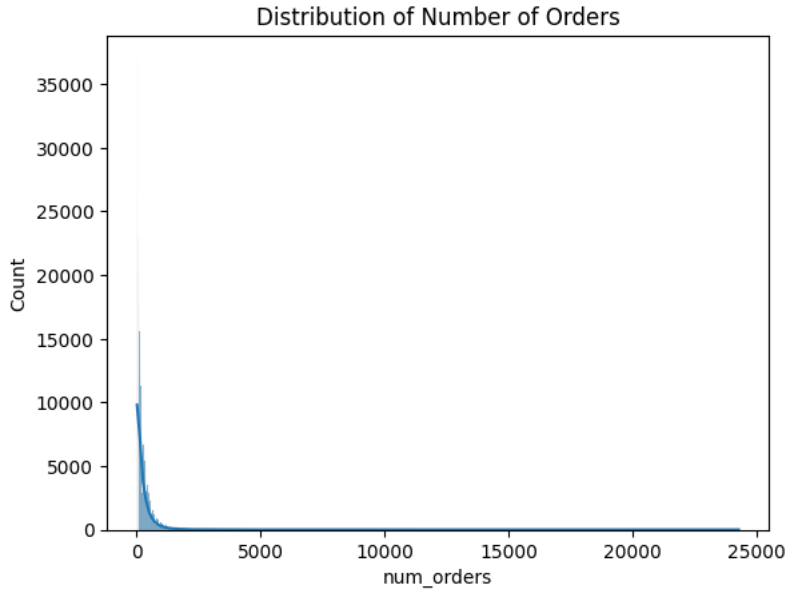
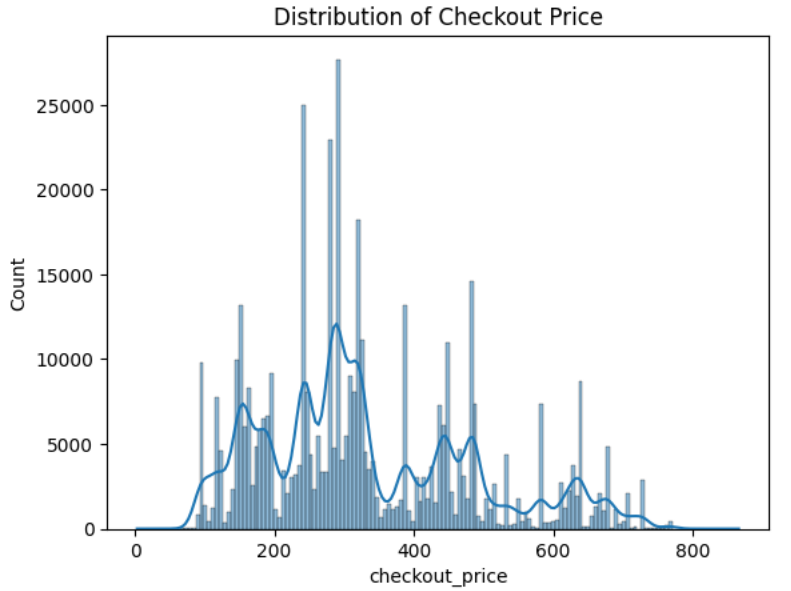
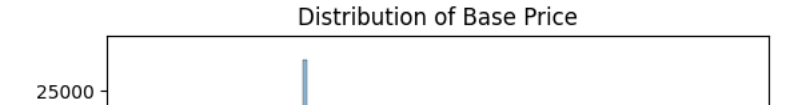
```
import pandas as pd

# Load the datasets
fulfilment_center_info = pd.read_csv('fulfilment_center_info.csv')
meal_info = pd.read_csv('meal_info.csv')
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')



# Merge datasets for a comprehensive view
train_data = train_data.merge(fulfilment_center_info, on='center_id', how='left')
train_data = train_data.merge(meal_info, on='meal_id', how='left')



# Check for missing values
missing_values = train_data.isnull().sum()

# If there are any missing values, decide on a strategy (e.g., imputation, deletion)
# For now, let's fill with 0 or a placeholder value if needed
train_data.fillna(0, inplace=True)



# Display summary statistics
train_data.describe()
```

	checkout_price
	456548.000000
	332.238933
	152.939723
	2.970000
	228.950000
	296.820000
	445.230000
	866.270000

