

```
[1]: import pandas as pd  
import numpy as np  
  
[2]: df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')  
  
[3]: print(df.info())  
print(df.describe())  
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   customerID      7043 non-null    object  
 1   gender          7043 non-null    object  
 2   SeniorCitizen   7043 non-null    int64  
 3   Partner         7043 non-null    object  
 4   Dependents     7043 non-null    object  
 5   tenure          7043 non-null    int64  
 6   PhoneService    7043 non-null    object  
 7   MultipleLines   7043 non-null    object  
 8   InternetService 7043 non-null    object  
 9   OnlineSecurity  7043 non-null    object  
 10  OnlineBackup    7043 non-null    object  
 11  DeviceProtection 7043 non-null    object  
 12  TechSupport    7043 non-null    object  
 13  StreamingTV     7043 non-null    object  
 14  StreamingMovies  7043 non-null    object  
 15  Contract        7043 non-null    object  
 16  PaperlessBilling 7043 non-null    object  
 17  PaymentMethod   7043 non-null    object  
 18  MonthlyCharges  7043 non-null    float64 
 19  TotalCharges    7043 non-null    object  
 20  Churn           7043 non-null    object  
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None
```

```
...  
    SeniorCitizen          tenure  MonthlyCharges  
count      7043.000000  7043.000000      7043.000000  
mean       0.162147     32.371149      64.761692  
std        0.368612     24.559481      30.090047  
min        0.000000     0.000000      18.250000  
25%       0.000000     9.000000      35.500000  
50%       0.000000    29.000000      70.350000  
75%       0.000000    55.000000      89.850000  
max       1.000000    72.000000     118.750000  
customerID          0  
gender            0  
SeniorCitizen      0  
Partner           0  
Dependents        0  
tenure            0  
PhoneService       0  
MultipleLines       0  
InternetService     0  
OnlineSecurity      0  
OnlineBackup        0  
DeviceProtection    0  
TechSupport         0  
StreamingTV         0  
StreamingMovies      0  
Contract           0  
PaperlessBilling    0  
PaymentMethod       0  
MonthlyCharges      0  
TotalCharges        0  
Churn              0  
dtype: int64
```

```
[4]: df['TotalCharges'] = df['TotalCharges'].replace(' ', np.nan).astype(float)
df['TotalCharges'] = df['TotalCharges'].fillna(df['MonthlyCharges'])

[5]: df['Churn'] = df['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)

[6]: import sqlite3

[7]: conn = sqlite3.connect('telecom_churn.db')
df.to_sql('customers', conn, if_exists='replace', index=False)

[7]: 7043

[8]: # Example SQL query
query = """
SELECT
    InternetService,
    AVG(MonthlyCharges) as avg_monthly_charge,
    SUM(Churn) as churn_count,
    COUNT(*) as total_customers,
    SUM(Churn)*100.0/COUNT(*) as churn_rate
FROM customers
GROUP BY InternetService
"""

pd.read_sql(query, conn)
```

```
[8]:
```

	InternetService	avg_monthly_charge	churn_count	total_customers	churn_rate
0	DSL	58.102169	459	2421	18.959108
1	Fiber optic	91.500129	1297	3096	41.892765
2	No	21.079194	113	1526	7.404980

