Project Title "Supermart Grocery Sales - Retail Analytics Dataset"

1. Business Objectives

These objectives are aligned with the grocery delivery application's business growth and optimization goals.

1.1 Increase Profitability

Identify which categories, subcategories, or regions generate the highest profits.

Determine the impact of discounts on sales and profit margins.

1.2 Improve Customer Targeting

Understand customer preferences by analyzing the most frequently purchased categories and subcategories.

Identify cities or regions with the highest sales to focus marketing campaigns.

1.3 Optimize Pricing and Discounts

Analyze the relationship between discounts, sales, and profit to recommend optimal discount rates.

Avoid over-discounting products that negatively impact profitability.

1.4 Regional Insights

Compare sales, profits, and discounts across regions (North, South, West) to allocate resources effectively.

The objective of this project is to analyze the grocery delivery application's order data for customers in Tamil Nadu, India, to provide actionable insights on profitability, customer preferences, and regional performance. The goal is to enhance decision-making in pricing strategies, marketing efforts, and resource allocation to maximize overall profitability and customer satisfaction.

Data Collection and Preparation

In [116…
```
import pandas as pd
import numpy as np
```

In [117…
```
df = pd.read_csv(r"C:\Users\chira\Downloads\Supermart Grocery Sales - Retail Analyt
data = df
```

In [118…
```
df.head()
```

Out[118]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	Profit	S
0	OD1	Harish	Oil & Masala	Masalas	Vellore	11-08-2017	North	1254	0.12	401.28	T N
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11-08-2017	South	749	0.18	149.80	T N
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	06-12-2017	West	2360	0.21	165.20	T N
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10-11-2016	South	896	0.25	89.60	T N
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10-11-2016	South	2355	0.26	918.45	T N

In [119…
```
df.tail()
```

Out[119]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount
9989	OD9990	Sudeep	Eggs, Meat & Fish	Eggs	Madurai	12/24/2015	West	945	0.16
9990	OD9991	Alan	Bakery	Biscuits	Kanyakumari	07-12-2015	West	1195	0.26
9991	OD9992	Ravi	Food Grains	Rice	Bodi	06-06-2017	West	1567	0.16
9992	OD9993	Peer	Oil & Masala	Spices	Pudukottai	10/16/2018	West	1659	0.15
9993	OD9994	Ganesh	Food Grains	Atta & Flour	Tirunelveli	4/17/2018	West	1034	0.28

In [120…
```
df.shape
```

Out[120]:
```
(9994, 11)
```

In [121…
```
df.isnull().sum()
```

Out[121]:
```
Order ID         0
Customer Name    0
Category         0
Sub Category     0
City             0
Order Date       0
Region           0
Sales            0
Discount         0
Profit           0
State            0
dtype: int64
```

In [122…
```
df.dropna(inplace=True)
```

In [123…
```
df.isnull().sum()
```

Out[123]:
```
Order ID         0
Customer Name    0
Category         0
Sub Category     0
City             0
Order Date       0
Region           0
Sales            0
Discount         0
Profit           0
State            0
dtype: int64
```

In [124…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Order ID       9994 non-null   object 
 1   Customer Name  9994 non-null   object 
 2   Category       9994 non-null   object 
 3   Sub Category   9994 non-null   object 
 4   City           9994 non-null   object 
 5   Order Date     9994 non-null   object 
 6   Region         9994 non-null   object 
 7   Sales          9994 non-null   int64  
 8   Discount       9994 non-null   float64
 9   Profit         9994 non-null   float64
 10  State          9994 non-null   object 
dtypes: float64(2), int64(1), object(8)
memory usage: 859.0+ KB
```

In [125…
```
df.describe()
```

Out[125]:

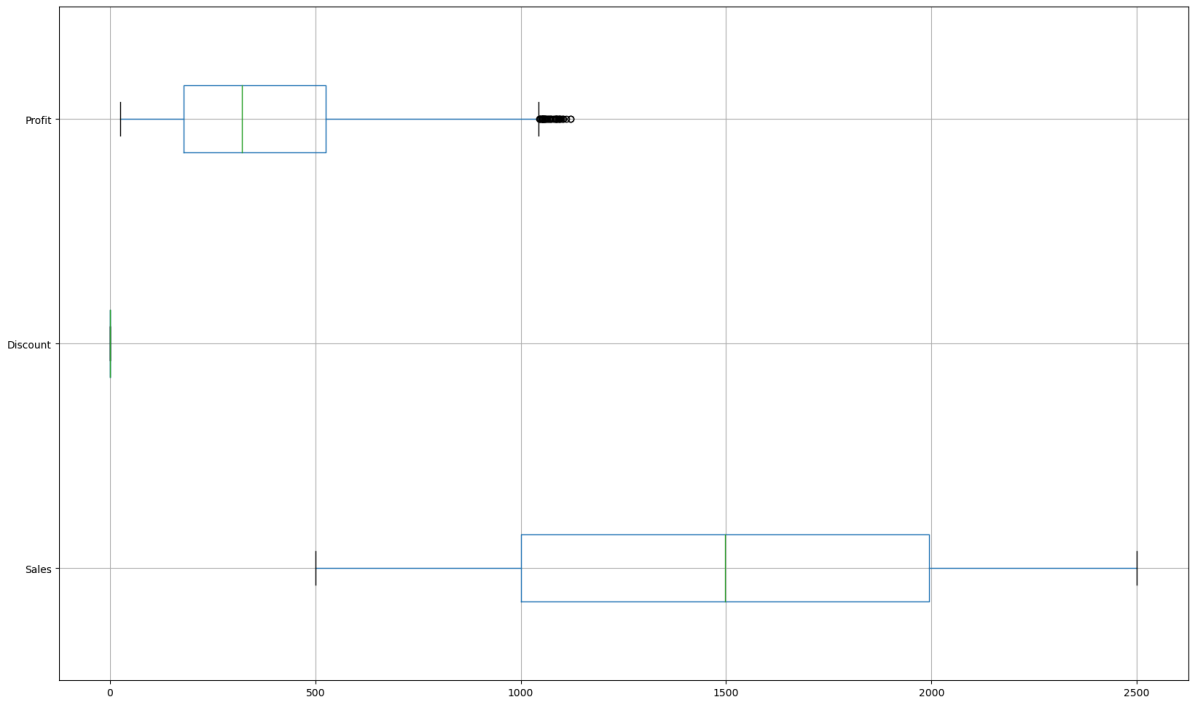
	Sales	Discount	Profit
count	9994.000000	9994.000000	9994.000000
mean	1496.596158	0.226817	374.937082
std	577.559036	0.074636	239.932881
min	500.000000	0.100000	25.250000
25%	1000.000000	0.160000	180.022500
50%	1498.000000	0.230000	320.780000
75%	1994.750000	0.290000	525.627500
max	2500.000000	0.350000	1120.950000

In [126…

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [127…

