In [1]:
```
#Importing Basic Libraries
import pandas as pd
import numpy as np
```

In [2]:
```
#Reading the dataset
df = pd.read_csv("telecom_churn.csv")
```

In [3]:
```
df
```

Out[3]:

	customer_id	telecom_partner	gender	age	state	city	pincode	date_of_regi
0	1	Reliance Jio	F	25	Karnataka	Kolkata	755597	202
1	2	Reliance Jio	F	55	Mizoram	Mumbai	125926	202
2	3	Vodafone	F	57	Arunachal Pradesh	Delhi	423976	202
3	4	BSNL	M	46	Tamil Nadu	Kolkata	522841	202
4	5	BSNL	F	26	Tripura	Delhi	740247	202
...	...	...	...	...	...	...	...	
243548	243549	Airtel	F	28	Mizoram	Kolkata	110295	202
243549	243550	Reliance Jio	F	52	Assam	Kolkata	713481	202
243550	243551	Reliance Jio	M	59	Tripura	Kolkata	520218	202
243551	243552	BSNL	M	49	Madhya Pradesh	Kolkata	387744	202
243552	243553	BSNL	F	37	Telangana	Hyderabad	139086	202

243553 rows × 14 columns

In [4]:
```
#Checking Null values
df.isnull().sum()
```

Out[4]:
```
customer_id             0
telecom_partner         0
gender                  0
age                     0
state                   0
city                    0
pincode                 0
date_of_registration    0
num_dependents          0
estimated_salary        0
calls_made              0
sms_sent                0
data_used               0
churn                   0
dtype: int64
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243553 entries, 0 to 243552
Data columns (total 14 columns):
 #   Column                Non-Null Count   Dtype 
---  ------                --------------   ----- 
 0   customer_id           243553 non-null  int64 
 1   telecom_partner       243553 non-null  object
 2   gender                243553 non-null  object
 3   age                   243553 non-null  int64 
 4   state                 243553 non-null  object
 5   city                  243553 non-null  object
 6   pincode               243553 non-null  int64 
 7   date_of_registration  243553 non-null  object
 8   num_dependents        243553 non-null  int64 
 9   estimated_salary      243553 non-null  int64 
 10  calls_made            243553 non-null  int64 
 11  sms_sent              243553 non-null  int64 
 12  data_used             243553 non-null  int64 
 13  churn                 243553 non-null  int64 
dtypes: int64(9), object(5)
memory usage: 26.0+ MB
```

In [6]:
```
#Statistical analysis of the features
df.describe(include="all")
```

Out[6]:

	customer_id	telecom_partner	gender	age	state	city	pinc
count	243553.000000	243553	243553	243553.000000	243553	243553	243553.00(
unique	NaN	4	2	NaN	28	6	
top	NaN	Reliance Jio	M	NaN	Uttarakhand	Chennai	
freq	NaN	61123	145977	NaN	8856	40749	
mean	121777.000000	NaN	NaN	46.077609	NaN	NaN	549501.27(
std	70307.839393	NaN	NaN	16.444029	NaN	NaN	259808.86(
min	1.000000	NaN	NaN	18.000000	NaN	NaN	100006.00(
25%	60889.000000	NaN	NaN	32.000000	NaN	NaN	324586.00(
50%	121777.000000	NaN	NaN	46.000000	NaN	NaN	548112.00(
75%	182665.000000	NaN	NaN	60.000000	NaN	NaN	774994.00(
max	243553.000000	NaN	NaN	74.000000	NaN	NaN	999987.00(

In [7]:
```
df.columns
```

Out[7]:
```
Index(['customer_id', 'telecom_partner', 'gender', 'age', 'state', 'city',
       'pincode', 'date_of_registration', 'num_dependents', 'estimated_salar
y',
       'calls_made', 'sms_sent', 'data_used', 'churn'],
      dtype='object')
```

In [8]:
```
#Gender class numbers
df["gender"].value_counts()
```

Out[8]:
```
M    145977
F     97576
Name: gender, dtype: int64
```

In [9]:
```
df.shape
```

Out[9]:
```
(243553, 14)
```

In [10]:
```
#Dropped useless column
df.drop(["customer_id"] ,axis=1, inplace=True)
```

