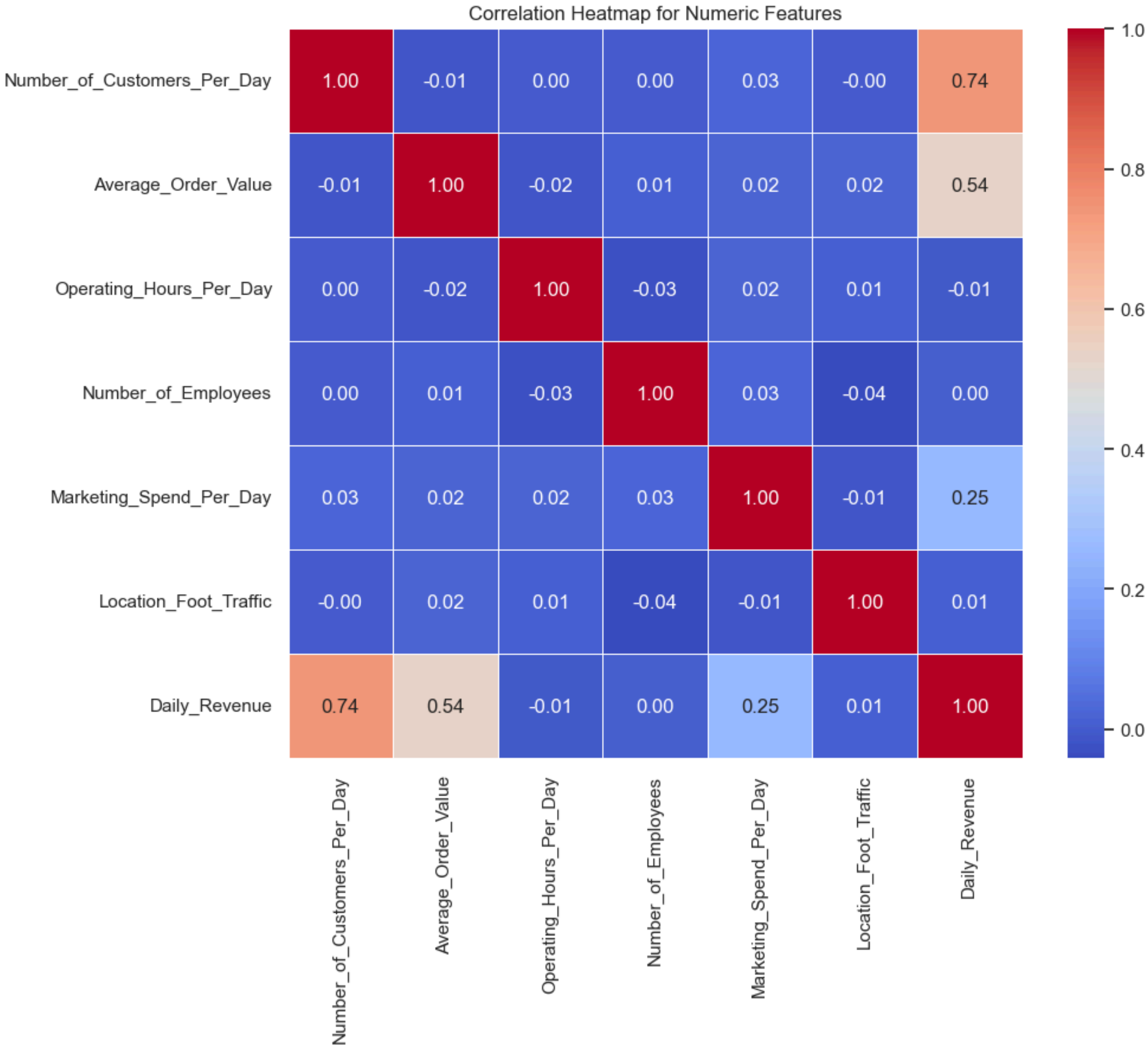
In [29]:
```
# Box plot for Daily_Revenue to inspect outliers
plt.figure(figsize=(8,5))
sns.boxplot(x=df['Daily_Revenue'])
plt.title('Distribution of Daily Revenue')
plt.xlabel('Daily Revenue')
plt.ylabel('Frequency')
plt.show()
```



In [31]:
```
# Pair plot: Exploring pairwise relationships beteen features
sns.pairplot(df)
plt.show()
```