```
# Create a box plot
plt.figure(figsize=(20,12))
df.boxplot(vert=0)
plt.show()
```



In [128…

```
def remove_outlier(col):
  # Convert column to numeric before sorting (handling potential errors)
  col = pd.to_numeric(col, errors='coerce')
  sorted_col = sorted(col)
  Q1, Q3 = np.percentile(sorted_col, [25, 75])
  IQR = Q3 - Q1
  lower_range = Q1 - (1.5 * IQR)
  upper_range = Q3 + (1.5 * IQR)
  return lower_range, upper_range

# Assuming 'df' is your DataFrame
for column in df.columns:
  lower, upper = remove_outlier(df[column])
  df[column] = np.where(df[column] > upper, upper, df[column])
  df[column] = np.where(df[column] < lower, lower, df[column])
```

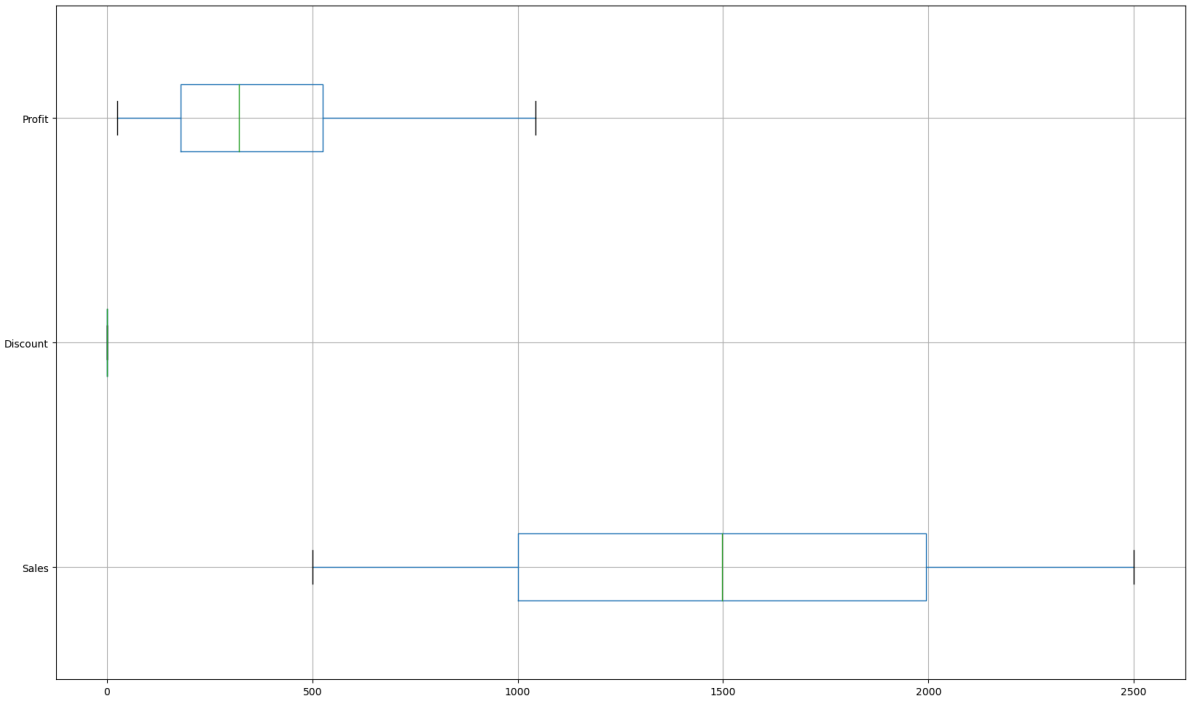
```
# Now 'df' has outliers removed (assuming numerical columns)
```

In [129…