In [11]:
```
df
```

Out[11]:

	telecom_partner	gender	age	state	city	pincode	date_of_registration	num_
0	Reliance Jio	F	25	Karnataka	Kolkata	755597	2020-01-01	
1	Reliance Jio	F	55	Mizoram	Mumbai	125926	2020-01-01	
2	Vodafone	F	57	Arunachal Pradesh	Delhi	423976	2020-01-01	
3	BSNL	M	46	Tamil Nadu	Kolkata	522841	2020-01-01	
4	BSNL	F	26	Tripura	Delhi	740247	2020-01-01	
...	...	...	...	...	...	...	...	...
243548	Airtel	F	28	Mizoram	Kolkata	110295	2023-05-03	
243549	Reliance Jio	F	52	Assam	Kolkata	713481	2023-05-03	
243550	Reliance Jio	M	59	Tripura	Kolkata	520218	2023-05-03	
243551	BSNL	M	49	Madhya Pradesh	Kolkata	387744	2023-05-03	
243552	BSNL	F	37	Telangana	Hyderabad	139086	2023-05-04	

243553 rows × 13 columns

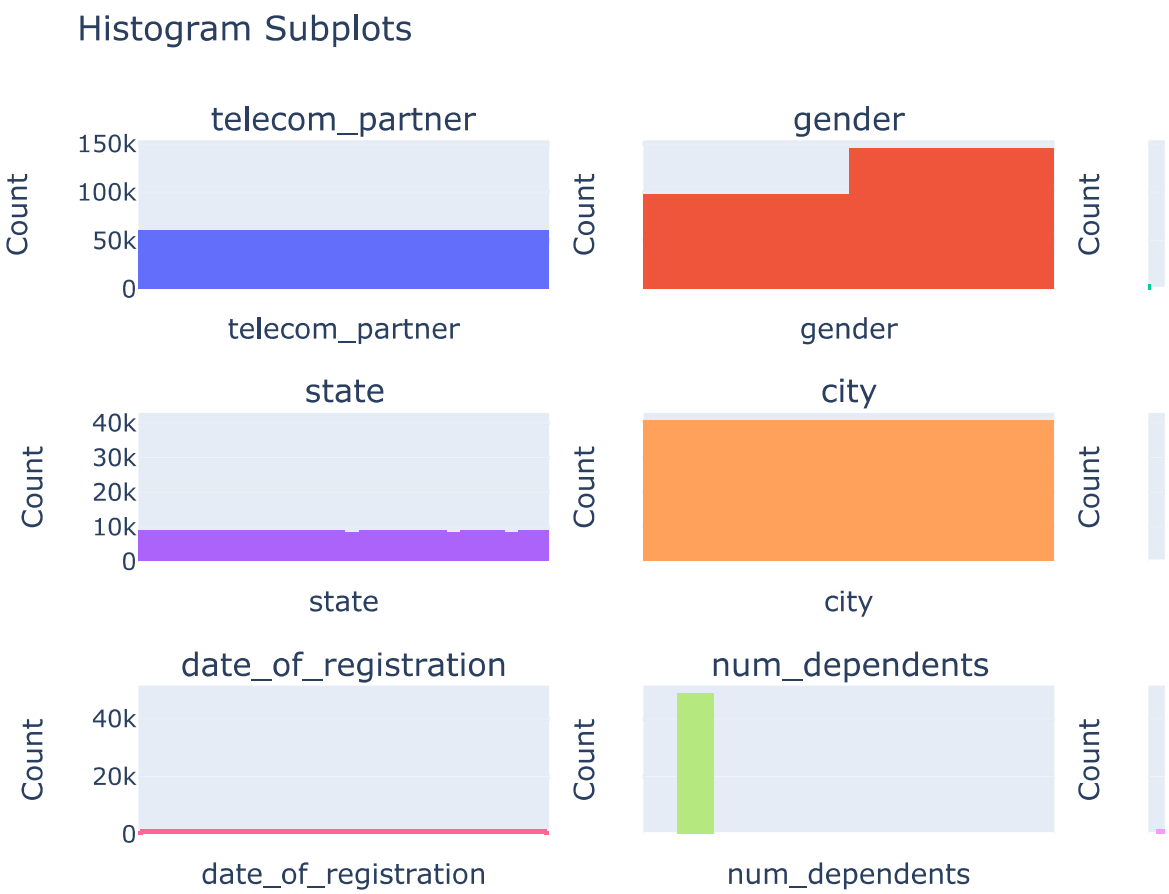
In [12]:
```
#Importing lib for eda
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
```

