```
import pandas as pd

# Load the dataset
df = pd.read_csv('https://github.com/dpak141/telco/blob/main/dataset_iran.csv?raw=true')



# Exploratory Data Analysis

# Display first 10 rows
print("First 10 rows of the DataFrame:")
print(df.head(10))
```

```
First 10 rows of the DataFrame:
   Call  Failure  Complains  Subscription  Length  Charge  Amount  \
0              8          0                    38               0
1              0          0                    39               0
2             10          0                    37               0
3             10          0                    38               0
4              3          0                    38               0
5             11          0                    38               1
6              4          0                    38               0
7             13          0                    37               2
8              7          0                    38               0
9              7          0                    38               1

   Seconds of Use  Frequency of use  Frequency of SMS  \
0            4370                71                 5
1             318                 5                 7
2            2453                60               359
3            4198                66                 1
4            2393                58                 2
5            3775                82                32
6            2360                39               285
7            9115               121               144
8           13773               169                 0
9            4515                83                 2

   Distinct Called Numbers  Age Group  Tariff Plan  Status  Age  \
0                       17          3            1       1   30
1                        4          2            1       2   25
2                       24          3            1       1   30
3                       35          1            1       1   15
4                       33          1            1       1   15
5                       28          3            1       1   30
6                       18          3            1       1   30
7                       43          3            1       1   30
8                       44          3            1       1   30
9                       25          3            1       1   30

   Customer Value  Churn
0         197.640      0
1          46.035      0
2        1536.520      0
3         240.020      0
4         145.805      0
5         282.280      0
6        1235.960      0
7         945.440      0
8         557.680      0
9         191.920      0
```

```
# Display shape
print("\nShape of the DataFrame (rows, columns):")
print(df.shape)
```

```

Shape of the DataFrame (rows, columns):
(3150, 14)
```

```
# Display summary statistics
print("\nSummary statistics of the DataFrame:")
print(df.describe())
```

```

Summary statistics of the DataFrame:
       Call  Failure    Complains  Subscription  Length  Charge  Amount  \
count    3150.000000  3150.000000           3150.000000     3150.000000
```

```
mean        7.627937     0.076508            32.541905        0.942857
std         7.263886     0.265851             8.573482        1.521072
min         0.000000     0.000000             3.000000        0.000000
25%         1.000000     0.000000            30.000000        0.000000
50%         6.000000     0.000000            35.000000        0.000000
75%        12.000000     0.000000            38.000000        1.000000
max        36.000000     1.000000            47.000000       10.000000

       Seconds of Use  Frequency of use  Frequency of SMS  \
count     3150.000000       3150.000000       3150.000000
mean      4472.459683         69.460635         73.174921
std       4197.908687         57.413308        112.237560
min          0.000000          0.000000          0.000000
25%       1391.250000         27.000000          6.000000
50%       2990.000000         54.000000         21.000000
75%       6478.250000         95.000000         87.000000
max      17090.000000        255.000000        522.000000

       Distinct Called Numbers    Age Group  Tariff Plan       Status  \
count              3150.000000  3150.000000  3150.000000  3150.000000
mean                 23.509841     2.826032     1.077778     1.248254
std                  17.217337     0.892555     0.267864     0.432069
min                   0.000000     1.000000     1.000000     1.000000
25%                  10.000000     2.000000     1.000000     1.000000
50%                  21.000000     3.000000     1.000000     1.000000
75%                  34.000000     3.000000     1.000000     1.000000
max                  97.000000     5.000000     2.000000     2.000000

                Age  Customer Value        Churn
count  3150.000000     3150.000000  3150.000000
mean     30.998413      470.972916     0.157143
std       8.831095      517.015433     0.363993
min      15.000000        0.000000     0.000000
25%      25.000000      113.801250     0.000000
50%      30.000000      228.480000     0.000000
75%      30.000000      788.388750     0.000000
max      55.000000     2165.280000     1.000000
```

```
# Display DataFrame info
print("\nInformation about the DataFrame:")
df.info()
```

```
Information about the DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Call  Failure            3150 non-null   int64
 1   Complains                3150 non-null   int64
 2   Subscription  Length     3150 non-null   int64
 3   Charge  Amount           3150 non-null   int64
 4   Seconds of Use           3150 non-null   int64
 5   Frequency of use         3150 non-null   int64
 6   Frequency of SMS         3150 non-null   int64
 7   Distinct Called Numbers  3150 non-null   int64
 8   Age Group                3150 non-null   int64
 9   Tariff Plan              3150 non-null   int64
 10  Status                   3150 non-null   int64
 11  Age                      3150 non-null   int64
 12  Customer Value           3150 non-null   float64
 13  Churn                    3150 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 344.7 KB
```

```
# Check for missing values
print("\nNumber of missing values for each column:")
print(df.isnull().sum())
```

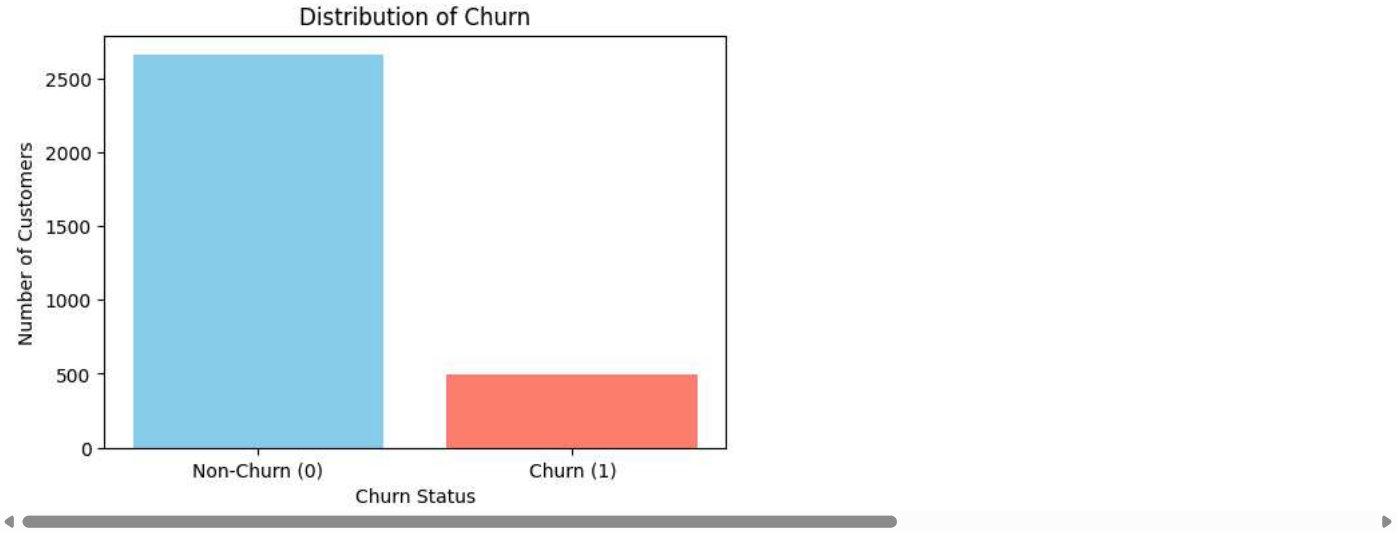
```
Number of missing values for each column:
Call  Failure              0
Complains                  0
Subscription  Length       0
Charge  Amount             0
Seconds of Use             0
Frequency of use           0
Frequency of SMS           0
Distinct Called Numbers    0
Age Group                  0
Tariff Plan                0
```

```
Status                    0
Age                       0
Customer Value            0
Churn                     0
dtype: int64
```

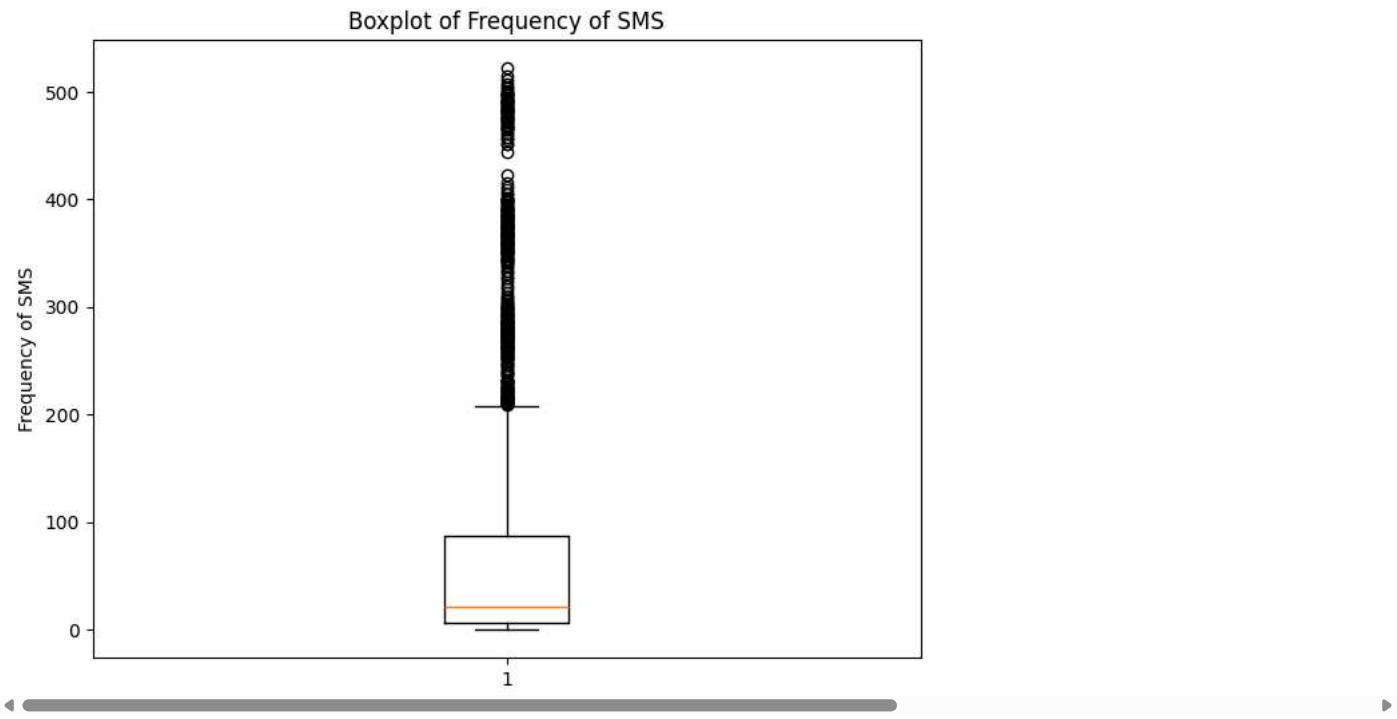
```
import matplotlib.pyplot as plt

# Plot churn distribution
print("\nBar plot for 'Churn' column:")
churn_counts = df['Churn'].value_counts()
plt.figure(figsize=(6, 4))
plt.bar(churn_counts.index.astype(str), churn_counts.values, color=['skyblue', 'salmon'])
plt.xticks([0, 1], ['Non-Churn (0)', 'Churn (1)'])
plt.xlabel('Churn Status')
plt.ylabel('Number of Customers')
plt.title('Distribution of Churn')
plt.show()
```