```
# Identification of Outliers using boxplot

plt.figure(figsize=(20,12))
df.boxplot(vert = 0)
```

Out[129]:

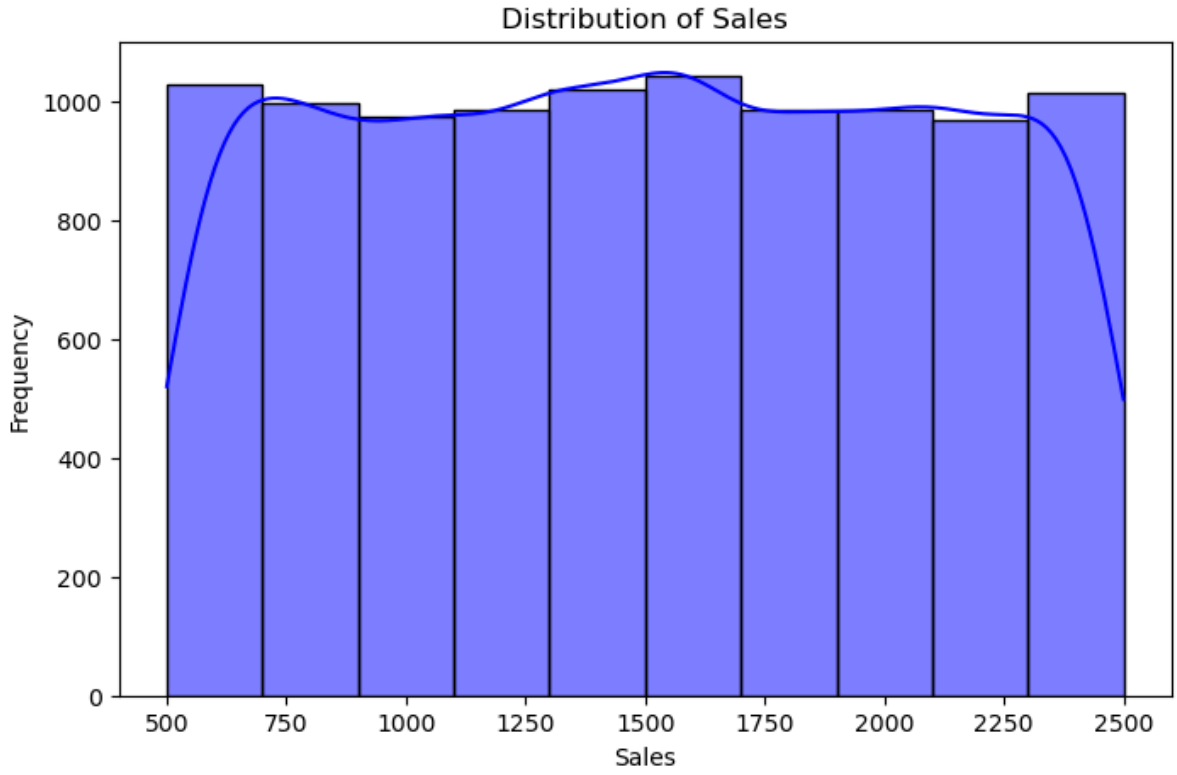
```
<Axes: >
```



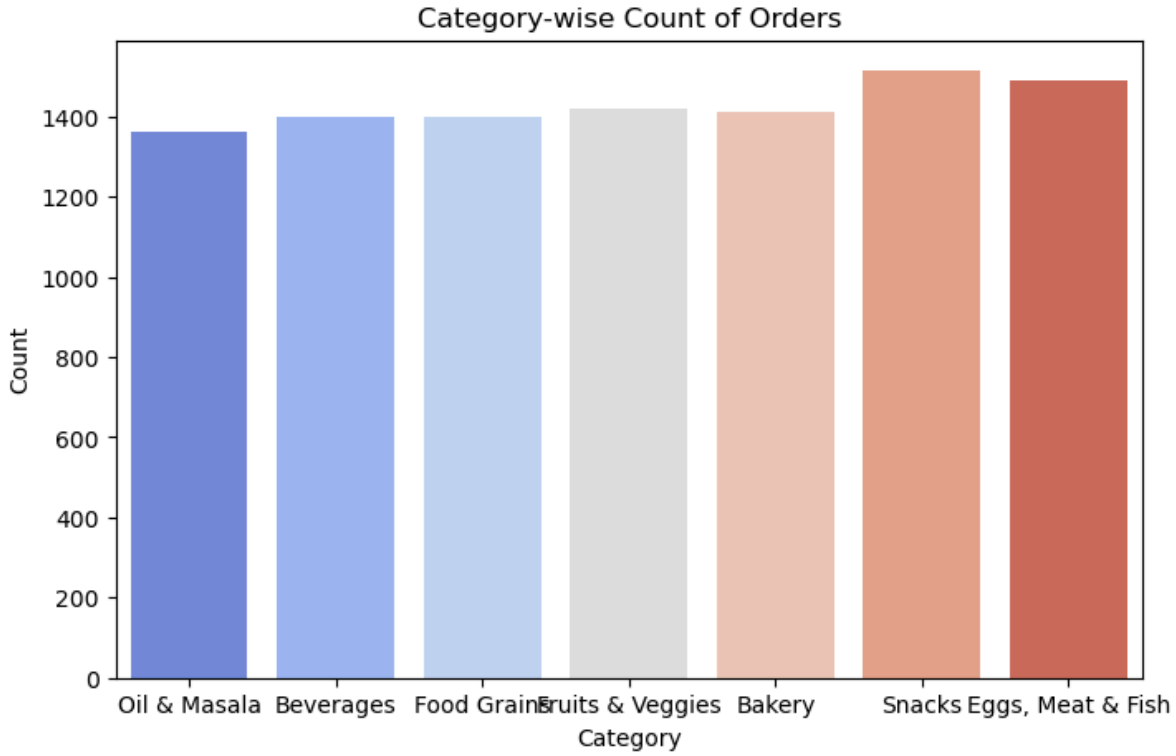
Explorartory Data Analysis

In [130…

```
# ---------------- Univariate Analysis ----------------
# 1. Distribution of Sales
plt.figure(figsize=(8, 5))
sns.histplot(df['Sales'], kde=True, bins=10, color='blue')
plt.title('Distribution of Sales')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()

# 2. Bar plot for Category Counts
plt.figure(figsize=(8, 5))
sns.countplot(x='Category', data=df, palette='coolwarm')
plt.title('Category-wise Count of Orders')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
```



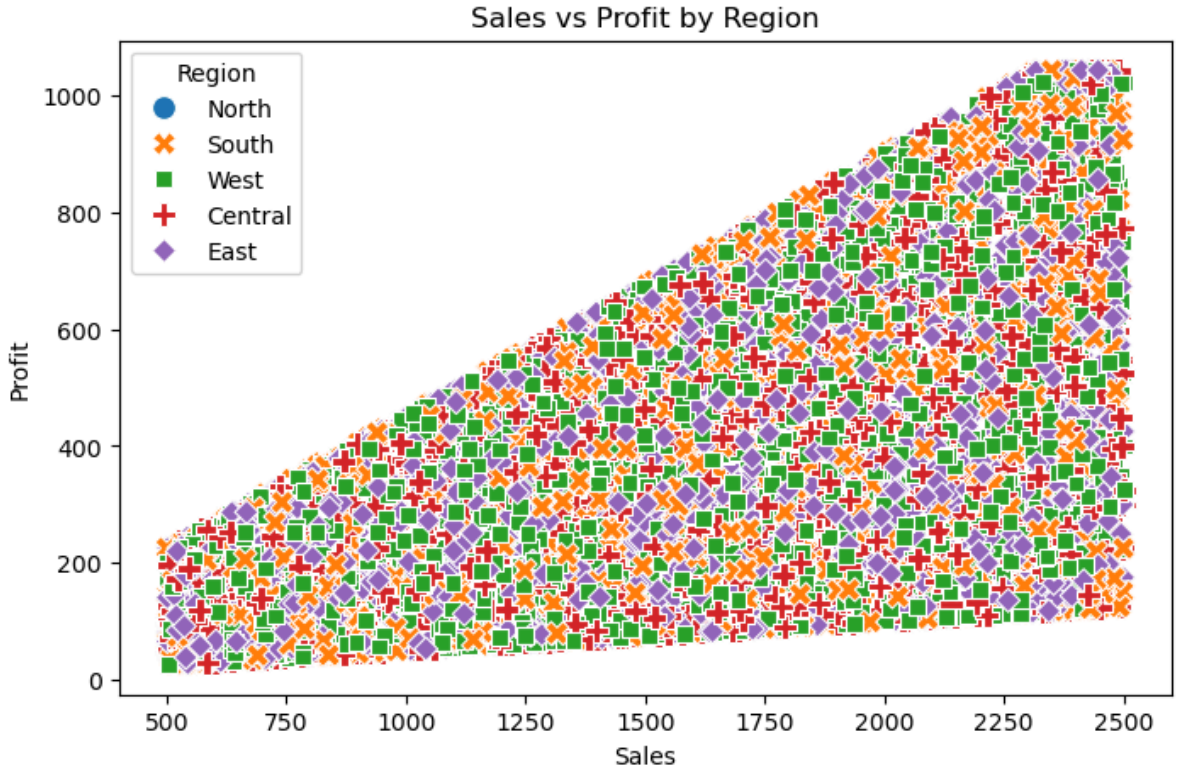