In [13]:
```
df["age"].value_counts().head()
```

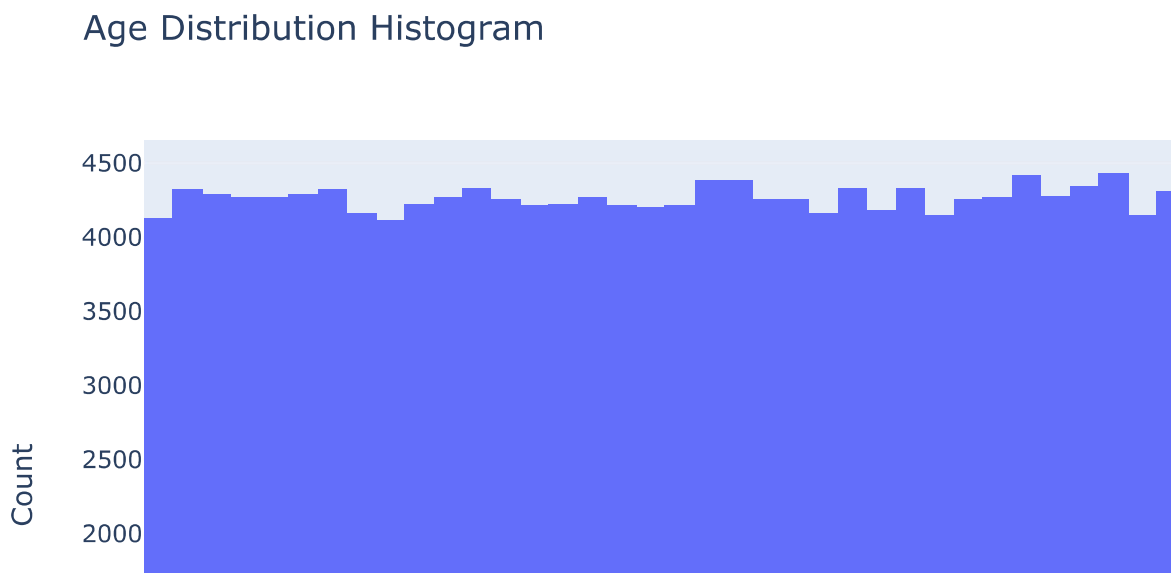
Out[13]:
```
60    4424
51    4423
48    4414
62    4402
38    4392
Name: age, dtype: int64
```

In [14]:
```
import plotly.subplots as sp
```

In [15]:
```
#Count plot for each feature
# Create a subplot layout
fig = sp.make_subplots(rows=5, cols=3, subplot_titles=df.columns[:-1], shared_

# Populating the subplots with histograms
for i, col in enumerate(df.columns[:-1]):
    trace = go.Histogram(x=df[col], name=col)
    fig.add_trace(trace, row=(i // 3) + 1, col=(i % 3) + 1)

# Update axis labels and titles
for i, col in enumerate(df.columns[:-1]):
    fig.update_xaxes(title_text=col, row=(i // 3) + 1, col=(i % 3) + 1)
    fig.update_yaxes(title_text='Count', row=(i // 3) + 1, col=(i % 3) + 1)

# Update layout
fig.update_layout(title='Histogram Subplots', height=800)


fig.show()
```


Histogram Subplots
telecom_partner
gender
state
city
date_of_registration
num_dependents
Count
150k
100k
50k
0
40k
30k
20k
10k

In [16]:
```
#Age distribution
fig = px.histogram(df, x='age', title='Age Distribution Histogram')

# Update axis labels
fig.update_layout(xaxis_title='Age', yaxis_title='Count')

# Show the plot
fig.show()
```



In [17]:
```
age_bins = [i for i in range(1, 101, 10)]
```

In [18]:
```
df['age_group'] = pd.cut(df['age'], bins=age_bins)
```

In [19]:
```
df
```

Out[19]:

	telecom_partner	gender	age	state	city	pincode	date_of_registration	num
0	Reliance Jio	F	25	Karnataka	Kolkata	755597	2020-01-01	
1	Reliance Jio	F	55	Mizoram	Mumbai	125926	2020-01-01	
2	Vodafone	F	57	Arunachal Pradesh	Delhi	423976	2020-01-01	
3	BSNL	M	46	Tamil Nadu	Kolkata	522841	2020-01-01	
4	BSNL	F	26	Tripura	Delhi	740247	2020-01-01	
...	...	...	...	...	...	...	...	
243548	Airtel	F	28	Mizoram	Kolkata	110295	2023-05-03	
243549	Reliance Jio	F	52	Assam	Kolkata	713481	2023-05-03	
243550	Reliance Jio	M	59	Tripura	Kolkata	520218	2023-05-03	
243551	BSNL	M	49	Madhya Pradesh	Kolkata	387744	2023-05-03	
243552	BSNL	F	37	Telangana	Hyderabad	139086	2023-05-04	

