```python
# install the required packages
%pip install -r requirements.txt

# import the required packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.preprocessing import StandardScaler

# Load dataset
file_path = "dataset.csv"
df = pd.read_csv(file_path)

# Convert specified columns to numeric, setting invalid parsing to NaN

numeric_columns = ["Tenure", "WarehouseToHome", "HourSpendOnApp",
                "OrderAmountHikeFromlastYear", "OrderCount",
"DaySinceLastOrder"]
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Replace zeros with NaN in specified columns

cols_to_replace_zeros = ["CashbackAmount", "CouponUsed"]
df[cols_to_replace_zeros] = df[cols_to_replace_zeros].replace(0,
np.nan)

# Identify numeric and categorical columns in the DataFrame

numeric_cols = df.select_dtypes(include=['number']).columns.tolist()
categorical_cols =
df.select_dtypes(exclude=['number']).columns.tolist()

# Fill missing values in numeric columns with the median value

df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].median())

# Fill missing values in categorical columns with the mode (most
frequent value)

for col in categorical_cols:
    df[col].fillna(df[col].mode()[0])


# Identify and drop columns with zero variance (i.e., columns where
all values are the same)

zero_variance_cols = [col for col in numeric_cols if df[col].nunique()
== 1]
```

```python
df = df.drop(columns=zero_variance_cols)
print("Dropped zero-variance columns:", zero_variance_cols)
```

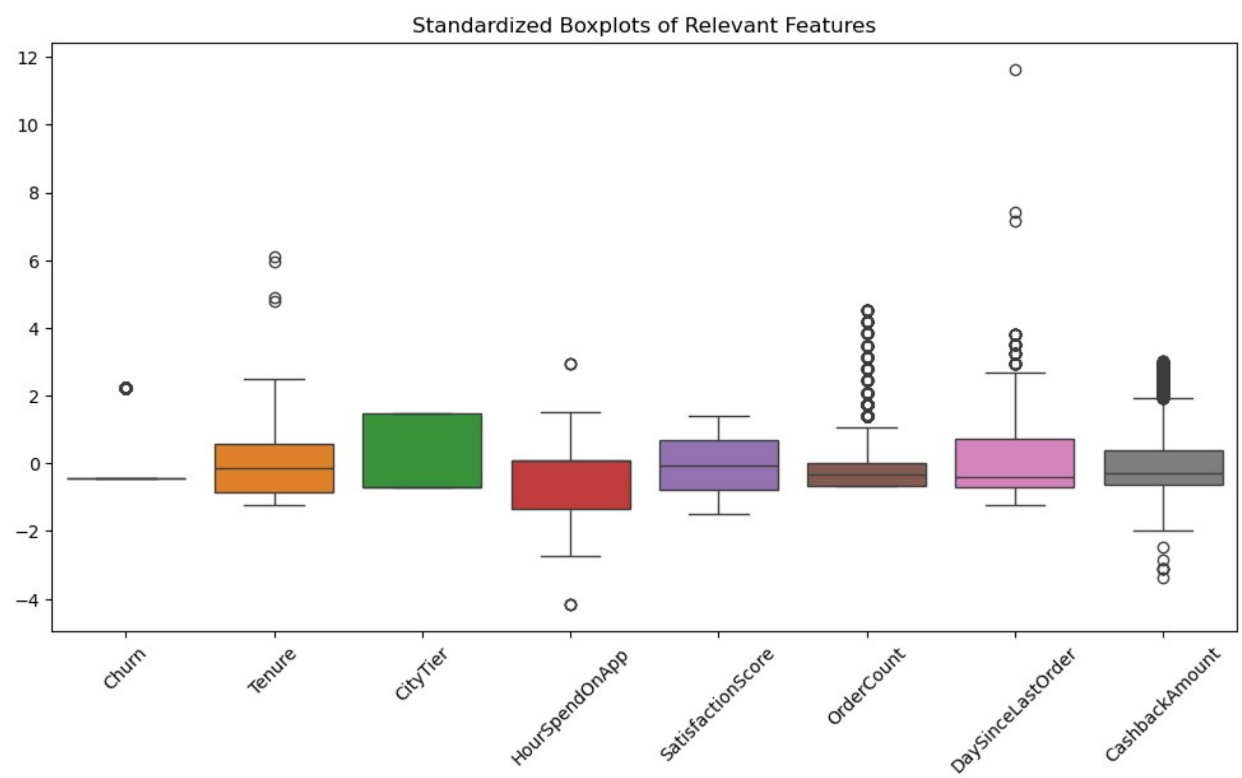
```
Dropped zero-variance columns: []
```

```python
# (a) Calculate mean, median, and standard deviation

# Select numeric columns from the DataFrame
numeric_cols = df.select_dtypes(include=['number'])

# Generate descriptive statistics for the numeric columns and
transpose the result
stats = numeric_cols.describe().T

# Add the median to the descriptive statistics
stats["median"] = numeric_cols.median()

# Print the summary statistics
print("\nSummary Statistics:")
print(stats)
```