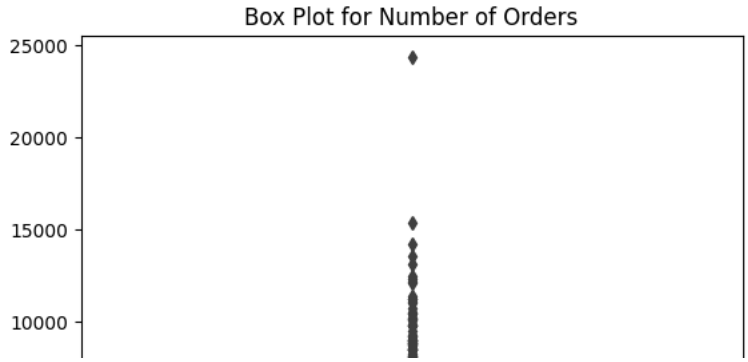
```
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution of Number of Orders
sns.histplot(train_data['num_orders'], kde=True)
plt.title('Distribution of Number of Orders')
plt.show()

# Distribution of Checkout Price
sns.histplot(train_data['checkout_price'], kde=True)
plt.title('Distribution of Checkout Price')
plt.show()

# Distribution of Base Price
sns.histplot(train_data['base_price'], kde=True)
plt.title('Distribution of Base Price')
plt.show()
```



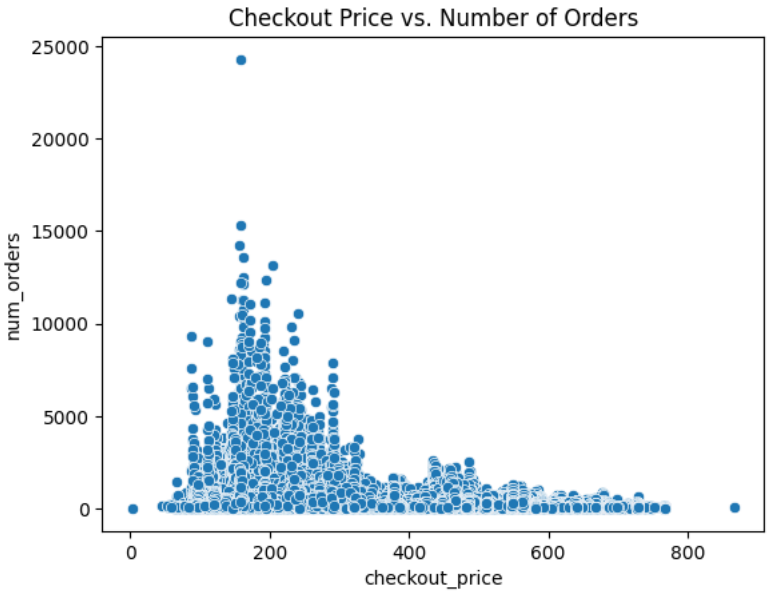





```
# Box plot for Number of Orders
sns.boxplot(train_data['num_orders'])
plt.title('Box Plot for Number of Orders')
plt.show()
```

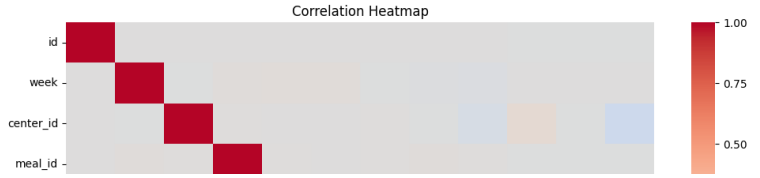



```
# Relationship between Checkout Price and Number of Orders
sns.scatterplot(x=train_data['checkout_price'], y=train_data['num_orders'])
plt.title('Checkout Price vs. Number of Orders')
plt.show()
```




```
# Compute correlation matrix
correlation_matrix = train_data.corr()

# Visualize correlations using a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap')
plt.show()
```