243553 rows × 14 columns

In [20]:
```
age_group_counts = df['age_group'].value_counts().sort_index()
```

In [21]:
```
age_group_counts
```

Out[21]:
```
(1, 11]         0
(11, 21]    16988
(21, 31]    42436
(31, 41]    42562
(41, 51]    42964
(51, 61]    42650
(61, 71]    43183
(71, 81]    12770
(81, 91]        0
Name: age_group, dtype: int64
```

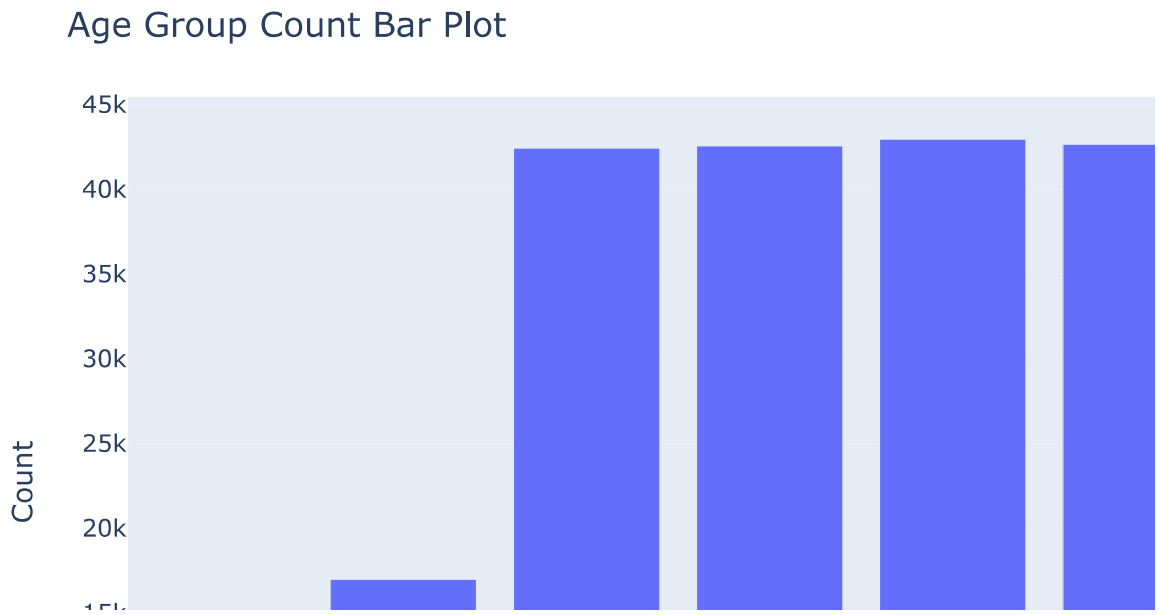
In [22]:
```
df_ag = pd.DataFrame(age_group_counts)
df_ag
```

Out[22]:

	age_group
(1, 11]	0
(11, 21]	16988
(21, 31]	42436
(31, 41]	42562
(41, 51]	42964
(51, 61]	42650
(61, 71]	43183
(71, 81]	12770
(81, 91]	0

In [23]:
```
fig = px.bar(x=[i for i in range(0,9)], y=df_ag['age_group'], labels={'x': 'Ag
```

In [24]:
```
#Age group wise analysis
fig.update_xaxes(tickvals=age_bins)

# Update layout
fig.update_layout(title='Age Group Count Bar Plot')
```

