SHREE RAMSWAROOP UNIVEERSITY

INTRODUCTION

SANIDHYA PRAJAPTI -202410101150105

VIVEK KUMAR -202410101150106

B.tech CS (DS+AI) 33,34

DATA ANALYTICS AND REPORTING

PROJECT:- Telecom Customers data analysis using Google COLAB.

LIBRARIES USED:-

1. **PANDAS**
2. **MATPLOTLIB.PYPLOT**
3. **OPENPYXL -WORKBOOK**
4. **OPENPYXL.CHART- BARCHART, REFERENCE**

STEP 1:-**Importing necessary libraries**

1 Pandas, MATPLOTLIB.PYPLOT, OPENPYXL -WORKBOOK, OPENPYXL.CHART- BARCHART, REFERENCE

```
import pandas as pd
import matplotlib.pyplot as plt
from openpyxl import workbook
from openpyxl.chart import BarChart, Reference
```

STEP 2:- Importing the telecom users data of 1000 users with pd.read_csv(“”)

```
df = pd.read_csv("/content/drive/MyDrive/project/TELECOM_USER_DATASET1000.csv")
df
```

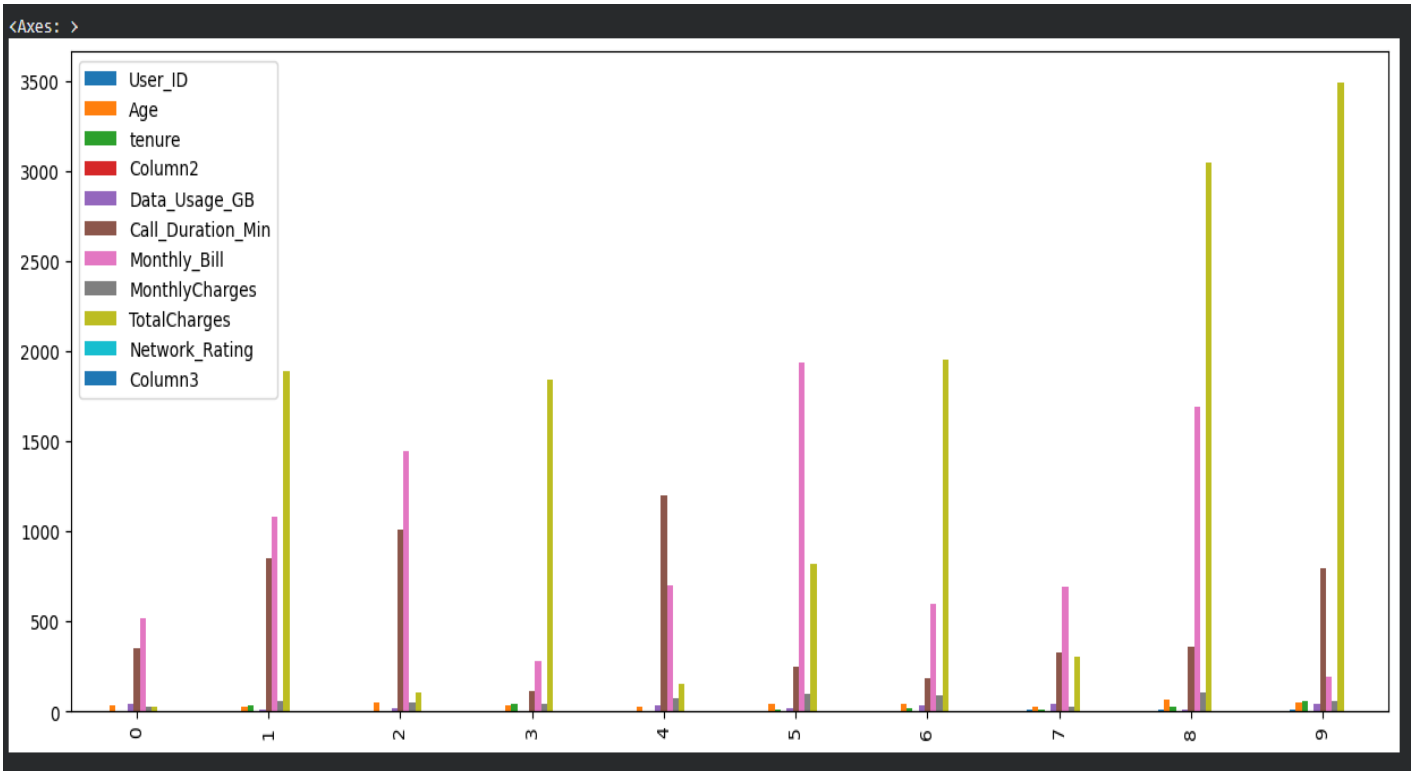
	customerID	User_ID	gender	Age	Region	tenure	Column2	MultipleLines	InternetService	PaymentMethod	Plan_Type
0	7590-VHVEG	1	Female	34	East	1.0	NaN	No phone service	DSL	Electronic check	Prepaid
1	5575-GNVDE	2	Male	26	South	34.0	NaN	No	DSL	Mailed check	Postpaid
2	3668-QPYBK	3	Male	50	North	2.0	NaN	No	DSL	Mailed check	Prepaid
3	7795-CFOCW	4	Male	37	West	45.0	NaN	No phone service	DSL	Bank transfer (automatic)	Postpaid
4	9237-HQITU	5	Female	30	South	2.0	NaN	No	Fiber optic	Electronic check	Postpaid
...	...	...	...	...	...	...	...	...	...	...	...
995	3842-IYKUE	996	Female	62	North	35.0	NaN	Yes	Fiber optic	Credit card (automatic)	Prepaid
996	6641-XRPSU	997	Female	52	East	34.0	NaN	No	Fiber optic	Credit card (automatic)	Postpaid
997	1374-DMZUI	998	Female	23	North	4.0	NaN	Yes	Fiber optic	Electronic check	Prepaid
998	2545-LXYVJ	999	Male	36	East	72.0	NaN	No	No	Bank transfer (automatic)	Prepaid
999	NaN	1000	NaN	47	South	NaN	NaN	NaN	NaN	NaN	Postpaid

