pandas

assingment 1

```
#1

import pandas as pd

churn = pd.read_csv('/content/customer_churn.csv')
print(churn.head())
```

```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
0  No phone service             DSL             No  ...               No
1                No             DSL            Yes  ...              Yes
2                No             DSL            Yes  ...               No
3  No phone service             DSL            Yes  ...              Yes
4                No     Fiber optic             No  ...               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

              PaymentMethod MonthlyCharges  TotalCharges Churn
0          Electronic check          29.85         29.85    No
1              Mailed check          56.95        1889.5    No
2              Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4          Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

```
#2

Cols = churn.columns.tolist()
newCols = churn.iloc[ : ,[3,7,9,20]]
print(Cols)
print(newCols)
```

```
['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', '
     Partner     MultipleLines OnlineSecurity Churn
0        Yes  No phone service             No    No
1         No                No            Yes    No
2         No                No            Yes   Yes
3         No  No phone service            Yes    No
4         No                No             No   Yes
...      ...               ...            ...   ...
7038     Yes               Yes            Yes    No
7039     Yes               Yes             No    No
7040     Yes  No phone service            Yes    No
7041     Yes               Yes             No   Yes
7042      No                No            Yes    No

[7043 rows x 4 columns]
```

```
churn[200:1001]
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
200	9323-HGFWY	Female	0	Yes	No	27	Yes	N
201	8544-GOQSH	Female	0	No	No	14	Yes	N
202	3363-DTIVD	Male	0	Yes	Yes	71	Yes	Ye
203	7018-WBJNK	Male	0	No	Yes	13	Yes	N
204	9142-KZXOP	Male	0	No	No	44	Yes	N
...	...	...	...	...	...	...	...	.
996	6641-XRPSU	Female	0	No	No	34	Yes	N
997	1374-DMZUI	Female	1	No	No	4	Yes	Ye
998	2545-LXYVJ	Male	0	Yes	No	72	Yes	N
999	3234-VKACU	Male	0	No	No	2	Yes	N
1000	8357-EQXFO	Female	0	No	No	7	Yes	N

801 rows × 21 columns

```
churn.tail(10)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
7033	9767-FFLEM	Male	0	No	No	38	Yes	N
7034	0639-TSIQW	Female	0	No	No	67	Yes	Ye
7035	8456-QDAVC	Male	0	No	No	19	Yes	N
7036	7750-EYXWZ	Female	0	No	No	12	No	No phone servic
7037	2569-WGERO	Female	0	No	No	72	Yes	N
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Ye
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Ye
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone servic
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Ye
7042	3186-AJIEK	Male	0	No	No	66	Yes	N

10 rows × 21 columns

```
churn[-1:]
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
7042	3186-AJIEK	Male	0	No	No	66	Yes	N

1 rows × 21 columns

```
churn.sort_values(by='tenure',ascending=False)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
1672	4737-AQCPU	Male	0	Yes	Yes	72	Yes	Ye
193	9680-NIAUV	Female	0	Yes	Yes	72	Yes	Ye
4553	5914-XRFQB	Male	0	Yes	No	72	Yes	Ye
483	5168-MQQCA	Female	0	Yes	No	72	Yes	Ye
3266	0464-WJTKO	Female	0	Yes	Yes	72	Yes	N
...	...	...	...	...	...	...	...	.
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Ye
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Ye
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	N
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Ye
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone servic

7043 rows × 21 columns

```
churn[(churn['tenure'] >50 ) & (churn['gender'] == 'Female')]
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
15	3655-SNQYZ	Female	0	Yes	Yes	69	Yes	Ye
16	8191-XWSZG	Female	0	No	No	52	Yes	N
23	3638-WEABW	Female	0	Yes	No	58	Yes	Ye
30	3841-NFECX	Female	1	Yes	No	71	Yes	Ye
35	6234-RAAPL	Female	0	Yes	Yes	72	Yes	Ye
...	...	...	...	...	...	...	...	.
7023	1035-IPQPU	Female	1	Yes	No	63	Yes	Ye
7028	9281-CEDRU	Female	0	Yes	No	68	Yes	N
7034	0639-TSIQW	Female	0	No	No	67	Yes	Ye
7037	2569-WGERO	Female	0	No	No	72	Yes	N
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Ye