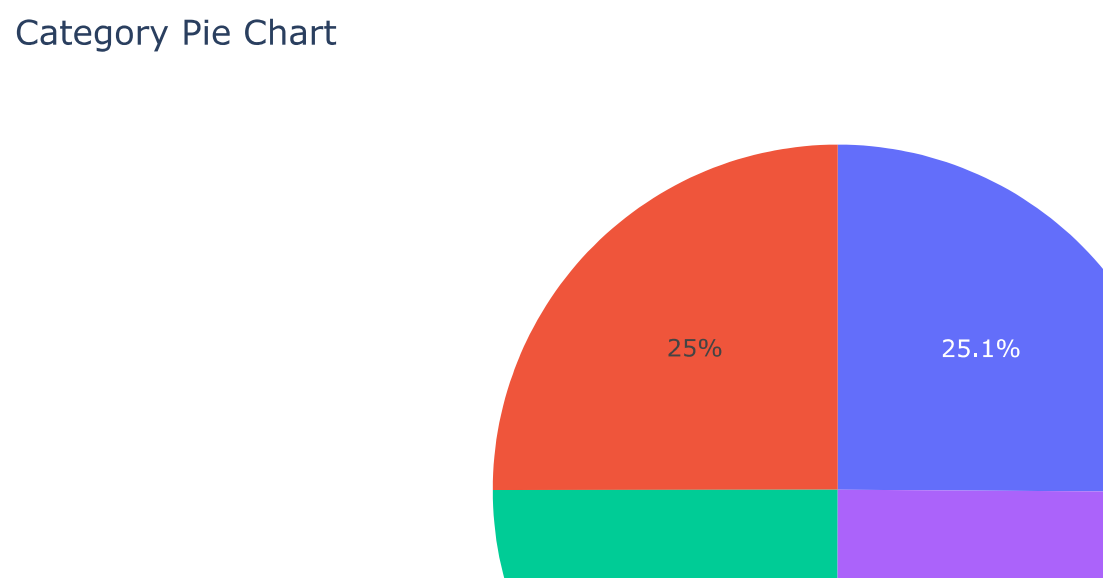


In [25]:
```
category_counts = df['telecom_partner'].value_counts()
sum = 0
for i in range(0,4):
    sum = sum + category_counts[i]
sum
```

Out[25]: 243553

In [26]:
```
fig = px.pie(values=(category_counts.values / sum), names=category_counts.inde
```

In [27]:
```
#Company wise distribution
fig.show()
```



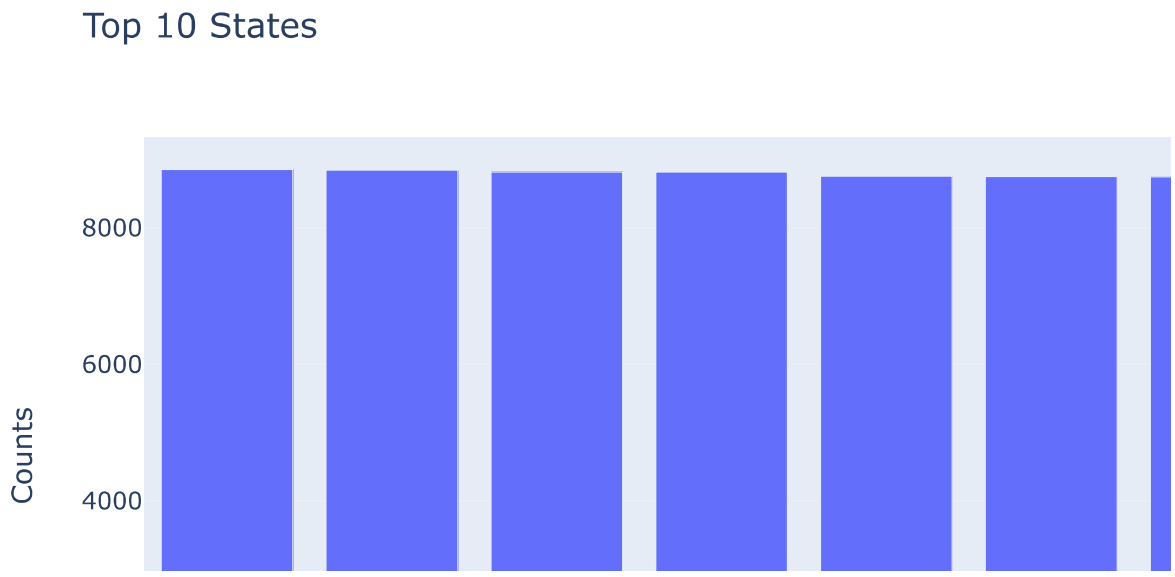
In [28]:
```
df["telecom_partner"].value_counts()
```

Out[28]:
```
Reliance Jio    61123
Airtel          60905
Vodafone        60802
BSNL            60723
Name: telecom_partner, dtype: int64
```