In [33]:
```
# Correlation heatmap: use only numeric columns
numeric_df = df.select_dtypes(include=[np.number])
if numeric_df.shape[1] >= 4:
    plt.figure(figsize=(10, 8))
    corr = numeric_df.corr()
    sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm', linewidths=0.5)
    plt.title('Correlation Heatmap for Numeric Features')
    plt.show()
else:
    print('Not enough numeric columns to display a correlation heatmap.')
```



Data Cleaning and Preprocessing

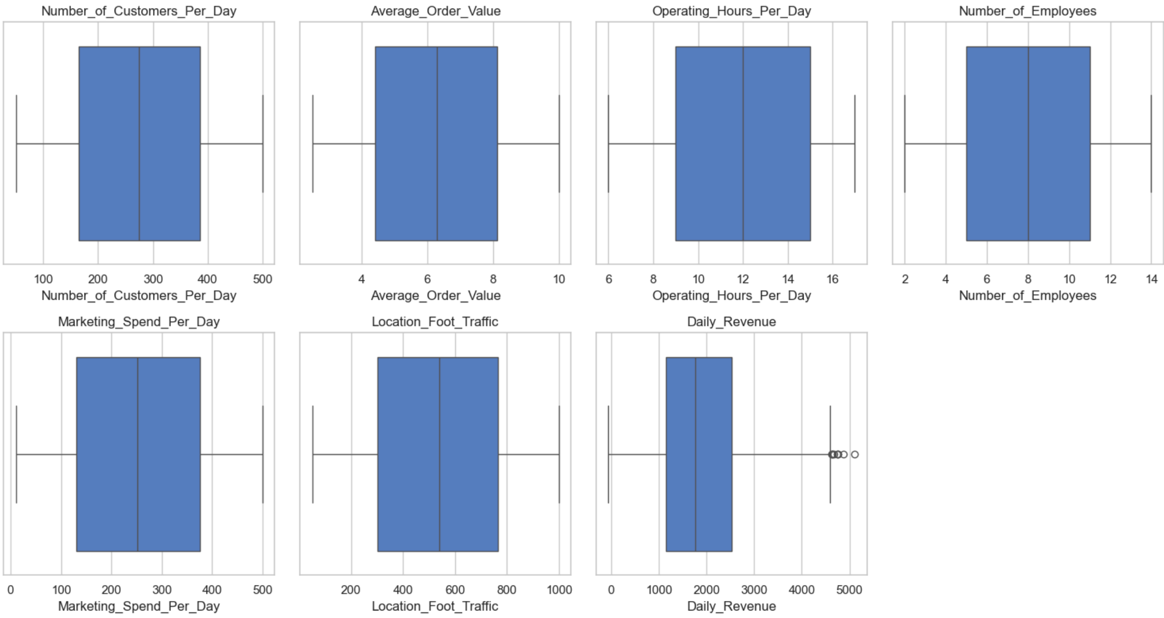
In the Data Cleaning phase, we check for missing values and outliers.For example, if missing values are encountered, we either drop or impute them based on domain knowledge. Errors similsr to encoding issue in CSV files are common and are handled by specifying the correct encoding during file read.

In [40]:
```
# Basic check for missing values
df.isnull().sum()
```

Out[40]:
```
Number_of_Customers_Per_Day    0
Average_Order_Value            0
Operating_Hours_Per_Day        0
Number_of_Employees            0
Marketing_Spend_Per_Day        0
Location_Foot_Traffic          0
Daily_Revenue                  0
dtype: int64
```

In [44]:
```
# Outlier detection for numeric features using box plots
numeric_cols = numeric_df.columns
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols):
    plt.subplot(2, (len(numeric_cols) + 1) // 2, i+1)
    sns.boxplot(x=df[col])
```

```
    plt.title(col)
plt.tight_layout()
plt.show()
```



Feature Engineering

At this stage, the dataset does not require complex feature engineering. However, potential improvements include creating interaction terms(for example, Marketing_Spend_Per_Day per Number_of Customers_per_Day) and normalizing features if needed. Feature version of this notebook could experiment with these techniques.

Predictive Modeling

Now, we build a predictor to forecast Daily_Revenue based on the available features. A Random Forest Regressor is used owing to its robustness and ability to capture non-linear relationships. We perform a train-test split and evaluate the model using metrics such as RMSE and R2 score.