1022 rows × 21 columns

```
churn[(churn['gender'] == 'Male') & (churn['SeniorCitizen'] == 0)]
#print(len(churn))
#print(len(churn[(churn['TechSupport']=='yes') & (churn[ (churn['gender'] == 'Male') & (churn['Sen
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
1	5575-GNVDE	Male	0	No	No	34	Yes	N
2	3668-QPYBK	Male	0	No	No	2	Yes	N
3	7795-CFOCW	Male	0	No	No	45	No	No phone servic
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Ye
9	6388-TABGU	Male	0	No	Yes	62	Yes	N
...	...	...	...	...	...	...	...	.
7027	0550-DCXLH	Male	0	No	No	13	Yes	N
7033	9767-FFLEM	Male	0	No	No	38	Yes	N
7035	8456-QDAVC	Male	0	No	No	19	Yes	N
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Ye
7042	3186-AJIEK	Male	0	No	No	66	Yes	N

2981 rows × 21 columns

```
count = 0

for i in range(0,len(churn)):
  if (churn.iloc[i]['TechSupport'] == 'Yes'):
    if (churn.iloc[i]['gender'] == 'Male') & (churn.iloc[i]['SeniorCitizen'] == 1):
      count += 1
print(count)
```

```
119
```

```
print(len(churn[(churn['TechSupport'] == 'yes') & (churn['gender'] == 'Male') & (churn['SeniorCiti
```

```
0
```

```
churn.iloc[20:200, 2:15]
```

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	O
20	1	No	No	1	No	No phone service	DSL	
21	0	Yes	No	12	Yes	No	No	N
22	0	No	No	1	Yes	No	No	N
23	0	Yes	No	58	Yes	Yes	DSL	
24	0	Yes	Yes	49	Yes	No	DSL	
...	...	...	...	...	...	...	...	
195	0	Yes	No	20	Yes	No	Fiber optic	
196	0	Yes	Yes	24	Yes	Yes	No	N
197	0	No	No	59	Yes	Yes	Fiber optic	
198	0	Yes	Yes	72	Yes	Yes	Fiber optic	
199	0	No	Yes	1	Yes	No	No	N

180 rows × 13 columns

```
churn.head(100)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No
...	...	...	...	...	...	...	...	...
95	8637-XJIVR	Female	0	No	No	12	Yes	Yes
96	9803-FTJCG	Male	0	Yes	Yes	71	Yes	Yes
97	0278-YXOOG	Male	0	No	No	5	Yes	No
98	3212-KXOCR	Male	0	No	No	52	Yes	No
99	4598-XLKNJ	Female	1	Yes	No	25	Yes	No