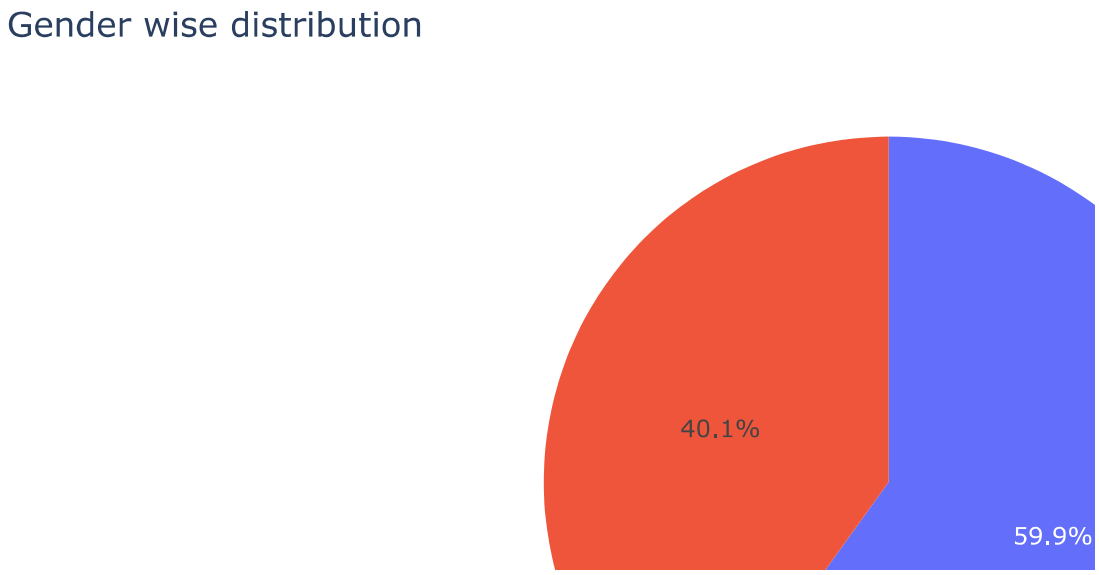
In [29]:
```
#Presenting top 10 state
top_10_state = df["state"].value_counts().sort_values(ascending=False).head(10
```

In [30]:
```
fig = px.bar(x=top_10_state.index , y = top_10_state.values , labels = {"x":"T
```

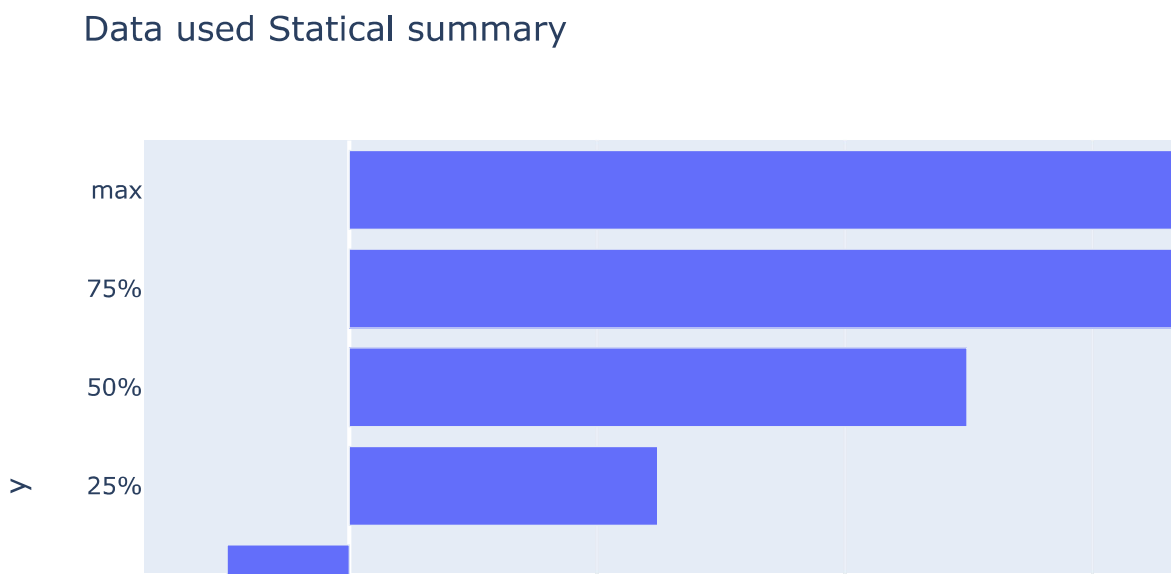
In [31]:
```
fig.show()
```



In [32]:
```
gender = df["gender"].value_counts()

fig = px.pie(values=gender.values , names=gender.index , title = "Gender wise
fig.show()
```



