```
import pandas as pd

try:
    df = pd.read_csv('/content/customer_churn_dataset-training-master.csv')
    print("CSV file read successfully!")
    display(df.head()) # Display the first 5 rows of the DataFrame
except FileNotFoundError:
    print("Error: File not found. Please ensure the file is uploaded and the path is correct.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
CSV file read successfully!
```

	CustomerID	Age	Gender	Tenure	Usage Frequency	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interaction	Churn
0	2.0	30.0	Female	39.0	14.0	5.0	18.0	Standard	Annual	932.0	17.0	1.0
1	3.0	65.0	Female	49.0	1.0	10.0	8.0	Basic	Monthly	557.0	6.0	1.0
2	4.0	55.0	Female	14.0	4.0	6.0	18.0	Basic	Quarterly	185.0	3.0	1.0
3	5.0	58.0	Male	38.0	21.0	7.0	7.0	Standard	Monthly	396.0	29.0	1.0
4	6.0	23.0	Male	32.0	20.0	5.0	8.0	Basic	Monthly	617.0	20.0	1.0

```
# Check for missing values in the specified columns
missing_values = df[['Total Spend', 'Last Interaction', 'Payment Delay']].isnull().sum()

# Display the number of missing values for each column
print("Missing values in specified columns:")
display(missing_values)
```

```
Missing values in specified columns:
```

	0
Total Spend	1
Last Interaction	1
Payment Delay	1

dtype: int64

```
# Perform one-hot encoding on the specified categorical columns
# This will create new columns for each category in 'Gender', 'Subscription Type', and 'Contract Length'
# The values in the new columns will be True or False, which can be interpreted as 1 and 0
df_encoded = pd.get_dummies(df, columns=['Gender', 'Subscription Type', 'Contract Length'])

# Convert the boolean columns created by one-hot encoding to integers (1 for True, 0 for False)
for col in ['Gender_Female', 'Gender_Male', 'Subscription Type_Basic', 'Subscription Type_Premium', 'Subscription Type_Standar
    if col in df_encoded.columns:
        df_encoded[col] = df_encoded[col].astype(int)

# Display the first few rows of the encoded DataFrame
print("DataFrame after one-hot encoding:")
display(df_encoded.head())
```

```
DataFrame after one-hot encoding:
```

	CustomerID	Age	Tenure	Usage Frequency	Support Calls	Payment Delay	Total Spend	Last Interaction	Churn	Gender_Female	Gender_Male	Subscription Type_Basic	Su Ty
0	2.0	30.0	39.0	14.0	5.0	18.0	932.0	17.0	1.0	1	0	0	
1	3.0	65.0	49.0	1.0	10.0	8.0	557.0	6.0	1.0	1	0	1	
2	4.0	55.0	14.0	4.0	6.0	18.0	185.0	3.0	1.0	1	0	1	
3	5.0	58.0	38.0	21.0	7.0	7.0	396.0	29.0	1.0	0	1	0	
4	6.0	23.0	32.0	20.0	5.0	8.0	617.0	20.0	1.0	0	1	1	

```
from sklearn.preprocessing import StandardScaler

# Initialize the StandardScaler
scaler = StandardScaler()

# Select the numerical columns to normalize
numerical_cols = ['Usage Frequency', 'Total Spend']

# Apply StandardScaler to the selected columns
df_encoded[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])
```

```
# Display the first few rows of the DataFrame with normalized features
print("DataFrame after normalizing 'Usage Frequency' and 'Total Spend':")
display(df_encoded.head())
```

```
DataFrame after normalizing 'Usage Frequency' and 'Total Spend':
```