In [48]:
```
# Scikit-learn libraries for modeling
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [50]:
```
# Prepare feature matrix X and target vector y
features = ['Number_of_Customers_Per_Day', 'Average_Order_Value', 'Operating_Hou
            'Number_of_Employees', 'Marketing_Spend_Per_Day', 'Location_Foot_Tra
target = 'Daily_Revenue'

X = df[features]
y = df[target]
```

In [52]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [54]:
```
# Initialize and train Random Forest Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

print('Model training completed.')
```

```
Model training completed.
```

Model Evaluation

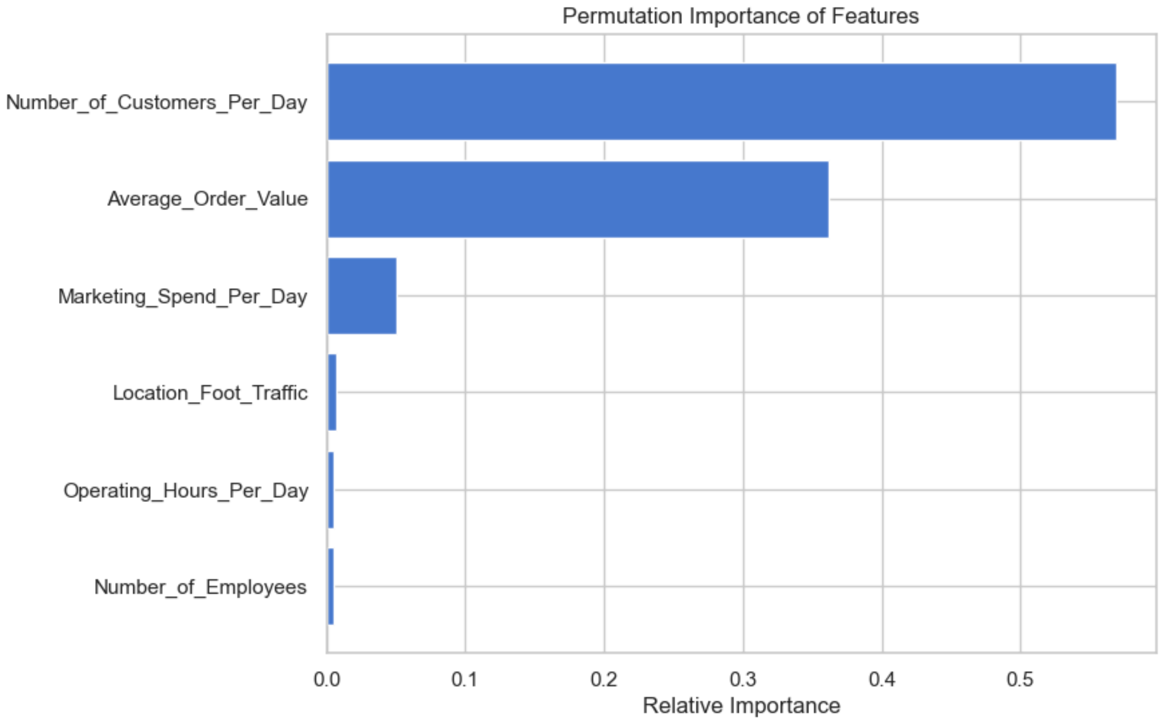
After training the model, we evaluate its performance on the test set. Here, we calculate the RMSE and R2 score. In addition, we visualize the feature importance using a horizontal bar chart, which can help in understanding the influential features.

In [57]:
```
# Predict on the test set
y_pred = model.predict(X_test)
```

In [59]:
```
# Evaluation metrics
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print('RMSE:', rmse)
print('R2 Score:', r2)
```

```
RMSE: 217.94851231361906
R2 Score: 0.9491618691981997
```

In [61]:
```
# Plotting feature importances
importances = model.feature_importances_
feature_names = features

indices = np.argsort(importances)

plt.figure(figsize=(8, 6))
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel('Relative Importance')
plt.title('Permutation Importance of Features')
plt.show()
```



Conclusion and Future Work

This notebook demonstrates a comprehensive approach to exploring and modeling a coffee shop's daily revenue data. We performed exploratory data analysis using various visualization techniques, processed the data to address missing values and outliers, and built a Random Forest Regressor to predict daily revenue with a reasonable performance.

The approach outlined here provides a strong starting point. Future analysis could include:

Advanced feature engineering to capture interactions between variables, Hyperparameter tuning for the Random Forest, or even trying alternative algorithms, Deployment of the predictor as a standalone application for real-time predictions, and An in-depth study of seasonal patterns for coffee shop revenue if time-based data becomes available. Thank you for reading this notebook. If you found the analysis helpful, please consider upvoting.

In [ ]:

```
```