```
C:\Users\chira\AppData\Local\Temp\ipykernel_11752\508860363.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='Category', data=df, palette='coolwarm')
```



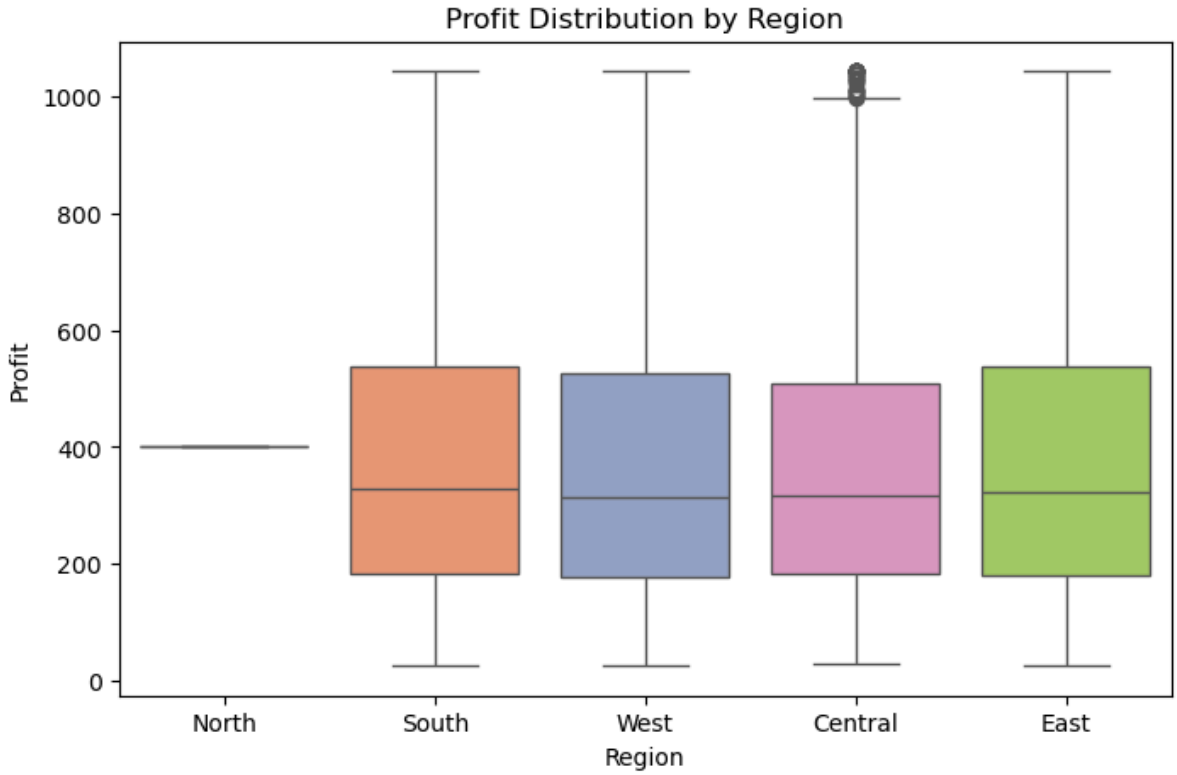
In [131…

```
# ----------------- Bivariate Analysis -----------------
# 3. Scatter plot: Sales vs Profit
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Sales', y='Profit', hue='Region', style='Region', s=100, data=df
plt.title('Sales vs Profit by Region')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.show()
```