	CustomerID	Age	Tenure	Usage Frequency	Support Calls	Payment Delay	Total Spend	Last Interaction	Churn	Gender_Female	Gender_Male	Subscription Type_Basic
0	2.0	30.0	39.0	-0.210511	5.0	18.0	1.247427	17.0	1.0	1	0	0
1	3.0	65.0	49.0	-1.724562	10.0	8.0	-0.309865	6.0	1.0	1	0	1
2	4.0	55.0	14.0	-1.375166	6.0	18.0	-1.854698	3.0	1.0	1	0	1
3	5.0	58.0	38.0	0.604748	7.0	7.0	-0.978462	29.0	1.0	0	1	0
4	6.0	23.0	32.0	0.488282	5.0	8.0	-0.060698	20.0	1.0	0	1	1

```
# Check for duplicate rows in the DataFrame
num_duplicates = df_encoded.duplicated().sum()
print(f"Number of duplicate rows before removal: {num_duplicates}")

# Remove duplicate rows
df_cleaned = df_encoded.drop_duplicates()
print(f"Number of rows after removing duplicates: {df_cleaned.shape[0]}")

# Check data types of columns
print("\nData types of columns after removing duplicates:")
df_cleaned.info()
```

```
Number of duplicate rows before removal: 0
Number of rows after removing duplicates: 440833

Data types of columns after removing duplicates:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440833 entries, 0 to 440832
Data columns (total 17 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   CustomerID                  440832 non-null  float64
 1   Age                         440832 non-null  float64
 2   Tenure                      440832 non-null  float64
 3   Usage Frequency             440832 non-null  float64
 4   Support Calls               440832 non-null  float64
 5   Payment Delay               440832 non-null  float64
 6   Total Spend                 440832 non-null  float64
 7   Last Interaction            440832 non-null  float64
 8   Churn                       440832 non-null  float64
 9   Gender_Female               440833 non-null  int64
 10  Gender_Male                 440833 non-null  int64
 11  Subscription Type_Basic     440833 non-null  int64
 12  Subscription Type_Premium   440833 non-null  int64
 13  Subscription Type_Standard  440833 non-null  int64
 14  Contract Length_Annual      440833 non-null  int64
 15  Contract Length_Monthly     440833 non-null  int64
 16  Contract Length_Quarterly   440833 non-null  int64
dtypes: float64(9), int64(8)
memory usage: 57.2 MB
```

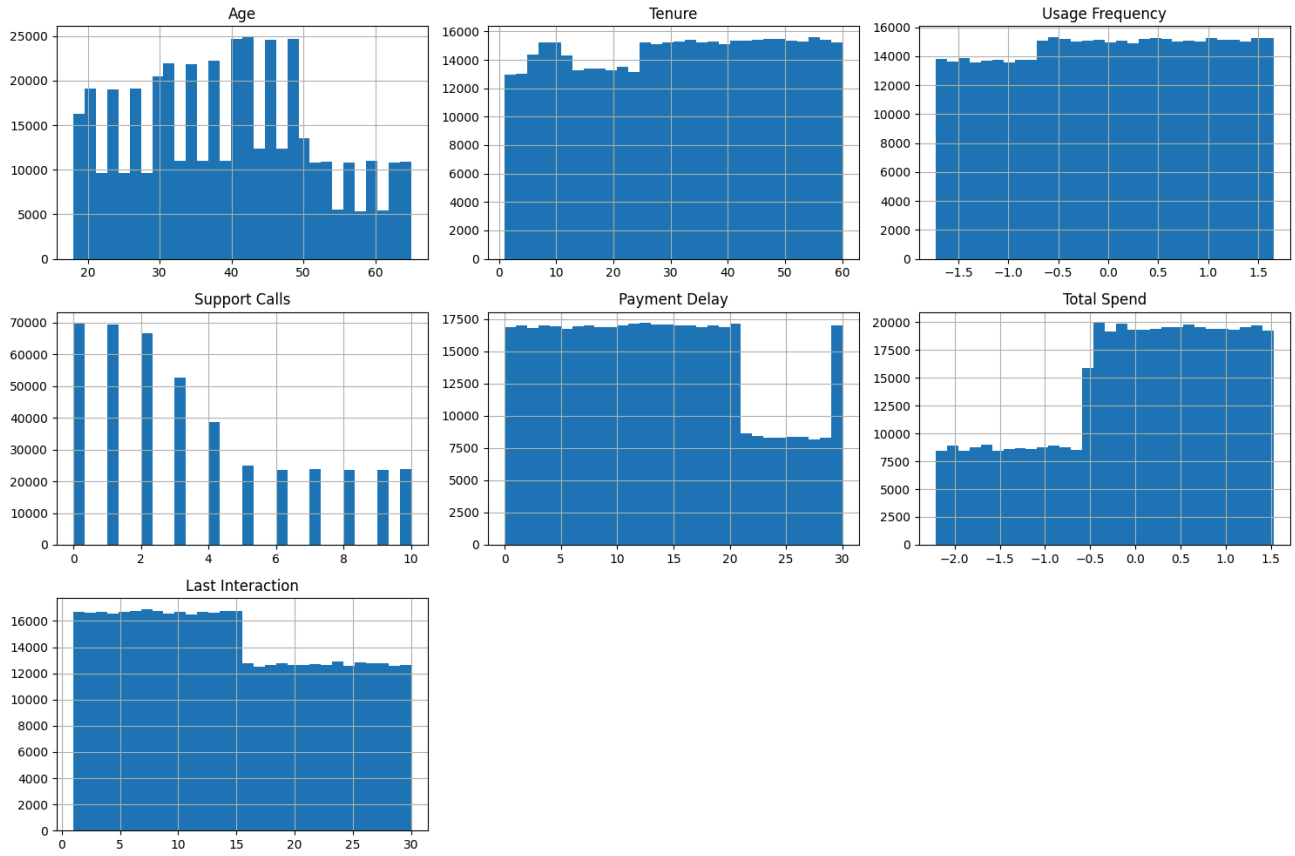
```
Start coding or generate with AI.
```

```
# Calculate and display summary statistics for numerical columns
print("Summary statistics for numerical columns:")
display(df_cleaned.describe())
```