100 rows × 21 columns

```
churn[(churn['TechSupport'] == 'Yes') & (churn['Churn'] == 'No')]
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
3	7795-CFOCW	Male	0	No	No	45	No	No phone servic
14	5129-JLPIS	Male	0	No	No	25	Yes	N
15	3655-SNQYZ	Female	0	Yes	Yes	69	Yes	Ye
23	3638-WEABW	Female	0	Yes	No	58	Yes	Ye
24	6322-HRPFA	Male	0	Yes	Yes	49	Yes	N
...	...	...	...	...	...	...	...	.
7027	0550-DCXLH	Male	0	No	No	13	Yes	N
7028	9281-CEDRU	Female	0	Yes	No	68	Yes	N
7036	7750-EYXWZ	Female	0	No	No	12	No	No phone servic
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Ye
7042	3186-AJIEK	Male	0	No	No	66	Yes	N

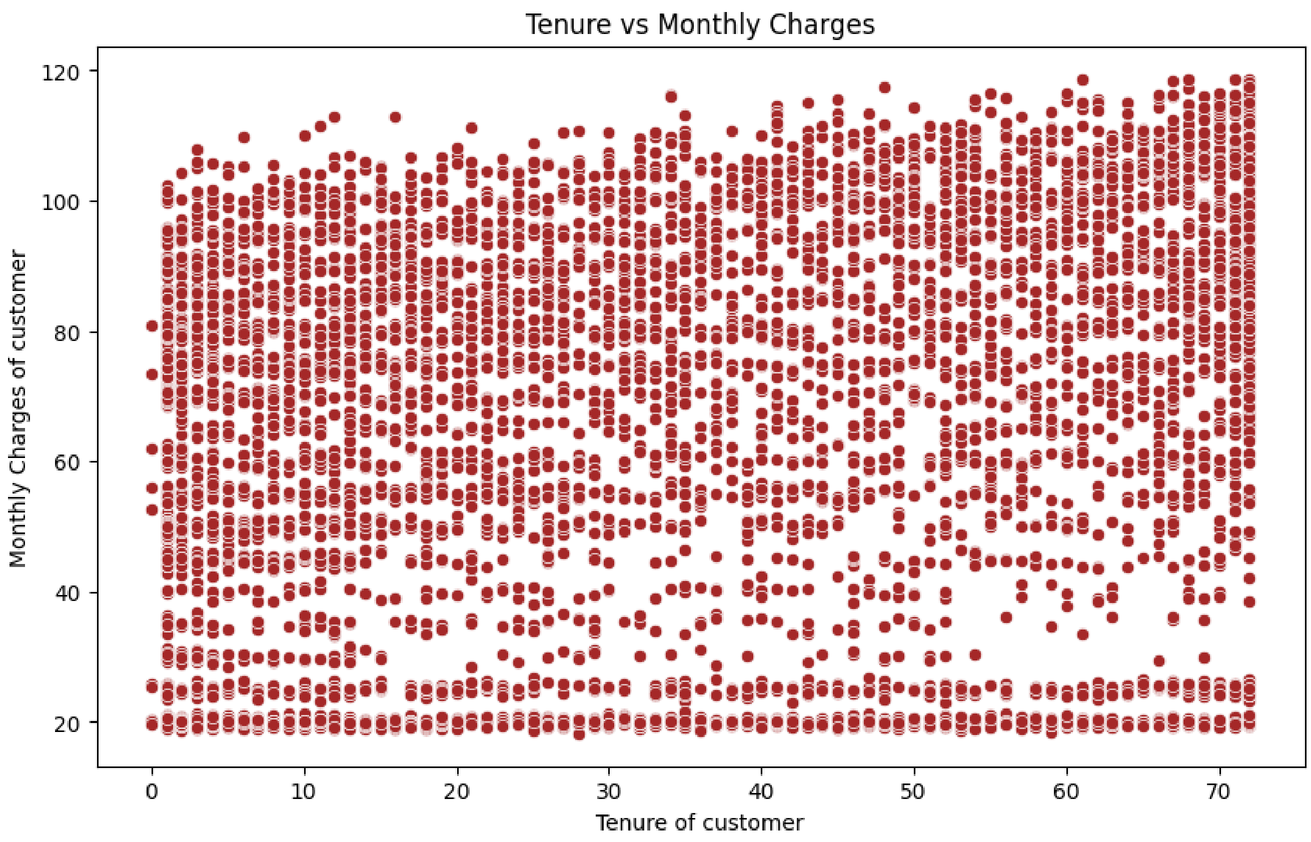