```
# 4. Box plot: Profit Distribution by Region
plt.figure(figsize=(8, 5))
sns.boxplot(x='Region', y='Profit', data=df, palette='Set2')
plt.title('Profit Distribution by Region')
plt.xlabel('Region')
plt.ylabel('Profit')
plt.show()
```

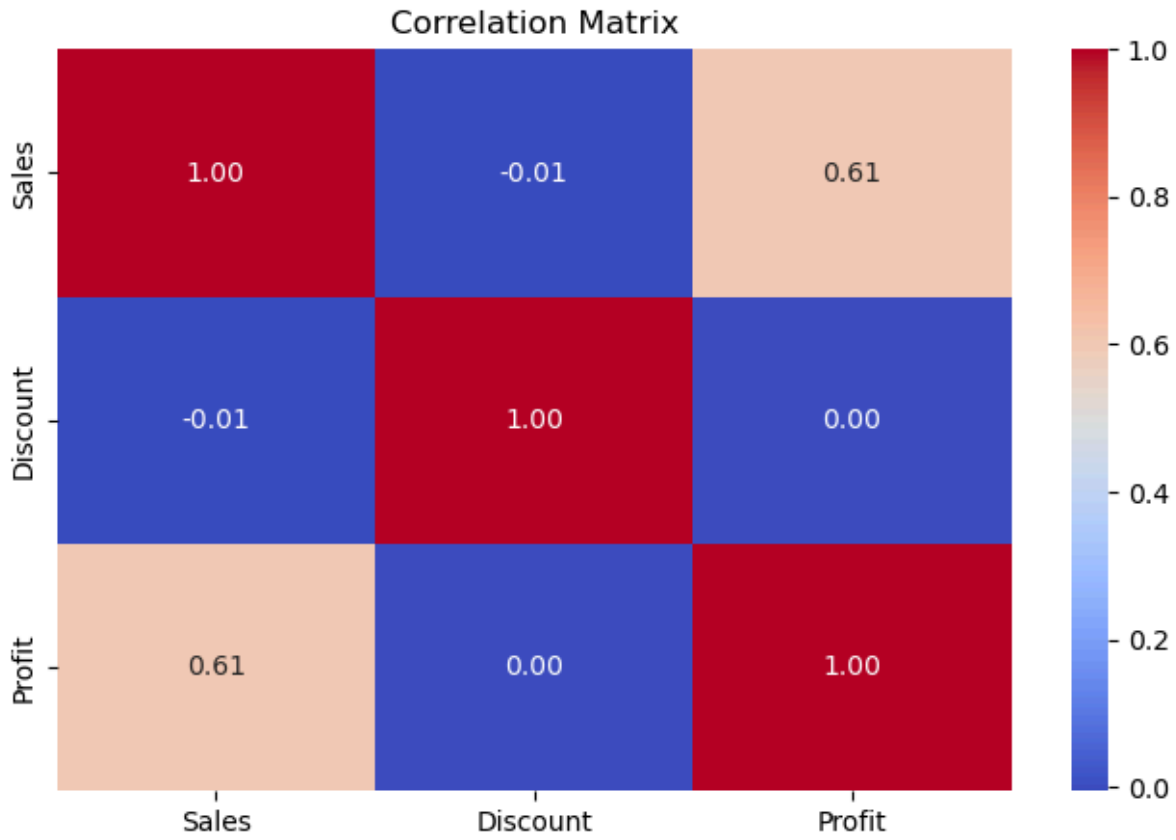



```
C:\Users\chira\AppData\Local\Temp\ipykernel_11752\238963201.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='Region', y='Profit', data=df, palette='Set2')
```

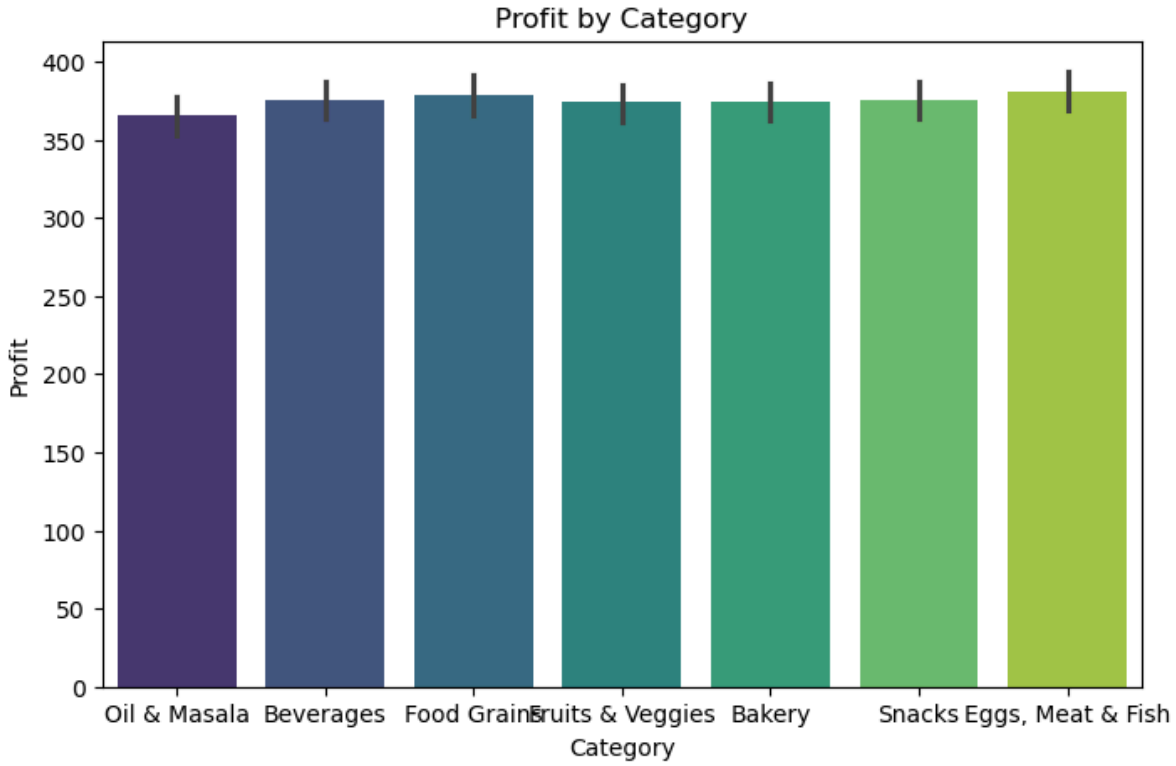


In [132…

```
# ----------------- Correlation Analysis -----------------
# 5. Heatmap for Numerical Features
plt.figure(figsize=(8, 5))
corr = df[['Sales', 'Discount', 'Profit']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()