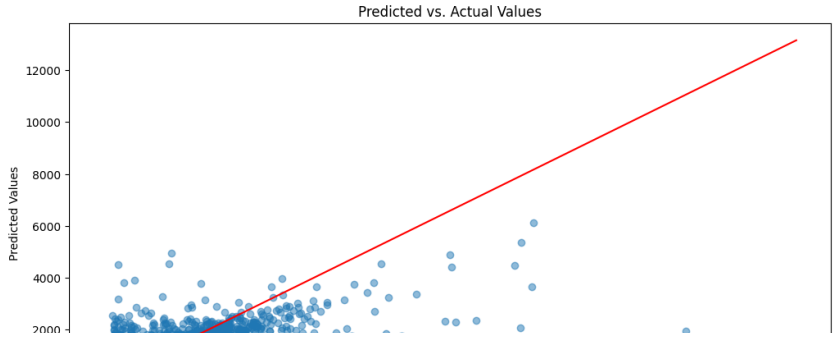
```
<ipython-input-8-82171283d45f>:2: FutureWarning: The default value of numeric_only
  correlation_matrix = train_data.corr()
```




```
# Feature Engineering:
# Convert 'center_type', 'category', and 'cuisine' into numerical format using one-hot encoding
train_data = pd.get_dummies(train_data, columns=['center_type', 'category', 'cuisine'])

# Lag Features: Representing the demand of past weeks
for i in range(1, 5):  # Lags for the previous 1, 2, 3, and 4 weeks
    train_data[f'lag_{i}_week_orders'] = train_data.groupby(['center_id', 'meal_id'])['num_orders'].shift(i)
    train_data[f'lag_{i}_week_orders'].fillna(0, inplace=True)

# Price-based Features
train_data['price_diff'] = train_data['checkout_price'] - train_data['base_price']
train_data['price_diff_percent'] = (train_data['checkout_price'] - train_data['base_price']) / train_data['base_price']

# Promotion Interaction Features
train_data['promo_price_interaction'] = train_data['emailer_for_promotion'] * train_data['checkout_price']
train_data['homepage_price_interaction'] = train_data['homepage_featured'] * train_data['checkout_price']

# Train-Test Split:
# Features and target variable
X = train_data.drop(columns=['id', 'num_orders', 'week', 'center_id', 'meal_id'])
y = train_data['num_orders']

# Splitting the data into training and validation sets (80-20 split)
train_size = int(0.8 * len(train_data))
X_train, X_val = X[:train_size], X[train_size:]
y_train, y_val = y[:train_size], y[train_size:]

# Model Training:
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Model Evaluation:
# Predict on the validation set
y_pred = model.predict(X_val)

# Compute the MAE and RMSE
mae = mean_absolute_error(y_val, y_pred)
rmse = np.sqrt(mean_squared_error(y_val, y_pred))

print(f"Mean Absolute Error: {mae}")
print(f"Root Mean Squared Error: {rmse}")
```

```
Mean Absolute Error: 100.69755243965186
Root Mean Squared Error: 211.24778722633408
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.scatter(y_val, y_pred, alpha=0.5)
plt.title('Predicted vs. Actual Values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.plot([min(y_val), max(y_val)], [min(y_val), max(y_val)], color='red')  # Diagonal line
plt.show()
```




```
residuals = y_val - y_pred

plt.figure(figsize=(12, 6))
plt.scatter(y_pred, residuals, alpha=0.5)
plt.title('Residuals vs. Predicted Values')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='red')
plt.show()
```