1000 rows × 19 columns

Data_Usage_GB	Call_Duration_Min	Monthly_Bill	MonthlyCharges	TotalCharges	Network_Rating	Column3	Churn
43.39	352	518.17	29.85	29.85	4	NaN	No
11.17	852	1084.72	56.95	1889.5	1	NaN	No
18.34	1013	1449.22	53.85	108.15	3	NaN	Yes
3.89	118	281.89	42.30	1840.75	4	NaN	No
38.28	1202	703.62	70.70	151.65	1	NaN	Yes
...	...	...	...	...	...	...	...
26.06	236	559.49	85.30	2917.5	2	NaN	Yes
39.43	781	909.40	70.00	2416.1	4	NaN	Yes
47.16	1263	1938.71	94.30	424.45	1	NaN	Yes
37.47	1461	1349.75	20.70	1492.1	5	NaN	No
36.17	478	1172.35	NaN	NaN	3	NaN	NaN

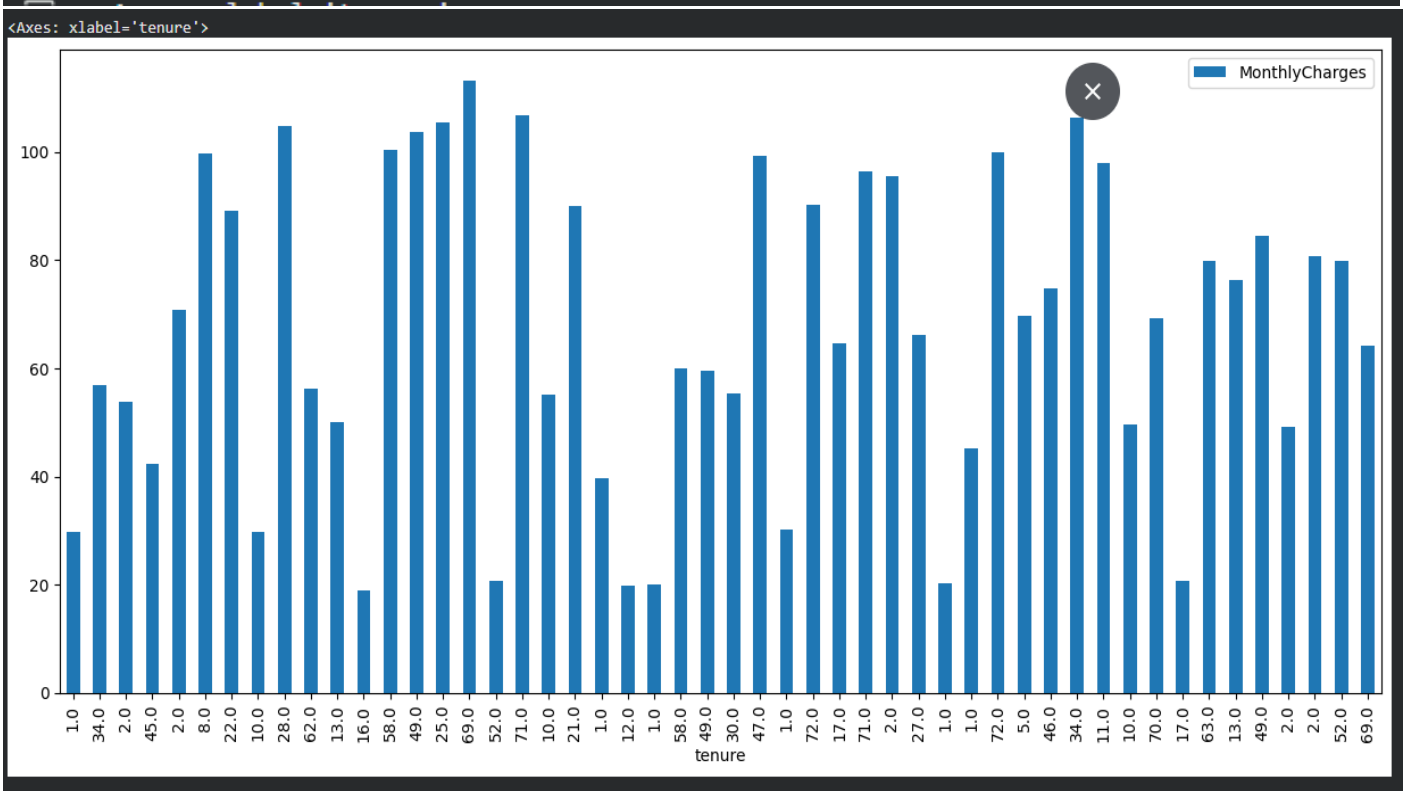
STEP 3:- Representing the data of 10 users in bar graph

```
df.head(10).plot(figsize=(15,6),kind="bar")
```



STEP 4:- Representing tenure and monthly charges in bar graph of 50 users

```
df.head(50).plot(x= "tenure", y= "MonthlyCharges",kind='bar',figsize=(15,7))
```



STEP 5:- Describing the telecom data and finding count mean std min and max of the data of all users

```
df.describe()
```

	User_ID	Age	tenure	Column2	Data_Usage_GB	Call_Duration_Min	Monthly_Bill	MonthlyCharges	Network_Rating	Column3
count	1000.000000	1000.00000	999.000000	0.0	1000.000000	1000.000000	1000.000000	999.000000	1000.000000	0.0
mean	500.500000	41.39200	32.249249	NaN	25.693840	781.861000	1028.508070	66.479980	3.030000	NaN
std	288.819436	13.68143	24.837729	NaN	14.325168	410.894656	552.897058	29.934448	1.418842	NaN
min	1.000000	18.00000	0.000000	NaN	0.510000	52.000000	100.350000	18.950000	1.000000	NaN
25%	250.750000	29.00000	8.000000	NaN	13.125000	435.750000	547.382500	42.000000	2.000000	NaN
50%	500.500000	42.00000	29.000000	NaN	26.525000	787.500000	1009.190000	74.250000	3.000000	NaN
75%	750.250000	53.00000	56.000000	NaN	38.215000	1132.250000	1513.987500	90.050000	4.000000	NaN
max	1000.000000	64.00000	72.000000	NaN	49.990000	1499.000000	1995.480000	116.250000	5.000000	NaN

STEP 6:- Finding the info of the 1000 users data using df.info().

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   customerID         999 non-null    object 
 1   User_ID            1000 non-null   int64  
 2   gender             999 non-null    object 
 3   Age                1000 non-null   int64  
 4   Region             1000 non-null   object 
 5   tenure             999 non-null    float64
 6   Column2            0 non-null      float64
 7   MultipleLines      999 non-null    object 
 8   InternetService    999 non-null    object 
 9   PaymentMethod      999 non-null    object 
 10  Plan_Type          1000 non-null   object 
 11  Data_Usage_GB      1000 non-null   float64
 12  Call_Duration_Min  1000 non-null   int64  
 13  Monthly_Bill       1000 non-null   float64
 14  MonthlyCharges     999 non-null    float64
 15  TotalCharges       999 non-null    object 
 16  Network_Rating     1000 non-null   int64  
 17  Column3            0 non-null      float64
 18  Churn              999 non-null    object 
dtypes: float64(6), int64(4), object(9)
memory usage: 148.6+ KB
```

STEP7:- Finding the missing values in the data.

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

	0
customerID	1
User_ID	0
gender	1
Age	0
Region	0
tenure	1
Column2	1000
MultipleLines	1
InternetService	1
PaymentMethod	1
Plan_Type	0
Data_Usage_GB	0
Call_Duration_Min	0
Monthly_Bill	0
MonthlyCharges	1
TotalCharges	1
Network_Rating	0
Column3	1000
Churn	1

dtype: int64

STEP8: Now we remove the missing values form the data of users if any

```
df.dropna()
df
```

-

	customerID	User_ID	gender	Age	Region	tenure	Column2	MultipleLines	InternetService	PaymentMethod	Plan_Type
0	7590-VHVEG	1	Female	34	East	1.0	NaN	No phone service	DSL	Electronic check	Prepaid
1	5575-GNVDE	2	Male	26	South	34.0	NaN	No	DSL	Mailed check	Postpaid
2	3668-QPYBK	3	Male	50	North	2.0	NaN	No	DSL	Mailed check	Prepaid
3	7795-CFOCW	4	Male	37	West	45.0	NaN	No phone service	DSL	Bank transfer (automatic)	Postpaid
4	9237-HQITU	5	Female	30	South	2.0	NaN	No	Fiber optic	Electronic check	Postpaid
...	...	...	...	...	...	...	...	...	...	...	...
995	3842-IYKUE	996	Female	62	North	35.0	NaN	Yes	Fiber optic	Credit card (automatic)	Prepaid
996	6641-XRPSU	997	Female	52	East	34.0	NaN	No	Fiber optic	Credit card (automatic)	Postpaid
997	1374-DMZUI	998	Female	23	North	4.0	NaN	Yes	Fiber optic	Electronic check	Prepaid
998	2545-LXYVJ	999	Male	36	East	72.0	NaN	No	No	Bank transfer (automatic)	Prepaid
999	NaN	1000	NaN	47	South	NaN	NaN	NaN	NaN	NaN	Postpaid

1000 rows x 19 columns

Data_Usage_GB	Call_Duration_Min	Monthly_Bill	MonthlyCharges	TotalCharges	Network_Rating	Column3	Churn
43.39	352	518.17	29.85	29.85	4	NaN	No
11.17	852	1084.72	56.95	1889.5	1	NaN	No
18.34	1013	1449.22	53.85	108.15	3	NaN	Yes
3.89	118	281.89	42.30	1840.75	4	NaN	No
38.28	1202	703.62	70.70	151.65	1	NaN	Yes
...	...	...	...	...	...	...	...
26.06	236	559.49	85.30	2917.5	2	NaN	Yes
39.43	781	909.40	70.00	2416.1	4	NaN	Yes
47.16	1263	1938.71	94.30	424.45	1	NaN	Yes
37.47	1461	1349.75	20.70	1492.1	5	NaN	No
36.17	478	1172.35	NaN	NaN	3	NaN	NaN

STEP 9:-Representation of the shape of data (rows and column).

```
print("shape of data",df.shape)
df.head()
```

shape of data (1000, 19)

	customerID	User_ID	gender	Age	Region	tenure	Column2	MultipleLines	InternetService	PaymentMethod	Plan_Type	Data_Usage_GB
0	7590-VHVEG	1	Female	34	East	1.0	NaN	No phone service	DSL	Electronic check	Prepaid	43.39
1	5575-GNVDE	2	Male	26	South	34.0	NaN	No	DSL	Mailed check	Postpaid	11.17
2	3668-QPYBK	3	Male	50	North	2.0	NaN	No	DSL	Mailed check	Prepaid	18.34
3	7795-CFOCW	4	Male	37	West	45.0	NaN	No phone service	DSL	Bank transfer (automatic)	Postpaid	3.89
4	9237-HQITU	5	Female	30	South	2.0	NaN	No	Fiber optic	Electronic check	Postpaid	38.28

STEP 10:- now Converting "TotalCharges" column to numeric, Filtering the rows ,Display filtered data and representing only the users of tenure 50

```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
Filtered_data = df[(df['TotalCharges'] > 100) & (df['tenure'] == 50)]
Filtered_data
```

	customerID	User_ID	gender	Age	Region	tenure	Column2	MultipleLines	InternetService	PaymentMethod	Plan_Type
68	3170-NMYVV	69	Female	56	East	50.0	NaN	No	No	Bank transfer (automatic)	Postpaid
231	3316-UWXUY	232	Male	43	South	50.0	NaN	No	Fiber optic	Credit card (automatic)	Prepaid
257	6655-LHBYW	258	Male	37	South	50.0	NaN	Yes	Fiber optic	Credit card (automatic)	Postpaid
266	6292-TOSSS	267	Male	32	West	50.0	NaN	Yes	No	Bank transfer (automatic)	Postpaid
548	4676-MQUEA	549	Male	50	East	50.0	NaN	Yes	Fiber optic	Bank transfer (automatic)	Postpaid
614	7138-GIRSH	615	Male	61	South	50.0	NaN	Yes	DSL	Bank transfer (automatic)	Postpaid
647	2391-IPLOP	648	Male	53	North	50.0	NaN	Yes	DSL	Electronic check	Postpaid
830	9101-BWFSS	831	Female	27	West	50.0	NaN	Yes	Fiber optic	Electronic check	Postpaid
891	3804-RVTGV	892	Male	20	North	50.0	NaN	Yes	Fiber optic	Bank transfer (automatic)	Prepaid
899	2267-FPIMA	900	Male	21	South	50.0	NaN	Yes	Fiber optic	Credit card (automatic)	Postpaid

Data_Usage_GB	Call_Duration_Min	Monthly_Bill	MonthlyCharges	TotalCharges	Network_Rating	Column3	Churn
12.99	770	747.04	20.15	930.90	2	NaN	No
5.71	269	855.98	93.50	4747.50	4	NaN	No
2.03	444	719.06	114.35	5791.10	1	NaN	No
18.13	840	1645.53	24.90	1195.25	3	NaN	No
35.65	1416	1930.00	101.90	5265.50	3	NaN	No
11.35	269	1428.74	69.50	3418.20	4	NaN	No
25.98	1479	700.55	69.65	3442.15	4	NaN	No
8.56	136	880.35	108.55	5610.70	2	NaN	Yes
22.81	79	1739.55	103.85	5017.90	1	NaN	Yes
22.73	655	1817.86	94.40	4914.90	3	NaN	No

STEP 11:-Representing only male users of totalcharges less than 100

```
Filtered_data = df[(df['TotalCharges'] > 100) & (df['gender'] == 'Male')]
display(Filtered_data)
```

	customerID	User_ID	gender	Age	Region	tenure	Column2	MultipleLines	InternetService	PaymentMethod	Plan_Type	Data_Usage_GB	C
1	5575-GNVDE	2	Male	26	South	34.0	NaN	No	DSL	Mailed check	Postpaid	11.17	
2	3668-QPYBK	3	Male	50	North	2.0	NaN	No	DSL	Mailed check	Prepaid	18.34	
3	7795-CFOCW	4	Male	37	West	45.0	NaN	No phone service	DSL	Bank transfer (automatic)	Postpaid	3.89	
6	1452-KIOVK	7	Male	46	West	22.0	NaN	Yes	Fiber optic	Credit card (automatic)	Postpaid	38.08	
9	6388-TABGU	10	Male	52	South	62.0	NaN	No	DSL	Bank transfer (automatic)	Prepaid	40.47	
...	...	...	...	...	...	...	...	...	...	...	...	...	
981	1106-HRLKZ	982	Male	46	South	40.0	NaN	No	No	Mailed check	Postpaid	39.36	
988	9046-JBFWA	989	Male	44	South	27.0	NaN	No	No	Mailed check	Postpaid	10.90	
989	3280-NMUVX	990	Male	58	South	34.0	NaN	No	No	Credit card (automatic)	Prepaid	14.41	
994	7277-KAMWT	995	Male	41	South	13.0	NaN	No	No	Credit card (automatic)	Prepaid	6.04	
998	2545-LXYVJ	999	Male	36	East	72.0	NaN	No	No	Bank transfer (automatic)	Prepaid	37.47	

435 rows × 19 columns

Call_Duration_Min	Monthly_Bill	MonthlyCharges	TotalCharges	Network_Rating	Column3	Churn
852	1084.72	56.95	1889.50	1	NaN	No
1013	1449.22	53.85	108.15	3	NaN	Yes
118	281.89	42.30	1840.75	4	NaN	No
189	598.45	89.10	1949.40	2	NaN	No
795	195.41	56.15	3487.95	4	NaN	No
...	...	...	...	...	...	...
1400	955.54	19.60	808.95	5	NaN	No
1214	1167.54	19.15	537.35	2	NaN	No
875	1375.67	19.60	678.80	1	NaN	No
857	1148.68	20.00	268.45	2	NaN	No
1461	1349.75	20.70	1492.10	5	NaN	No

STEP 12:- Representing only female users of totalcharges less than 100

```
Filtered_data = df[(df['TotalCharges'] < 1000) & (df['gender'] == 'Female')]
display(Filtered_data)
```

	customerID	User_ID	gender	Age	Region	tenure	Column2	MultipleLines	InternetService	PaymentMethod	Plan_Type	D
0	7590-VHVEG	1	Female	34	East	1.0	NaN	No phone service	DSL	Electronic check	Prepaid	
4	9237-HQITU	5	Female	30	South	2.0	NaN	No	Fiber optic	Electronic check	Postpaid	
5	9305-CDSKC	6	Female	45	East	8.0	NaN	Yes	Fiber optic	Electronic check	Postpaid	
7	6713-OKOMC	8	Female	30	West	10.0	NaN	No phone service	DSL	Mailed check	Postpaid	
18	4190-MFLUW	19	Female	62	West	10.0	NaN	No	DSL	Credit card (automatic)	Prepaid	
...	...	...	...	...	...	...	...	...	...	...	...	
980	3318-ISQFQ	981	Female	30	South	20.0	NaN	No	No	Bank transfer (automatic)	Postpaid	
986	2604-XVDAM	987	Female	30	South	12.0	NaN	No	DSL	Bank transfer (automatic)	Prepaid	
992	4883-KCPZJ	993	Female	48	East	22.0	NaN	Yes	No	Credit card (automatic)	Prepaid	
993	9108-EQPNQ	994	Female	51	West	10.0	NaN	Yes	No	Credit card (automatic)	Postpaid	
997	1374-DMZUI	998	Female	23	North	4.0	NaN	Yes	Fiber optic	Electronic check	Prepaid	

205 rows x 19 columns

Data_Usage_GB	Call_Duration_Min	Monthly_Bill	MonthlyCharges	TotalCharges	Network_Rating	Column3	Churn
43.39	352	518.17	29.85	29.85	4	NaN	No
38.28	1202	703.62	70.70	151.65	1	NaN	Yes
20.71	245	1940.48	99.65	820.50	3	NaN	Yes
45.23	332	694.80	29.75	301.90	2	NaN	No
11.72	1462	856.63	55.20	528.35	1	NaN	Yes
...	...	...	...	...	...	...	...
25.25	194	1303.20	19.50	413.00	5	NaN	No
23.74	849	1855.74	43.80	540.95	3	NaN	No
20.73	929	1667.13	25.25	555.40	3	NaN	No
46.24	512	1408.74	26.10	225.55	1	NaN	No
47.16	1263	1938.71	94.30	424.45	1	NaN	Yes

STEP 13:- Representing the total charges according to region using pivot table

```
pivot_table = pd.pivot_table(df, values='TotalCharges', index='Region', aggfunc='sum')
pivot_table
```

Region	TotalCharges
East	484288.8
North	561794.6
South	680529.6
West	605303.0

STEP 14:- Representing the Data usage according to region using pivot table

```
pivot_table = pd.pivot_table(df, values='Data_Usage_GB', index='Region', aggfunc='sum')
pivot_table
```

Region	Data_Usage_GB
East	6089.25
North	6693.37
South	6662.50
West	6248.72

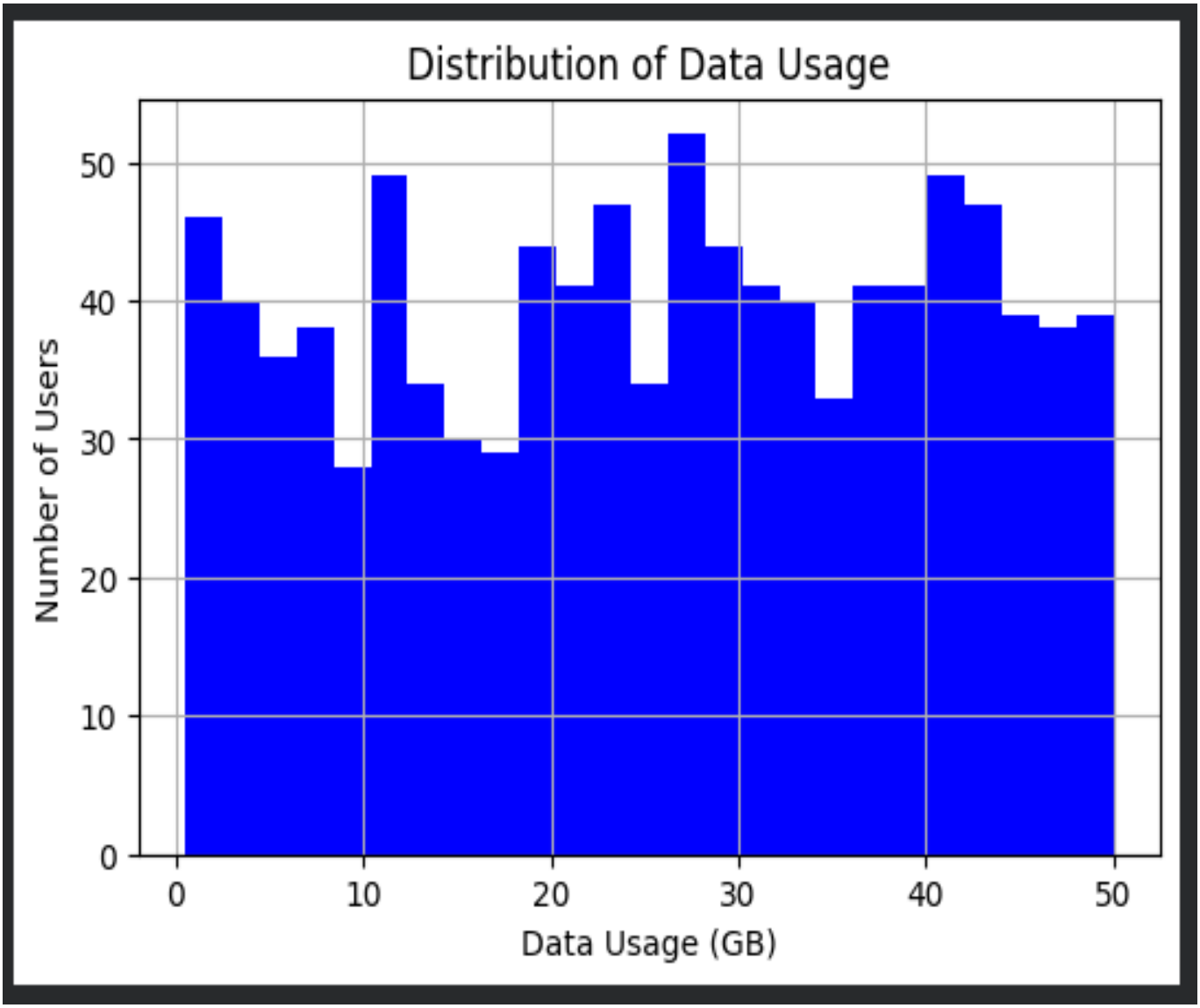
STEP15:- Calculated the percentage distribution of customers who churned and who did not churn using value_counts (normalize=True).

```
churn_rate = df['Churn'].value_counts(normalize=True) * 100
print("Churn Rate (%):\n", churn_rate)
```

```
Churn Rate (%):
 Churn
No     74.374374
Yes    25.625626
Name: proportion, dtype: float64
```

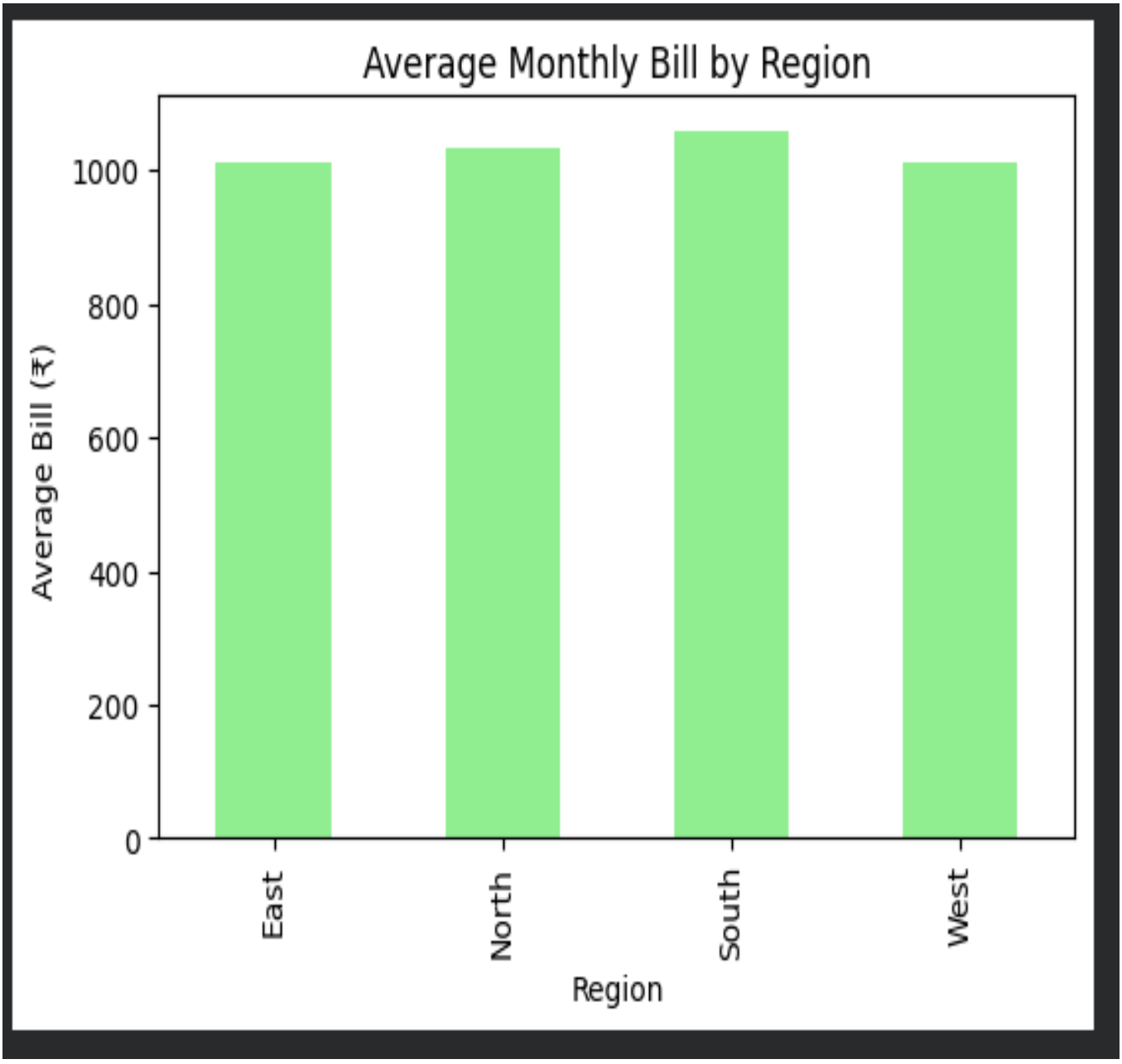
STEP 16:- Plotted a histogram to visualize the distribution of users based on their data usage in GB

```
plt.figure(figsize=(6,4))
df['Data_Usage_GB'].hist(bins=25, color='blue')
plt.title('Distribution of Data Usage')
plt.xlabel('Data Usage (GB)')
plt.ylabel('Number of Users')
plt.show()
```



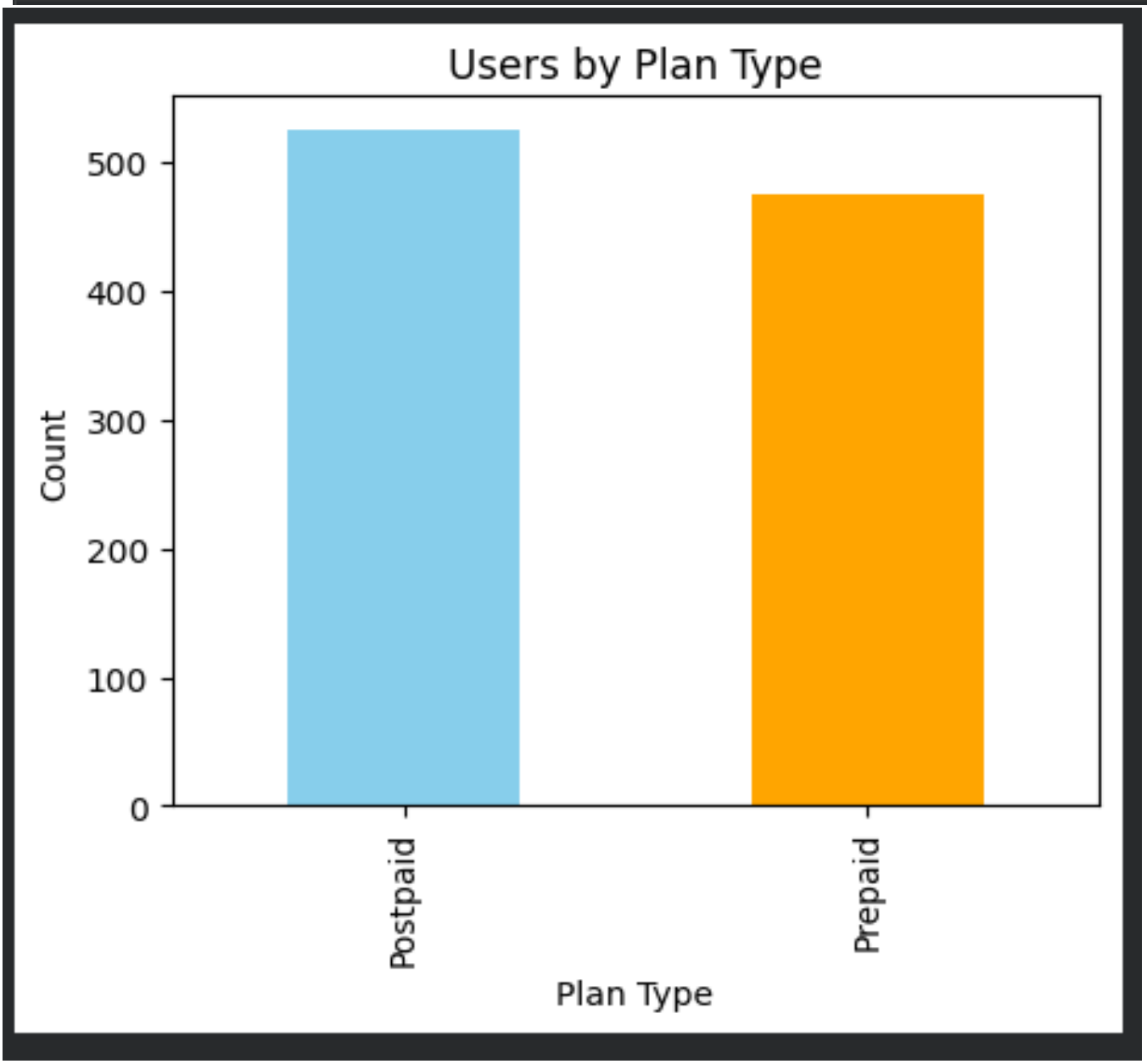
STEP 17:-representing the monthly bill average according to the region of the users data

```
plt.figure(figsize=(6,4))
df.groupby('Region')['Monthly_Bill'].mean().plot(kind='bar', color='lightgreen')
plt.title('Average Monthly Bill by Region')
plt.ylabel('Average Bill (₹)')
plt.show()
```



STEP 18:- Representing the users data according to the plan types the users is having in bar graph where postpaid is in blue and prepaid in yellow

```
plt.figure(figsize=(5,4))
df['Plan_Type'].value_counts().plot(kind='bar', color=['skyblue','orange'])
plt.title('Users by Plan Type')
plt.xlabel('Plan Type')
plt.ylabel('Count')
plt.show()
```



STEP 19:- Filling the missing Age values using the mean age of the column

```
df['Age'].fillna(df['Age'].mean(),inplace=True)
df
```

```
/tmp/ipython-input-2368695939.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].mean(),inplace=True)
```

	customerID	User_ID	gender	Age	Region	tenure	Column2	MultipleLines	InternetService	PaymentMethod	Plan_Type	Data_Usage_GB	Call_Duration_Min	Monthly_Bill	MonthlyCharges	TotalCharges	Network_Rating	Column3	Churn
0	7590-VHVEG	1	Female	34	East	1.0	NaN	No phone service	DSL	Electronic check	Prepaid	43.39	352	518.17	29.85	29.85	4	NaN	No
1	5575-GNVDE	2	Male	26	South	34.0	NaN	No	DSL	Mailed check	Postpaid	11.17	852	1084.72	56.95	1889.50	1	NaN	No
2	3668-QPYBK	3	Male	50	North	2.0	NaN	No	DSL	Mailed check	Prepaid	18.34	1013	1449.22	53.85	108.15	3	NaN	Yes
3	7795-CFOCW	4	Male	37	West	45.0	NaN	No phone service	DSL	Bank transfer (automatic)	Postpaid	3.89	118	281.89	42.30	1840.75	4	NaN	No
4	9237-HQITU	5	Female	30	South	2.0	NaN	No	Fiber optic	Electronic check	Postpaid	38.28	1202	703.62	70.70	151.65	1	NaN	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	3842-IYKUE	996	Female	62	North	35.0	NaN	Yes	Fiber optic	Credit card (automatic)	Prepaid	26.06	236	559.49	85.30	2917.50	2	NaN	Yes
996	6641-XRPSU	997	Female	52	East	34.0	NaN	No	Fiber optic	Credit card (automatic)	Postpaid	39.43	781	909.40	70.00	2416.10	4	NaN	Yes
997	1374-DMZUI	998	Female	23	North	4.0	NaN	Yes	Fiber optic	Electronic check	Prepaid	47.16	1263	1938.71	94.30	424.45	1	NaN	Yes
998	2545-LXYVJ	999	Male	36	East	72.0	NaN	No	No	Bank transfer (automatic)	Prepaid	37.47	1461	1349.75	20.70	1492.10	5	NaN	No
999	NaN	1000	NaN	47	South	NaN	NaN	NaN	NaN	NaN	Postpaid	36.17	478	1172.35	NaN	NaN	3	NaN	NaN

1000 rows × 19 columns

STEP 20:- Finding the mean of the data of all the users.

```
mean = df.mean(numeric_only=True)
mean
```

	0
User_ID	500.50000
Age	41.39200
tenure	32.21700
Column2	0.00000
Data_Usage_GB	25.69384
Call_Duration_Min	781.86100
Monthly_Bill	1028.50807
MonthlyCharges	66.41350
TotalCharges	2331.91600
Network_Rating	3.03000
Column3	0.00000

dtype: float64

STEP 21:-Finding the median of the data of all the users

```
median = df.median(numeric_only=True)
median
```

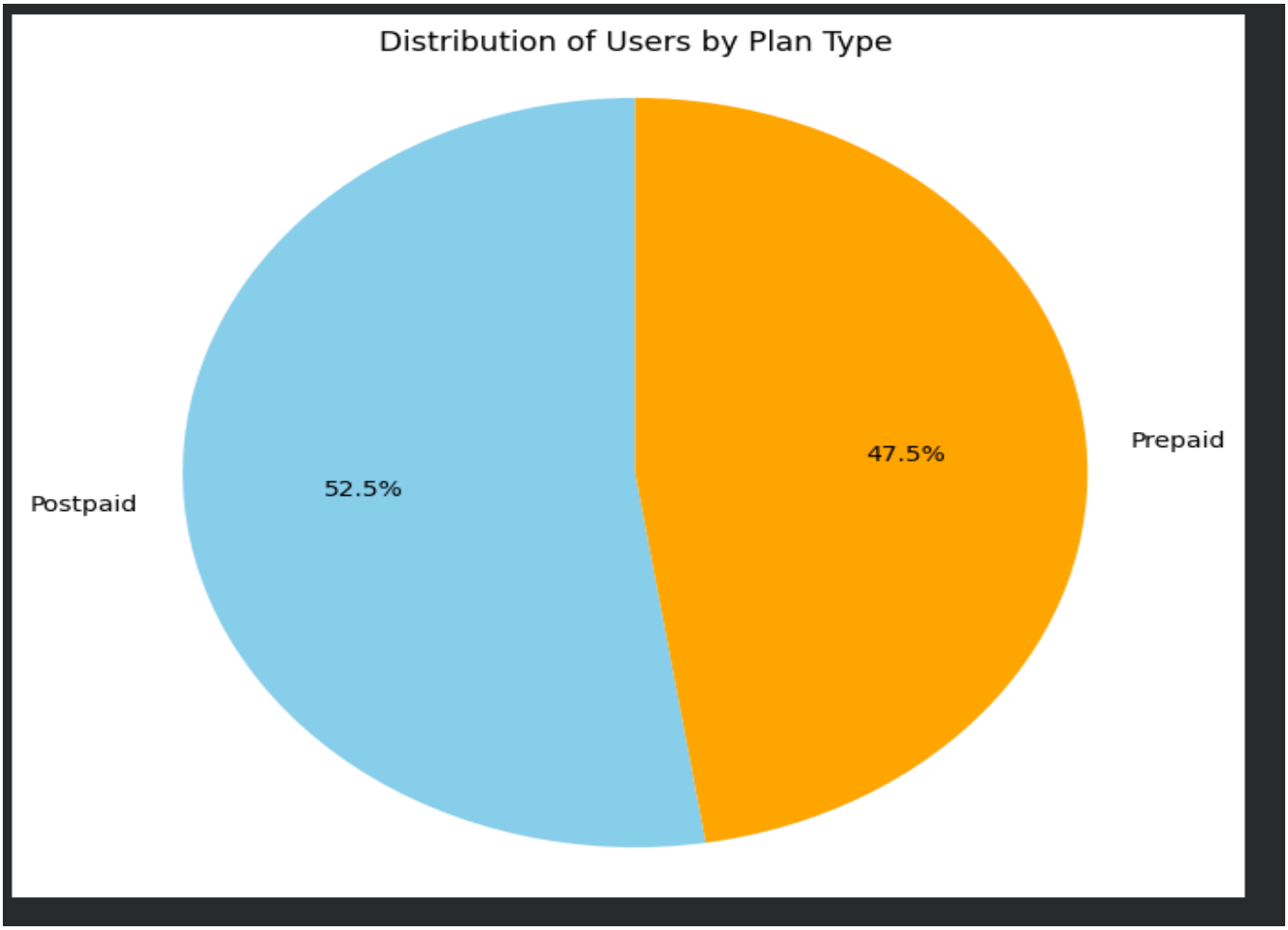
	0
User_ID	500.500
Age	42.000
tenure	29.000
Column2	0.000
Data_Usage_GB	26.525
Call_Duration_Min	787.500
Monthly_Bill	1009.190
MonthlyCharges	74.250
TotalCharges	1448.875
Network_Rating	3.000
Column3	0.000

dtype: float64

STEP 22:- Shows the percentage distribution of different Plan Types using a pie chart.

1. Count the number of users in each (Plan_Type) category.
2. Create a pie chart to visualize the proportion of users in each plan type.
3. Display percentage values on each slice and rotate the chart for better readability.
4. Set colors, title, and ensure the pie chart appears as a perfect circle.
5. Finally, render the chart to show the distribution clearly.

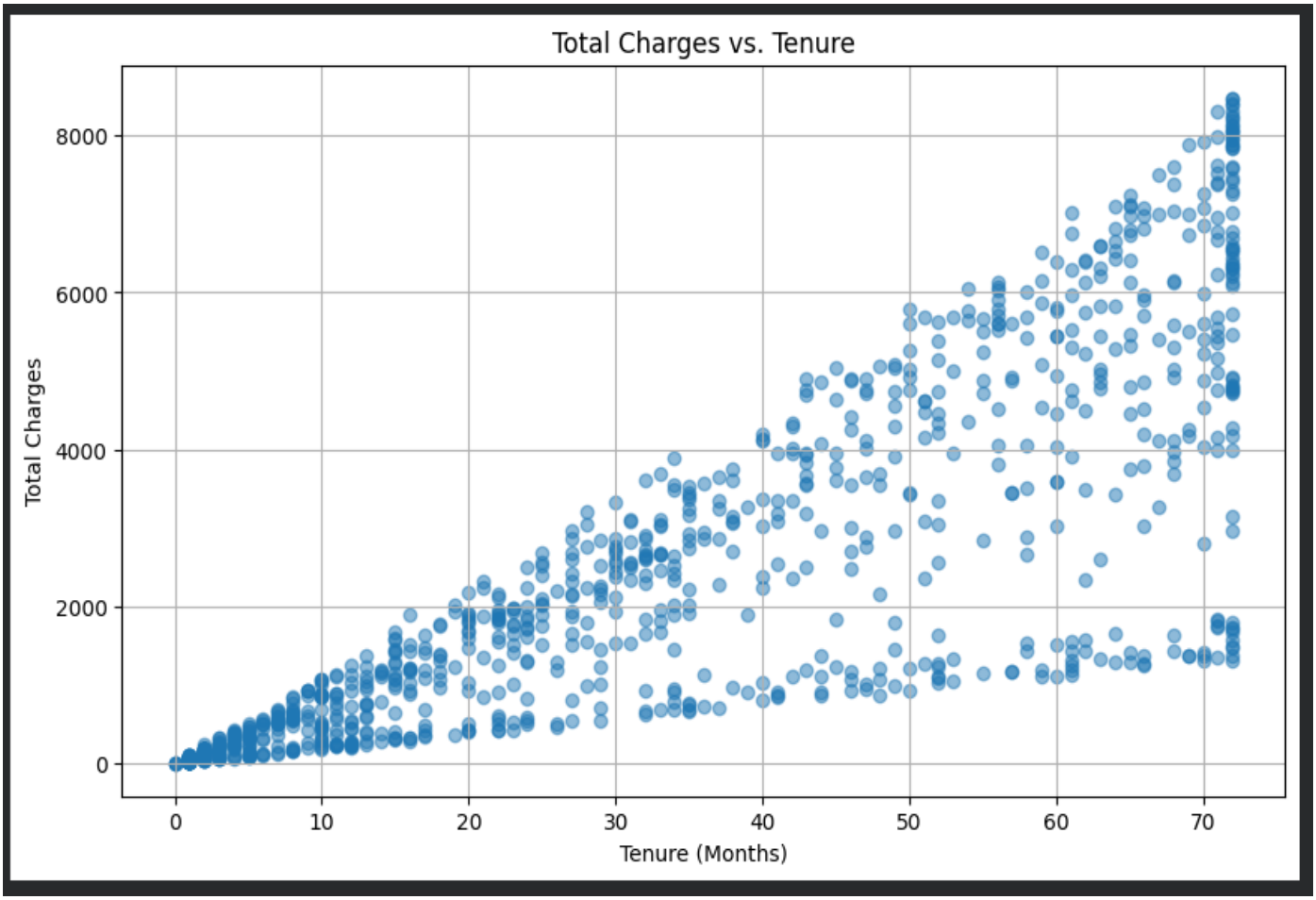
```
plan_type_counts = df['Plan_Type'].value_counts()
plt.figure(figsize=(7, 7))
plt.pie(plan_type_counts, labels=plan_type_counts.index, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'orange'])
plt.title('Distribution of Users by Plan Type')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



STEP 23:- Visualizes the relationship between customer tenure and total charges using a scatter plot.

2 **Set figure size: [**plt.figure(figsize=(10, 6)**)]** creates a larger plot for better visibility.
3 **Plot data points: [**plt.scatter(df['tenure'], df['TotalCharges'], alpha=0.5)**]** creates a scatter plot of TotalCharges vs. tenure. The alpha=0.5 makes points semi-transparent to reduce overlap clutter.
4 **Label axes: -[**plt.xlabel and plt.ylabel**]** add descriptive labels for X and Y axes.
5 **Add title: [**plt.title('Total Charges vs. Tenure') **]**provides a clear title explaining the relationship being visualized.
6 **Add grid: [**plt.grid(True)**]** helps in better visual estimation of values.
7 **Display plot:[** plt.show()**]** renders the scatter plot to the screen.

```
plt.figure(figsize=(10, 6))
plt.scatter(df['tenure'], df['TotalCharges'], alpha=0.5)
plt.xlabel('Tenure (Months)')
plt.ylabel('Total Charges')
plt.title('Total Charges vs. Tenure')
plt.grid(True)
plt.show()
```



STEP 24:- Creates a new Excel workbook, renames the sheet, prints the dataframe columns, and saves the file as 'Revenue_by_product.xlsx'

```
import openpyxl
from openpyxl import workbook
from openpyxl.chart import BarChart, Reference

Wb = openpyxl.Workbook()
Ws = Wb.active
Ws.title = "Revenue by Product"
print(df.columns)
Wb.save("Revenue_by_product.xlsx")
```

```
Index(['customerID', 'User_ID', 'gender', 'Age', 'Region', 'tenure', 'Column2',
       'MultipleLines', 'InternetService', 'PaymentMethod', 'Plan_Type',
       'Data_Usage_GB', 'Call_Duration_Min', 'Monthly_Bill', 'MonthlyCharges',
       'TotalCharges', 'Network_Rating', 'Column3', 'Churn'],
      dtype='object')
```