# 6. Profit by Category (Bar Plot)
plt.figure(figsize=(8, 5))
sns.barplot(x='Category', y='Profit', data=df, palette='viridis')
plt.title('Profit by Category')
plt.xlabel('Category')
plt.ylabel('Profit')
plt.show()
```



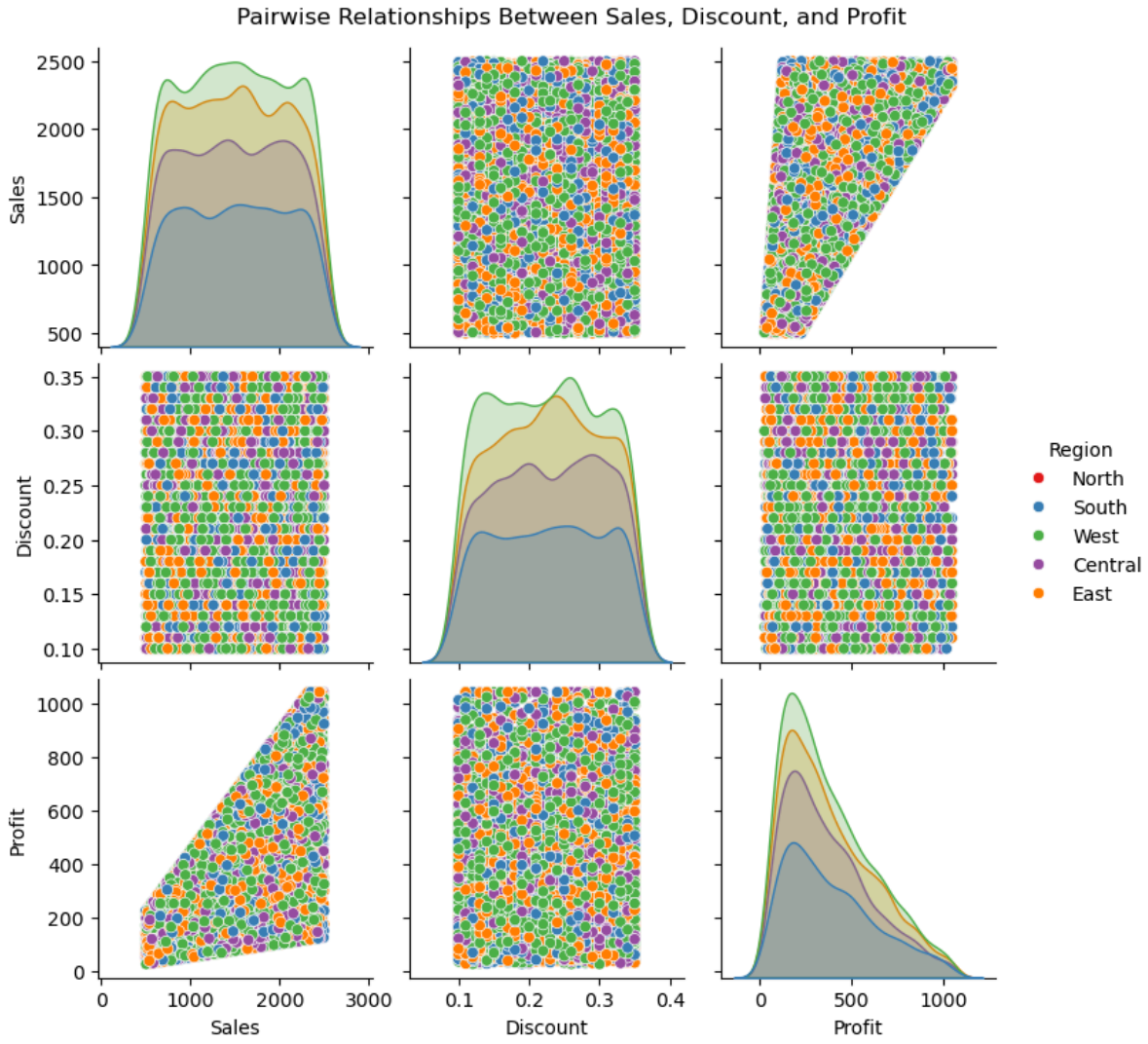

```
C:\Users\chira\AppData\Local\Temp\ipykernel_11752\2579700842.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Category', y='Profit', data=df, palette='viridis')
```



In [133…

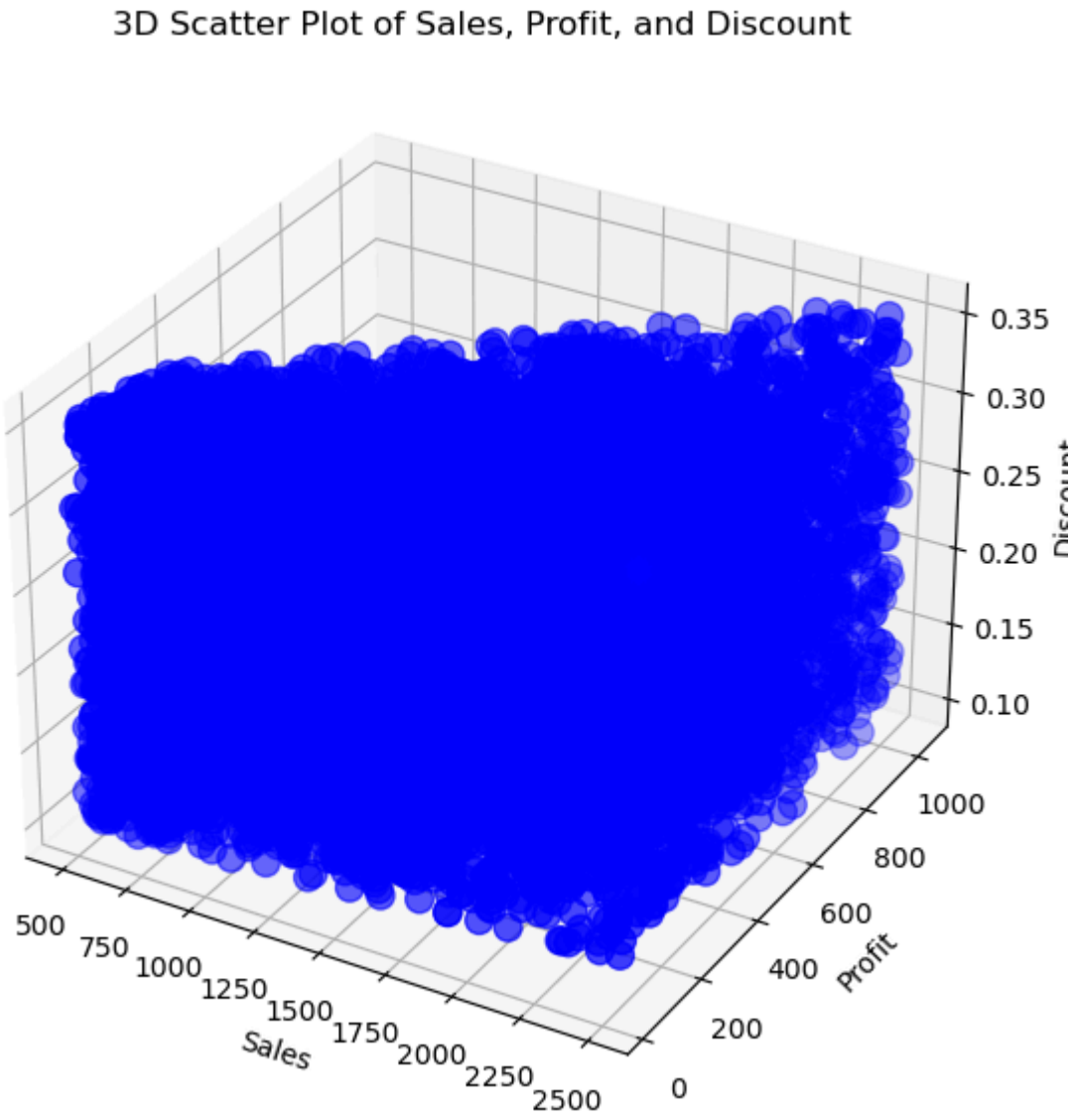
```
# ---------------- 1. Pair Plot ----------------
# Pair plot to analyze pairwise relationships between numerical features
sns.pairplot(df, vars=['Sales', 'Discount', 'Profit'], hue='Region', palette='Set1'
plt.suptitle('Pairwise Relationships Between Sales, Discount, and Profit', y=1.02)
plt.show()
```