1734 rows × 21 columns

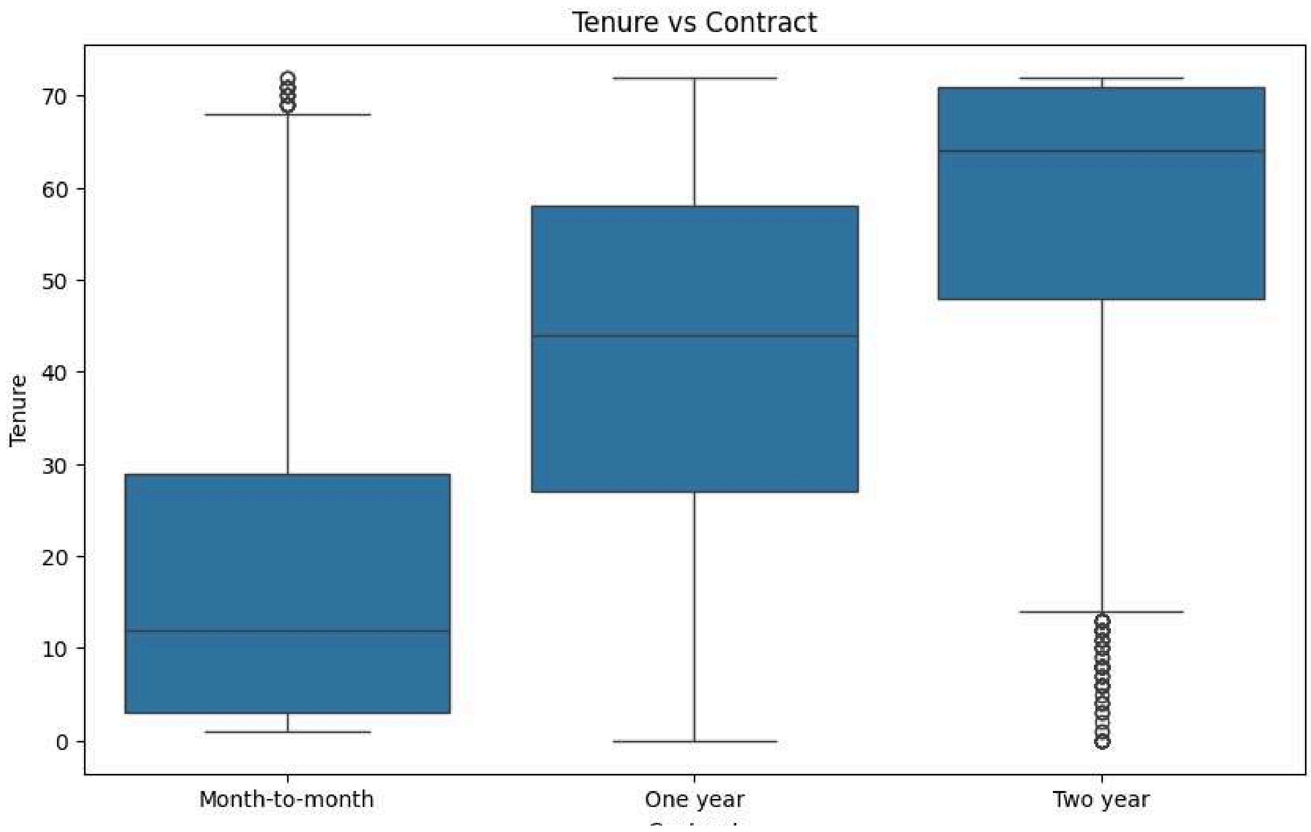
```
churn[(churn['Contract'] == 'Month-to-month') & (churn['Churn'] == 'Yes')]
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
2	3668-QPYBK	Male	0	No	No	2	Yes	N
4	9237-HQITU	Female	0	No	No	2	Yes	N
5	9305-CDSKC	Female	0	No	No	8	Yes	Ye
8	7892-POOKP	Female	0	Yes	No	28	Yes	Ye
13	0280-XJGEX	Male	0	No	No	49	Yes	Ye
...	...	...	...	...	...	...	...	.
7018	1122-JWTJW	Male	0	Yes	Yes	1	Yes	N
7026	8775-CEBBJ	Female	0	No	No	9	Yes	N
7032	6894-LFHLY	Male	1	No	No	1	Yes	Ye
7034	0639-TSIQW	Female	0	No	No	67	Yes	Ye
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Ye

