```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
#from google.colab import drive
#drive.mount('/content/drive')
```

```
## Display all the columns of the dataframe
pd.pandas.set_option('display.max_columns',None)
```

```
df=pd.read_csv('/content/drive/MyDrive/PROJ-406-Capstone/telecom_customer_churn.csv')
```

```
df.head()
```

ing sic	Unlimited Data	Contract	Paperless Billing	Payment Method	Monthly Charge	Total Charges	Total Refunds	Total Extra Data Charges	Total Long Distance Charges	Total Revenue	Customer Status	Churn Category	Churn Reason
No	Yes	One Year	Yes	Credit Card	65.6	593.30	0.00	0	381.51	974.81	Stayed	NaN	NaN
Yes	No	Month-to-Month	No	Credit Card	-4.0	542.40	38.33	10	96.21	610.28	Stayed	NaN	NaN
No	Yes	Month-to-Month	Yes	Bank Withdrawal	73.9	280.85	0.00	0	134.60	415.45	Churned	Competitor	Competitor had better devices
No	Yes	Month-to-Month	Yes	Bank Withdrawal	98.0	1237.85	0.00	0	361.66	1599.51	Churned	Dissatisfaction	Product dissatisfaction
No	Yes	Month-to-Month	Yes	Credit Card	83.9	267.40	0.00	0	22.14	289.54	Churned	Dissatisfaction	Network reliability

```
df.columns
```

```
Index(['Customer ID', 'Gender', 'Age', 'Married', 'Number of Dependents',
       'City', 'Zip Code', 'Latitude', 'Longitude', 'Number of Referrals',
       'Tenure in Months', 'Offer', 'Phone Service',
       'Avg Monthly Long Distance Charges', 'Multiple Lines',
       'Internet Service', 'Internet Type', 'Avg Monthly GB Download',
       'Online Security', 'Online Backup', 'Device Protection Plan',
       'Premium Tech Support', 'Streaming TV', 'Streaming Movies',
       'Streaming Music', 'Unlimited Data', 'Contract', 'Paperless Billing',
       'Payment Method', 'Monthly Charge', 'Total Charges', 'Total Refunds',
       'Total Extra Data Charges', 'Total Long Distance Charges',
       'Total Revenue', 'Customer Status', 'Churn Category', 'Churn Reason',
       'No. of Service'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 39 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Customer ID                        7043 non-null   object 
 1   Gender                             7043 non-null   object 
 2   Age                                7043 non-null   int64  
 3   Married                            7043 non-null   object 
 4   Number of Dependents               7043 non-null   int64  
 5   City                               7043 non-null   object 
 6   Zip Code                           7043 non-null   int64  
 7   Latitude                           7043 non-null   float64
 8   Longitude                          7043 non-null   float64
 9   Number of Referrals                7043 non-null   int64  
 10  Tenure in Months                   7043 non-null   int64  
 11  Offer                              3166 non-null   object 
 12  Phone Service                      7043 non-null   object 
 13  Avg Monthly Long Distance Charges  6361 non-null   float64
```

```
 14  Multiple Lines                 6361 non-null   object
 15  Internet Service               7043 non-null   object
 16  Internet Type                  5517 non-null   object
 17  Avg Monthly GB Download        5517 non-null   float64
 18  Online Security                5517 non-null   object
 19  Online Backup                  5517 non-null   object
 20  Device Protection Plan         5517 non-null   object
 21  Premium Tech Support           5517 non-null   object
 22  Streaming TV                   5517 non-null   object
 23  Streaming Movies               5517 non-null   object
 24  Streaming Music                5517 non-null   object
 25  Unlimited Data                 5517 non-null   object
 26  Contract                       7043 non-null   object
 27  Paperless Billing              7043 non-null   object
 28  Payment Method                 7043 non-null   object
 29  Monthly Charge                 7043 non-null   float64
 30  Total Charges                  7043 non-null   float64
 31  Total Refunds                  7043 non-null   float64
 32  Total Extra Data Charges       7043 non-null   int64
 33  Total Long Distance Charges    7043 non-null   float64
 34  Total Revenue                  7043 non-null   float64
 35  Customer Status                7043 non-null   object
 36  Churn Category                 1869 non-null   object
 37  Churn Reason                   1869 non-null   object
 38  No. of Service                 7043 non-null   int64
dtypes: float64(9), int64(7), object(23)
memory usage: 2.1+ MB
```

```
df.describe()
```

igitude	Number of Referrals	Tenure in Months	Avg Monthly Long Distance Charges	Avg Monthly GB Download	Monthly Charge	Total Charges	Total Refunds	Total Extra Data Charges	Total Long Distance Charges	Total Revenue	
.000000	7043.000000	7043.000000	6361.000000	5517.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	704
.756684	1.951867	32.386767	25.420517	26.189958	63.596131	2280.381264	1.962182	6.860713	749.099262	3034.379056	
.154425	3.001199	24.542061	14.200374	19.586585	31.204743	2266.220462	7.902614	25.104978	846.660055	2865.204542	
.301372	0.000000	1.000000	1.010000	2.000000	-10.000000	18.800000	0.000000	0.000000	0.000000	21.360000	
.788090	0.000000	9.000000	13.050000	13.000000	30.400000	400.150000	0.000000	0.000000	70.545000	605.610000	1
.595293	0.000000	29.000000	25.690000	21.000000	70.050000	1394.550000	0.000000	0.000000	401.440000	2108.640000	1
.969795	3.000000	55.000000	37.680000	30.000000	89.750000	3786.600000	0.000000	0.000000	1191.100000	4801.145000	1
.192901	11.000000	72.000000	49.990000	85.000000	118.750000	8684.800000	49.790000	150.000000	3564.720000	11979.340000	1

Lets begin our data analysis by checkin ght edimension of our data frame.

```
df.shape
```

```
(7043, 39)
```

Now, lets check if we have any missing values in our data frame.

```
df.isnull()
```

	Customer ID	Gender	Age	Married	Number of Dependents	City	Zip Code	Latitude	Longitude	Number of Referrals	Tenure in Months	Offer	Phone Service	Avg Monthly Long Distance Charges	Multipl Line
0	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
7038	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fals
7039	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fals
7040	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fals
7041	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fals
7042	False	False	False	False	False	False	False	False	False	False	False	True	False	True	Tru

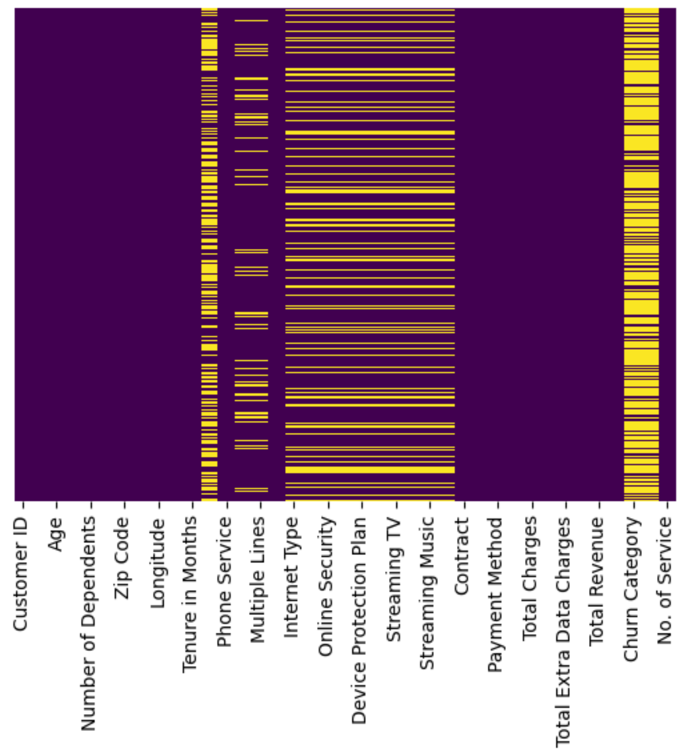
7043 rows × 39 columns

```
df.isnull().sum()
```

```
Customer ID                          0
Gender                               0
Age                                  0
Married                              0
Number of Dependents                 0
City                                 0
Zip Code                             0
Latitude                             0
Longitude                            0
Number of Referrals                  0
Tenure in Months                     0
Offer                             3877
Phone Service                        0
Avg Monthly Long Distance Charges  682
Multiple Lines                     682
Internet Service                     0
Internet Type                     1526
Avg Monthly GB Download           1526
Online Security                   1526
Online Backup                     1526
Device Protection Plan            1526
Premium Tech Support              1526
Streaming TV                      1526
Streaming Movies                  1526
Streaming Music                   1526
Unlimited Data                    1526
Contract                             0
Paperless Billing                    0
Payment Method                       0
Monthly Charge                       0
Total Charges                        0
Total Refunds                        0
Total Extra Data Charges             0
Total Long Distance Charges          0
Total Revenue                        0
Customer Status                      0
Churn Category                    5174
Churn Reason                      5174
No. of Service                       0
dtype: int64
```

```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

```
<Axes: >
```




```
# Lets check for the data types of all columns of our data frame
df.dtypes
```

```
Customer ID                          object
Gender                               object
Age                                   int64
Married                              object
Number of Dependents                  int64
City                                 object
Zip Code                              int64
Latitude                            float64
Longitude                           float64
Number of Referrals                   int64
Tenure in Months                      int64
Offer                                object
Phone Service                        object
Avg Monthly Long Distance Charges   float64
Multiple Lines                       object
Internet Service                     object
Internet Type                        object
Avg Monthly GB Download             float64
Online Security                      object
Online Backup                        object
Device Protection Plan               object
Premium Tech Support                 object
Streaming TV                         object
Streaming Movies                     object
Streaming Music                      object
Unlimited Data                       object
Contract                             object
Paperless Billing                    object
Payment Method                       object
Monthly Charge                      float64
Total Charges                       float64
Total Refunds                       float64
Total Extra Data Charges              int64
Total Long Distance Charges         float64
Total Revenue                       float64
Customer Status                      object
Churn Category                       object
Churn Reason                         object
No. of Service                        int64
dtype: object
```

Lets find the current status of all the customer and easily visualize it using pie chart.

```
df_CurrentStatus = df['Customer Status'].value_counts().index
df_CurrentStatus
```

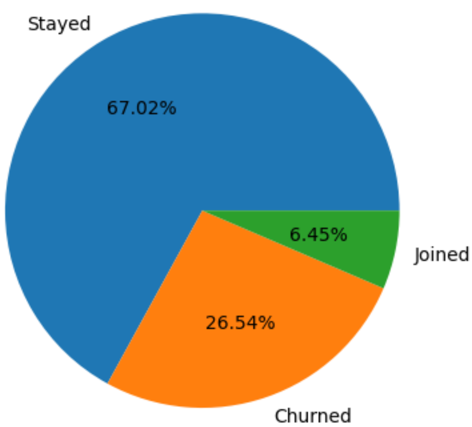
```
Index(['Stayed', 'Churned', 'Joined'], dtype='object', name='Customer Status')
```

```
df_CurrentStatus_values = df['Customer Status'].value_counts().values
df_CurrentStatus_values
```

```
array([4720, 1869,  454])
```

```
plt.pie(df_CurrentStatus_values[:3], labels = df_CurrentStatus[:3], autopct='%1.2f%%')
```

```
([<matplotlib.patches.Wedge at 0x780144050190>,
  <matplotlib.patches.Wedge at 0x780144050070>,
  <matplotlib.patches.Wedge at 0x780144050d60>],
 [Text(-0.5604479811676993, 0.9465189170877941, 'Stayed'),
  Text(0.3586221713095951, -1.0398991000309556, 'Churned'),
  Text(1.0775211031566014, -0.22124256428676015, 'Joined')],
 [Text(-0.30569889881874507, 0.5162830456842513, '67.02%'),
  Text(0.1956120934415973, -0.5672176909259757, '26.54%'),
  Text(0.5877387835399643, -0.1206777623382328, '6.45%')])
```



Observation - It looks like majority of customers are staying with company but company is still loosing customer as 26.54% are churned and only 6.45% only are joining. In a way, company is effectively loosing its customer base by almost 20%.

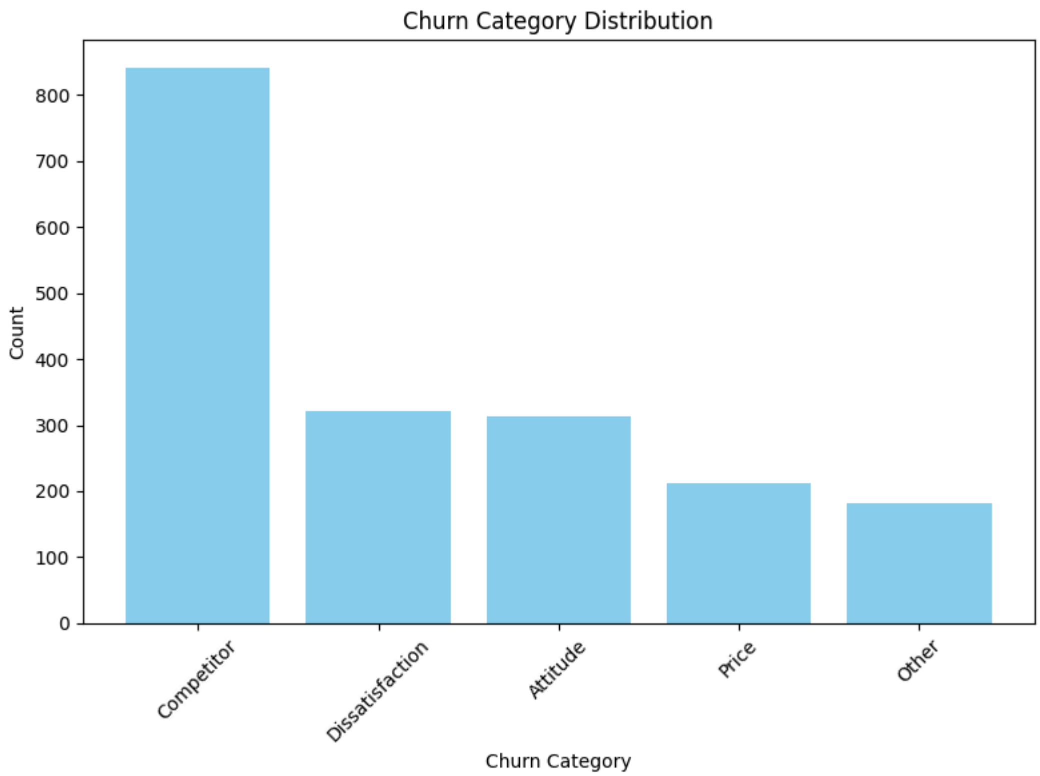
```
df_ChurnCategory = df['Churn Category'].value_counts().index
df_ChurnCategory
```

```
Index(['Competitor', 'Dissatisfaction', 'Attitude', 'Price', 'Other'], dtype='object', name='Churn Category')
```

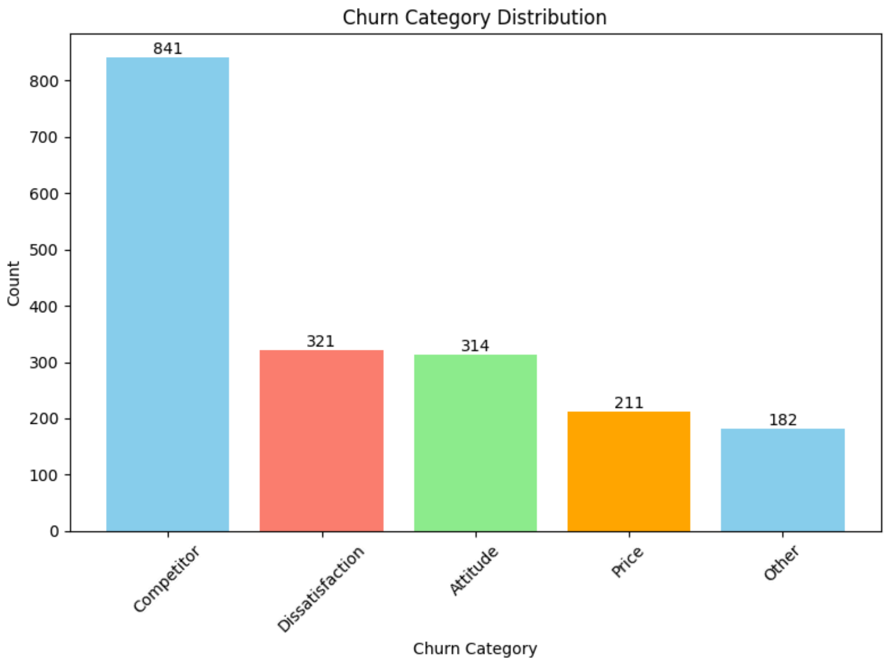
```
df_ChurnCategory_values = df['Churn Category'].value_counts().values
df_ChurnCategory_values
```

```
array([841, 321, 314, 211, 182])
```

```
plt.figure(figsize=(8, 6))
plt.bar(df_ChurnCategory, df_ChurnCategory_values, color='skyblue')
plt.xlabel('Churn Category')
plt.ylabel('Count')
plt.title('Churn Category Distribution')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```




```
colors = ['skyblue', 'salmon', 'lightgreen', 'orange']

# Plotting
plt.figure(figsize=(8, 6))
bars = plt.bar(df_ChurnCategory, df_ChurnCategory_values, color=colors)

# Add value labels to each bar
for bar, value in zip(bars, df_ChurnCategory_values):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), str(value),
             ha='center', va='bottom', color='black')

plt.xlabel('Churn Category')
plt.ylabel('Count')
plt.title('Churn Category Distribution')

# Customize x-axis labels rotation for better readability
plt.xticks(rotation=45)

plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```




```
df.isna().sum()
```

```
Customer ID                          0
Gender                               0
Age                                  0
Married                              0
Number of Dependents                 0
City                                 0
Zip Code                             0
Latitude                             0
Longitude                            0
Number of Referrals                  0
Tenure in Months                     0
Offer                             3877
Phone Service                        0
Avg Monthly Long Distance Charges  682
Multiple Lines                     682
Internet Service                     0
Internet Type                     1526
Avg Monthly GB Download           1526
Online Security                   1526
Online Backup                     1526
Device Protection Plan            1526
Premium Tech Support              1526
Streaming TV                      1526
Streaming Movies                  1526
Streaming Music                   1526
Unlimited Data                    1526
Contract                             0
Paperless Billing                    0
Payment Method                       0
Monthly Charge                       0
Total Charges                        0
Total Refunds                        0
Total Extra Data Charges             0
Total Long Distance Charges          0
Total Revenue                        0
Customer Status                      0
Churn Category                    5174
Churn Reason                      5174
No. of Service                       0
dtype: int64
```

Lets analyze how much our customers are paying per month and what is their tenure based on their status to have a better insights on their monthly expenses.

```
df.groupby('Customer Status')[['Monthly Charge', 'Tenure in Months']].agg(['min', 'max', 'mean'])
```

	Monthly Charge			Tenure in Months		
	min	max	mean	min	max	mean
Customer Status						
Churned	-10.0	118.35	73.347592	1	72	17.979133
Joined	-8.0	107.95	42.775991	1	3	1.720264
Stayed	-10.0	118.75	61.737415	4	72	41.041525

```
# I am creating one more dataframe to ensure that my original dataframe data is intact
dataset = df
```

```
dataset
```

	Customer ID	Gender	Age	Married	Number of Dependents	City	Zip Code	Latitude	Longitude	Number of Referrals	Tenure in Months	Offer	Phone Service	Avg Monthly Long Distance Charges	Mu
0	0002-ORFBO	Female	37	Yes	0	Frazier Park	93225	34.827662	-118.999073	2	9	NaN	Yes	42.39	
1	0003-MKNFE	Male	46	No	0	Glendale	91206	34.162515	-118.203869	0	9	NaN	Yes	10.69	
2	0004-TLHLJ	Male	50	No	0	Costa Mesa	92627	33.645672	-117.922613	0	4	Offer E	Yes	33.65	
3	0011-IGKFF	Male	78	Yes	0	Martinez	94553	38.014457	-122.115432	1	13	Offer D	Yes	27.82	
4	0013-EXCHZ	Female	75	Yes	0	Camarillo	93010	34.227846	-119.079903	3	3	NaN	Yes	7.38	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	9987-LUTYD	Female	20	No	0	La Mesa	91941	32.759327	-116.997260	0	13	Offer D	Yes	46.68	
7039	9992-RRAMN	Male	40	Yes	0	Riverbank	95367	37.734971	-120.954271	1	22	Offer D	Yes	16.20	
7040	9992-UJOEL	Male	22	No	0	Elk	95432	39.108252	-123.645121	0	2	Offer E	Yes	18.62	
7041	9993-LHIEB	Male	21	Yes	0	Solana Beach	92075	33.001813	-117.263628	5	67	Offer A	Yes	2.12	
7042	9995-HOTOH	Male	36	Yes	0	Sierra City	96125	39.600599	-120.636358	1	63	NaN	No	NaN	

7043 rows × 39 columns

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 39 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Customer ID                        7043 non-null   object
 1   Gender                             7043 non-null   object
 2   Age                                7043 non-null   int64
 3   Married                            7043 non-null   object
 4   Number of Dependents               7043 non-null   int64
 5   City                               7043 non-null   object
 6   Zip Code                           7043 non-null   int64
 7   Latitude                           7043 non-null   float64
 8   Longitude                          7043 non-null   float64
 9   Number of Referrals                7043 non-null   int64
 10  Tenure in Months                   7043 non-null   int64
 11  Offer                              3166 non-null   object
 12  Phone Service                      7043 non-null   object
```

```
 13  Avg Monthly Long Distance Charges  6361 non-null   float64
 14  Multiple Lines                     6361 non-null   object
 15  Internet Service                   7043 non-null   object
 16  Internet Type                      5517 non-null   object
 17  Avg Monthly GB Download            5517 non-null   float64
 18  Online Security                    5517 non-null   object
 19  Online Backup                      5517 non-null   object
 20  Device Protection Plan             5517 non-null   object
 21  Premium Tech Support               5517 non-null   object
 22  Streaming TV                       5517 non-null   object
 23  Streaming Movies                   5517 non-null   object
 24  Streaming Music                    5517 non-null   object
 25  Unlimited Data                     5517 non-null   object
 26  Contract                           7043 non-null   object
 27  Paperless Billing                  7043 non-null   object
 28  Payment Method                     7043 non-null   object
 29  Monthly Charge                     7043 non-null   float64
 30  Total Charges                      7043 non-null   float64
 31  Total Refunds                      7043 non-null   float64
 32  Total Extra Data Charges           7043 non-null   int64
 33  Total Long Distance Charges        7043 non-null   float64
 34  Total Revenue                      7043 non-null   float64
 35  Customer Status                    7043 non-null   object
 36  Churn Category                     1869 non-null   object
 37  Churn Reason                       1869 non-null   object
 38  No. of Service                     7043 non-null   int64
dtypes: float64(9), int64(7), object(23)
memory usage: 2.1+ MB
```

```
## Lets check the percentage of nan values present in each feature
## 1 -step make the list of features which has missing values
features_with_na=[features for features in dataset.columns if dataset[features].isnull().sum()>1]
## 2- step print the feature name and the percentage of missing values

for feature in features_with_na:
    print(feature, np.round(dataset[feature].isnull().mean(), 4),  ' % missing values')
```

```
Offer 0.5505  % missing values
Avg Monthly Long Distance Charges 0.0968  % missing values
Multiple Lines 0.0968  % missing values
Internet Type 0.2167  % missing values
Avg Monthly GB Download 0.2167  % missing values
Online Security 0.2167  % missing values
Online Backup 0.2167  % missing values
Device Protection Plan 0.2167  % missing values
Premium Tech Support 0.2167  % missing values
Streaming TV 0.2167  % missing values
Streaming Movies 0.2167  % missing values
Streaming Music 0.2167  % missing values
Unlimited Data 0.2167  % missing values
Churn Category 0.7346  % missing values
Churn Reason 0.7346  % missing values
```

```
# Check for missing values in each column
missing_values = dataset.isnull().sum()

# Print the number of missing values in each column
print("Missing values in each column:")
for column, missing_count in missing_values.items():
    print(f"{column}: {missing_count}")

# Check for missing values in the entire dataset
total_missing = dataset.isnull().sum().sum()
print("\nTotal missing values in the dataset:", total_missing)
```

```
Missing values in each column:
Customer ID: 0
Gender: 0
Age: 0
Married: 0
Number of Dependents: 0
City: 0
Zip Code: 0
Latitude: 0
Longitude: 0
Number of Referrals: 0
Tenure in Months: 0
Offer: 3877
Phone Service: 0
Avg Monthly Long Distance Charges: 682
Multiple Lines: 682
```

```
Internet Service: 0
Internet Type: 1526
Avg Monthly GB Download: 1526
Online Security: 1526
Online Backup: 1526
Device Protection Plan: 1526
Premium Tech Support: 1526
Streaming TV: 1526
Streaming Movies: 1526
Streaming Music: 1526
Unlimited Data: 1526
Contract: 0
Paperless Billing: 0
Payment Method: 0
Monthly Charge: 0
Total Charges: 0
Total Refunds: 0
Total Extra Data Charges: 0
Total Long Distance Charges: 0
Total Revenue: 0
Customer Status: 0
Churn Category: 5174
Churn Reason: 5174
No. of Service: 0

Total missing values in the dataset: 30849
```

```
# Check for duplicate rows in the entire dataset
duplicate_rows = dataset[dataset.duplicated()]
print("Duplicate rows in the dataset:")
print(duplicate_rows)
```

```
Duplicate rows in the dataset:
Empty DataFrame
Columns: [Customer ID, Gender, Age, Married, Number of Dependents, City, Zip Code, Latitude, Longitude, Number of Referrals, Tenure in M
Index: []
```

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import cross_val_score
```

```
# Define class variables
class_variable = "Customer Status"
```

```
dataset
```

	Customer ID	Gender	Age	Married	Number of Dependents	City	Zip Code	Latitude	Longitude	Number of Referrals	Tenure in Months	Offer	Phone Service	Avg Monthly Long Distance Charges	Mu
0	0002-ORFBO	Female	37	Yes	0	Frazier Park	93225	34.827662	-118.999073	2	9	NaN	Yes	42.39	
1	0003-MKNFE	Male	46	No	0	Glendale	91206	34.162515	-118.203869	0	9	NaN	Yes	10.69	
2	0004-TLHLJ	Male	50	No	0	Costa Mesa	92627	33.645672	-117.922613	0	4	Offer E	Yes	33.65	
3	0011-IGKFF	Male	78	Yes	0	Martinez	94553	38.014457	-122.115432	1	13	Offer D	Yes	27.82	
4	0013-EXCHZ	Female	75	Yes	0	Camarillo	93010	34.227846	-119.079903	3	3	NaN	Yes	7.38	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	9987-LUTYD	Female	20	No	0	La Mesa	91941	32.759327	-116.997260	0	13	Offer D	Yes	46.68	
7039	9992-RRAMN	Male	40	Yes	0	Riverbank	95367	37.734971	-120.954271	1	22	Offer D	Yes	16.20	
7040	9992-UJOEL	Male	22	No	0	Elk	95432	39.108252	-123.645121	0	2	Offer E	Yes	18.62	
7041	9993-LHIEB	Male	21	Yes	0	Solana Beach	92075	33.001813	-117.263628	5	67	Offer A	Yes	2.12	
7042	9995-HOTOH	Male	36	Yes	0	Sierra City	96125	39.600599	-120.636358	1	63	NaN	No	NaN	

7043 rows × 39 columns

```
irrelevant_variables = ["Customer ID","City","Zip Code","Latitude","Longitude","Phone Service","Avg Monthly Long Distance Charges","Multiple
                        "Internet Service","Internet Type","Avg Monthly GB Download","Online Security","Online Backup","Device Protection Pla
                        "Premium Tech Support","Streaming TV","Streaming Movies","Streaming Music","Paperless Billing","Payment Method"]

dataset.drop(irrelevant_variables, axis=1, inplace=True)
```

```
# Encode categorical variables
label_encoders = {}
for column in dataset.select_dtypes(include=["object"]).columns:
    label_encoders[column] = LabelEncoder()
    dataset[column] = label_encoders[column].fit_transform(dataset[column])


# Define features (X) and target variable (y)
X = dataset.drop(class_variable, axis=1)
y = dataset[class_variable]


# Split the 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)
```

```
# Create and train classification models
models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "Support Vector Machine": SVC()
}


for name, model in models.items():
    model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
    # Evaluate model
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    cv_score = cross_val_score(model, X, y, cv=5).mean()
    print(f"Model: {name}")
    print(f"Accuracy: {accuracy}")
    print(f"Cross-Validation Score: {cv_score}")
    print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
    print("----------------------------------------------")
```

```
Model: Support Vector Machine
Accuracy: 0.7111426543647977
Cross-Validation Score: 0.7238395420672301
Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.39      0.41       373
           1       0.00      0.00      0.00        97
           2       0.79      0.91      0.85       939

    accuracy                           0.71      1409
   macro avg       0.41      0.43      0.42      1409
weighted avg       0.64      0.71      0.67      1409

----------------------------------------------
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
```

```
# Increase max_iter for Logistic Regression
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),  # Adjust max_iter as needed
    "Random Forest": RandomForestClassifier(),
    "Support Vector Machine": SVC()
}

for name, model in models.items():
    model.fit(X_train, y_train)

    # Evaluate model
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    cv_score = cross_val_score(model, X, y, cv=5).mean()
    print(f"Model: {name}")
    print(f"Accuracy: {accuracy}")
    print(f"Cross-Validation Score: {cv_score}")
    print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
    print("----------------------------------------------")
```

```
Model: Random Forest
Accuracy: 1.0
Cross-Validation Score: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       373
           1       1.00      1.00      1.00        97
           2       1.00      1.00      1.00       939

    accuracy                           1.00      1409
   macro avg       1.00      1.00      1.00      1409
weighted avg       1.00      1.00      1.00      1409

----------------------------------------------
Model: Support Vector Machine
Accuracy: 0.7111426543647977
Cross-Validation Score: 0.7238395420672301
Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.39      0.41       373
           1       0.00      0.00      0.00        97
           2       0.79      0.91      0.85       939

    accuracy                           0.71      1409
   macro avg       0.41      0.43      0.42      1409
weighted avg       0.64      0.71      0.67      1409

----------------------------------------------
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are il
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.model_selection import GridSearchCV

# Define SVM with hyperparameter grid for tuning
svm_params = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly']
}
svm_model = SVC()
svm_grid_search = GridSearchCV(svm_model, svm_params, cv=5)
svm_grid_search.fit(X_train, y_train)

# Best parameters for SVM
print("Best parameters for SVM:", svm_grid_search.best_params_)

# Evaluate best SVM model
best_svm_model = svm_grid_search.best_estimator_
best_svm_model.fit(X_train, y_train)
y_pred_svm = best_svm_model.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
cv_score_svm = cross_val_score(best_svm_model, X, y, cv=5).mean()

print("Model: Support Vector Machine (Tuned)")
print(f"Accuracy: {accuracy_svm}")
print(f"Cross-Validation Score: {cv_score_svm}")
print(f"Classification Report:\n{classification_report(y_test, y_pred_svm)}")
print("----------------------------------------------")
```

```
Best parameters for SVM: {'C': 10, 'kernel': 'linear'}
Model: Support Vector Machine (Tuned)
Accuracy: 1.0
Cross-Validation Score: 0.9998579545454545
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       373
           1       1.00      1.00      1.00        97
           2       1.00      1.00      1.00       939

    accuracy                           1.00      1409
   macro avg       1.00      1.00      1.00      1409
weighted avg       1.00      1.00      1.00      1409
```

```
    ---------------------------------------------

import numpy as np

# Define metrics
metrics = ['Precision', 'Recall', 'F1-score']
classes = ['Churned', 'Stayed']

# Define values for Logistic Regression
lr_values = [[0.85, 0.91, 0.88], [1.0, 0.94, 0.97]]

# Define values for Random Forest
rf_values = [[1.0, 1.0, 1.0], [1.0, 1.0, 1.0]]

# Define values for Support Vector Machine (Tuned)
svm_values = [[1.0, 1.0, 1.0], [1.0, 1.0, 1.0]]

# Plot metrics for each class
for i, metric in enumerate(metrics):
    plt.figure(figsize=(10, 5))

    # Combine values for each class
    lr_class_values = np.array(lr_values)[:, i]
    rf_class_values = np.array(rf_values)[:, i]
    svm_class_values = np.array(svm_values)[:, i]

    # Plot bar chart
    x = np.arange(len(classes))
    width = 0.2
    plt.bar(x - width, lr_class_values, width, label='Logistic Regression', color='b')
    plt.bar(x, rf_class_values, width, label='Random Forest', color='g')
    plt.bar(x + width, svm_class_values, width, label='SVM (Tuned)', color='r')

    plt.xlabel('Class')
    plt.ylabel(metric)
    plt.title(f'{metric} for Churned and Stayed Customers')
    plt.xticks(x, classes)
    plt.legend()
    plt.show()
```