```
C:\Users\chira\anaconda3\Lib\site-packages\seaborn\axisgrid.py:123: UserWarning: T
he figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [134…

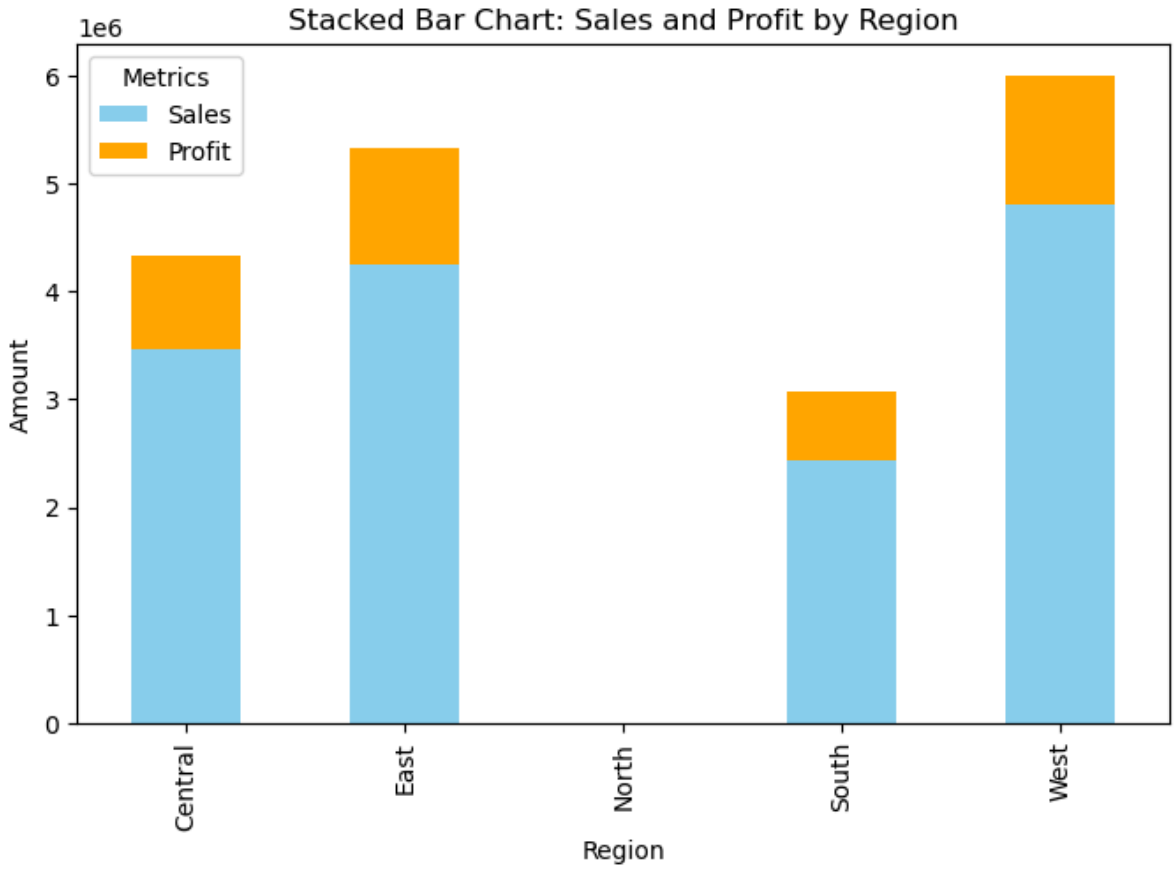
```
# ----------------- 2. 3D Scatter Plot -----------------
# 3D scatter plot: Sales, Profit, and Discount
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df['Sales'], df['Profit'], df['Discount'], c='blue', marker='o', s=100)

ax.set_title('3D Scatter Plot of Sales, Profit, and Discount')
ax.set_xlabel('Sales')
ax.set_ylabel('Profit')
ax.set_zlabel('Discount')
plt.show()
```



In [135…
```
df.columns
```

Out[135]:
```
Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
       'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State'],
      dtype='object')
```

In [136…
```
# ----------------- 3. Stacked Bar Chart -----------------
# Stacked bar chart: Sales and Profit by Region
region_group = df.groupby('Region')[['Sales', 'Profit']].sum()

region_group.plot(kind='bar', stacked=True, figsize=(8, 5), color=['skyblue', 'orar
plt.title('Stacked Bar Chart: Sales and Profit by Region')
plt.xlabel('Region')
plt.ylabel('Amount')
plt.legend(title='Metrics')
plt.show()
```