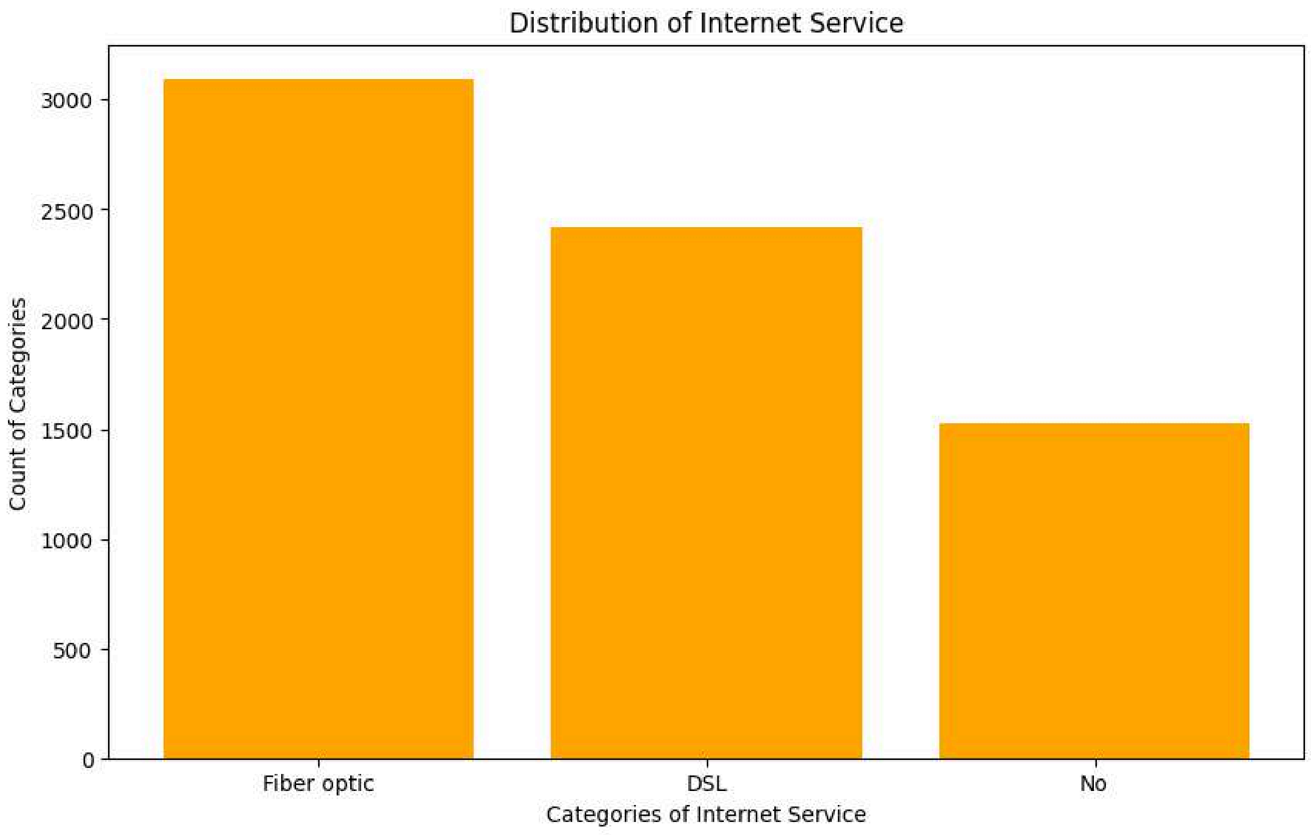
1655 rows × 21 columns

```
# prompt: Assign the points a color of ‘brown’
# b. Set the x-axis label to ‘Tenure of customer’
# c. Set the y-axis label to ‘Monthly Charges of customer’
# d. Set the title to ‘Tenure vs Monthly Charges’
# e. Build a box-plot between ‘tenure’ & ‘Contract’. Map ‘tenure’ on the
# y-axis &
# f. ‘Contract’ on the x-axis.

import matplotlib.pyplot as plt
import seaborn as sns

# a. Assign the points a color of ‘brown’
# b. Set the x-axis label to ‘Tenure of customer’
# c. Set the y-axis label to ‘Monthly Charges of customer’
# d. Set the title to ‘Tenure vs Monthly Charges’
plt.figure(figsize=(10, 6))
sns.scatterplot(x='tenure', y='MonthlyCharges', data=churn, color='brown')
plt.xlabel('Tenure of customer')
plt.ylabel('Monthly Charges of customer')
plt.title('Tenure vs Monthly Charges')
plt.show()


# e. Build a box-plot between ‘tenure’ & ‘Contract’. Map ‘tenure’ on the y-axis &
# f. ‘Contract’ on the x-axis.
plt.figure(figsize=(10, 6))
sns.boxplot(x='Contract', y='tenure', data=churn)
plt.xlabel('Contract')
plt.ylabel('Tenure')
plt.title('Tenure vs Contract')
plt.show()
```

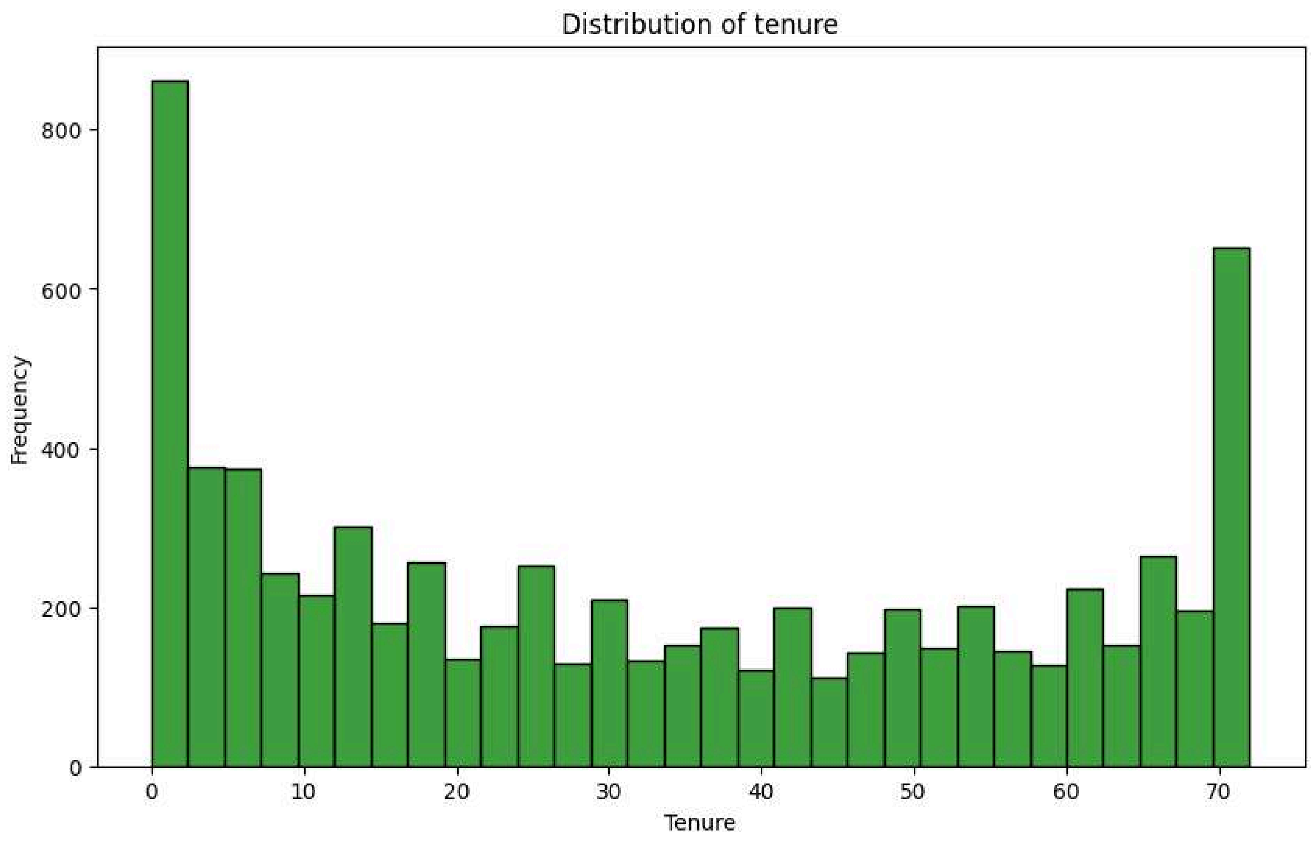




```
# prompt: Set x-axis label to ‘Categories of Internet Service’
# b. Set y-axis label to ‘Count of Categories’
# c. Set the title of plot to be ‘Distribution of Internet Service’
# d. Set the color of the bars to be ‘orange

# ... (Your existing code)

# Assuming 'InternetService' is the column you want to analyze
internet_service_counts = churn['InternetService'].value_counts()

plt.figure(figsize=(10, 6))
plt.bar(internet_service_counts.index, internet_service_counts.values, color='orange')
plt.xlabel('Categories of Internet Service')
plt.ylabel('Count of Categories')
plt.title('Distribution of Internet Service')
plt.show()
```



```
# prompt: Set the number of bins to be 30
# b. Set the color of the bins to be ‘green’
# c. Assign the title ‘Distribution of tenure

# ... (Your existing code)

# Histogram for tenure distribution
plt.figure(figsize=(10, 6))
sns.histplot(churn['tenure'], bins=30, color='green')
plt.xlabel('Tenure')
plt.ylabel('Frequency')
plt.title('Distribution of tenure')
plt.show()
```



```
# ... (Your existing code)

# Convert 'TotalCharges' to numeric, handling errors
churn['TotalCharges'] = pd.to_numeric(churn['TotalCharges'], errors='coerce')

# Drop rows with missing values in 'TotalCharges' after conversion
churn = churn.dropna(subset=['TotalCharges'])

# Separate features and target variable
X = churn[['tenure', 'MonthlyCharges']]  # Replace with your features
y = churn['TotalCharges']  # Replace with your target

# ... (Rest of your code)


  # prompt: Build a simple logistic regression model where dependent variable is
# ‘Churn’ and independent variable is ‘MonthlyCharges’:
# a. Divide the dataset in 65:35 ratio
# b. Build the model on train set and predict the values on test set
# c. Build the confusion matrix and get the accuracy score
# d. Build a multiple logistic regression model where dependent variable
# is ‘Churn’ and independent variables are ‘tenure’ and
# ‘MonthlyCharges’
# e. Divide the dataset in 80:20 ratio
# f. Build the model on train set and predict the values on test set
# g. Build the confusion matrix and get the accuracy score

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