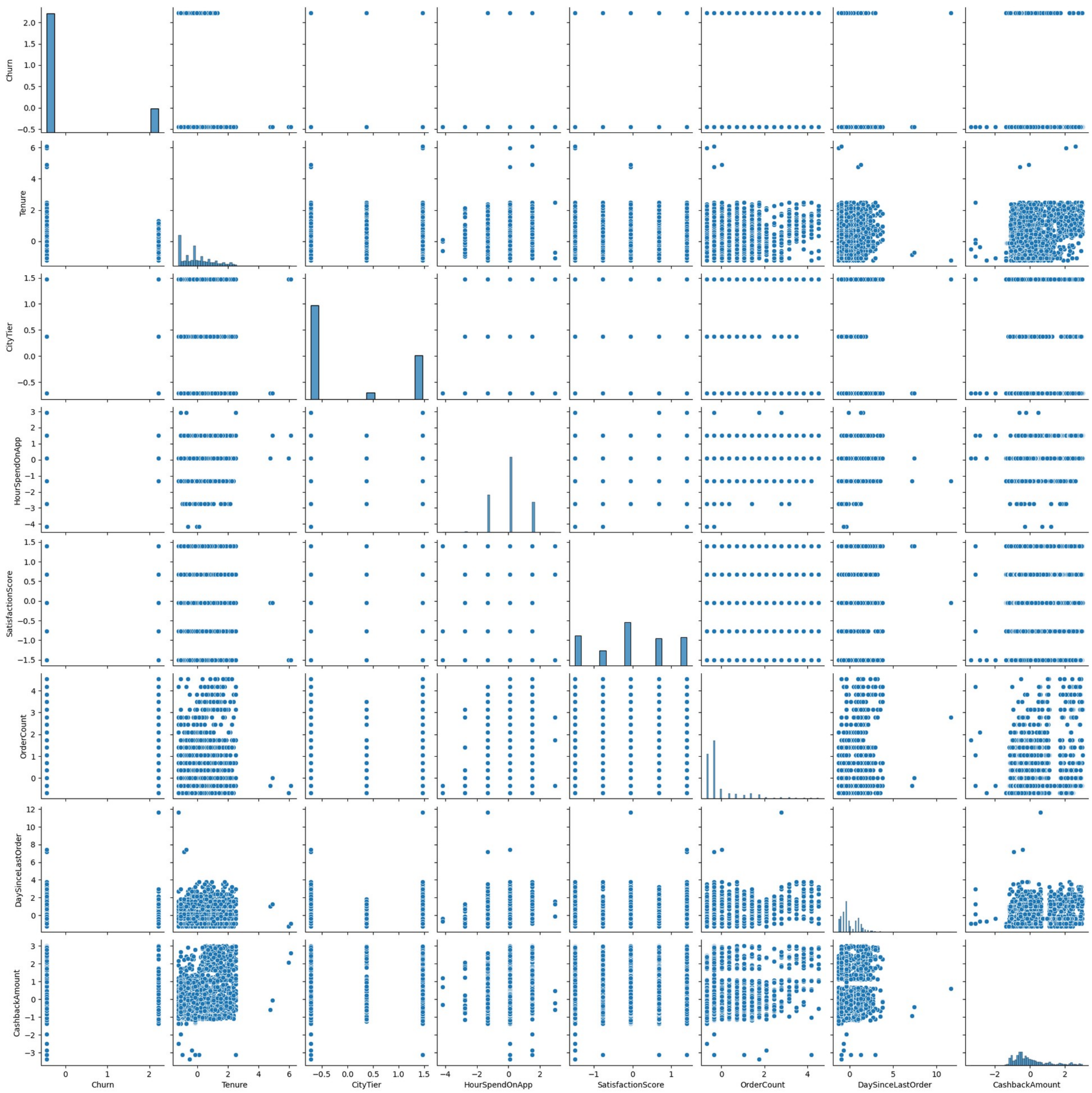
```
Summary Statistics:
                             count          mean          std
min  \
CustomerID                  5630.0  52815.500000  1625.385339
50001.0
Churn                       5630.0      0.168384     0.374240
0.0
Tenure                      5630.0     10.134103     8.357951
0.0
CityTier                    5630.0      1.654707     0.915389
1.0
WarehouseToHome             5630.0     15.566785     8.345961
5.0
HourSpendOnApp              5630.0      2.934636     0.705528
0.0
NumberOfDeviceRegistered    5630.0      3.688988     1.023999
1.0
SatisfactionScore           5630.0      3.066785     1.380194
1.0
NumberOfAddress             5630.0      4.214032     2.583586
1.0
Complain                    5630.0      0.284902     0.451408
0.0
OrderAmountHikeFromlastYear 5630.0     15.674600     3.591058
11.0
CouponUsed                  5630.0      2.128242     1.654433
1.0
OrderCount                  5630.0      2.961812     2.879248
```

```
1.0
DaySinceLastOrder            5630.0      4.459325     3.570626
0.0
CashbackAmount               5630.0    177.337300    48.967834
12.0

                                25%       50%       75%       max
median
CustomerID                 51408.25  52815.5  54222.75  55630.0
52815.5
Churn                          0.00      0.0      0.00      1.0
0.0
Tenure                         3.00      9.0     15.00     61.0
9.0
CityTier                       1.00      1.0      3.00      3.0
1.0
WarehouseToHome                9.00     14.0     20.00    127.0
14.0
HourSpendOnApp                 2.00      3.0      3.00      5.0
3.0
NumberOfDeviceRegistered       3.00      4.0      4.00      6.0
4.0
SatisfactionScore              2.00      3.0      4.00      5.0
3.0
NumberOfAddress                2.00      3.0      6.00     22.0
3.0
Complain                       0.00      0.0      1.00      1.0
0.0
OrderAmountHikeFromlastYear   13.00     15.0     18.00     26.0
15.0
CouponUsed                     1.00      2.0      2.00     16.0
2.0
OrderCount                     1.00      2.0      3.00     16.0
2.0
DaySinceLastOrder              2.00      3.0      7.00     46.0
3.0
CashbackAmount               146.00    163.0    196.00    325.0
163.0

# Select specific features from the DataFrame for further analysis

selected_features = ["Churn", "Tenure", "CityTier", "HourSpendOnApp",