Summary statistics for numerical columns:

	CustomerID	Age	Tenure	Usage Frequency	Support Calls	Payment Delay	Total Spend	Last Interaction	4408
count	440832.000000	440832.000000	440832.000000	4.408320e+05	440832.000000	440832.000000	4.408320e+05	440832.000000	
mean	225398.667955	39.373153	31.256336	-1.727873e-17	3.604437	12.965722	5.054674e-17	14.480868	
std	129531.918550	12.442369	17.255727	1.000001e+00	3.070218	8.258063	1.000001e+00	8.596208	
min	2.000000	18.000000	1.000000	-1.724562e+00	0.000000	0.000000	-2.207684e+00	1.000000	
25%	113621.750000	29.000000	16.000000	-7.928383e-01	1.000000	6.000000	-6.296283e-01	7.000000	
50%	226125.500000	39.000000	32.000000	2.242036e-02	3.000000	12.000000	1.220243e-01	14.000000	
75%	337739.250000	48.000000	46.000000	8.376790e-01	6.000000	19.000000	8.238436e-01	22.000000	
max	449999.000000	65.000000	60.000000	1.652938e+00	10.000000	30.000000	1.529816e+00	30.000000	

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

# Select numerical columns for visualization (excluding CustomerID and Churn for now)
numerical_cols_viz = ['Age', 'Tenure', 'Usage Frequency', 'Support Calls', 'Payment Delay', 'Total Spend', 'Last Interaction']

# Create histograms for numerical features
df_cleaned[numerical_cols_viz].hist(bins=30, figsize=(15, 10))
plt.tight_layout()
plt.show()

# Alternatively, you can use box plots to visualize distributions and outliers
# plt.figure(figsize=(15, 10))
# sns.boxplot(data=df_cleaned[numerical_cols_viz])
# plt.title('Box Plots of Numerical Features')
# plt.show()
```




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

# Select the original categorical columns for exploration
categorical_cols = ['Gender', 'Subscription Type', 'Contract Length']
```

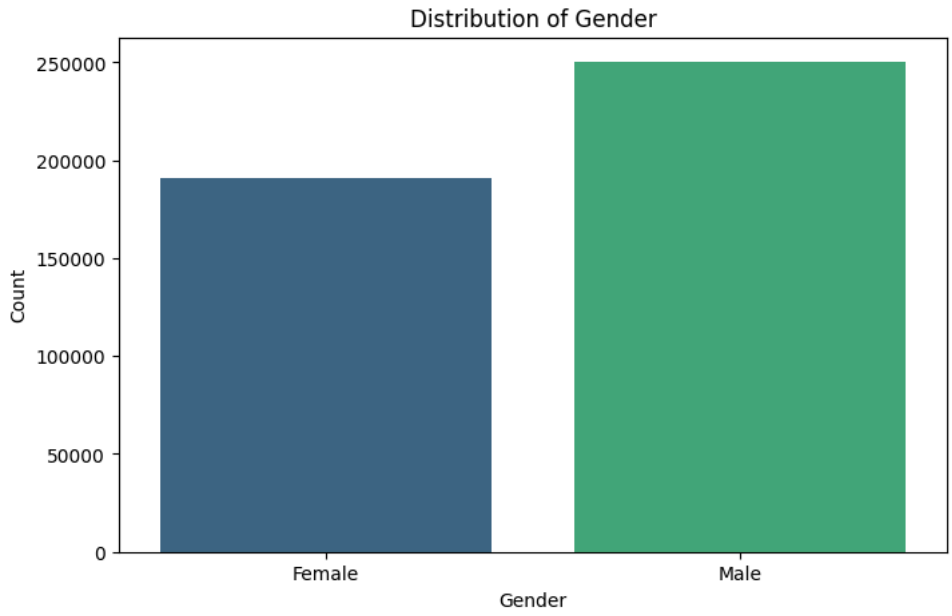
```
# Explore the distribution of each categorical variable
for col in categorical_cols:
    print(f"\nValue counts for {col}:")
    display(df[col].value_counts())

    plt.figure(figsize=(8, 5))
    sns.countplot(data=df, x=col, palette='viridis')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.show()
```

```
Value counts for Gender:
```

Gender	count
Male	250252
Female	190580

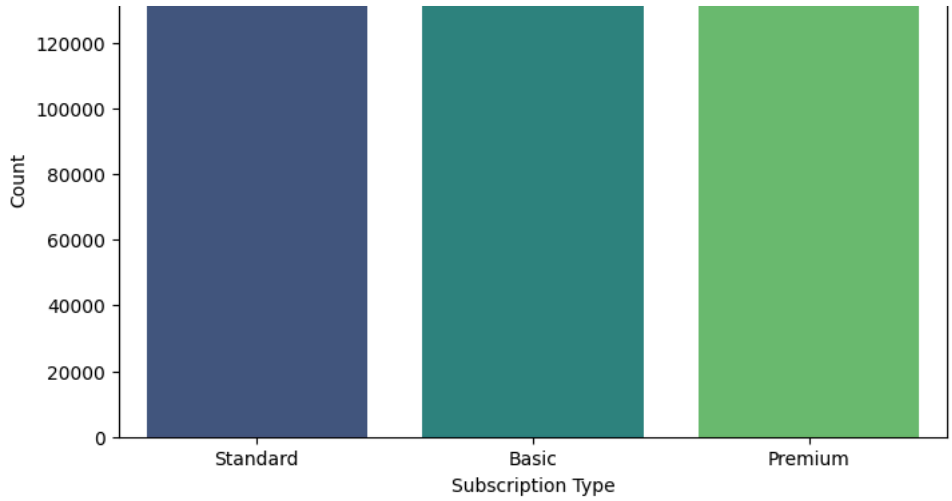
dtype: int64

```
/tmp/ipython-input-4145144485.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and se

  sns.countplot(data=df, x=col, palette='viridis')
```

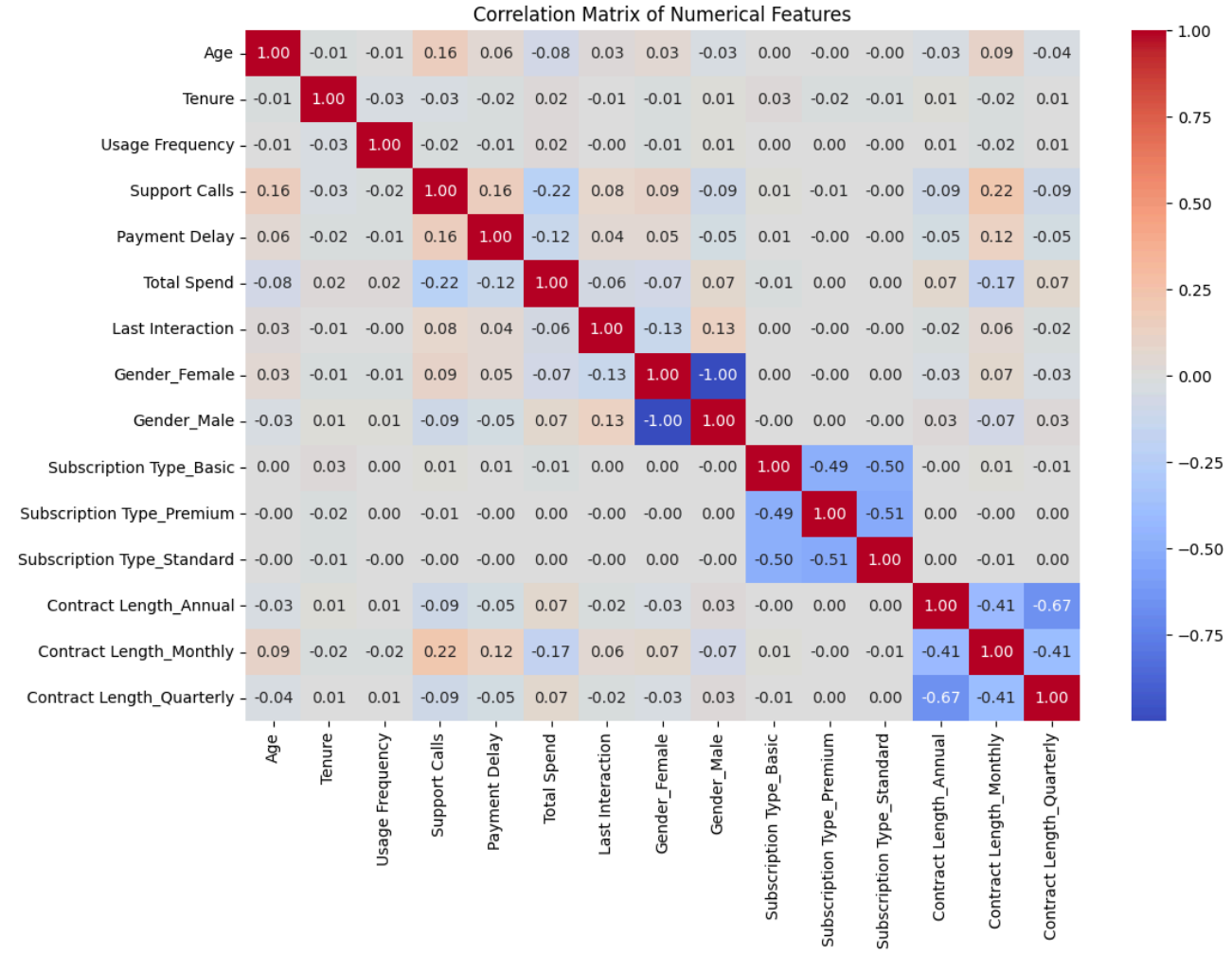



```
Value counts for Subscription Type:
```

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

# Select only numerical columns for correlation analysis, excluding the target variable 'Churn' and 'CustomerID'
numerical_cols_for_corr = df_cleaned.select_dtypes(include=['float64', 'int64']).columns.tolist()
if 'Churn' in numerical_cols_for_corr:
    numerical_cols_for_corr.remove('Churn')
if 'CustomerID' in numerical_cols_for_corr:
    numerical_cols_for_corr.remove('CustomerID')


# Calculate the correlation matrix
correlation_matrix = df_cleaned[numerical_cols_for_corr].corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```




```
Value counts for Contract Length:
```

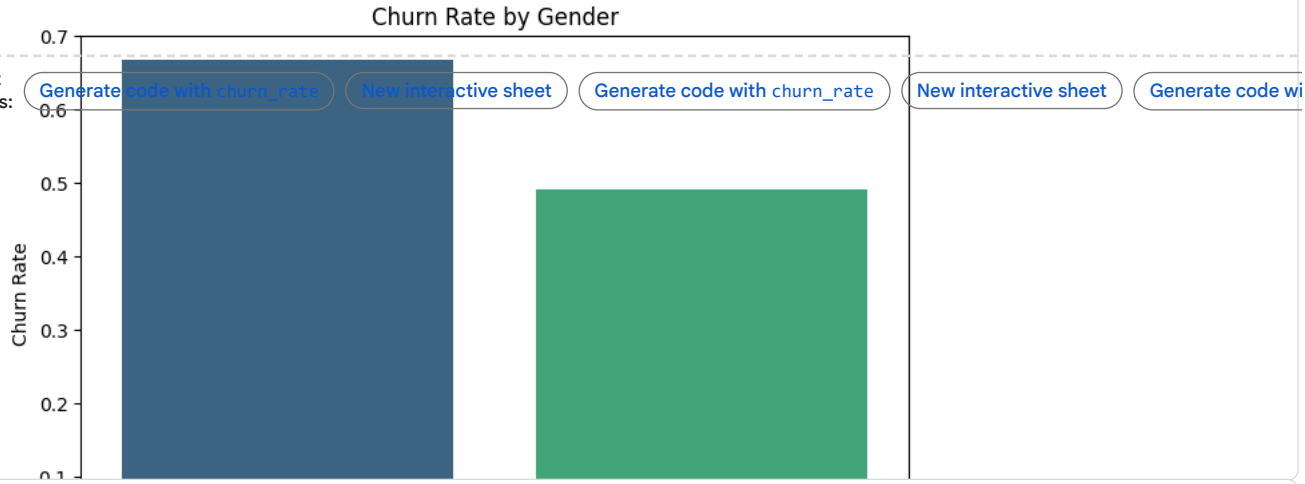



```
import matplotlib.pyplot as plt
import seaborn as sns

# Select the categorical columns and the target variable 'Churn'
categorical_cols = ['Gender', 'Subscription Type', 'Contract Length']
target = 'Churn'

# Analyze churn rate by each categorical feature
for col in categorical_cols:
    print(f"\nChurn rate by {col}:")
    churn_rate = df.groupby(col)[target].mean().reset_index()
    display(churn_rate)

    plt.figure(figsize=(8, 5))
    sns.barplot(data=churn_rate, x=col, y=target, palette='viridis')
    plt.title(f'Churn Rate by {col}')
    plt.xlabel(col)
    plt.ylabel('Churn Rate')
    plt.show()
```

```
Churn rate by Gender:
```

	Gender	Churn
0	Female	0.666691
1	Male	0.491269

```
/tmp/ipython-input-4207242613.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and se

  sns.barplot(data=churn_rate, x=col, y=target, palette='viridis')
```




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

# Churn rate by contract type and tenure
print("Churn rate by Contract Length and Tenure (mean tenure for each contract type and churn combination):")
churn_tenure_contract = df.groupby(['Contract Length', 'Churn'])['Tenure'].mean().reset_index()
display(churn_tenure_contract)

plt.figure(figsize=(10, 6))
sns.barplot(data=churn_tenure_contract, x='Contract Length', y='Tenure', hue='Churn', palette='viridis')
plt.title('Mean Tenure by Contract Length and Churn')
plt.xlabel('Contract Length')
plt.ylabel('Mean Tenure')
plt.show()

# Boxplots for spend vs churn
print("\nBoxplots for Total Spend vs Churn:")
plt.figure(figsize=(8, 6))
sns.boxplot(data=df_cleaned, x='Churn', y='Total Spend', palette='viridis')
plt.title('Total Spend vs Churn')
plt.xlabel('Churn (0: No, 1: Yes)')
plt.ylabel('Total Spend (Normalized)')
plt.xticks([0, 1], ['No Churn', 'Churn'])
plt.show()

# Heatmap of feature correlations (revisiting the previous correlation analysis for completeness)
print("\nHeatmap of feature correlations:")
# Select only numerical columns for correlation analysis, excluding the target variable 'Churn' and 'CustomerID'
numerical_cols_for_corr = df_cleaned.select_dtypes(include=['float64', 'int64']).columns.tolist()
if 'Churn' in numerical_cols_for_corr:
    numerical_cols_for_corr.remove('Churn')
if 'CustomerID' in numerical_cols_for_corr:
    numerical_cols_for_corr.remove('CustomerID')

# Calculate the correlation matrix
correlation_matrix = df_cleaned[numerical_cols_for_corr].corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()


Churn rate by Contract Length:
```

	Contract Length	Churn
0	Annual	0.460761
1	Monthly	1.000000
2	Quarterly	0.460256

```
/tmp/ipython-input-4207242613.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and se

  sns.barplot(data=churn_rate, x=col, y=target, palette='viridis')
```