# a. Build a simple logistic regression model
# Separate features and target variable
X = churn[['MonthlyCharges']]
y = churn['Churn']
```

```
# Convert 'Churn' to numerical (Yes=1, No=0)
y = y.map({'Yes': 1, 'No': 0})

# Split data into training and testing sets (65:35 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=42)

# Build the model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Build the confusion matrix and get the accuracy score
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print("Confusion Matrix:\n", cm)
print("Accuracy:", accuracy)


# d. Build a multiple logistic regression model
# Separate features and target variable
X = churn[['tenure', 'MonthlyCharges']]
y = churn['Churn']

# Convert 'Churn' to numerical (Yes=1, No=0)
y = y.map({'Yes': 1, 'No': 0})

# Split data into training and testing sets (80:20 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Build the confusion matrix and get the accuracy score
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print("\nConfusion Matrix (Multiple Logistic Regression):\n", cm)
print("Accuracy (Multiple Logistic Regression):", accuracy)
```

```
Confusion Matrix:
 [[1824    0]
 [ 638    0]]
Accuracy: 0.7408610885458976

Confusion Matrix (Multiple Logistic Regression):
 [[938  95]
 [215 159]]
Accuracy (Multiple Logistic Regression): 0.7796730632551528
```

```
# prompt: Build a decision tree model where dependent variable is ‘Churn’ and
# independent variable is ‘tenure’:
# a. Divide the dataset in 80:20 ratio
# b. Build the model on train set and predict the values on test set
# c. Build the confusion matrix and calculate the accuracy

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Separate features and target variable
```

```
X = churn[['tenure']]
```