                    "SatisfactionScore", "OrderCount",
"DaySinceLastOrder", "CashbackAmount"]
df_filtered = df[selected_features]

# Standardize the selected features

scaler = StandardScaler()
```

```
df_scaled = pd.DataFrame(scaler.fit_transform(df_filtered),
columns=selected_features)

# (b) Plot the boxplots

# Set the figure size for the plot
plt.figure(figsize=(12, 6))

# Create a boxplot for the standardized features in the DataFrame
sns.boxplot(data=df_scaled)

# Rotate the x-axis labels by 45 degrees for better readability
plt.xticks(rotation=45)

# Set the title of the plot
plt.title("Standardized Boxplots of Relevant Features")

# Display the plot
plt.show()
```




```
#(c) Draw pairplots

# Draw pairplots for the standardized features
sns.pairplot(df_scaled)
plt.show()
```




```
# (d) Calculate Pearson's correlation for numerical features with
'Churn'

# Initialize an empty dictionary to store the correlations
correlations = {}

# Iterate over each numeric column in the DataFrame
for col in df.select_dtypes(include=['number']).columns:
    # Skip the 'Churn' column as we don't want to calculate its
correlation with itself
    if col != "Churn":
        # Calculate the Pearson correlation coefficient between the
```

```
current column and 'Churn'
        corr, _ = pearsonr(df[col], df["Churn"])
        # Store the correlation coefficient in the dictionary
        correlations[col] = corr
        # Print the correlation coefficient
        print(f"Pearson correlation between {col} and Churn:
{corr:.3f}")
```

```
Pearson correlation between CustomerID and Churn: -0.019
Pearson correlation between Tenure and Churn: -0.338
Pearson correlation between CityTier and Churn: 0.085
Pearson correlation between WarehouseToHome and Churn: 0.070
Pearson correlation between HourSpendOnApp and Churn: 0.019
Pearson correlation between NumberOfDeviceRegistered and Churn: 0.108
Pearson correlation between SatisfactionScore and Churn: 0.105
Pearson correlation between NumberOfAddress and Churn: 0.044
Pearson correlation between Complain and Churn: 0.250
Pearson correlation between OrderAmountHikeFromlastYear and Churn: -
0.007
Pearson correlation between CouponUsed and Churn: -0.004
Pearson correlation between OrderCount and Churn: -0.024
Pearson correlation between DaySinceLastOrder and Churn: -0.156
Pearson correlation between CashbackAmount and Churn: -0.156
```

Findings: Columns for Regression, Classification, and Clustering

Regression: We can predict the following column values using regression:

Column	Description
WarehouseToHome	Distance (e.g., kilometers/miles)
HourSpendOnApp	Time spent (e.g., hours)
OrderAmountHikeFromlastYear	Percentage increase (e.g., 15.5%)
CashbackAmount	Monetary value (e.g., $25.30)
Tenure	Duration (e.g., 6.5 months)
DaySinceLastOrder	Continuous measure (e.g., 30.5 days)

Classification: We can perform classification on the below columns:

Column	Description
Churn (target variable)	Predict if a user churns or not
PreferredLoginDevice	Preferred login device of customer
PreferredPaymentMode	Preferred payment method of customer
PreferedOrderCat	Preferred order category of customer in last month
MaritalStatus	Marital status of customer
SatisfactionScore	Satisfactory score of customer on service
NumberOfDeviceRegistered	Number of devices (e.g., 2 devices)

Column	Description
NumberOfAddress	Number of addresses (e.g., 3 addresses)
CouponUsed	Number of coupons used (e.g., 5 coupons)
OrderCount	Number of orders (e.g., 10 orders)
Complain	0 (No) / 1 (Yes)

Clustering: Group using features: All continuous + encoded categorical/discrete columns

Group customers based on Tenure, CityTier, and CashbackAmount.

Key Learnings and Difficulties

What Did We Learn from These Steps?

This assignment emphasized the importance of thorough data cleaning and understanding data distributions.

- Handling missing values required careful consideration of appropriate imputation methods.
- Visualizations revealed challenges in interpreting boxplots with limited data points.
- Difficulties included determining appropriate features for zero replacement and managing overlapping visualizations.
- The exercise highlighted that EDA is a repetitive process and showed how important it is to understand the subject area when cleaning and preparing data.

The columns showing the highest correlation with the target variable (Churn) are:

- Tenure
- OrderCount
- HourSpendOnApp

These columns can be used to build predictive models for customer churn. High correlation indicates that changes in these features are strongly associated with changes in the target variable. For instance, Tenure can help identify long-term customers who are less likely to churn, while Orderount and HourSpendOnApp can provide insights into customer engagement and purchasing behavior.