Features Engineering and selection

1.1. Handle Date/Time Features

In [157…
```
df.head()
```

Out[157]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	Profit	State	Y
0	OD1	Harish	5	14	21	2017-11-08	2	1254.0	0.12	401.28	0	2(
1	OD2	Sudha	1	13	8	2017-11-08	3	749.0	0.18	149.80	0	2(
2	OD3	Hussain	3	0	13	2017-06-12	4	2360.0	0.21	165.20	0	2(
3	OD4	Jackson	4	12	4	2016-10-11	3	896.0	0.25	89.60	0	2(
4	OD5	Ridhesh	3	18	12	2016-10-11	3	2355.0	0.26	918.45	0	2(

In [159…
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

```
import seaborn as sns


# Feature Engineering & Selection
df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')
df['Year'], df['Month'], df['Day'] = df['Order Date'].dt.year, df['Order Date'].dt.
```

In [160…
```
# Label Encoding
label_columns = ['Category', 'Sub Category', 'City', 'Region', 'State']
df[label_columns] = df[label_columns].apply(LabelEncoder().fit_transform)
```

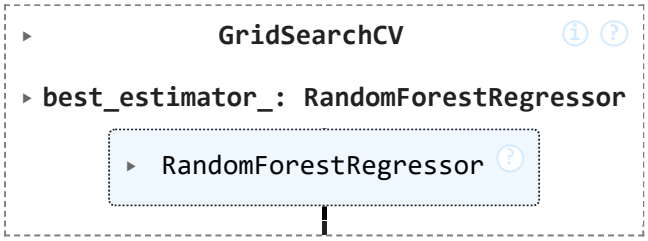
In [161…
```
# Define Features and Target
X = df[['Category', 'Sub Category', 'City', 'Region', 'State', 'Sales', 'Discount',
y = df['Profit']
```

In [162…
```
# Data Preprocessing: Scaling Features
X_scaled = StandardScaler().fit_transform(X)
```

In [165…
```
# Model Building & Hyperparameter Tuning
model = RandomForestRegressor(random_state=42)
```

In [166…
```
param_grid = {'n_estimators': [50, 100], 'max_depth': [10, 20], 'min_samples_split'
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error
grid_search.fit(X_scaled, y)
```

Out[166]:



In [167…
```
# Best Model
best_model = grid_search.best_estimator_
```

In [168…
```
# Model Evaluation
y_pred = best_model.predict(X_scaled)
mse, r2 = mean_squared_error(y, y_pred), r2_score(y, y_pred)
print(f"MSE: {mse:.2f}, R²: {r2:.2f}")
```

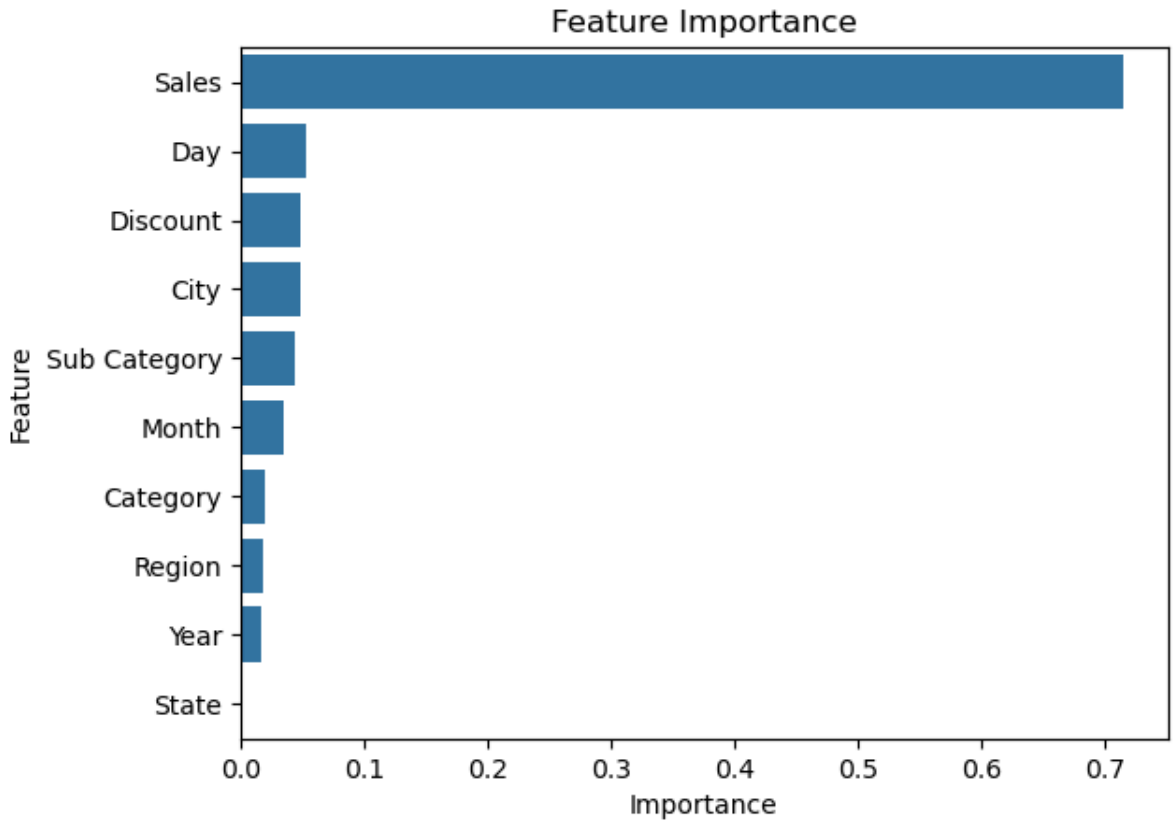
```
MSE: 24440.93, R²: 0.57
```

In [169…
```
# Cross-Validation
cv_scores = cross_val_score(best_model, X_scaled, y, cv=5, scoring='neg_mean_square
print(f"Average Cross-Validation MSE: {np.mean(cv_scores):.2f}")
```

```
Average Cross-Validation MSE: -37190.47
```

In [170…
```
# Feature Importance Plot
feature_importances = best_model.feature_importances_
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importanc
```

In [171…
```
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title("Feature Importance")
plt.show()
```


Feature Importance
Sales
Day
Discount
City
Sub Category
Month
Category
Region
Year
State
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Importance

In [ ]:

In [ ]: