```
import pandas as pd
```

```
from google.colab import files
uploaded = files.upload()
```

Dosyaları Seç Dosya seçilmedi Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Telco-Customer-Churn.csv to Telco-Customer-Churn.csv
```

```
df = pd.read_csv('Telco-Customer-Churn.csv')
```

```
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...

5 rows × 21 columns

```
df.tail()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes

5 rows × 21 columns

```
df.info()
```

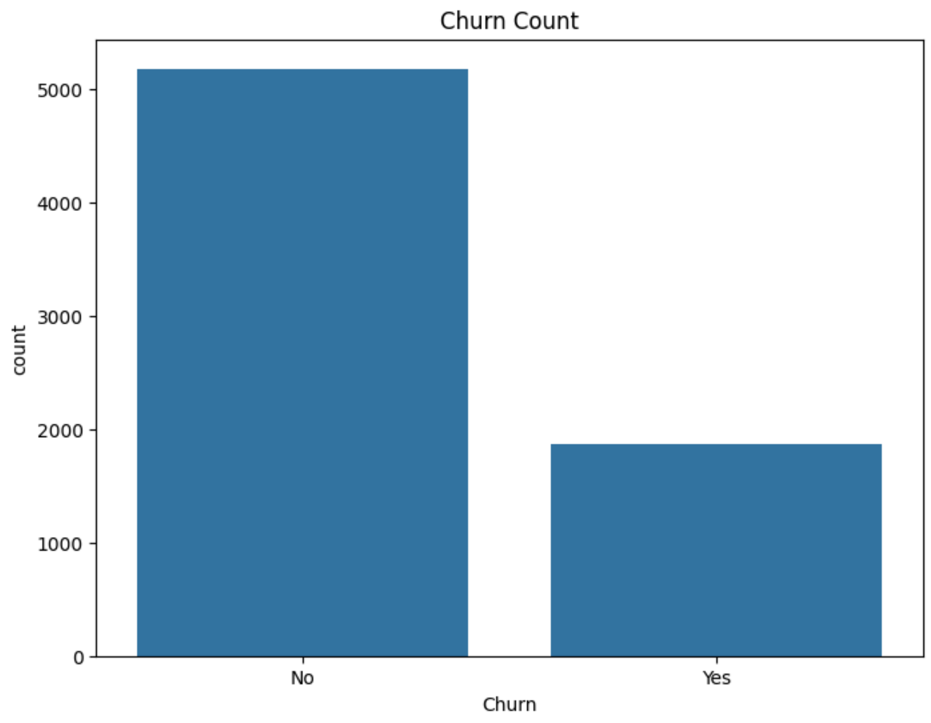
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   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
```

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

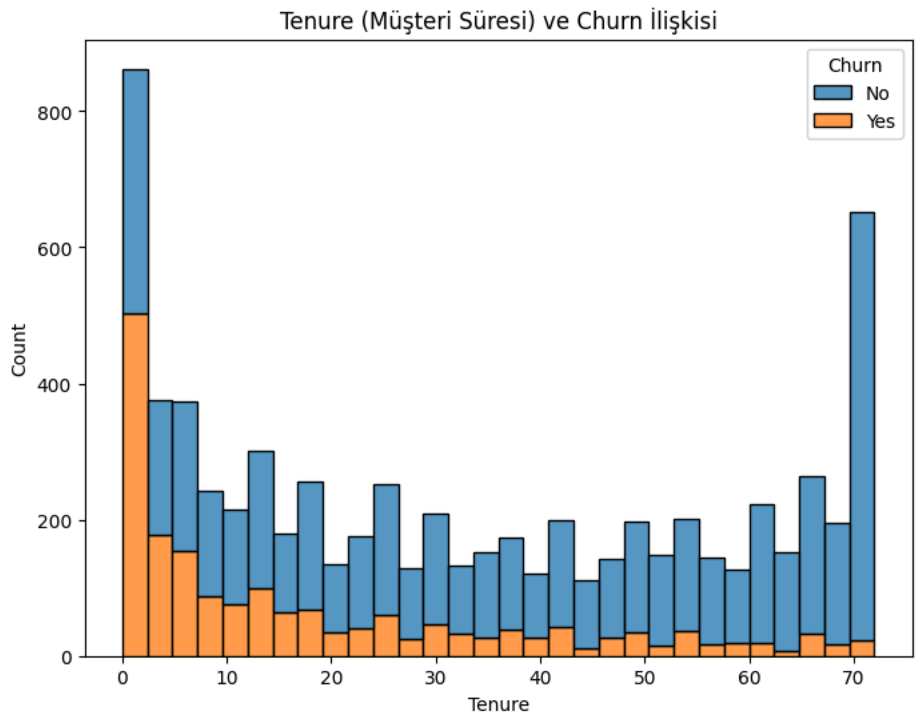
```
                  0
customerID        0
gender            0
SeniorCitizen     0
Partner           0
Dependents        0
tenure            0
PhoneService      0
MultipleLines     0
InternetService   0
OnlineSecurity    0
OnlineBackup      0
DeviceProtection  0
TechSupport       0
StreamingTV       0
StreamingMovies   0
Contract          0
PaperlessBilling  0
PaymentMethod     0
MonthlyCharges    0
TotalCharges      0
Churn             0

dtype: int64
```

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

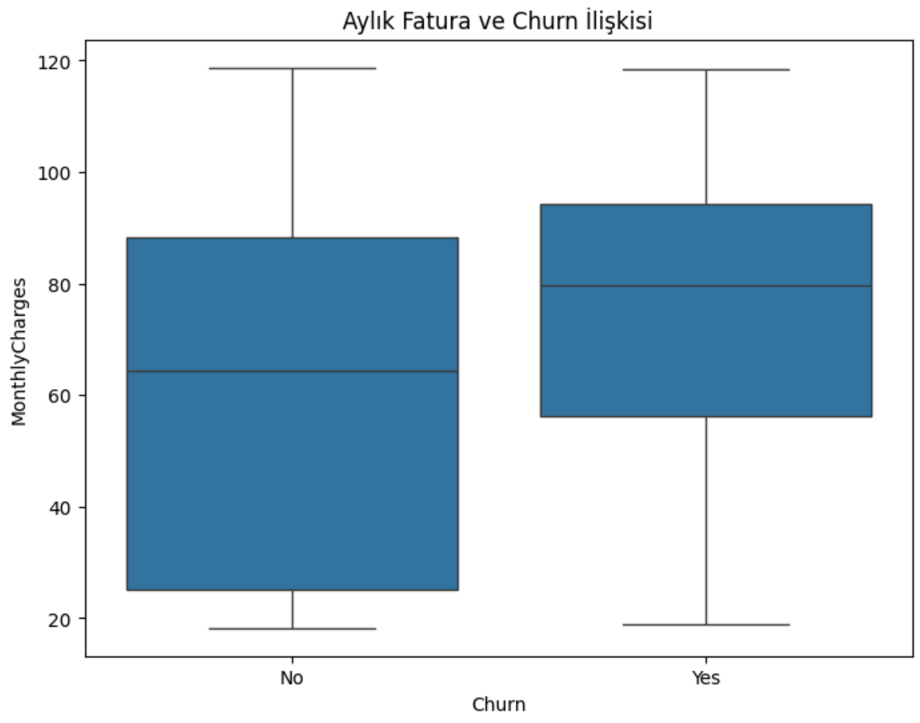
```
plt.figure(figsize=(8,6))
sns.countplot(x = 'Churn', data = df)
plt.title('Churn Count')
plt.show()
```



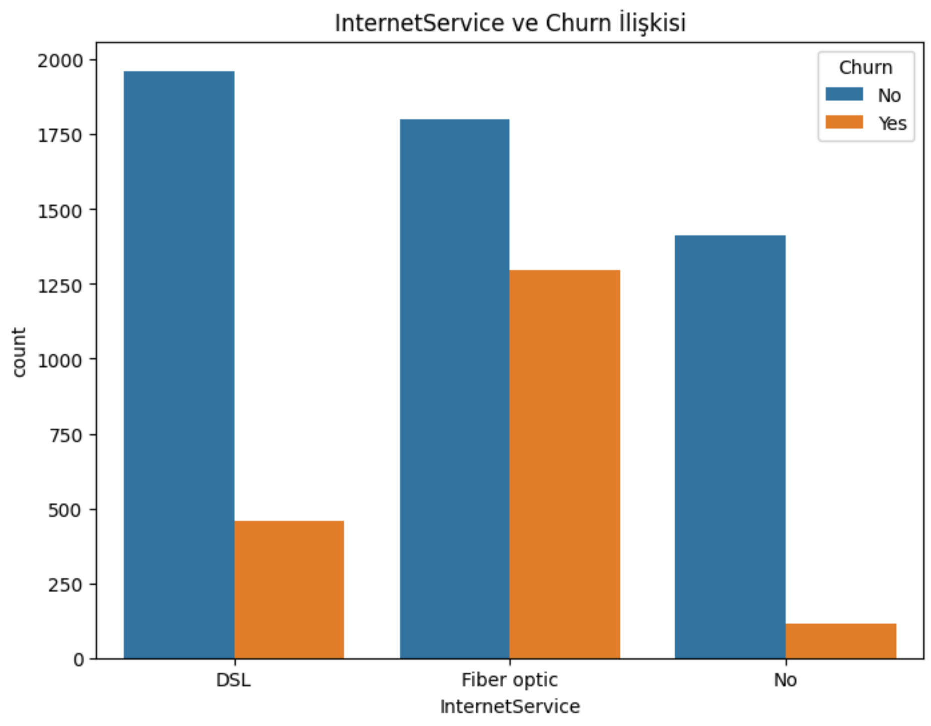
```
plt.figure(figsize = (8,6))
sns.histplot(x = 'tenure', hue = 'Churn', multiple = 'stack', data = df, bins = 30) # Changed 'Stack' to 'stack'
plt.title('Tenure (Müşteri Süresi) ve Churn İlişkisi')
plt.xlabel('Tenure')
plt.ylabel('Count')
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
plt.title('Aylık Fatura ve Churn İlişkisi')
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.countplot(x='InternetService', hue='Churn', data=df)
plt.title('InternetService ve Churn İlişkisi')
plt.show()
```



```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


X = pd.get_dummies(df.drop('Churn', axis=1), drop_first=True)
y = df['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)


X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42)


logreg = LogisticRegression(max_iter = 1000)
logreg.fit(X_train, y_train)
```

```
▾     LogisticRegression
LogisticRegression(max_iter=1000)
```

```
y_pred = logreg.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)


print(f"Doğruluk: {accuracy}")
print("Karışıklık Matrisi:")
print(conf_matrix)
print("Sınıflandırma Raporu:")
print(class_report)
```

```
Doğruluk: 0.8246983676366217
Karışıklık Matrisi:
[[941  95]
 [152 221]]
Sınıflandırma Raporu:
              precision    recall  f1-score   support

           0       0.86      0.91      0.88      1036
           1       0.70      0.59      0.64       373

    accuracy                           0.82      1409
   macro avg       0.78      0.75      0.76      1409
weighted avg       0.82      0.82      0.82      1409
```

Kodlamaya başlayın veya yapay zeka ile kod oluşturun.