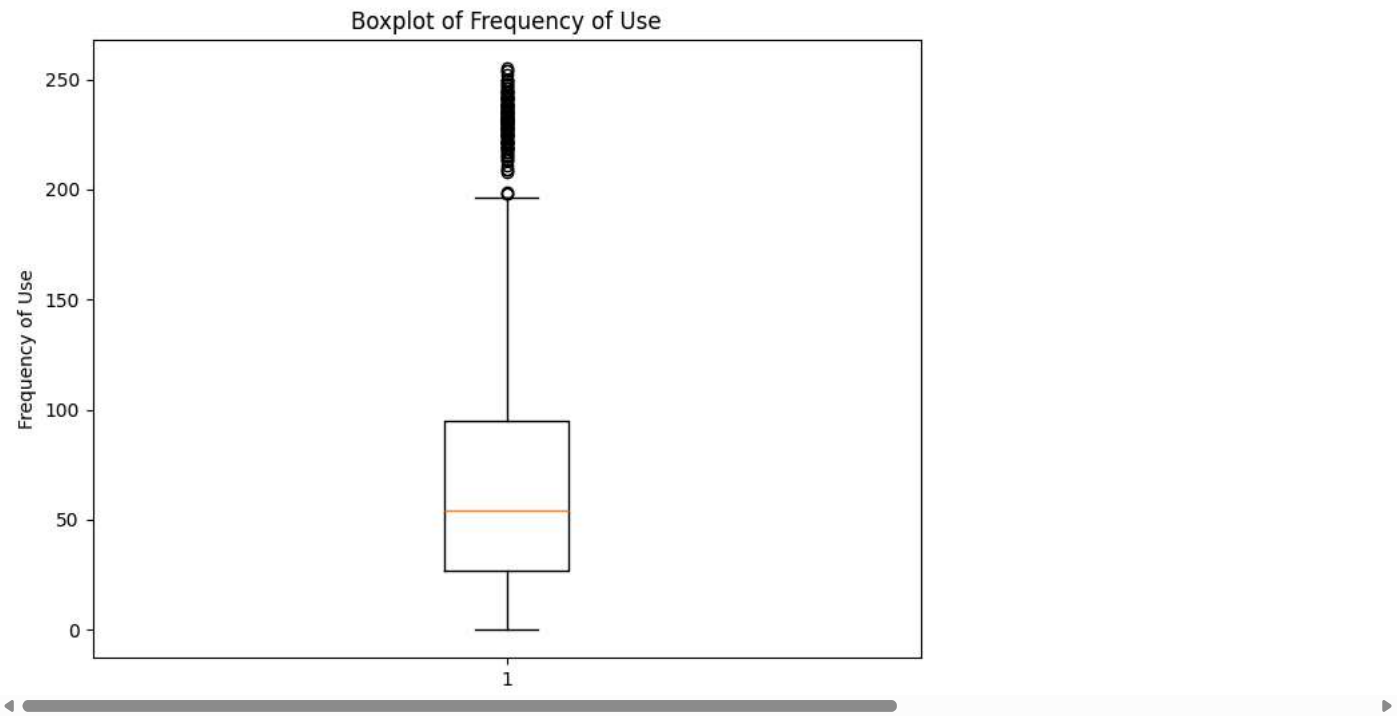
```
Bar plot for 'Churn' column:
```



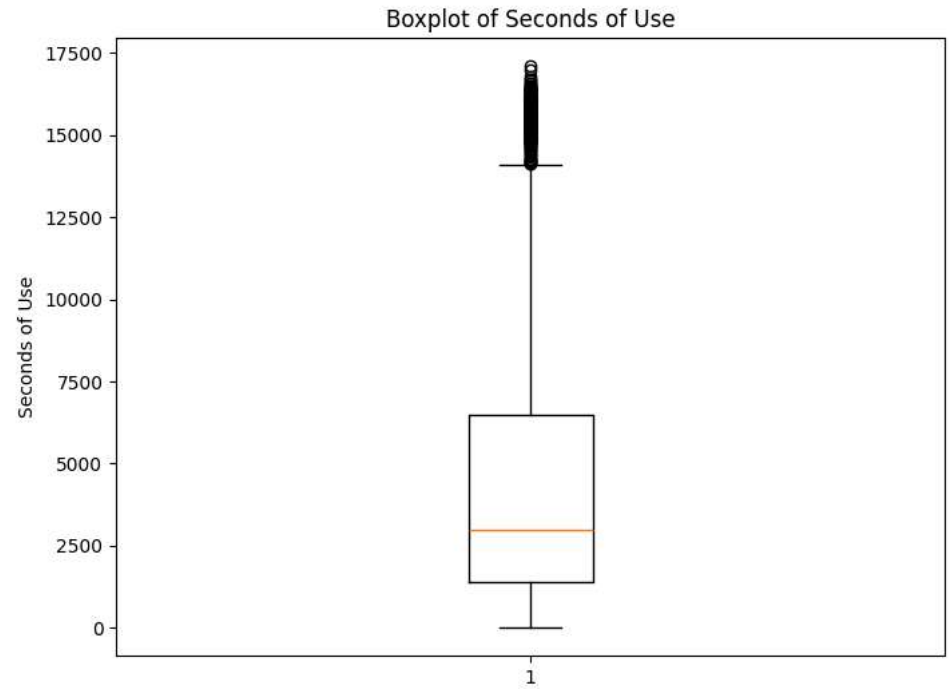

```
# Boxplot of 'Frequency of SMS'
plt.figure(figsize=(8, 6))
plt.boxplot(df['Frequency of SMS'])
plt.title('Boxplot of Frequency of SMS')
plt.ylabel('Frequency of SMS')
plt.show()
```




```
# Boxplot of 'Frequency of use'
plt.figure(figsize=(8, 6))
plt.boxplot(df['Frequency of use'])
plt.title('Boxplot of Frequency of Use')
plt.ylabel('Frequency of Use')
plt.show()
```

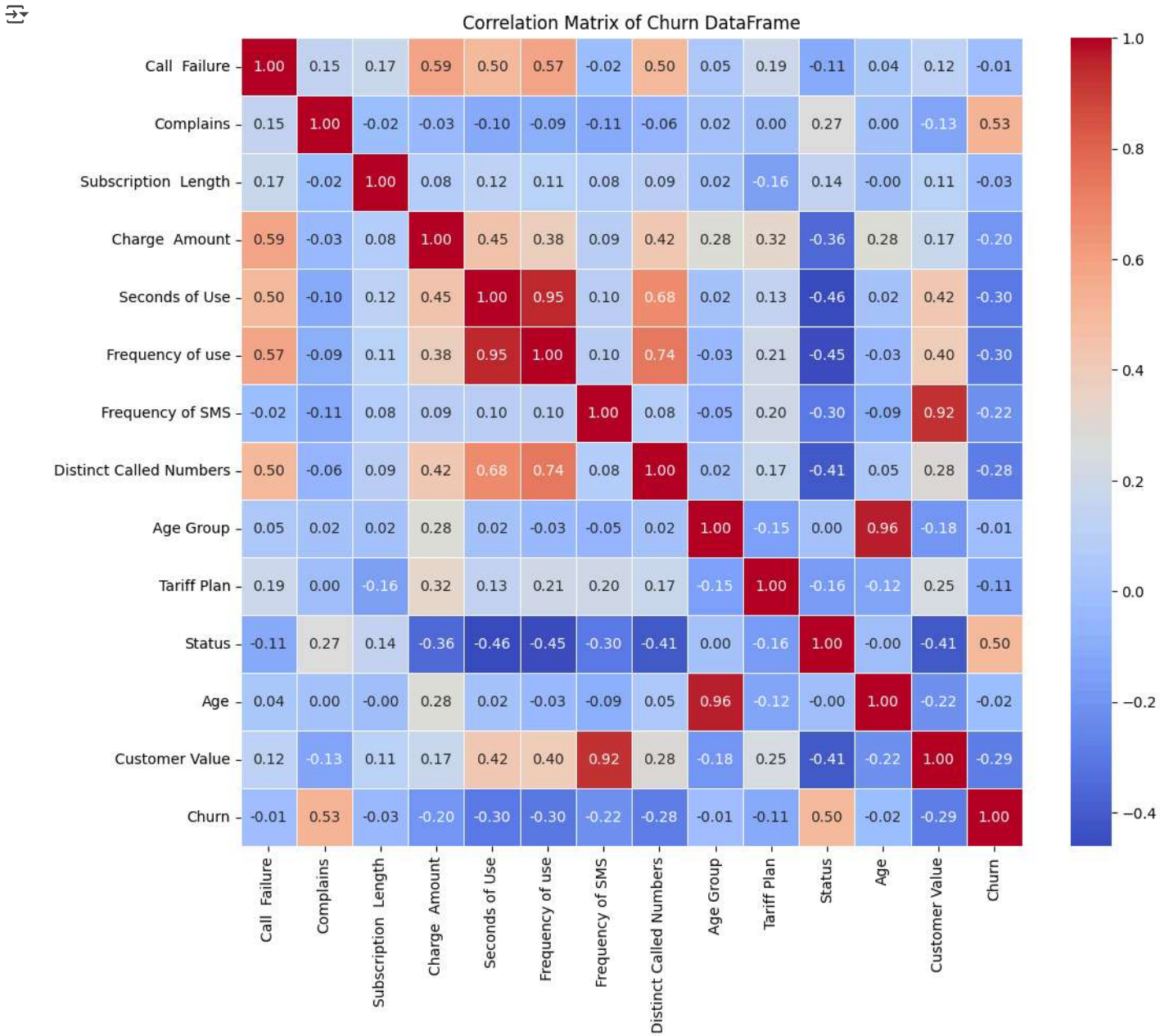



```
# Boxplot of 'Seconds of Use'
plt.figure(figsize=(8, 6))
plt.boxplot(df['Seconds of Use'])
plt.title('Boxplot of Seconds of Use')
plt.ylabel('Seconds of Use')
plt.show()
```




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

# Plot correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix of Churn DataFrame')
plt.show()
```




```
# Feature Engineering

# Drop highly correlated columns
df.drop(["Age Group", "Frequency of use"], inplace=True, axis=1)
print("First 5 rows of the DataFrame after dropping columns:")
print(df.head())
```

```
First 5 rows of the DataFrame after dropping columns:
   Call  Failure  Complains  Subscription  Length  Charge  Amount  \
0              8          0                    38               0
1              0          0                    39               0
2             10          0                    37               0
3             10          0                    38               0
4              3          0                    38               0

   Seconds of Use  Frequency of SMS  Distinct Called Numbers  Tariff Plan  \
0            4370                 5                       17            1
1             318                 7                        4            1
2            2453               359                       24            1
3            4198                 1                       35            1
4            2393                 2                       33            1

   Status  Age  Customer Value  Churn
0       1   30         197.640      0
```

```
1       2   25          46.035      0
2       1   30        1536.520      0
3       1   15         240.020      0
4       1   15         145.805      0
```

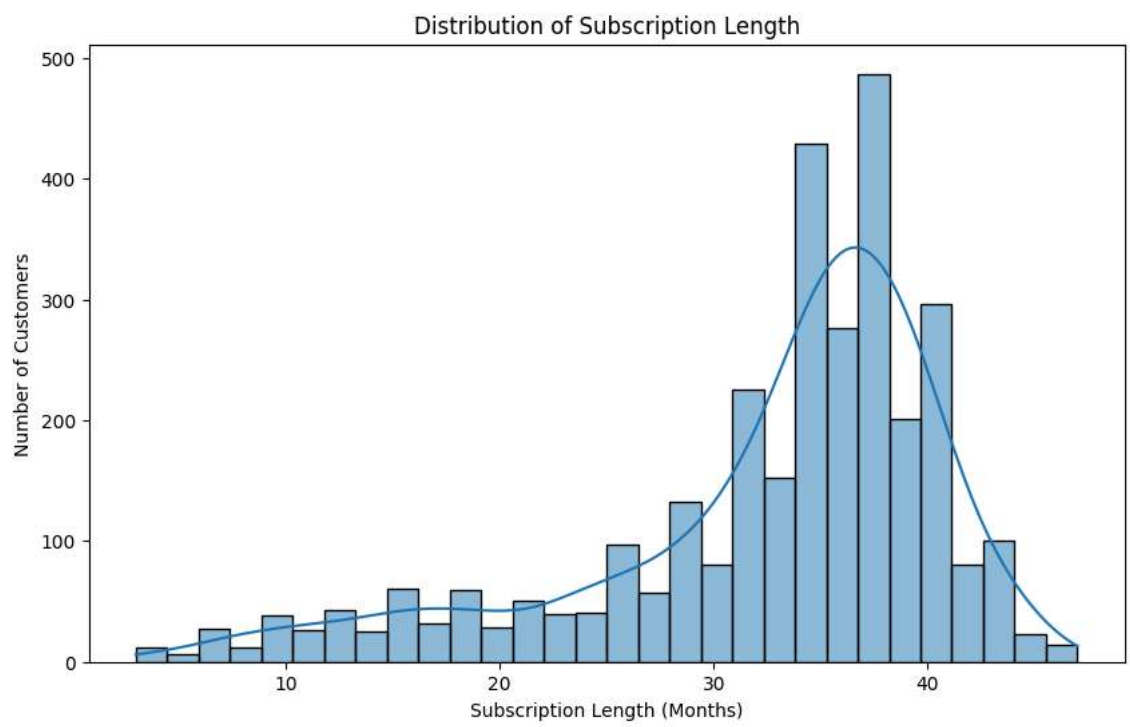
```
# more eda
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('https://github.com/dpak141/telco/blob/main/dataset_iran.csv?raw=true')

print("--- New Exploratory Data Analysis ---")
```

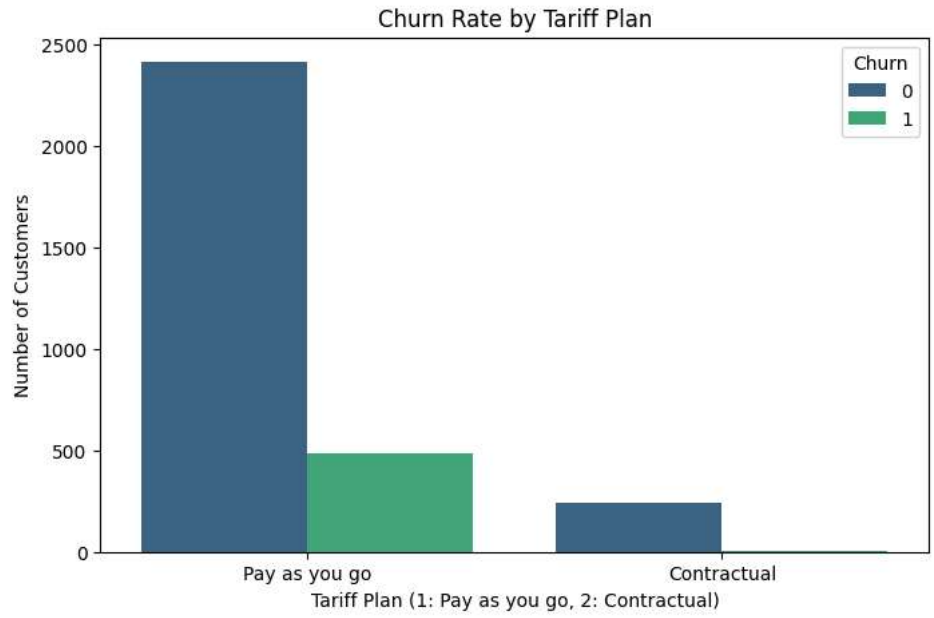
```
--- New Exploratory Data Analysis ---
```

```
# 1. Distribution of 'Subscription Length'
plt.figure(figsize=(10, 6))
sns.histplot(df['Subscription  Length'], bins=30, kde=True)
plt.title('Distribution of Subscription Length')
plt.xlabel('Subscription Length (Months)')
plt.ylabel('Number of Customers')
plt.show()
print("Observation: The majority of customers have shorter subscription lengths, with a peak around 10-20 months. There's a decreasing trend
```



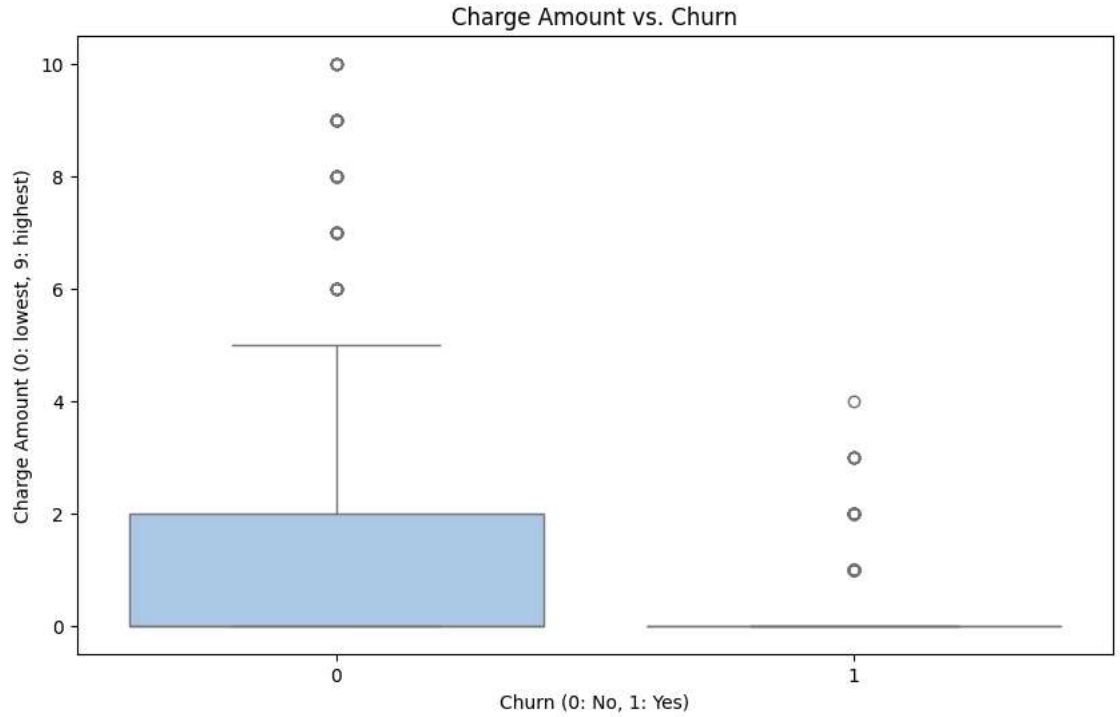

```
Observation: The majority of customers have shorter subscription lengths, with a peak around 10-20 months. There's a decreasing trend as
```

```
# 2. Churn Rate by 'Tariff Plan'
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='Tariff Plan', hue='Churn', palette='viridis')
plt.title('Churn Rate by Tariff Plan')
plt.xlabel('Tariff Plan (1: Pay as you go, 2: Contractual)')
plt.ylabel('Number of Customers')
plt.xticks(ticks=[0, 1], labels=['Pay as you go', 'Contractual'])
plt.show()
print("Observation: Customers on 'Pay as you go' tariff plan (1) appear to have a higher absolute number of churned customers compared to 'C
```



Observation: Customers on 'Pay as you go' tariff plan (1) appear to have a higher absolute number of churned customers compared to 'Cont

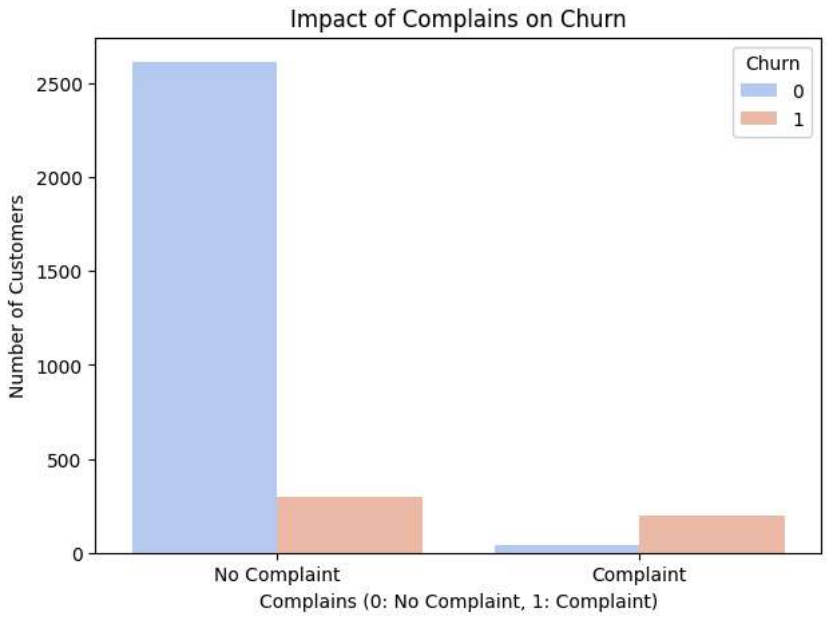
```
# 3. Relationship between 'Charge Amount' and 'Churn'
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Churn', y='Charge  Amount', palette='pastel')
plt.title('Charge Amount vs. Churn')
plt.xlabel('Churn (0: No, 1: Yes)')
plt.ylabel('Charge Amount (0: lowest, 9: highest)')
plt.show()
print("Observation: Churned customers tend to have a slightly lower median 'Charge Amount' compared to non-churned customers, although there's
```



Observation: Churned customers tend to have a slightly lower median 'Charge Amount' compared to non-churned customers, although there's

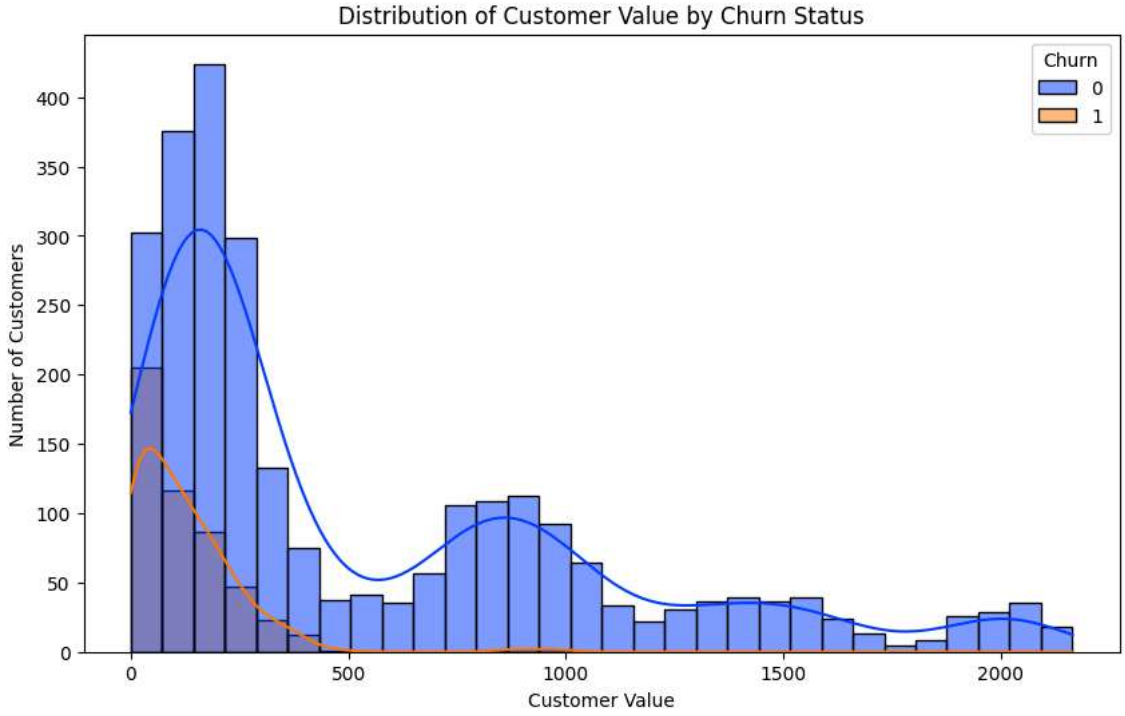
```
# 4. Impact of 'Complaints' on 'Churn'
plt.figure(figsize=(7, 5))
sns.countplot(data=df, x='Complains', hue='Churn', palette='coolwarm')
plt.title('Impact of Complains on Churn')
plt.xlabel('Complains (0: No Complaint, 1: Complaint)')
plt.ylabel('Number of Customers')
plt.xticks(ticks=[0, 1], labels=['No Complaint', 'Complaint'])
plt.show()
```

```
print("Observation: A significantly higher proportion of customers who registered a complaint (1) have churned compared to those who had no c
```




```
Observation: A significantly higher proportion of customers who registered a complaint (1) have churned compared to those who had no com
```

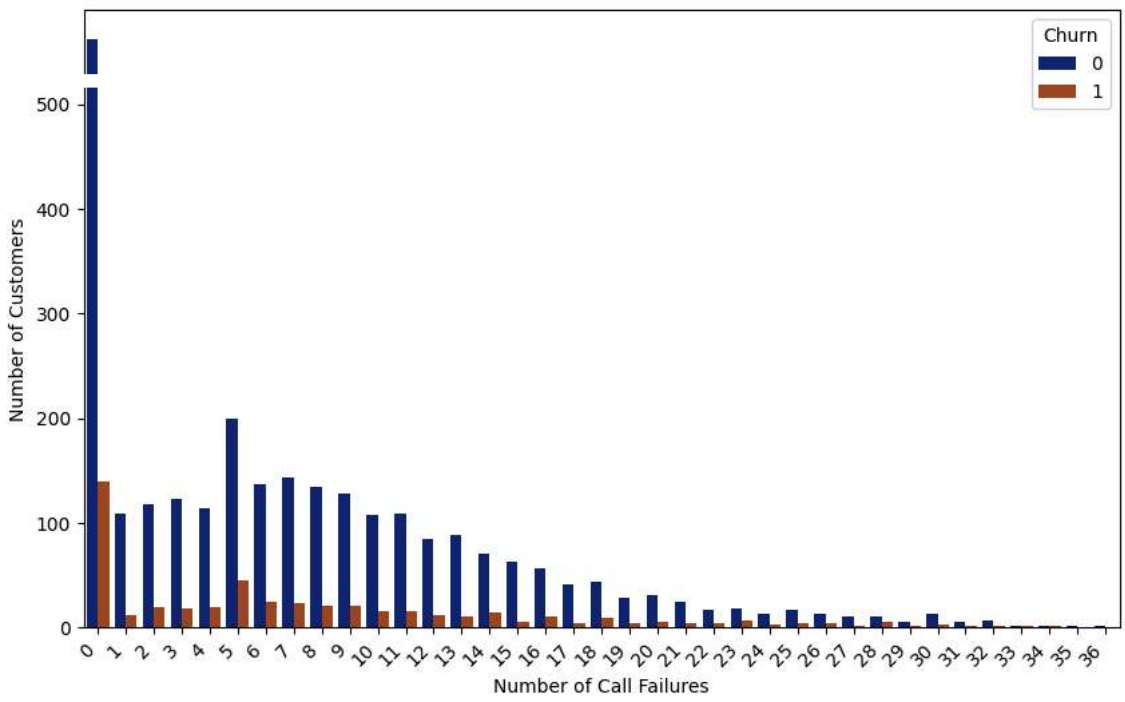
```
# 5. Distribution of 'Customer Value' for Churned vs. Non-Churned
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Customer Value', hue='Churn', bins=30, kde=True, palette='bright')
plt.title('Distribution of Customer Value by Churn Status')
plt.xlabel('Customer Value')
plt.ylabel('Number of Customers')
plt.show()
print("Observation: Non-churned customers generally have higher 'Customer Value' compared to churned customers, whose customer values are mo
```




```
Observation: Non-churned customers generally have higher 'Customer Value' compared to churned customers, whose customer values are more
```

```
# 6. 'Call Failure' vs. 'Churn' (considering discrete nature for countplot)
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Call  Failure', hue='Churn', palette='dark')
plt.title('Call Failure vs. Churn')
plt.xlabel('Number of Call Failures')
```

```
plt.ylabel('Number of Customers')
plt.xticks(rotation=45, ha='right')
plt.show()
print("Observation: As the number of call failures increases, the proportion of churned\n customers tends to rise. This indicates a strong r
```




```
Observation: As the number of call failures increases, the proportion of churned
 customers tends to rise. This indicates a strong relationship between service quality issues (represented by call failures) and custome
```

```
from sklearn.preprocessing import StandardScaler

# Define features to scale
features_to_scale = ['Age', 'Customer Value', 'Distinct Called Numbers', 'Frequency of SMS', 'Seconds of Use', 'Subscription  Length', 'Call

# Apply standard scaling
scaler = StandardScaler()
df[features_to_scale] = scaler.fit_transform(df[features_to_scale])
print("First 5 rows of the DataFrame after feature scaling:")
print(df.head())
```

```
First 5 rows of the DataFrame after feature scaling:
   Call  Failure  Complains  Subscription  Length  Charge  Amount  \
0       0.051229          0              0.636726               0
1      -1.050285          0              0.753384               0
2       0.326608          0              0.520069               0
3       0.326608          0              0.636726               0
4      -0.637217          0              0.636726               0

   Seconds of Use  Frequency of use  Frequency of SMS  \
0       -0.024411                71         -0.607513
1       -0.989807                 5         -0.589691
2       -0.481140                60          2.547012
3       -0.065390                66         -0.643157
4       -0.495435                58         -0.634246

   Distinct Called Numbers  Age Group  Tariff Plan  Status       Age  \
0                -0.378158          3            1       1 -0.113074
1                -1.133331          2            1       2 -0.679346
2                 0.028473          3            1       1 -0.113074
3                 0.667466          1            1       1 -1.811888
4                 0.551285          1            1       1 -1.811888

   Customer Value  Churn
0       -0.528759      0
1       -0.822036      0
2        2.061285      0
3       -0.446775      0
4       -0.629033      0
```

```
# Model Development

from sklearn.model_selection import train_test_split

# Separate target and features
y = df['Churn']
X = df.drop('Churn', axis=1)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (2520, 13)
Shape of X_test: (630, 13)
Shape of y_train: (2520,)
Shape of y_test: (630,)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import warnings
warnings.filterwarnings('ignore')

print("--- Logistic Regression ---")
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
lr_preds = lr.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, lr_preds))
print("Accuracy:", accuracy_score(y_test, lr_preds))
print("Classification Report:\n", classification_report(y_test, lr_preds))

from sklearn import svm
print("\n--- Support Vector Machine (SVM) ---")
Svm = svm.SVC(random_state=42)
Svm.fit(X_train, y_train)
svm_pred = Svm.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, svm_pred))
print("Accuracy:", accuracy_score(y_test, svm_pred))
print("Classification Report:\n", classification_report(y_test, svm_pred))

from sklearn.naive_bayes import GaussianNB
print("\n--- Gaussian Naive Bayes ---")
Gnb = GaussianNB()
Gnb.fit(X_train, y_train)
gnb_pred = Gnb.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, gnb_pred))
print("Accuracy:", accuracy_score(y_test, gnb_pred))
print("Classification Report:\n", classification_report(y_test, gnb_pred))

from sklearn.tree import DecisionTreeClassifier
print("\n--- Decision Tree Classifier ---")
DT = DecisionTreeClassifier(random_state=42)
DT.fit(X_train, y_train)
DT_pred = DT.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, DT_pred))
print("Accuracy:", accuracy_score(y_test, DT_pred))
print("Classification Report:\n", classification_report(y_test, DT_pred))
```

```
Classification Report:
                precision    recall  f1-score   support

             0       0.88      0.97      0.92       520
             1       0.73      0.40      0.52       110

      accuracy                           0.87       630
     macro avg       0.81      0.68      0.72       630
  weighted avg       0.86      0.87      0.85       630


--- Support Vector Machine (SVM) ---
```

	precision	recall	f1-score	support
0	0.83	1.00	0.90	520
1	0.00	0.00	0.00	110
accuracy			0.83	630
macro avg	0.41	0.50	0.45	630
weighted avg	0.68	0.83	0.75	630