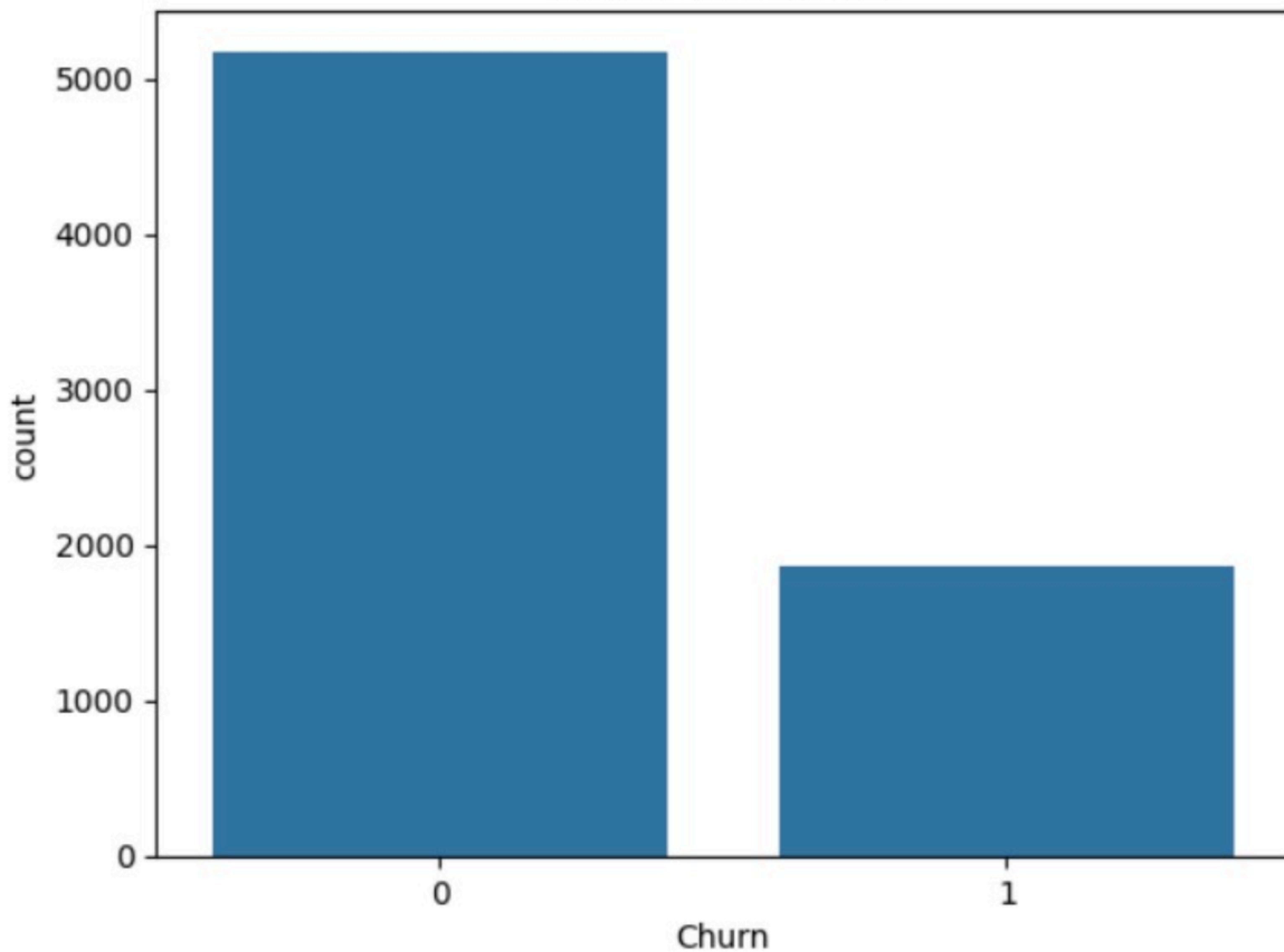
```
[9]:
```

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

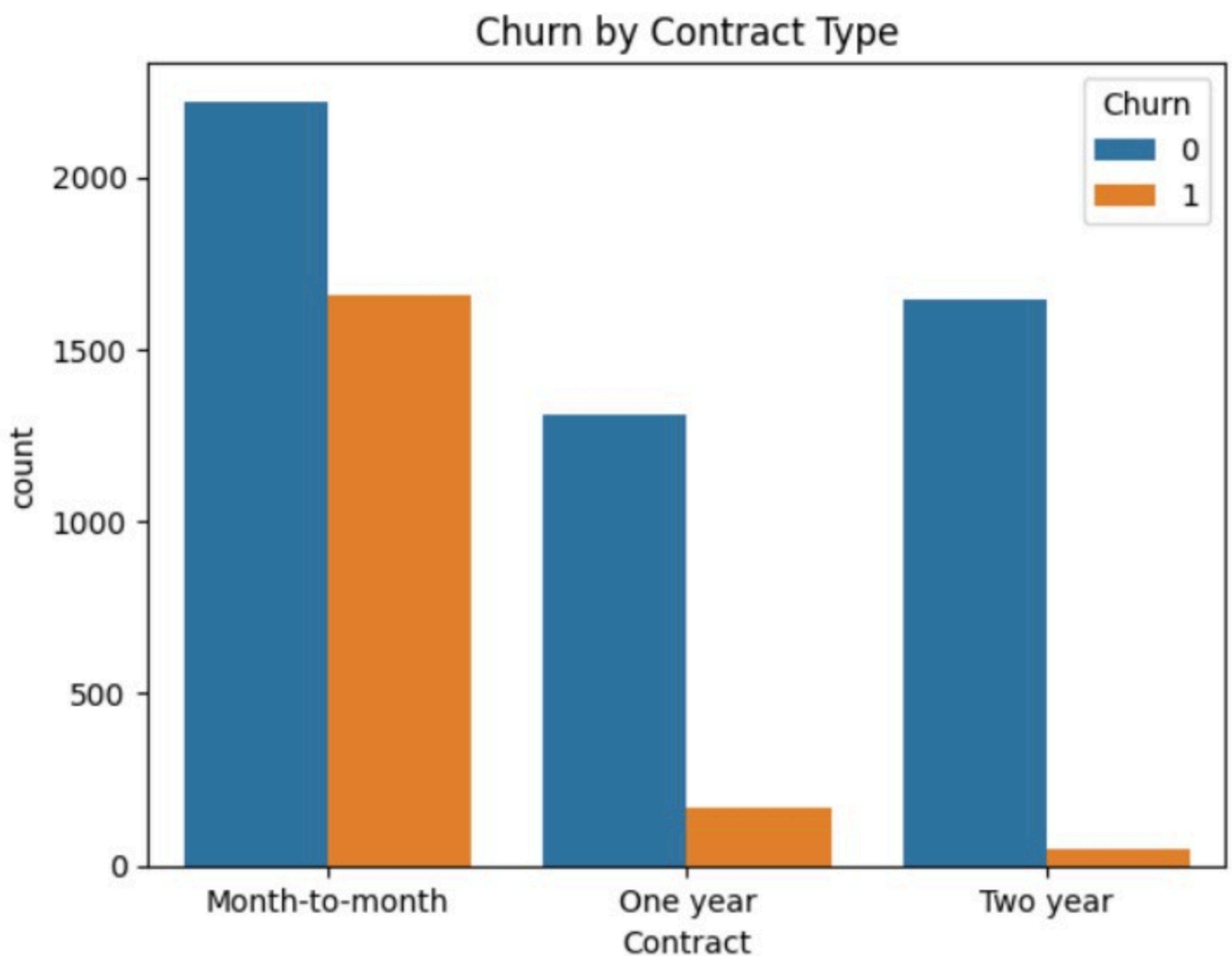
```
[10]:
```

```
sns.countplot(x='Churn', data=df)
plt.title('Churn Distribution')
plt.show()
```

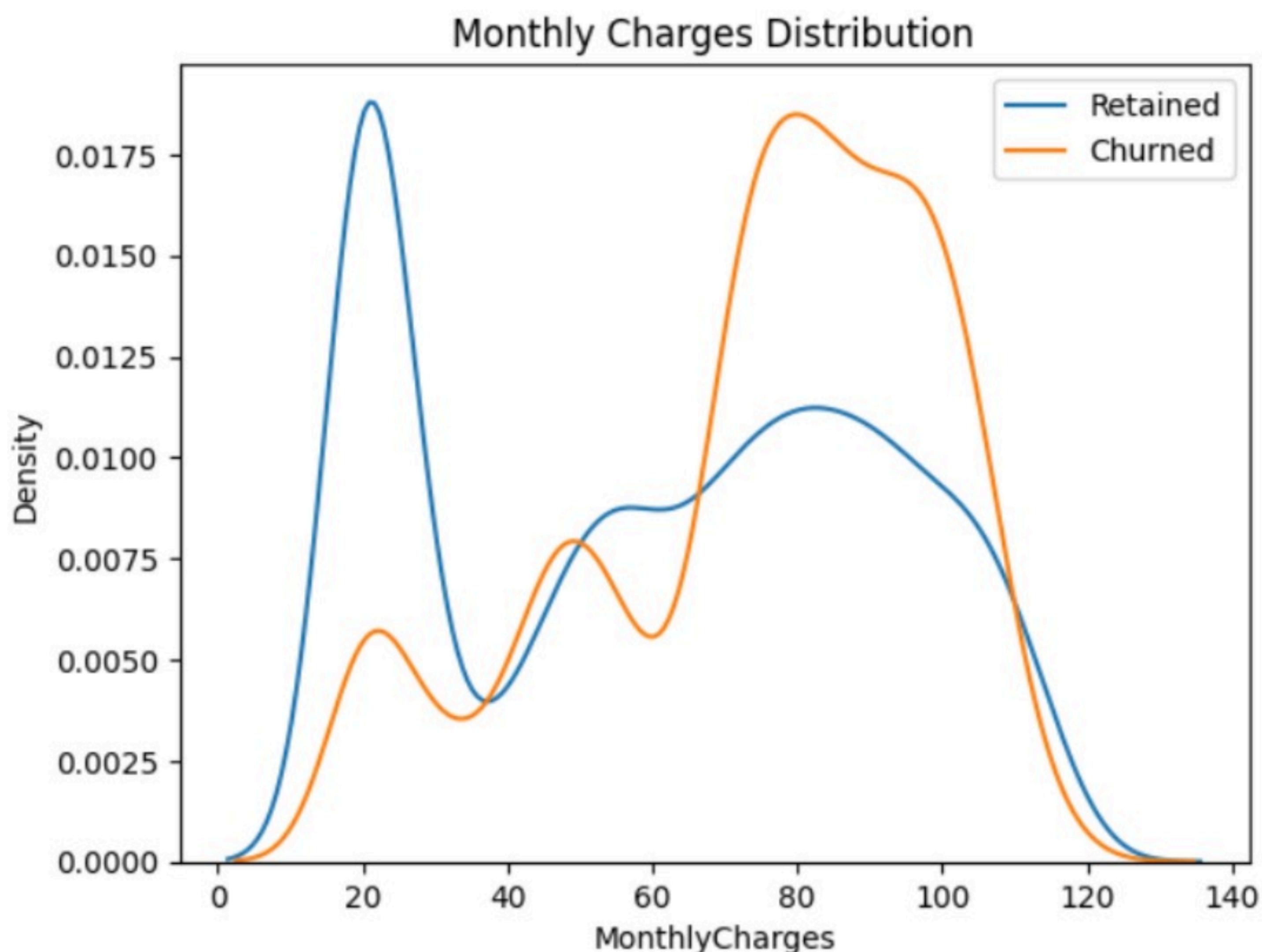
Churn Distribution



```
11]: sns.countplot(x='Contract', hue='Churn', data=df)
plt.title('Churn by Contract Type')
plt.show()
```



```
[12]: # Monthly charges distribution for churned vs retained
sns.kdeplot(df[df['Churn']==0]['MonthlyCharges'], label='Retained')
sns.kdeplot(df[df['Churn']==1]['MonthlyCharges'], label='Churned')
plt.title('Monthly Charges Distribution')
plt.legend()
plt.show()
```



```
[13]: from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

[14]: # Encode categorical variables
cat_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
            'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
            'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
            'PaperlessBilling', 'PaymentMethod']

le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

[15]: # Select features and target
X = df.drop(['customerID', 'Churn'], axis=1)
y = df['Churn']

[16]: # Scale numerical features
scaler = StandardScaler()
num_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
X[num_cols] = scaler.fit_transform(X[num_cols])

[17]: # Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[19]: from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score
```

```
[20]: # Random Forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print("\nRandom Forest:")
print(classification_report(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1]))
```

Random Forest:

	precision	recall	f1-score	support
0	0.83	0.91	0.87	1036
1	0.66	0.48	0.55	373
accuracy			0.80	1409
macro avg	0.74	0.69	0.71	1409
weighted avg	0.78	0.80	0.78	1409

ROC-AUC: 0.8326091794590453

```
[23]: # Add predicted probabilities to dataframe
df['Churn_Probability'] = rf.predict_proba(X)[:, 1]
```

```
[24]: # Create segments
def create_segments(row):
    if row['Churn_Probability'] > 0.7:
        return 'High Risk'
    elif row['Churn_Probability'] > 0.4:
        return 'Medium Risk'
    elif row['tenure'] > 24:
        return 'Loyal'
    else:
        return 'Low Risk'

df['Segment'] = df.apply(create_segments, axis=1)
```

```
[25]: # Analyze segments
segment_analysis = df.groupby('Segment').agg({
    'Churn': 'mean',
    'MonthlyCharges': 'mean',
    'tenure': 'mean',
    'customerID': 'count'
}).rename(columns={'customerID': 'Count'})
```