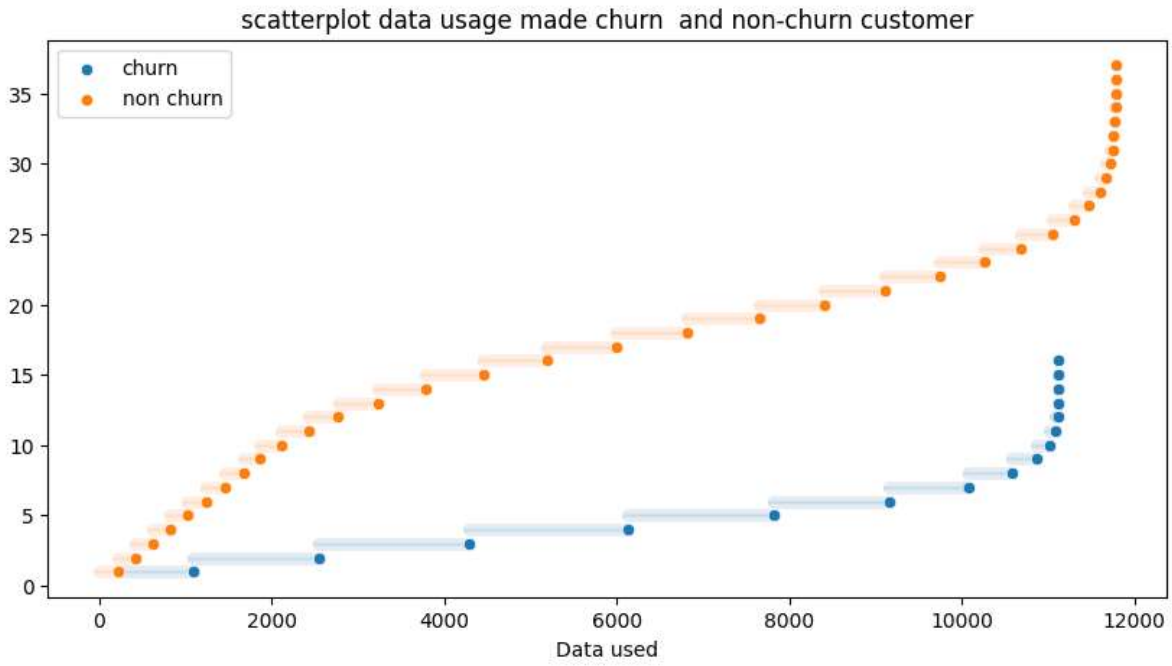
In [33]:
```
#Statistics of Data used feature
data_used=df["data_used"].describe()
fig = px.bar(x=data_used.values[1:] , y = data_used.index[1:] , title = "Data
fig.show()
```



In [34]:
```
churn_data_data_used=df[df['churn']==1]['data_used']
non_churn_data_used=df[df['churn']==0]['data_used']
```

In [35]:
```
churn_value_calls_made=list(churn_data_data_used.value_counts().sort_values())
non_churn_value_calls_made=list(non_churn_data_used.value_counts().sort_values
plt.figure(figsize=(10,5))
sns.scatterplot(churn_value_calls_made,label="churn")
sns.scatterplot(non_churn_value_calls_made,label="non churn")
plt.xlabel("Data used")
plt.title("scatterplot data usage made churn  and non-churn customer")
```

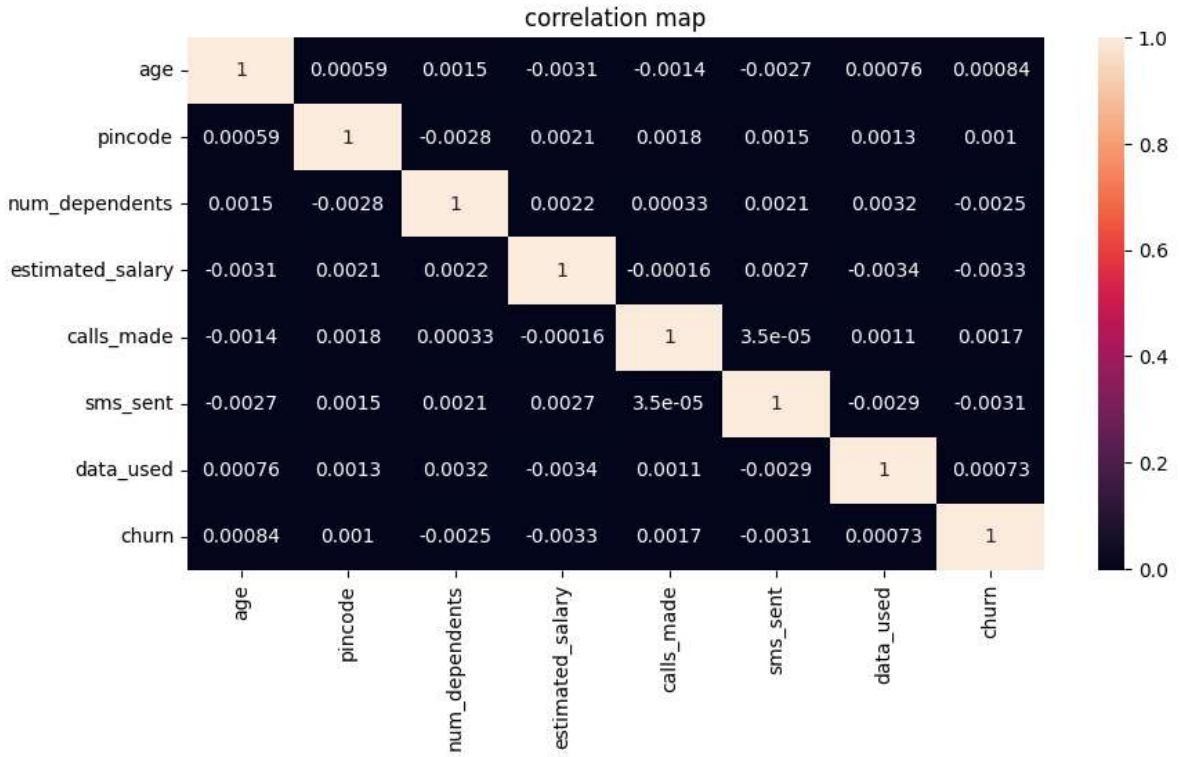
Out[35]:
```
Text(0.5, 1.0, 'scatterplot data usage made churn  and non-churn customer')
```



In [36]:
```
df = df.drop("age_group" , axis =1)
```

In [37]:
```
#Heatmap
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(),annot=True)
plt.title("correlation map")
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_18904\4288540805.py:3: FutureWarni
ng:

The default value of numeric_only in DataFrame.corr is deprecated. In a futur
e version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
```



In [38]:
```
#Label encoding categorical columns to numerical
from sklearn.preprocessing import LabelEncoder
model=LabelEncoder()
for i in df.columns:
    if df[i].dtype=="object":
        df[i]=model.fit_transform(df[i])
```

In [39]:
```
df
```

Out[39]:

	telecom_partner	gender	age	state	city	pincode	date_of_registration	num_dependent
0	2	0	25	10	4	755597	0	
1	2	0	55	16	5	125926	0	
2	3	0	57	1	2	423976	0	
3	1	1	46	22	4	522841	0	
4	1	0	26	24	2	740247	0	
...	...	...	...	...	...	...	...	.
243548	0	0	28	16	4	110295	1218	
243549	2	0	52	2	4	713481	1218	
243550	2	1	59	24	4	520218	1218	
243551	1	1	49	12	4	387744	1218	
243552	1	0	37	23	3	139086	1219	

243553 rows × 13 columns

In [40]:
```
#Train Test Split
from sklearn.model_selection import train_test_split
X=df.drop("churn",axis=1)
y=df['churn']
```

In [41]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size=0
```

In [42]:
```
#Feature scaling
from sklearn.preprocessing import StandardScaler

model=StandardScaler()

X_train=pd.DataFrame(model.fit_transform(X_train),columns=X_train.columns)
X_test=pd.DataFrame(model.fit_transform(X_test),columns=X_test.columns)
```

In [43]:
```
#Initialising the model
from sklearn.linear_model import LogisticRegression

Model = LogisticRegression()

#Fitting to Logistic regression model
Model.fit(X_train,y_train)
```

Out[43]:
```
LogisticRegression
LogisticRegression()
```

In [44]:
```
y_pred = Model.predict(X_test)
```

In [45]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [46]:
```
#Evaluating the result
accuracy=accuracy_score(y_test,y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7991624068485558
```

In [ ]:

In [ ]: