In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [2]:
```
path=r"C:\Users\Sruth\Downloads\telecom_churn_data.csv"
churn_data=pd.read_csv(path)
churn_data
```

Out[2]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	409-8743	Female	36	62	no
1	2015	100643	340-5930	Female	39	149	no
2	2015	100756	372-3750	Female	65	126	no
3	2015	101595	331-4902	Female	24	131	no
4	2015	101653	351-8398	Female	40	191	no
...	...	...	...	...	...	...	...
1995	2015	997132	385-7387	Female	54	75	no
1996	2015	998086	383-9255	Male	45	127	no
1997	2015	998474	353-2080	NaN	53	94	no
1998	2015	998934	359-7788	Male	40	94	no
1999	2015	999961	414-1496	Male	37	73	no

2000 rows × 16 columns

In [3]:
```
churn_data.isnull().sum()
```

Out[3]:
```
year                      0
customer_id               0
phone_no                  0
gender                   24
age                       0
no_of_days_subscribed     0
multi_screen              0
mail_subscribed           0
weekly_mins_watched       0
minimum_daily_mins        0
maximum_daily_mins        0
weekly_max_night_mins     0
videos_watched            0
maximum_days_inactive    28
customer_support_calls    0
churn                    35
dtype: int64
```

Filling missing values

In [59]:
```
mode_churn=churn_data['churn'].mode()
mode_churn[0] # pass the only value
```

Out[59]:
```
0.0
```

In [60]:
```
churn_data['churn'].fillna(mode_churn[0],inplace=True)
```

In [6]:
```
# cahnge the data into int
```

In [61]:
```
# col1:gender(object),col2:maximum_days_inactive(float),col3:churn(float)  (cont
churn_data['churn'].astype(int)
```

Out[61]:
```
0       0
1       0
2       1
3       0
4       0
       ..
1995    0
1996    0
1997    0
1998    0
1999    1
Name: churn, Length: 2000, dtype: int32
```

In [62]:
```
mode_gender=churn_data['gender'].mode()
mode_gender[0]
```

Out[62]:
```
'Male'
```

In [63]:
```
churn_data['churn'].fillna(mode_gender[0],inplace=True)
```

In [64]:
```
mode_max_days=churn_data['maximum_days_inactive'].mode()
mode_max_days[0]
```

Out[64]:
```
3.0
```

In [65]:
```
churn_data['churn'].fillna(mode_max_days[0],inplace=True)
```

In [66]:
```
churn_data=pd.read_csv(path)
mode_max_days=churn_data['maximum_days_inactive'].mode()
churn_data['maximum_days_inactive'].fillna(mode_max_days[0],inplace=True)
mode_gender=churn_data['gender'].mode()
churn_data['gender'].fillna(mode_gender[0],inplace=True)
mode_churn=churn_data['churn'].mode()
churn_data['churn'].fillna(mode_churn[0],inplace=True)
```

changing astype

In [13]:
```
churn_data['churn'].astype(int)
```

Out[13]:
```
0       0
1       0
2       1
3       0
4       0
       ..
1995    3
1996    0
1997    0
1998    0
1999    1
Name: churn, Length: 2000, dtype: int32
```

In [67]:
```
churn_data
```

Out[67]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	409-8743	Female	36	62	no
1	2015	100643	340-5930	Female	39	149	no
2	2015	100756	372-3750	Female	65	126	no
3	2015	101595	331-4902	Female	24	131	no
4	2015	101653	351-8398	Female	40	191	no
...	...	...	...	...	...	...	...
1995	2015	997132	385-7387	Female	54	75	no
1996	2015	998086	383-9255	Male	45	127	no
1997	2015	998474	353-2080	Male	53	94	no
1998	2015	998934	359-7788	Male	40	94	no
1999	2015	999961	414-1496	Male	37	73	no

2000 rows × 16 columns

In [68]:
```
churn_data['churn'].astype(int)
churn_data.dtypes
```

Out[68]:
```
year                      int64
customer_id               int64
phone_no                 object
gender                   object
age                       int64
no_of_days_subscribed     int64
multi_screen             object
mail_subscribed          object
weekly_mins_watched     float64
minimum_daily_mins      float64
maximum_daily_mins      float64
weekly_max_night_mins     int64
videos_watched            int64
maximum_days_inactive   float64
customer_support_calls    int64
churn                   float64
dtype: object
```

In [69]:
```
churn_data.isnull().sum()
```

Out[69]:
```
year                    0
customer_id             0
phone_no                0
gender                  0
age                     0
no_of_days_subscribed   0
multi_screen            0
mail_subscribed         0
weekly_mins_watched     0
minimum_daily_mins      0
maximum_daily_mins      0
weekly_max_night_mins   0
videos_watched          0
maximum_days_inactive   0
customer_support_calls  0
churn                   0
dtype: int64
```

In [34]:
```
churn_data.size
```

Out[34]:
```
32000
```

In [35]:
```
churn_data.shape
```

Out[35]:
```
(2000, 16)
```

In [36]:
```
churn_data.columns
```

Out[36]:
```
Index(['year', 'customer_id', 'phone_no', 'gender', 'age',
       'no_of_days_subscribed', 'multi_screen', 'mail_subscribed',
       'weekly_mins_watched', 'minimum_daily_mins', 'maximum_daily_mins',
       'weekly_max_night_mins', 'videos_watched', 'maximum_days_inactive',
       'customer_support_calls', 'churn'],
      dtype='object')
```

- It is a supervised learning problem
- supervised learning problem means data has a target column

- Here data has target column called : **churn**
- Churn means the customer stay with the company or not stay with the company
- stay with company represents with True or Yes : 1
- Not stay with the company represents with False or No : 0
- Tommorw a new input will come our model will say the churn is Yes or No
- It is a classification problem
- Means we are classifying the customers
- We have one type under supervised learning
- It is regression problem
- It is kind of a forecasting , the output data is represent with numbers
- Mostly continues data
- Ex: predicting the ICICI bank share price , predicting the sales of a company

In [37]:
```
churn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   year                    2000 non-null   int64  
 1   customer_id             2000 non-null   int64  
 2   phone_no                2000 non-null   object 
 3   gender                  2000 non-null   object 
 4   age                     2000 non-null   int64  
 5   no_of_days_subscribed   2000 non-null   int64  
 6   multi_screen            2000 non-null   object 
 7   mail_subscribed         2000 non-null   object 
 8   weekly_mins_watched     2000 non-null   float64
 9   minimum_daily_mins      2000 non-null   float64
 10  maximum_daily_mins      2000 non-null   float64
 11  weekly_max_night_mins   2000 non-null   int64  
 12  videos_watched          2000 non-null   int64  
 13  maximum_days_inactive   2000 non-null   float64
 14  customer_support_calls  2000 non-null   int64  
 15  churn                   2000 non-null   float64
dtypes: float64(5), int64(7), object(4)
memory usage: 250.1+ KB
```

cat_cols and num_cols

In [39]:
```
cat_cols=churn_data.select_dtypes(include='object').columns
cat_cols
```

Out[39]:
```
Index(['phone_no', 'gender', 'multi_screen', 'mail_subscribed'], dtype='objec
t')
```

In [40]:
```
num_cols=churn_data.select_dtypes(exclude='object').columns
num_cols
```

Out[40]:
```
Index(['year', 'customer_id', 'age', 'no_of_days_subscribed',
       'weekly_mins_watched', 'minimum_daily_mins', 'maximum_daily_mins',
       'weekly_max_night_mins', 'videos_watched', 'maximum_days_inactive',
       'customer_support_calls', 'churn'],
      dtype='object')
```

All missing values are filled

In [41]:
```
churn_data.isnull().sum()
```

Out[41]:
```
year                    0
customer_id             0
phone_no                0
gender                  0
age                     0
no_of_days_subscribed   0
multi_screen            0
mail_subscribed         0
weekly_mins_watched     0
minimum_daily_mins      0
maximum_daily_mins      0
weekly_max_night_mins   0
videos_watched          0
maximum_days_inactive   0
customer_support_calls  0
churn                   0
dtype: int64
```

cat_cols

In [42]:
```
cat_cols
```

Out[42]:
```
Index(['phone_no', 'gender', 'multi_screen', 'mail_subscribed'], dtype='objec
t')
```

In [44]:
```
churn_data[['phone_no']]
```

Out[44]:

	phone_no
0	409-8743
1	340-5930
2	372-3750
3	331-4902
4	351-8398
...	...
1995	385-7387
1996	383-9255
1997	353-2080
1998	359-7788
1999	414-1496

2000 rows × 1 columns

In [45]:
```
churn_data['phone_no'].unique()
```

Out[45]:
```
array(['409-8743', '340-5930', '372-3750', ..., '353-2080', '359-7788',
       '414-1496'], dtype=object)
```

In [46]:
```
churn_data['phone_no'].value_counts()
```

Out[46]:
```
phone_no
409-8743    1
419-5505    1
418-9385    1
347-1914    1
360-6309    1
           ..
330-8142    1
357-5801    1
420-5990    1
390-2891    1
414-1496    1
Name: count, Length: 2000, dtype: int64
```

In [47]:
```
churn_data['gender'].value_counts()
```

Out[47]:
```
gender
Male      1077
Female     923
Name: count, dtype: int64
```

In [48]:
```
churn_data['multi_screen'].value_counts()
```

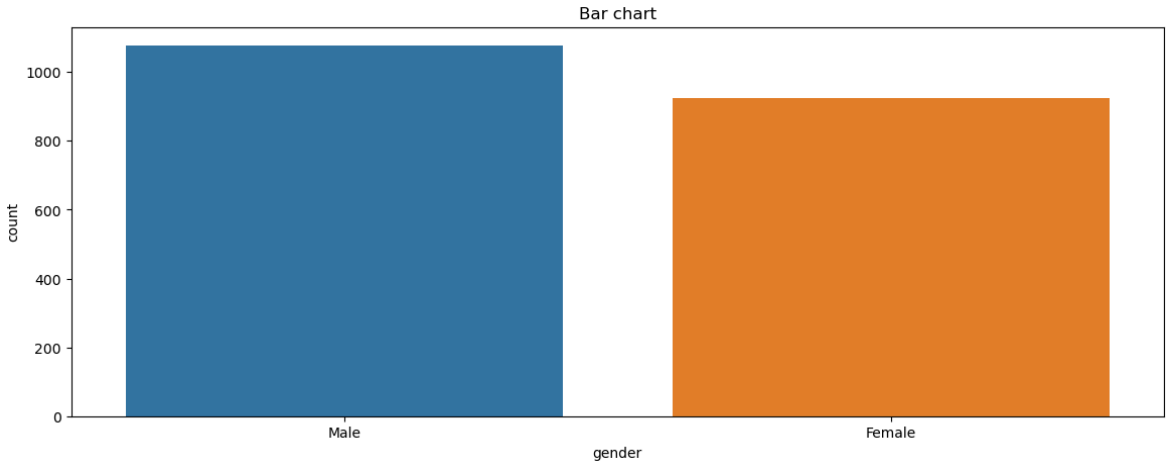
Out[48]:
```
multi_screen
no     1802
yes     198
Name: count, dtype: int64
```

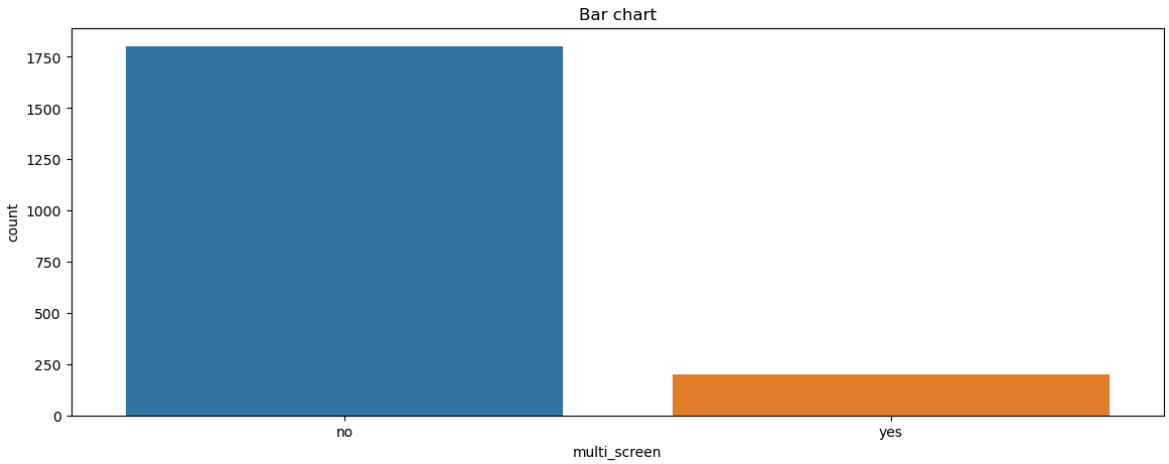
In [49]:
```
churn_data['mail_subscribed'].value_counts()
```

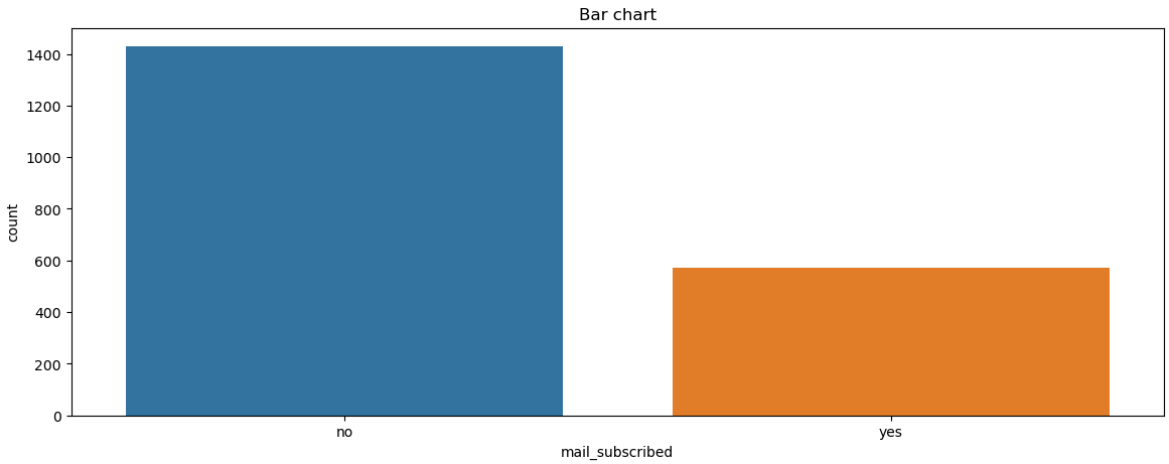
Out[49]:
```
mail_subscribed
no     1430
yes     570
Name: count, dtype: int64
```

count plot

In [51]:
```
import seaborn as sns
for i in cat_cols[1:]:
    order_phone_no=churn_data[i].value_counts().keys()
    plt.figure(figsize=(14,5))
    sns.countplot(data=churn_data,
                  x=i,
                  order=order_phone_no)
    plt.title('Bar chart')
```







pie chart

In [70]:
```
churn_data.isnull().sum()
```

Out[70]:
```
year                    0
customer_id             0
phone_no                0
gender                  0
age                     0
no_of_days_subscribed   0
multi_screen            0
mail_subscribed         0
weekly_mins_watched     0
minimum_daily_mins      0
maximum_daily_mins      0
weekly_max_night_mins   0
videos_watched          0
maximum_days_inactive   0
customer_support_calls  0
churn                   0
dtype: int64
```

In [71]:
```
churn_data
```

Out[71]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	409-8743	Female	36	62	no
1	2015	100643	340-5930	Female	39	149	no
2	2015	100756	372-3750	Female	65	126	no
3	2015	101595	331-4902	Female	24	131	no
4	2015	101653	351-8398	Female	40	191	no
...	...	...	...	...	...	...	...
1995	2015	997132	385-7387	Female	54	75	no
1996	2015	998086	383-9255	Male	45	127	no
1997	2015	998474	353-2080	Male	53	94	no
1998	2015	998934	359-7788	Male	40	94	no
1999	2015	999961	414-1496	Male	37	73	no

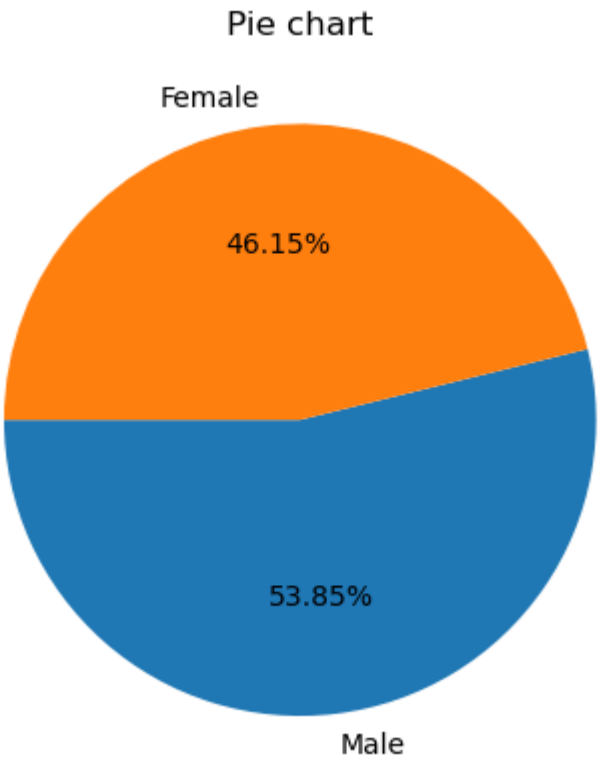
2000 rows × 16 columns

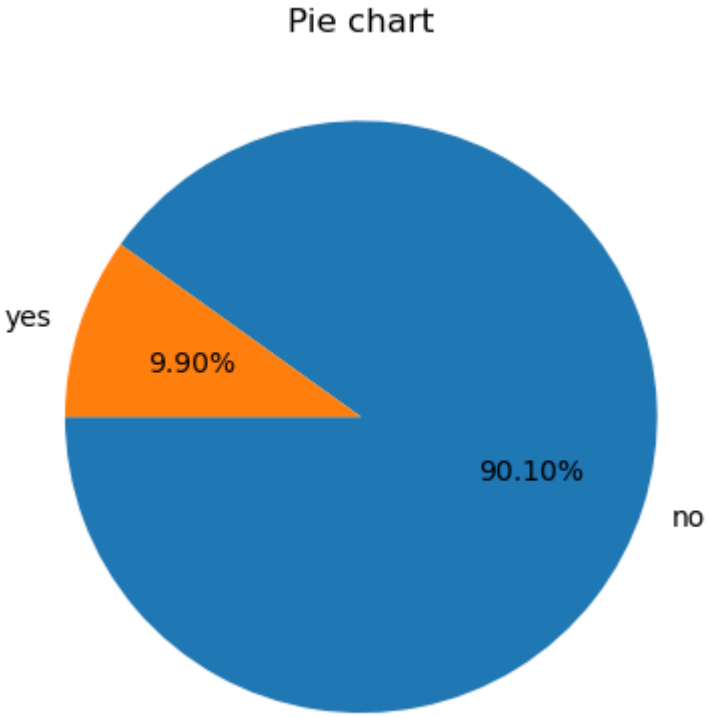
In [72]:

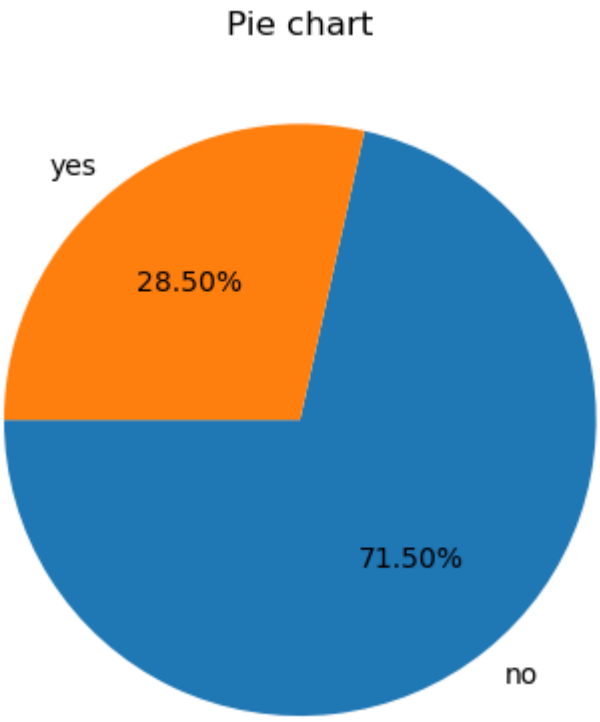
```
for i in cat_cols[1:]:
    keys=churn_data[i].value_counts().keys()
    values=churn_data[i].value_counts().values
    plt.pie(values,
            labels=keys,

            autopct="%0.2f%%",
            startangle=180
           )
    plt.title('Pie chart')

    plt.show()
```


Pie chart
Female
46.15%
53.85%
Male


Pie chart
yes
9.90%
90.10%
no



Numerical Data Analysis

In [73]:
```
num_cols
```

Out[73]:
```
Index(['year', 'customer_id', 'age', 'no_of_days_subscribed',
       'weekly_mins_watched', 'minimum_daily_mins', 'maximum_daily_mins',
       'weekly_max_night_mins', 'videos_watched', 'maximum_days_inactive',
       'customer_support_calls', 'churn'],
      dtype='object')
```

Describe function

In [74]:
```
churn_data.describe()
```

Out[74]:

	year	customer_id	age	no_of_days_subscribed	weekly_mins_watched
count	2000.0	2000.000000	2000.00000	2000.000000	2000.000000
mean	2015.0	554887.157500	38.69050	99.750000	270.178425
std	0.0	261033.690318	10.20641	39.755386	80.551627
min	2015.0	100198.000000	18.00000	1.000000	0.000000
25%	2015.0	328634.750000	32.00000	73.000000	218.212500
50%	2015.0	567957.500000	37.00000	99.000000	269.925000
75%	2015.0	773280.250000	44.00000	127.000000	324.675000
max	2015.0	999961.000000	82.00000	243.000000	526.200000

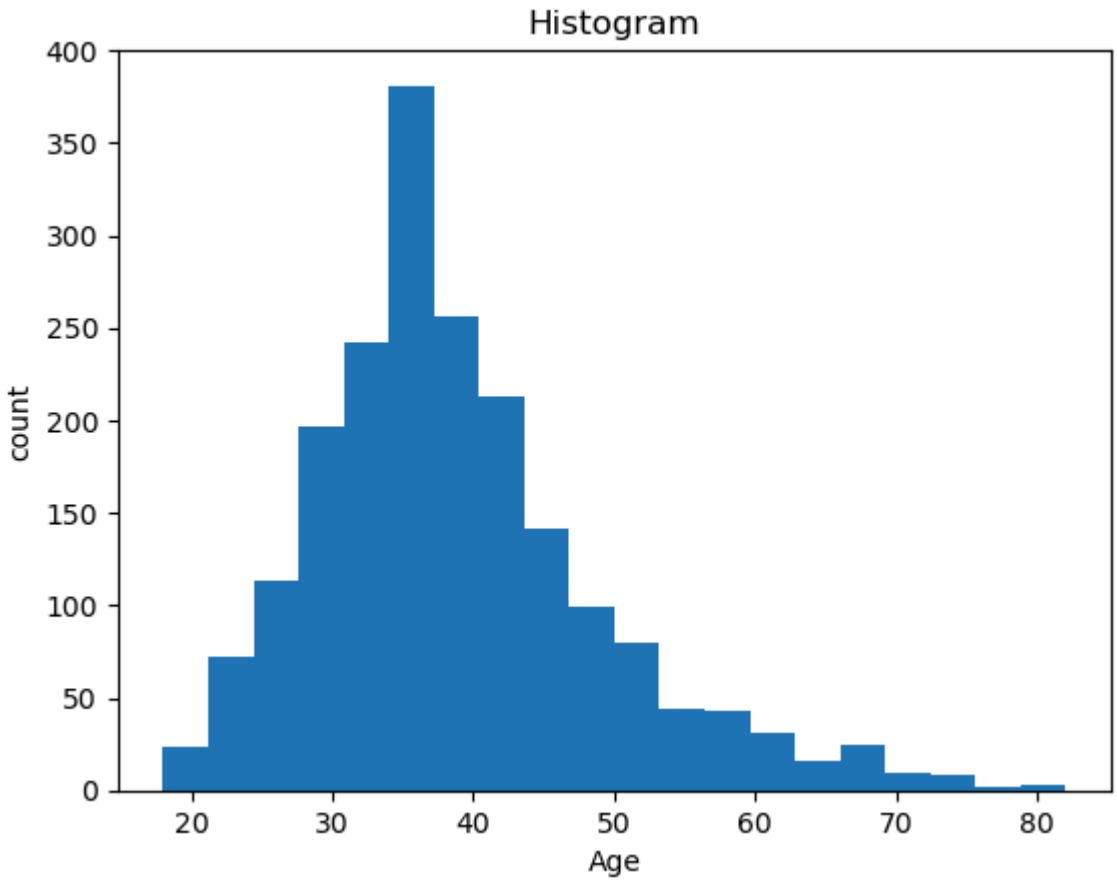
Histogram

In [75]:
```
num_cols
```

Out[75]:
```
Index(['year', 'customer_id', 'age', 'no_of_days_subscribed',
       'weekly_mins_watched', 'minimum_daily_mins', 'maximum_daily_mins',
       'weekly_max_night_mins', 'videos_watched', 'maximum_days_inactive',
       'customer_support_calls', 'churn'],
      dtype='object')
```

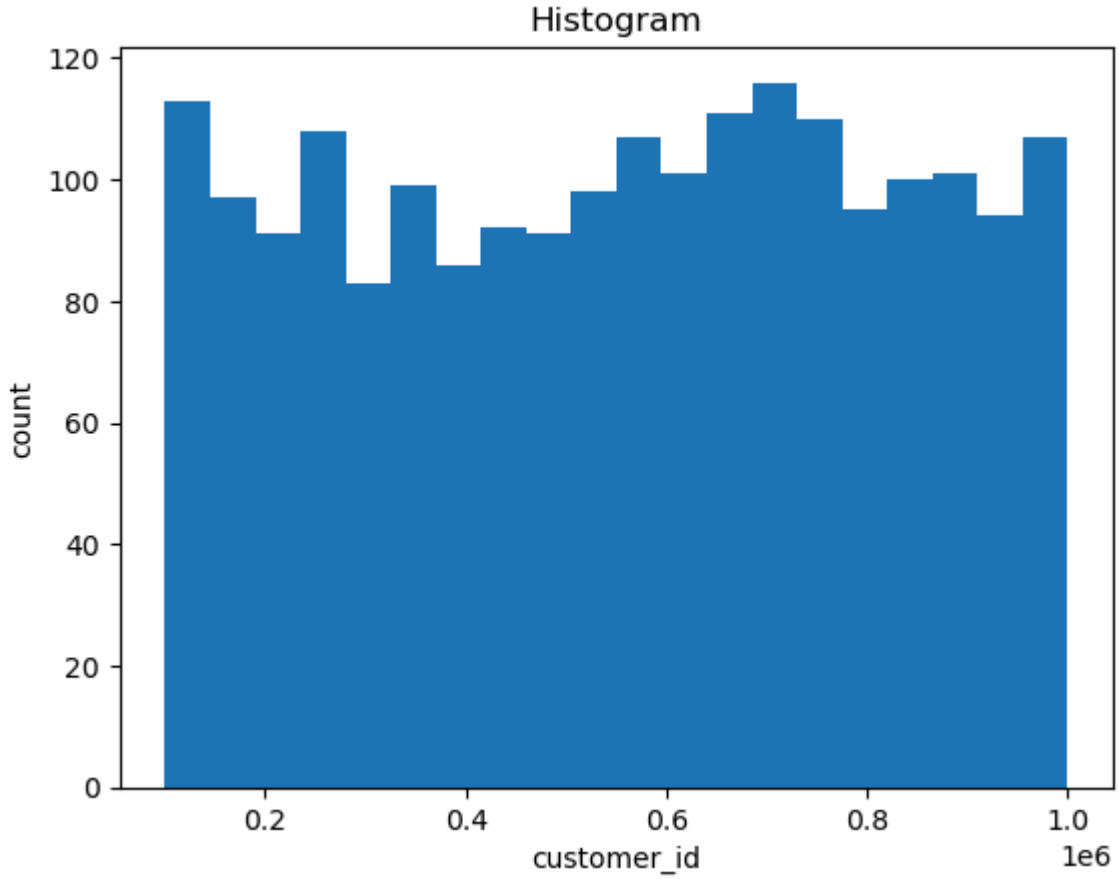
In [76]:
```
age_data=churn_data['age']
count,bins,x=plt.hist(age_data,bins=20)
plt.xlabel('Age')
plt.ylabel('count')
plt.title('Histogram')
print(len(count))
print(len(bins))
print(x)
plt.show()
```

```
20
21
<BarContainer object of 20 artists>
```

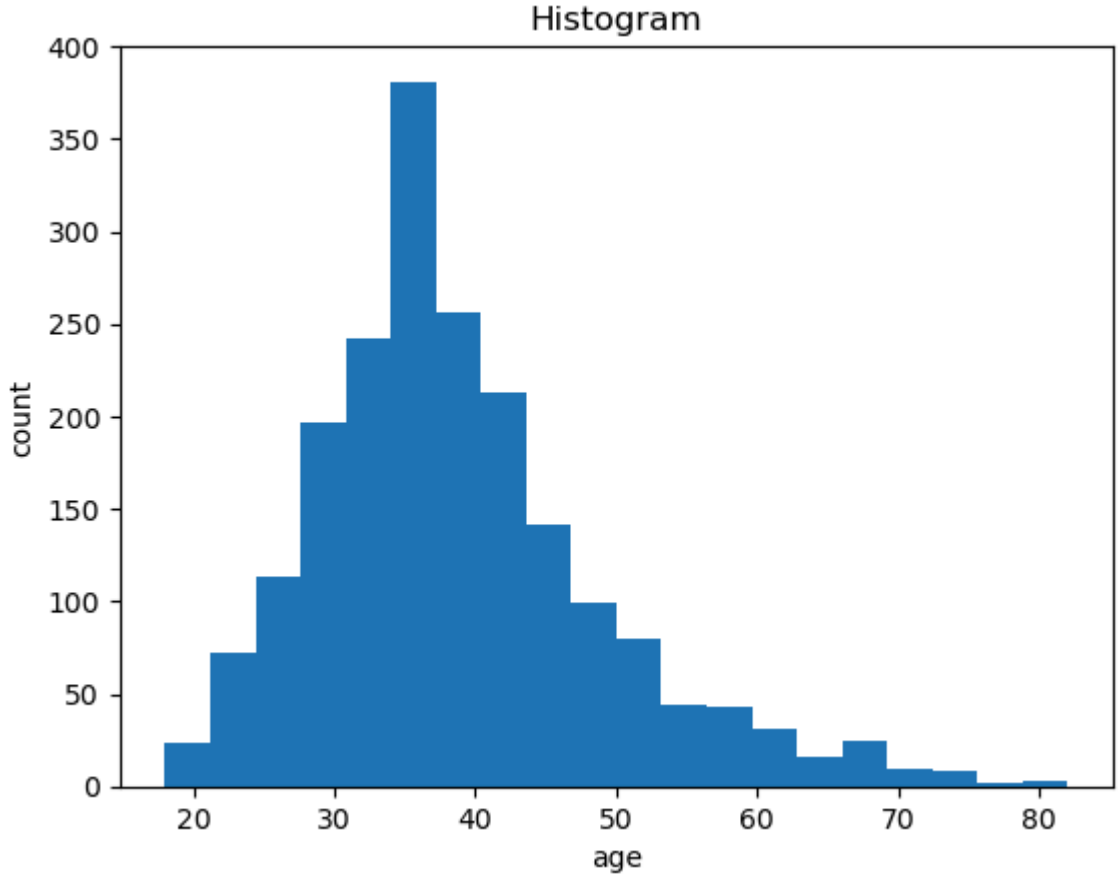


In [77]:
```
for i in num_cols[1:]:
    age_data=churn_data[i]
    count,bins,x=plt.hist(age_data,bins=20)
    plt.xlabel(i)
    plt.ylabel('count')
    plt.title('Histogram')
    print(len(count))
    print(len(bins))
    print(x)
    plt.show()
```

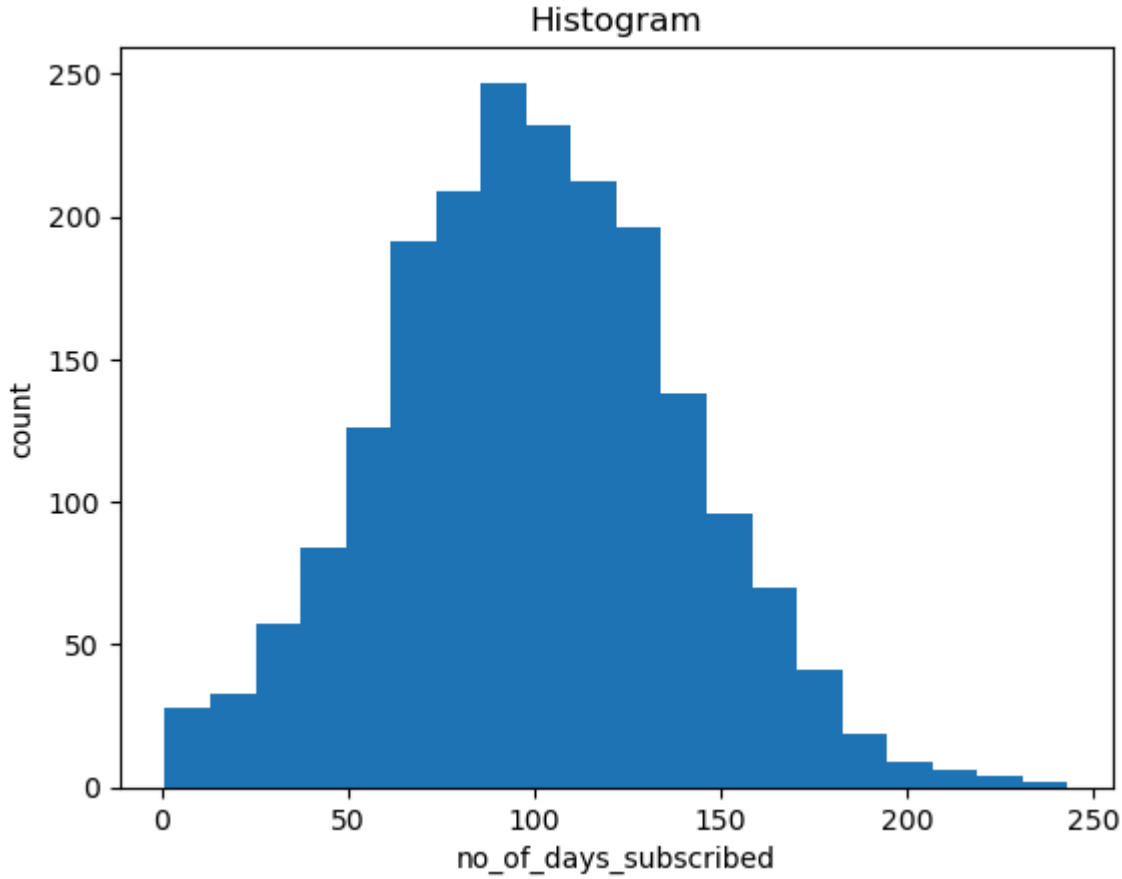
```
20
21
<BarContainer object of 20 artists>
```



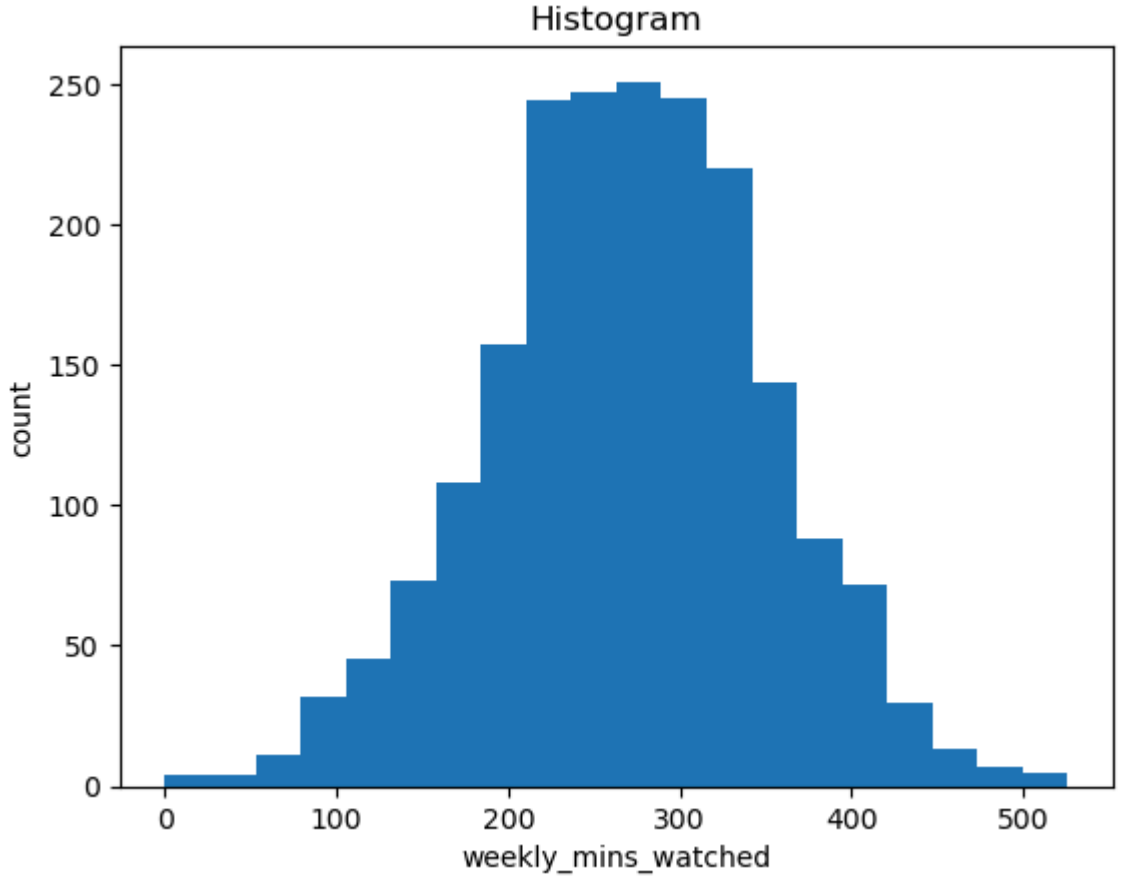

```
20
21
<BarContainer object of 20 artists>
```



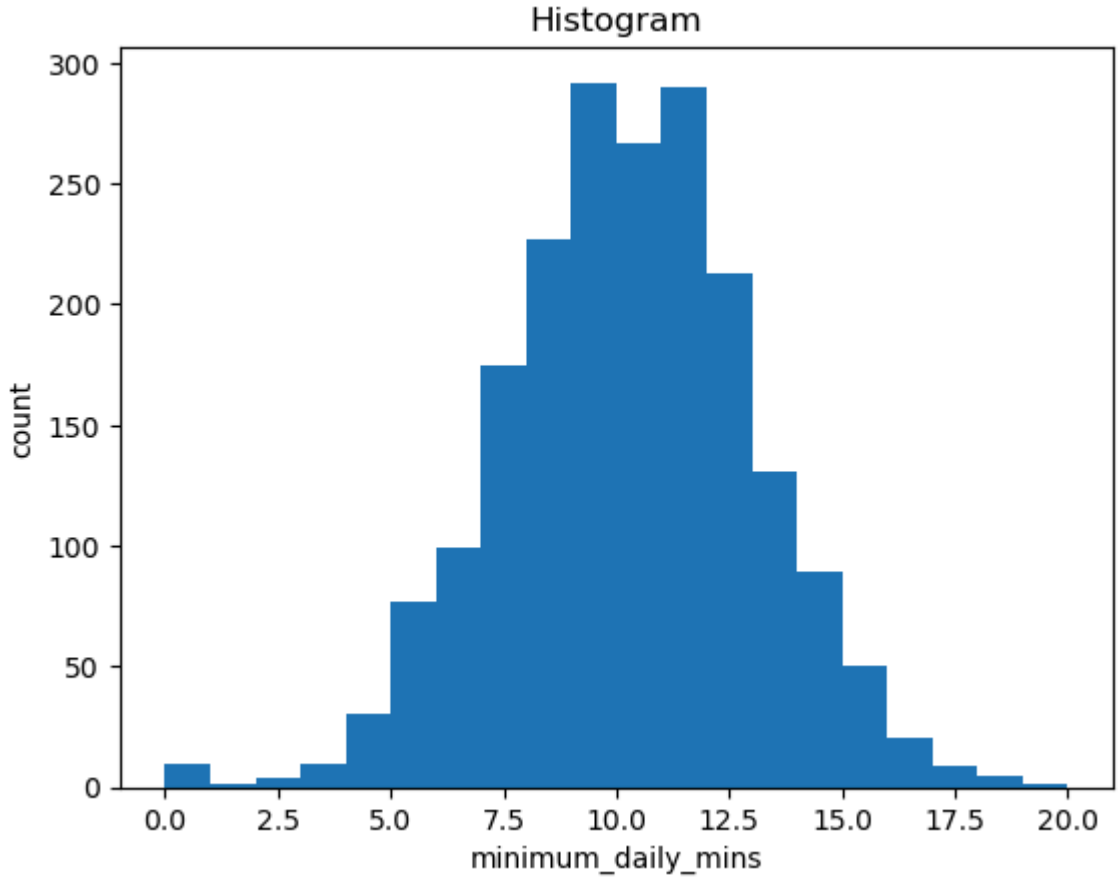
```
20
21
<BarContainer object of 20 artists>
```



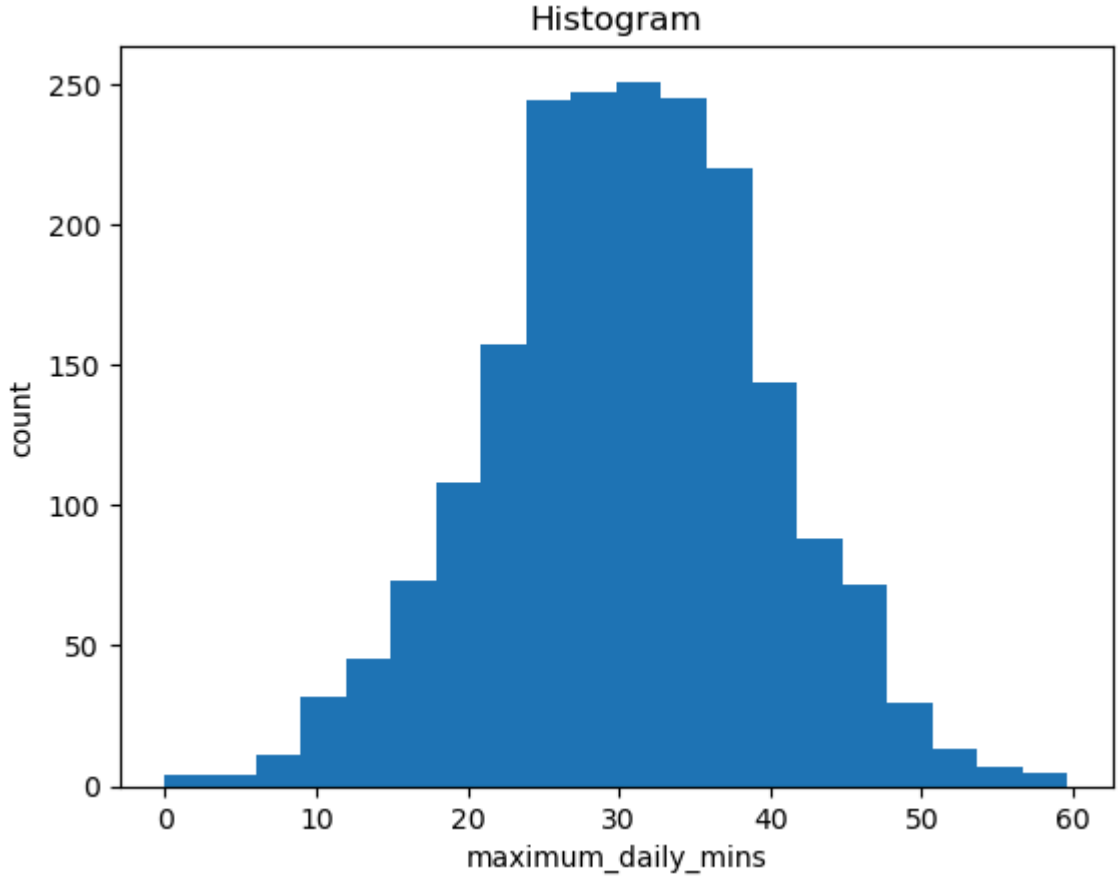

```
20
21
<BarContainer object of 20 artists>
```



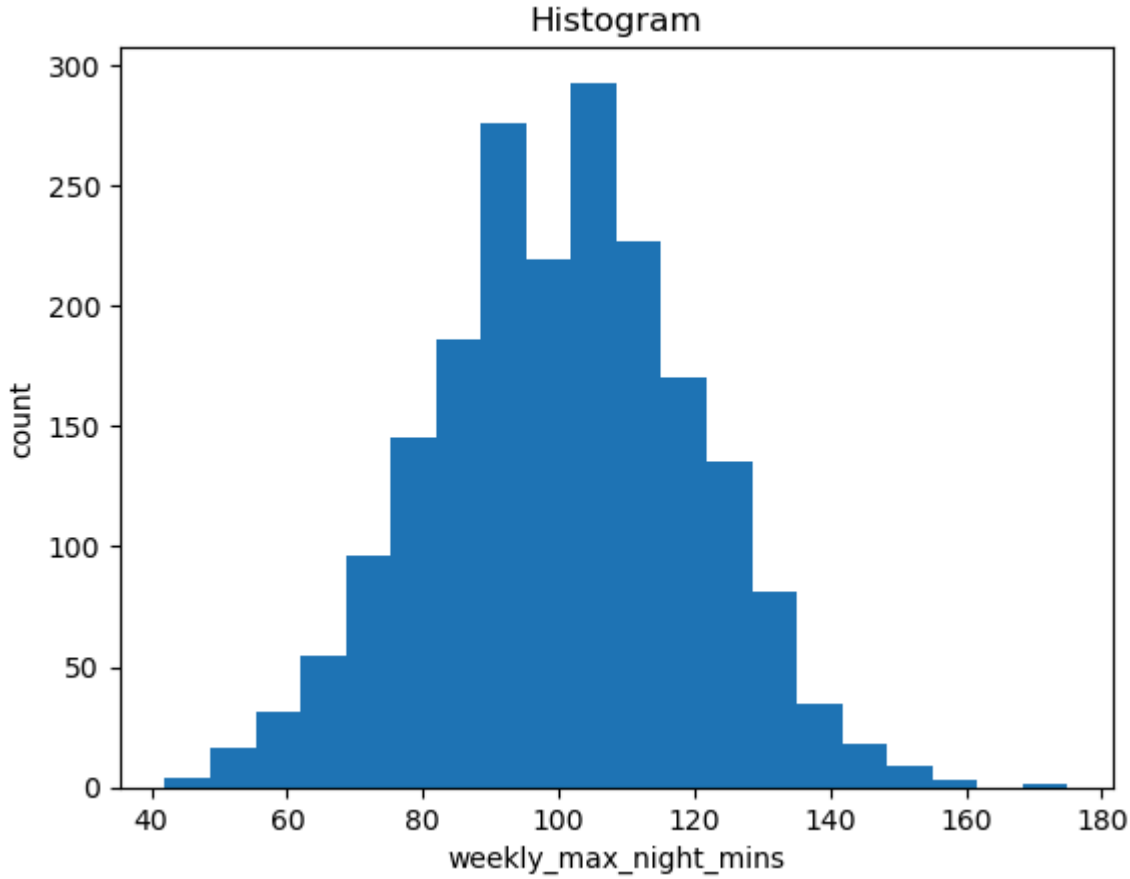
```
20
21
<BarContainer object of 20 artists>
```



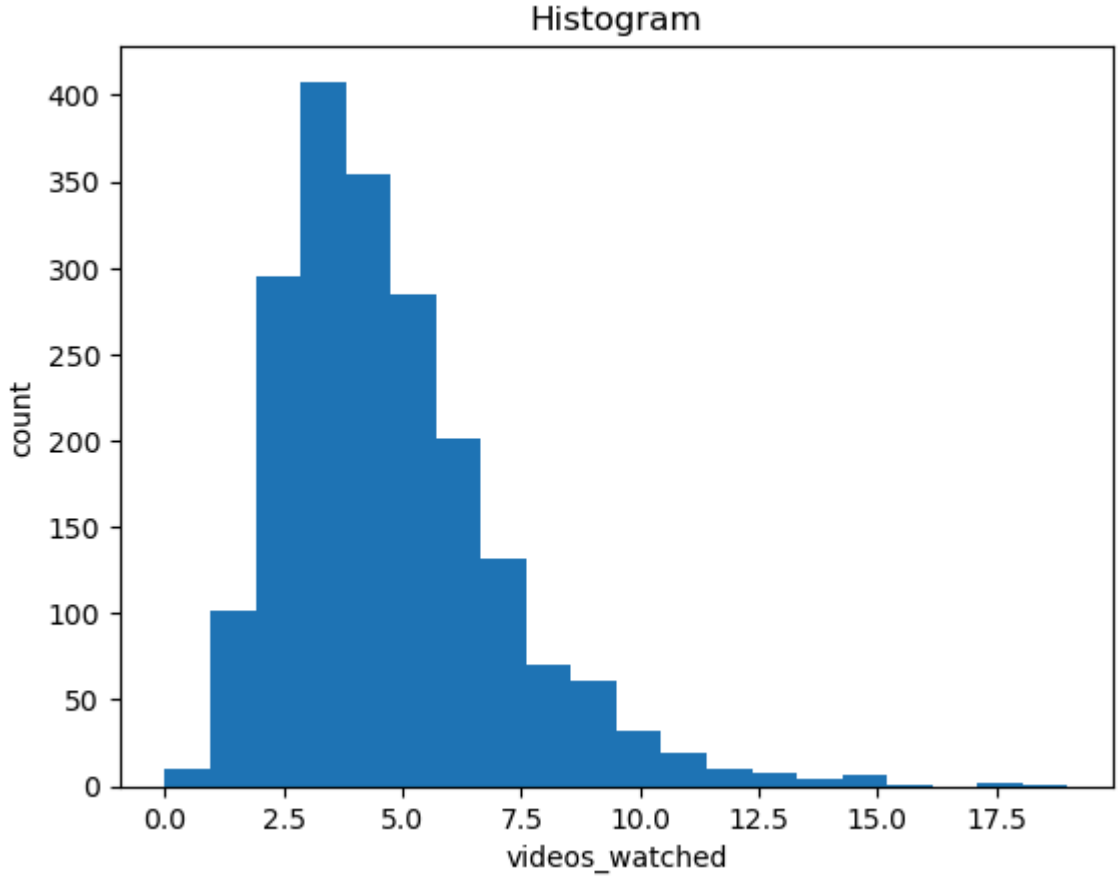

```
20
21
<BarContainer object of 20 artists>
```



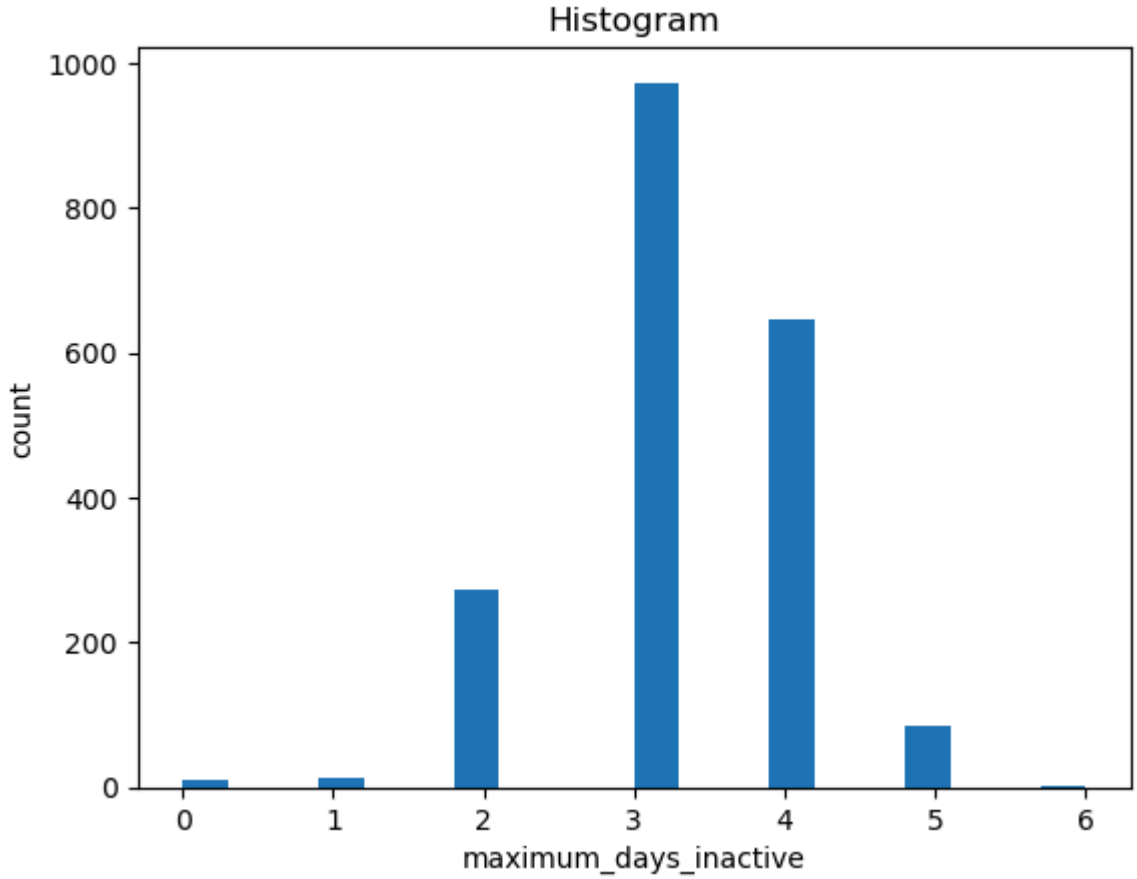
```
20
21
<BarContainer object of 20 artists>
```



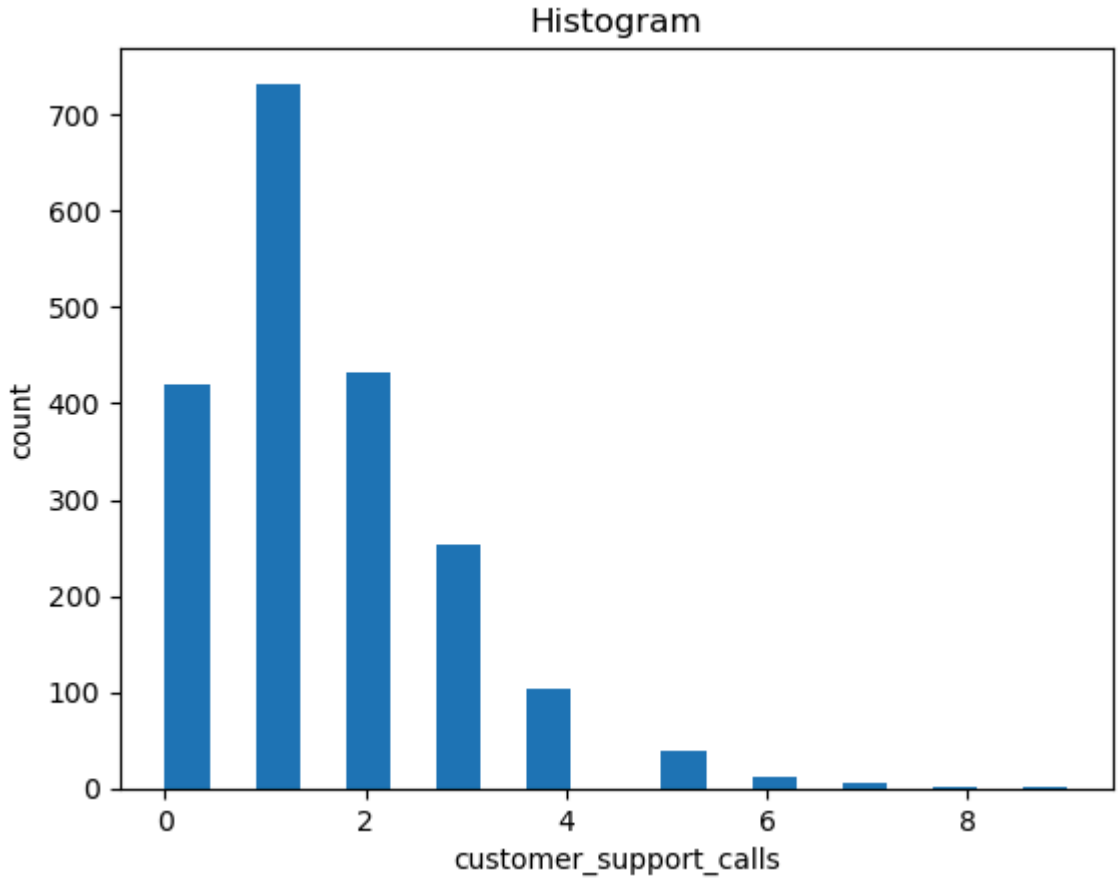

```
20
21
<BarContainer object of 20 artists>
```



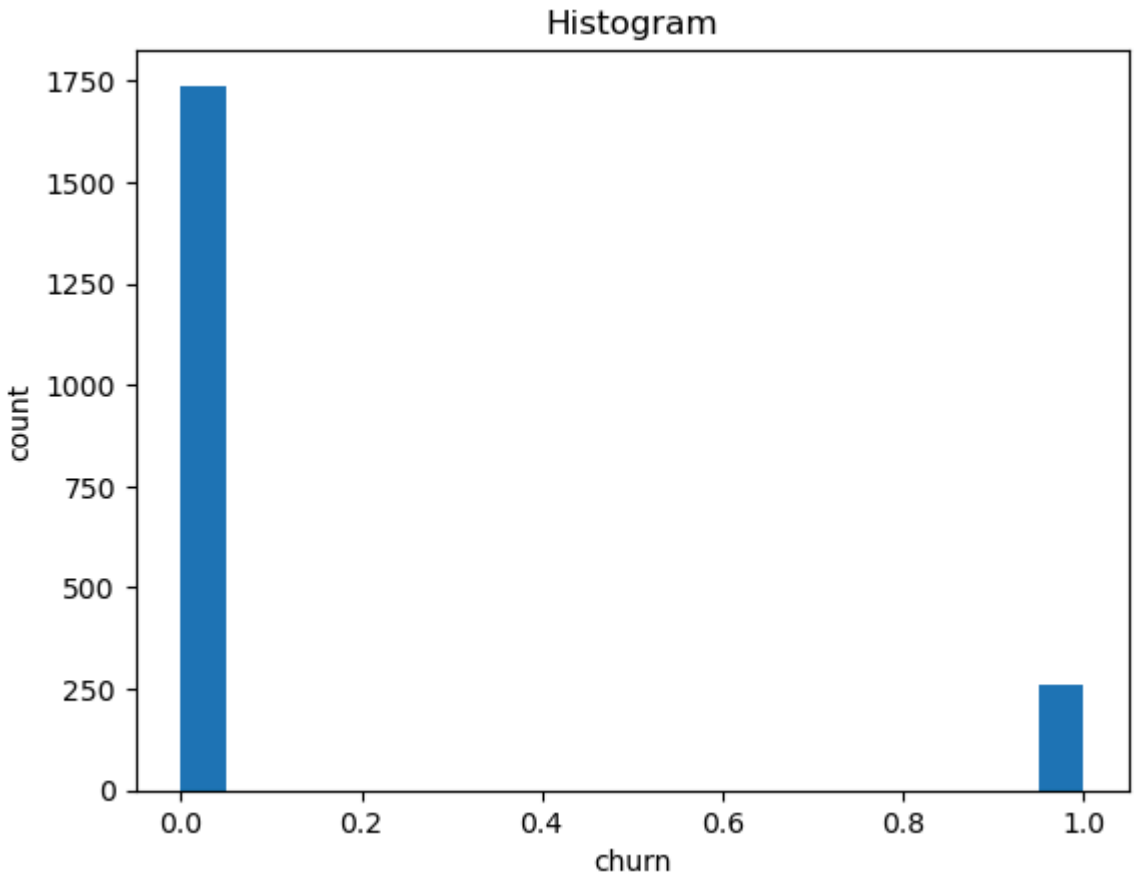
```
20
21
<BarContainer object of 20 artists>
```




```
20
21
<BarContainer object of 20 artists>
```

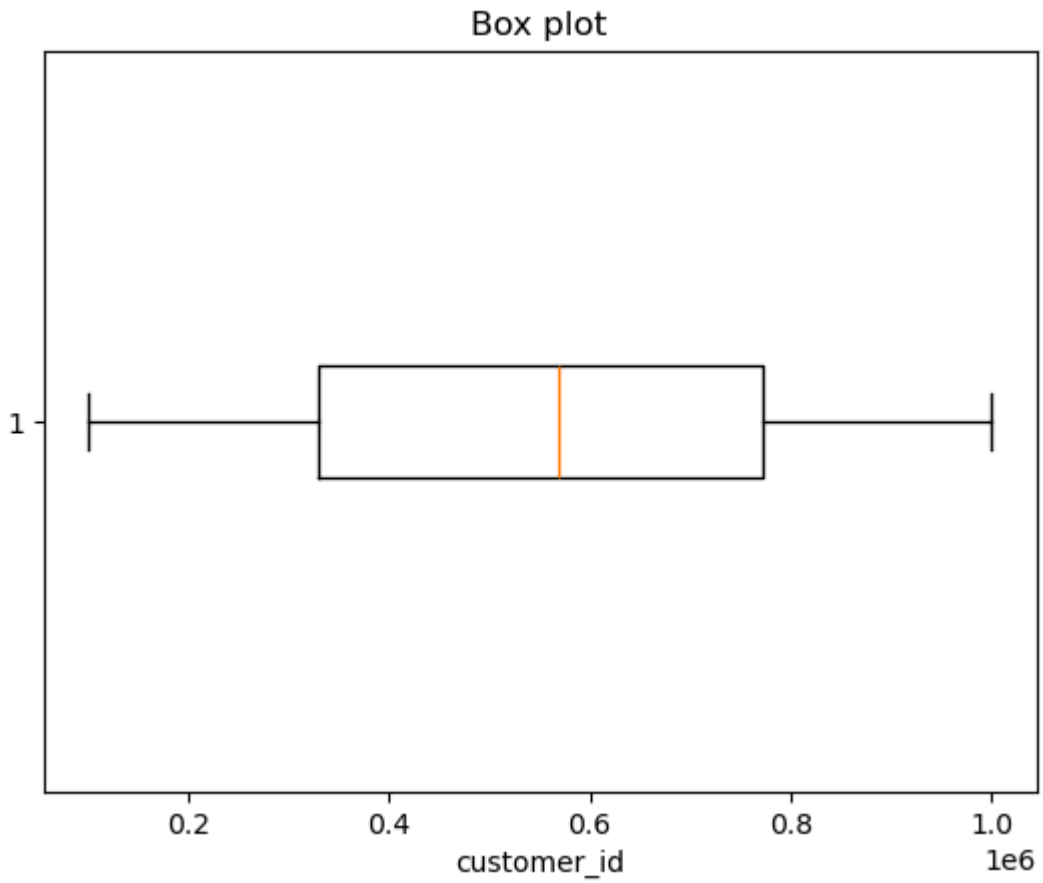


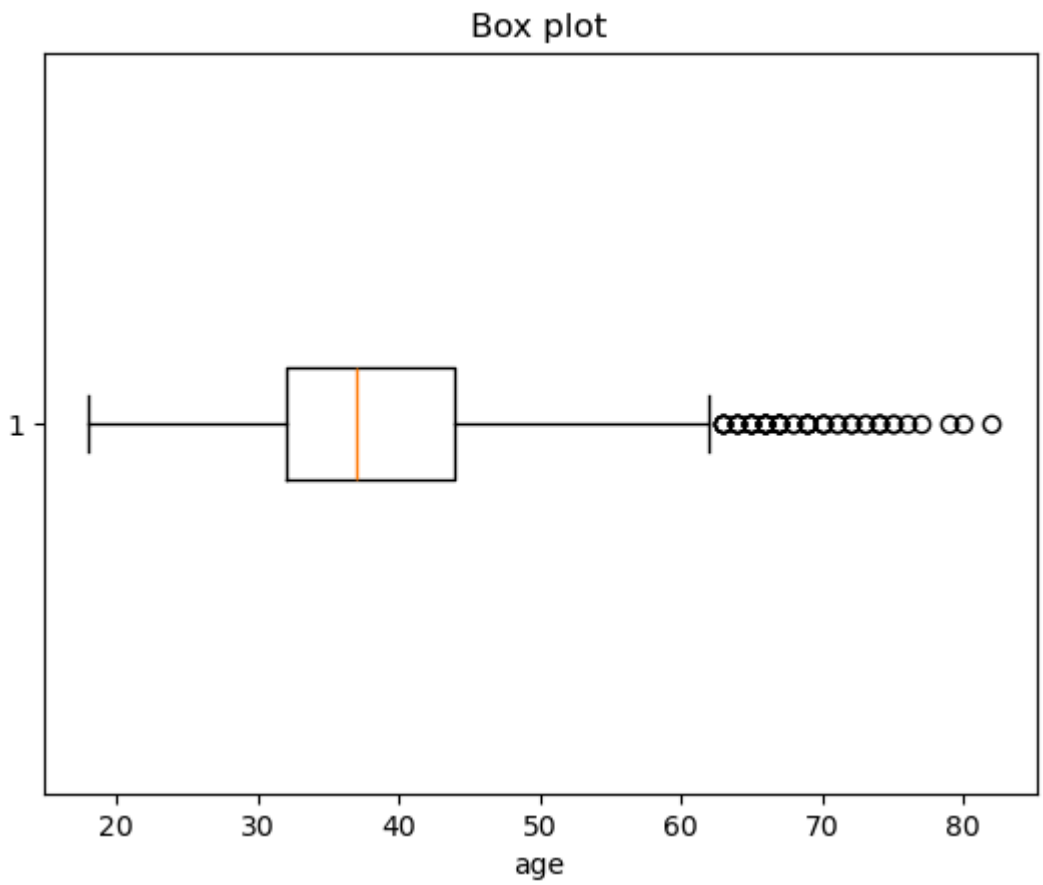
```
20
21
<BarContainer object of 20 artists>
```

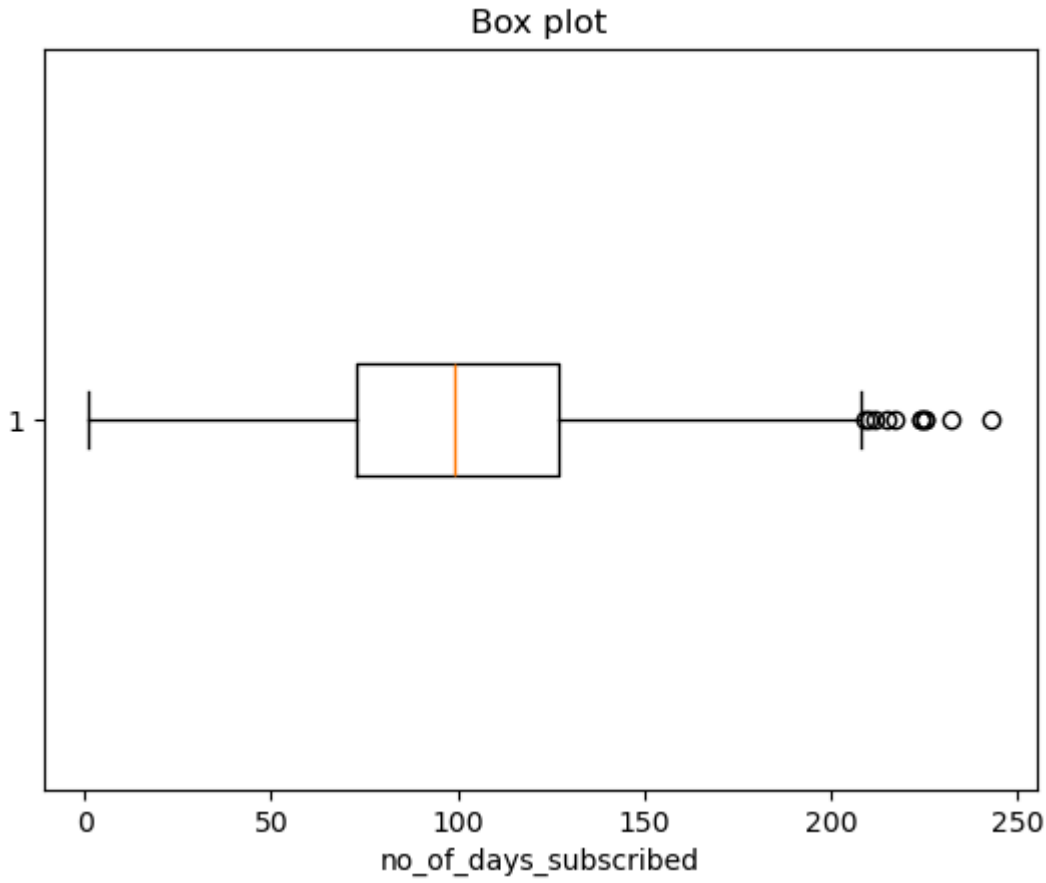


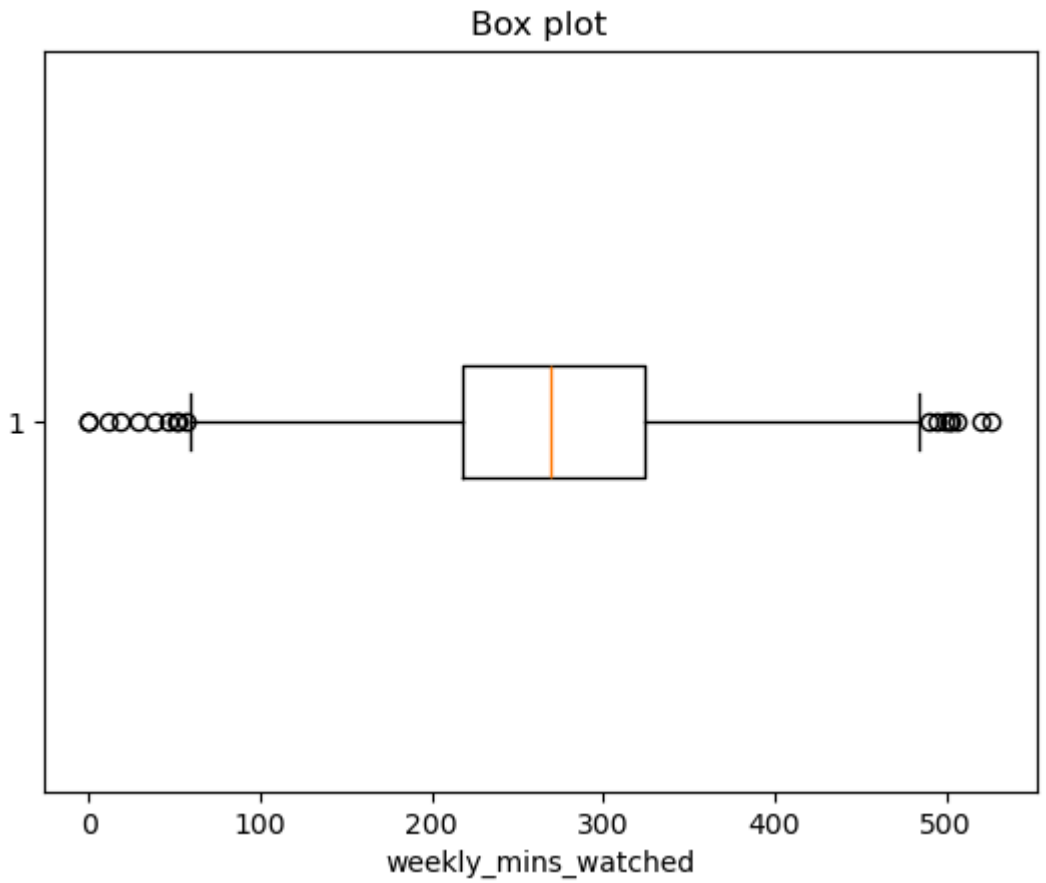
step-7:Box plot creation Outlier analysis

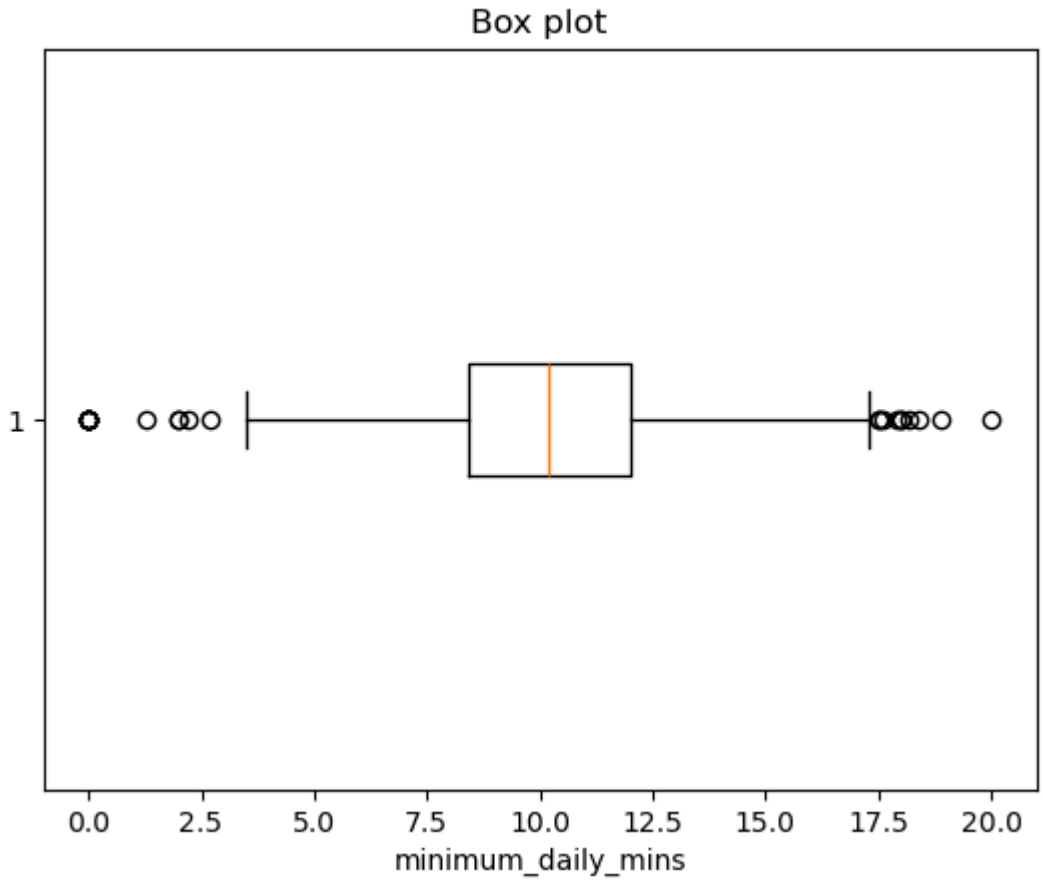
In [79]:
```
for i in num_cols[1:]:
    wage_data=churn_data[i]
    plt.boxplot(wage_data,vert=False)
    plt.title("Box plot")
    plt.xlabel(i)
    plt.show()
```

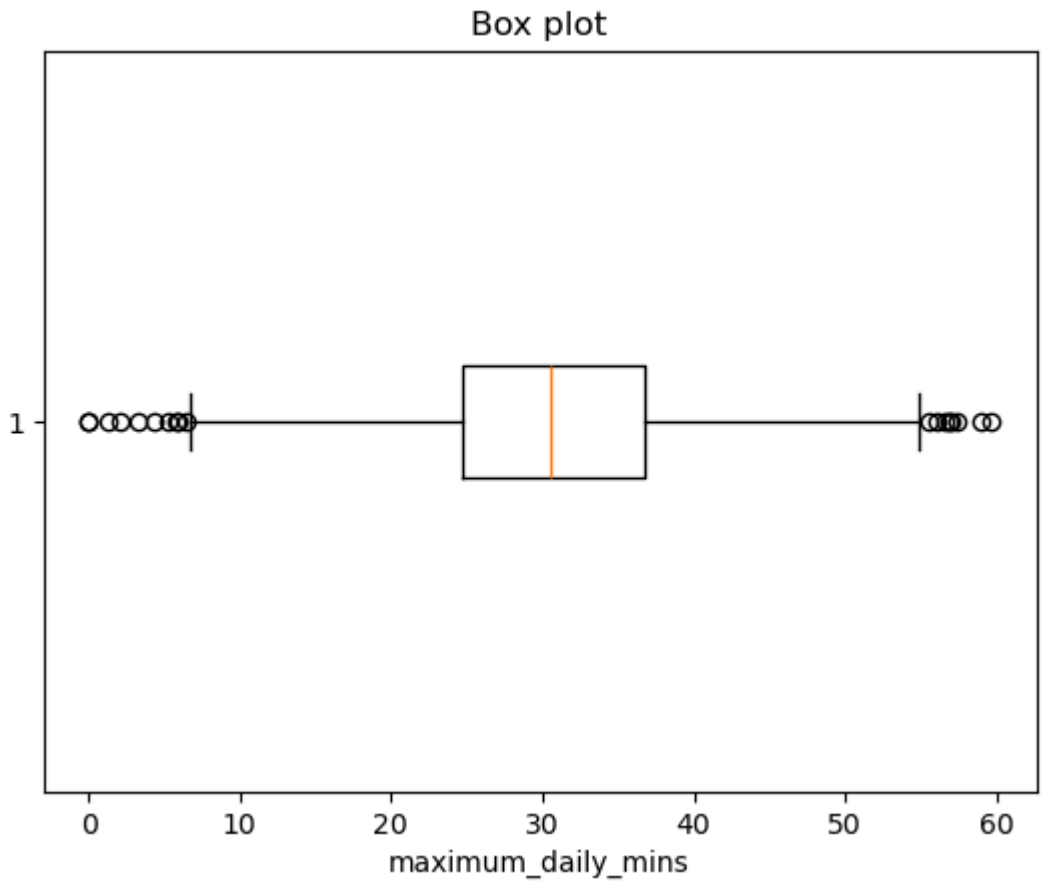

Box plot
1
0.2
0.4
0.6
0.8
1.0
customer_id
1e6

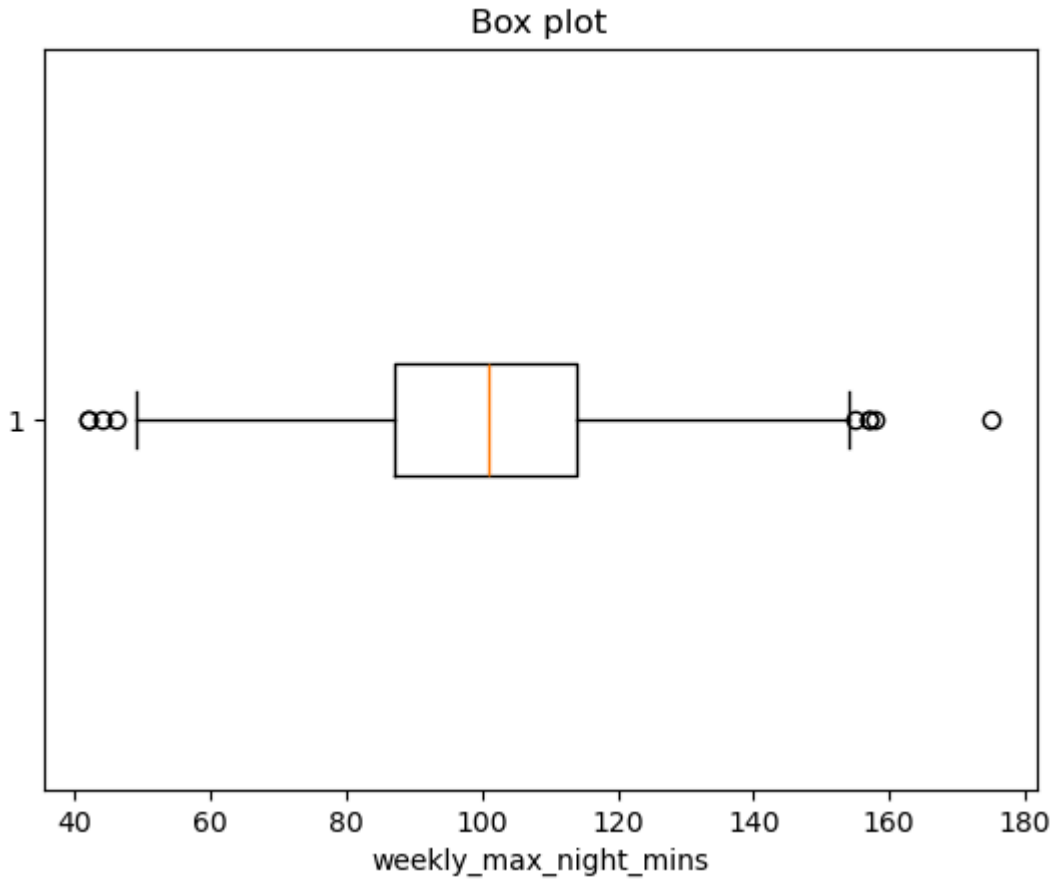

Box plot
1
20
30
40
50
60
70
80
age

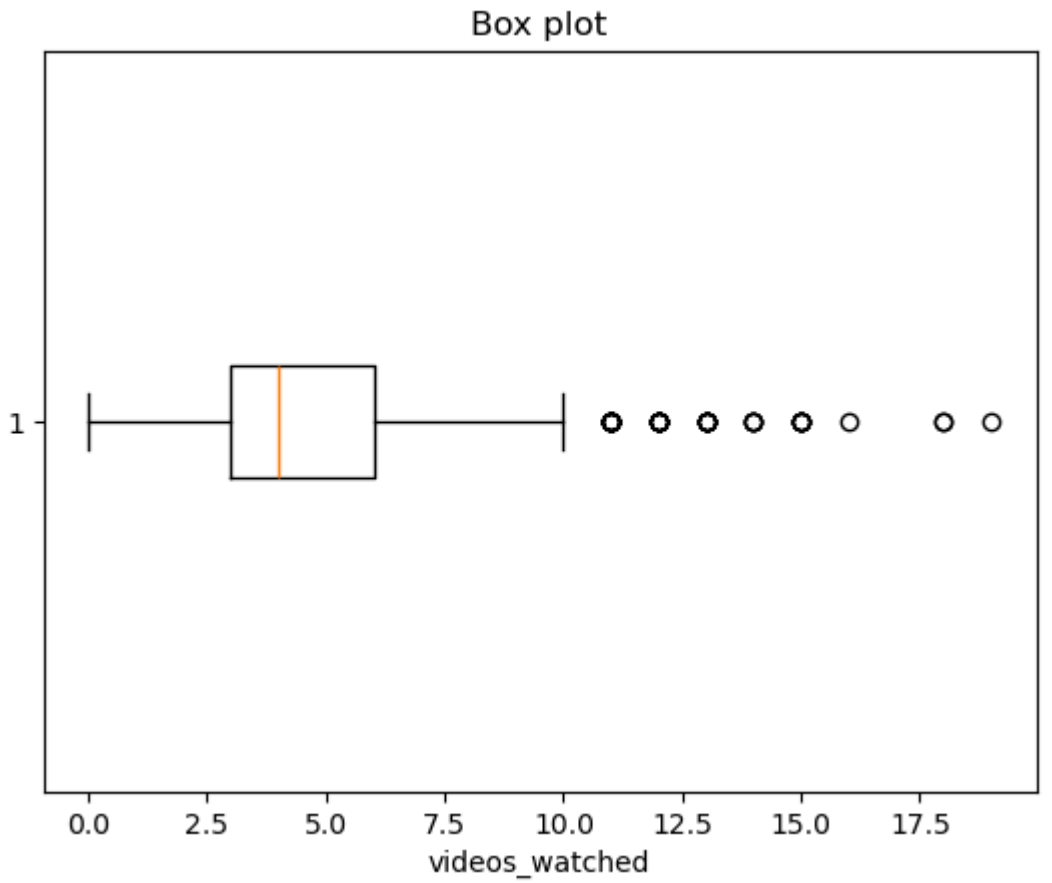

Box plot
1
0
50
100
150
200
250
no_of_days_subscribed

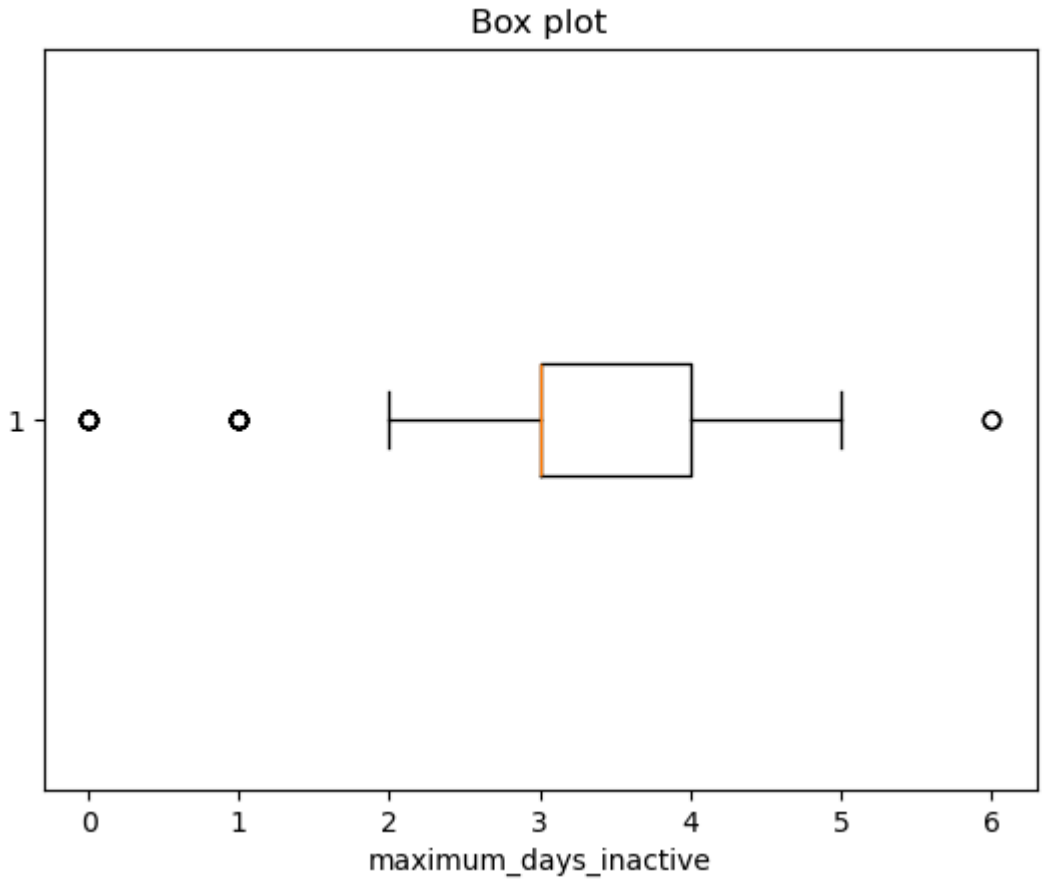

Box plot
1
0
100
200
300
400
500
weekly_mins_watched

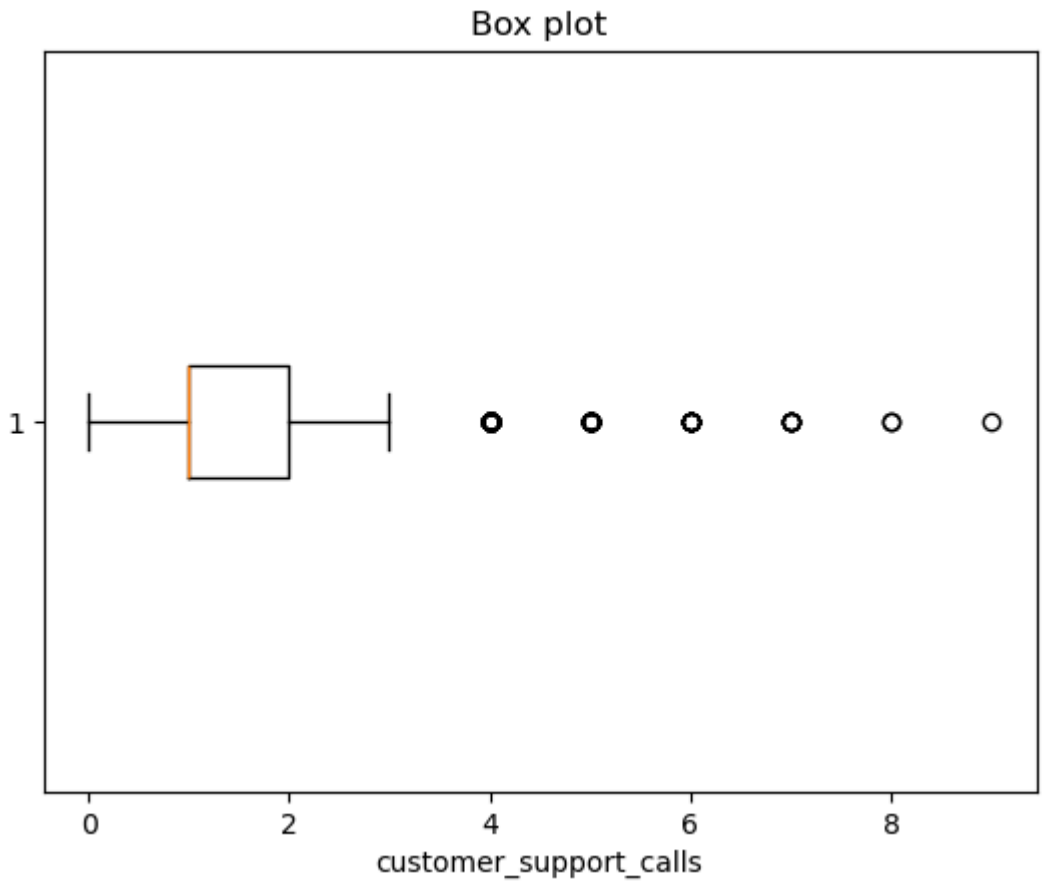

Box plot
1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
minimum_daily_mins

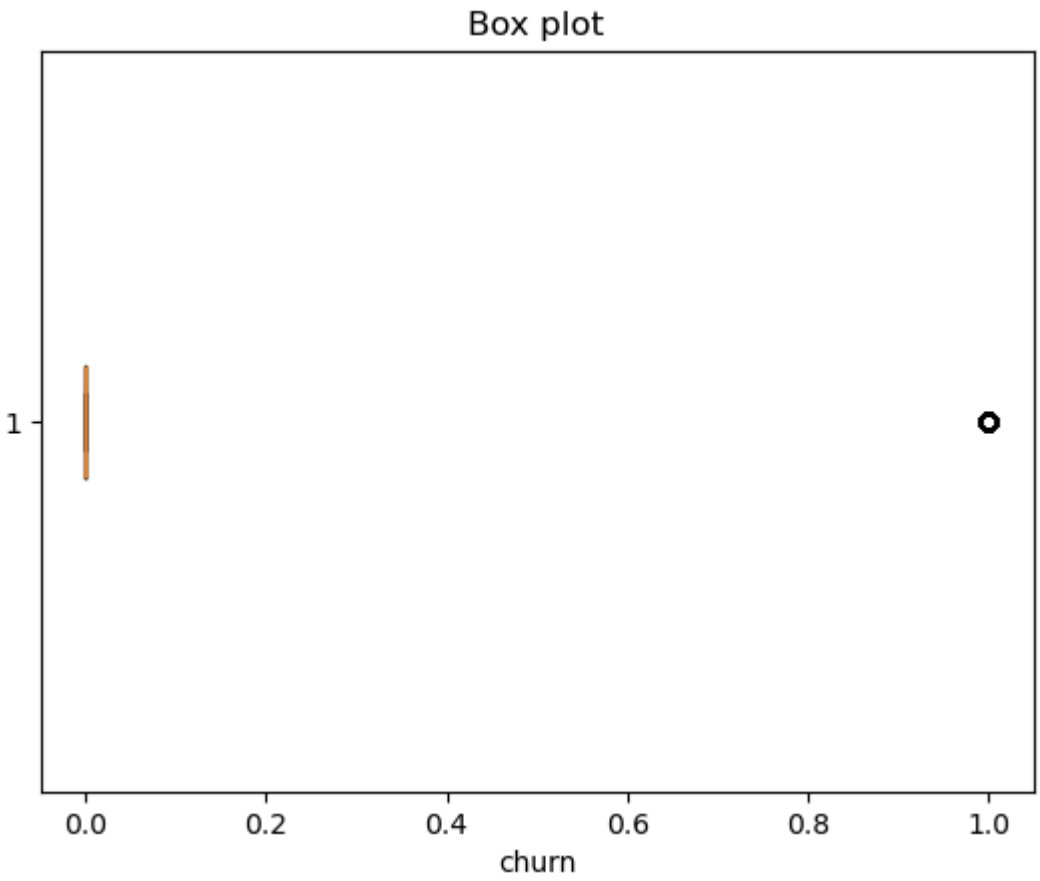

Box plot
1
0
10
20
30
40
50
60
maximum_daily_mins


Box plot
1
40
60
80
100
120
140
160
180
weekly_max_night_mins


Box plot
1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
videos_watched


Box plot
1
0
1
2
3
4
5
6
maximum_days_inactive


Box plot
1
0
2
4
6
8
customer_support_calls



Transformation methods

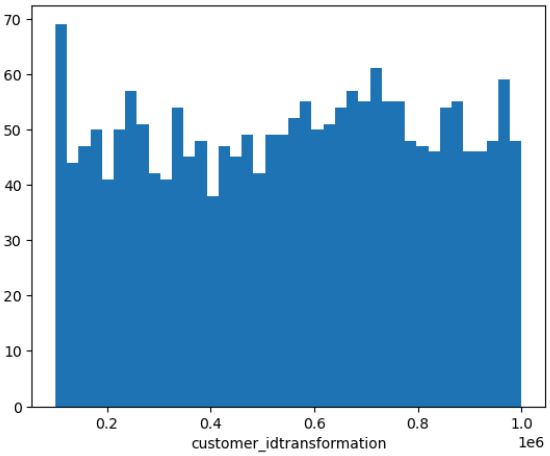
using power transformation

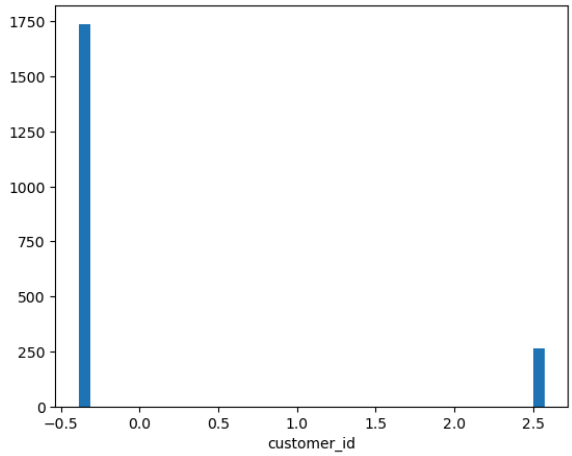
In [83]:
```
from sklearn.preprocessing import PowerTransformer
for i in num_cols[1:]:
    pt=PowerTransformer()
    transform_wage=pt.fit_transform(churn_data[[i]])
    transform_wage
```

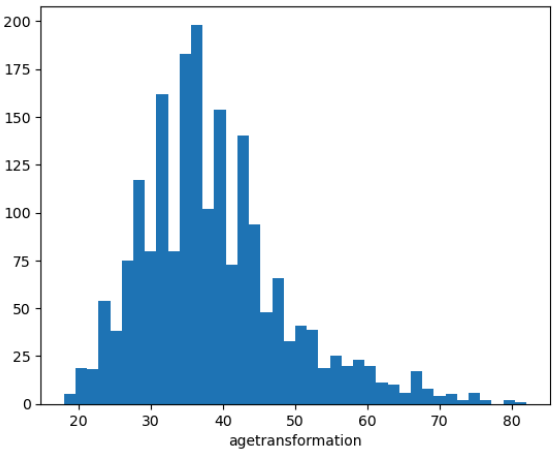
In [84]:
```
transform_wage
```

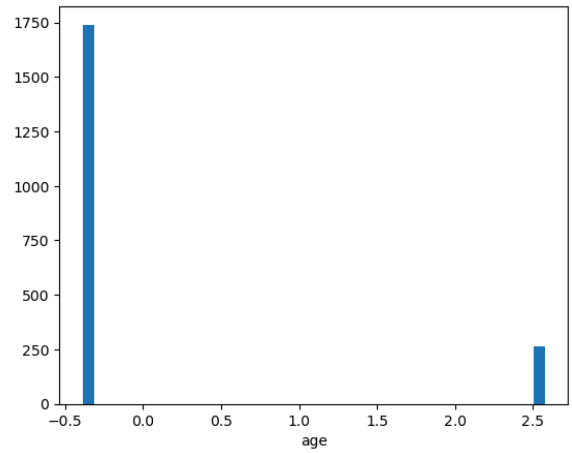
Out[84]:
```
array([[-0.38826278],
       [-0.38826278],
       [ 2.57557523],
       ...,
       [-0.38826278],
       [-0.38826278],
       [ 2.57557523]])
```

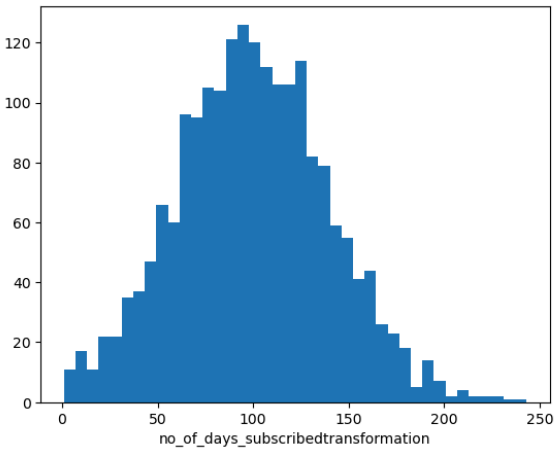
In [85]:
```
for i in num_cols[1:]:
    plt.figure(figsize=(14,5))
    plt.subplot(1,2,1).hist(churn_data[i],bins=40)
    plt.xlabel(f"{i}transformation")
    plt.subplot(1,2,2).hist(transform_wage,bins=40)
    plt.xlabel(i)
    plt.show()
```

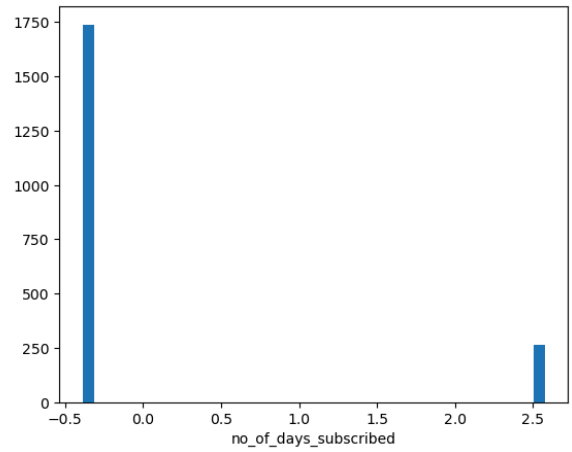

customer_idtransformation
1e6

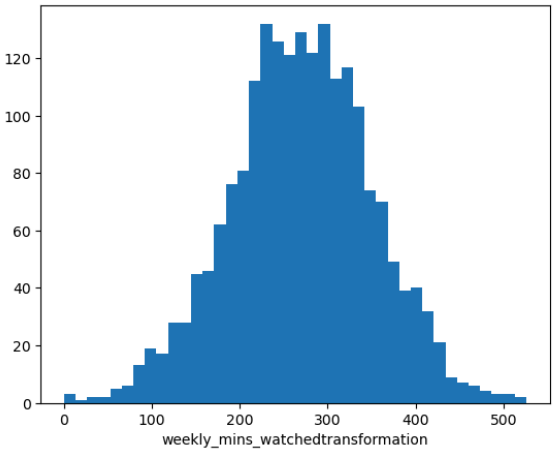

customer_id

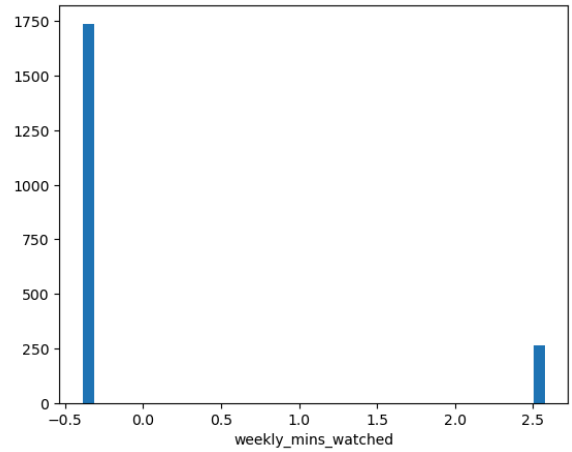

agetransformation

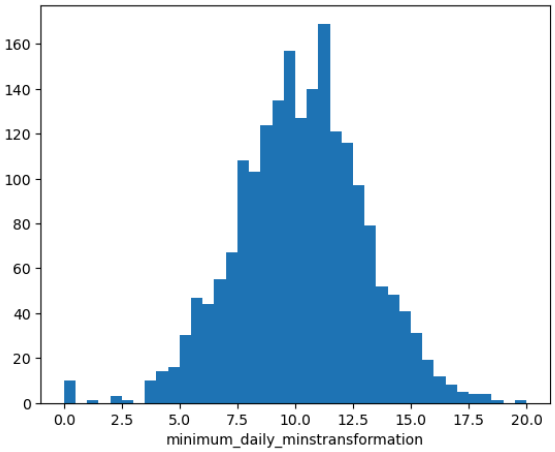

age

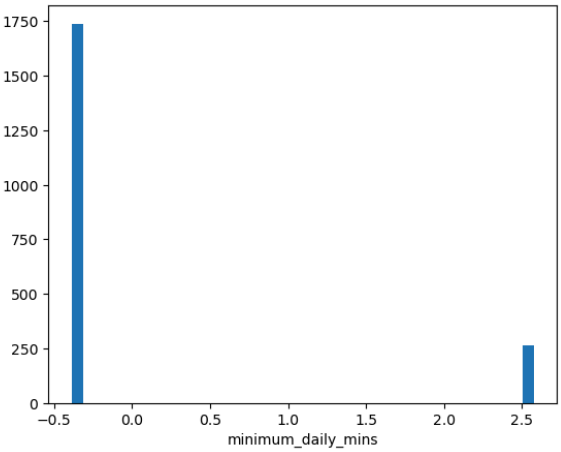

no_of_days_subscribedtransformation

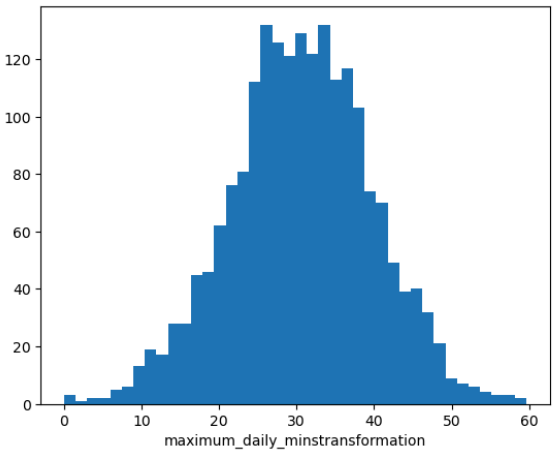

no_of_days_subscribed

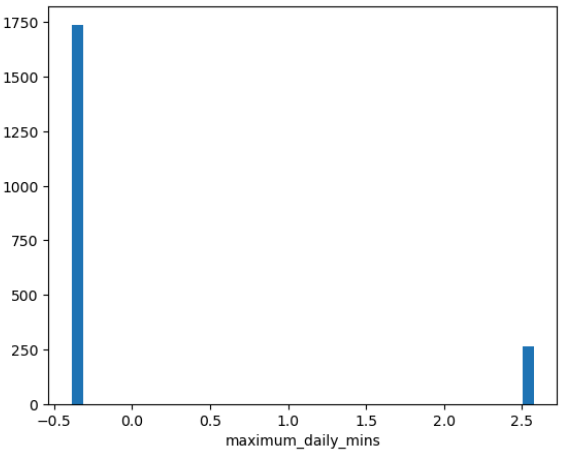

weekly_mins_watchedtransformation

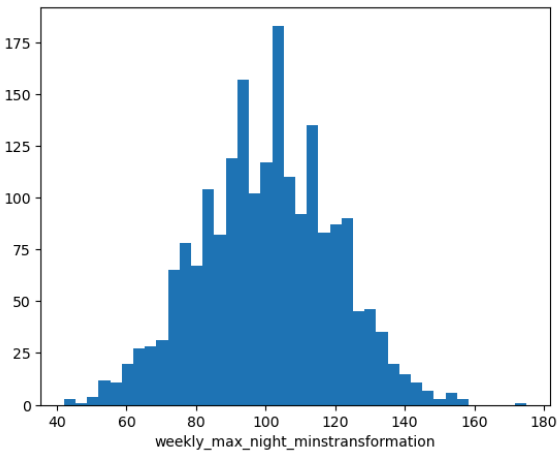

weekly_mins_watched

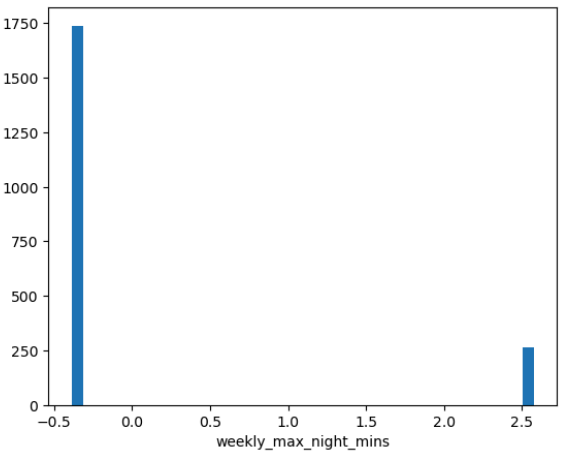

minimum_daily_minstransformation

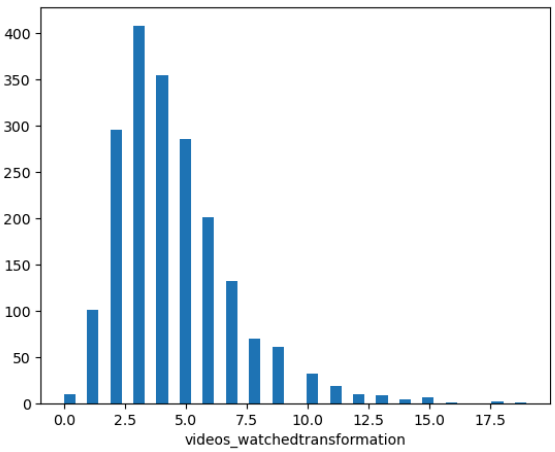

minimum_daily_mins

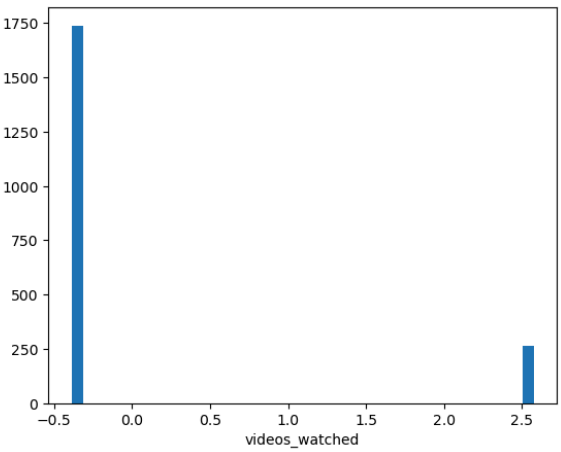

maximum_daily_minstransformation

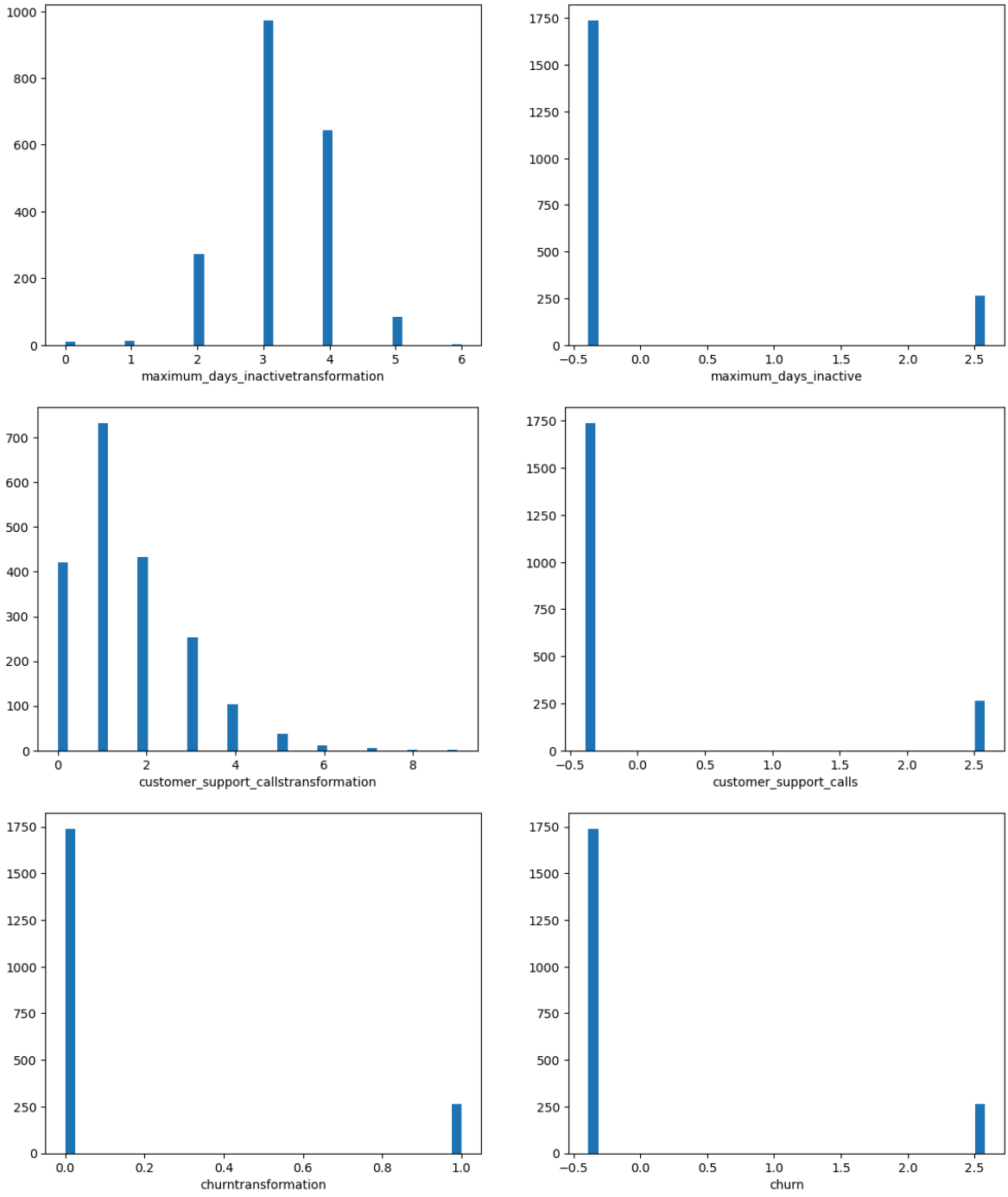

maximum_daily_mins


weekly_max_night_minstransformation


weekly_max_night_mins


videos_watchedtransformation


videos_watched



Step-9: Encoding methods map np.where LabelEncoder onehot encoder

Mapping

step-1 :read the column step-2: get unique labels step-3: make a dictionary step-4: apply the mapping

In [86]:
```
cat_cols
```

Out[86]:
```
Index(['phone_no', 'gender', 'multi_screen', 'mail_subscribed'], dtype='objec
t')
```

In [87]:
```
unique_labels=churn_data['gender'].unique()
list1=[i for i in range(len(unique_labels))]
unique_labels,list1
dict1=dict(zip(unique_labels,list1))
dict1
```

```
churn_data['gender1']=churn_data['gender'].map(dict1)
churn_data
```

Out[87]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	409-8743	Female	36	62	no
1	2015	100643	340-5930	Female	39	149	no
2	2015	100756	372-3750	Female	65	126	no
3	2015	101595	331-4902	Female	24	131	no
4	2015	101653	351-8398	Female	40	191	no
...	...	...	...	...	...	...	...
1995	2015	997132	385-7387	Female	54	75	no
1996	2015	998086	383-9255	Male	45	127	no
1997	2015	998474	353-2080	Male	53	94	no
1998	2015	998934	359-7788	Male	40	94	no
1999	2015	999961	414-1496	Male	37	73	no

2000 rows × 17 columns

In [88]:

```
for i in cat_cols:

    unique_labels=churn_data[i].unique()
    list1=[i for i in range(len(unique_labels))]
    unique_labels,list1
    dict1=dict(zip(unique_labels,list1))
    dict1
    churn_data[i]=churn_data[i].map(dict1)
churn_data
```

Out[88]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	0	0	36	62	0
1	2015	100643	1	0	39	149	0
2	2015	100756	2	0	65	126	0
3	2015	101595	3	0	24	131	0
4	2015	101653	4	0	40	191	0
...	...	...	...	...	...	...	...
1995	2015	997132	1995	0	54	75	0
1996	2015	998086	1996	1	45	127	0
1997	2015	998474	1997	1	53	94	0
1998	2015	998934	1998	1	40	94	0
1999	2015	999961	1999	1	37	73	0

2000 rows × 17 columns

LabelEncoder

In [91]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in cat_cols:
   churn_data[i]=le.fit_transform(churn_data[i])
churn_data
```

Out[91]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	0	0	36	62	0
1	2015	100643	1	0	39	149	0
2	2015	100756	2	0	65	126	0
3	2015	101595	3	0	24	131	0
4	2015	101653	4	0	40	191	0
...	...	...	...	...	...	...	...
1995	2015	997132	1995	0	54	75	0
1996	2015	998086	1996	1	45	127	0
1997	2015	998474	1997	1	53	94	0
1998	2015	998934	1998	1	40	94	0
1999	2015	999961	1999	1	37	73	0

2000 rows × 17 columns

np.where

In [96]:
```
churn_data=pd.read_csv(path)
mode_max_days=churn_data['maximum_days_inactive'].mode()
churn_data['maximum_days_inactive'].fillna(mode_max_days[0],inplace=True)
mode_gender=churn_data['gender'].mode()
churn_data['gender'].fillna(mode_gender[0],inplace=True)
mode_churn=churn_data['churn'].mode()
churn_data['churn'].fillna(mode_churn[0],inplace=True)
con=churn_data['age']=='Male'  #0 and other 1
churn_data['Male']=np.where(con, 0 ,1)
churn_data
```

Out[96]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	409-8743	Female	36	62	no
1	2015	100643	340-5930	Female	39	149	no
2	2015	100756	372-3750	Female	65	126	no
3	2015	101595	331-4902	Female	24	131	no
4	2015	101653	351-8398	Female	40	191	no
...	...	...	...	...	...	...	...
1995	2015	997132	385-7387	Female	54	75	no
1996	2015	998086	383-9255	Male	45	127	no
1997	2015	998474	353-2080	Male	53	94	no
1998	2015	998934	359-7788	Male	40	94	no
1999	2015	999961	414-1496	Male	37	73	no

2000 rows × 17 columns

In [99]:
```
churn_data=pd.read_csv(path)
churn_data
```

Out[99]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen
0	2015	100198	409-8743	Female	36	62	no
1	2015	100643	340-5930	Female	39	149	no
2	2015	100756	372-3750	Female	65	126	no
3	2015	101595	331-4902	Female	24	131	no
4	2015	101653	351-8398	Female	40	191	no
...	...	...	...	...	...	...	...
1995	2015	997132	385-7387	Female	54	75	no
1996	2015	998086	383-9255	Male	45	127	no
1997	2015	998474	353-2080	NaN	53	94	no
1998	2015	998934	359-7788	Male	40	94	no
1999	2015	999961	414-1496	Male	37	73	no

2000 rows × 16 columns

In [100…

```
churn_data=pd.read_csv(path)
mode_max_days=churn_data['maximum_days_inactive'].mode()
churn_data['maximum_days_inactive'].fillna(mode_max_days[0],inplace=True)
mode_gender=churn_data['gender'].mode()
churn_data['gender'].fillna(mode_gender[0],inplace=True)
mode_churn=churn_data['churn'].mode()
churn_data['churn'].fillna(mode_churn[0],inplace=True)
```

one-hot encoder

In [101…

```
df=pd.get_dummies(churn_data['gender'],dtype=int)
df['gender']=churn_data['gender'].values
df
```

Out[101…

	Female	Male	gender
0	1	0	Female
1	1	0	Female
2	1	0	Female
3	1	0	Female
4	1	0	Female
...	...	...	...
1995	1	0	Female
1996	0	1	Male
1997	0	1	Male
1998	0	1	Male
1999	0	1	Male

2000 rows × 3 columns

In [103…

```
churn_data=pd.read_csv(path)
mode_max_days=churn_data['maximum_days_inactive'].mode()
churn_data['maximum_days_inactive'].fillna(mode_max_days[0],inplace=True)
mode_gender=churn_data['gender'].mode()
churn_data['gender'].fillna(mode_gender[0],inplace=True)
mode_churn=churn_data['churn'].mode()
churn_data['churn'].fillna(mode_churn[0],inplace=True)
```

In [104…

```
pd.get_dummies(churn_data,dtype='int')
```

Out[104...

	year	customer_id	age	no_of_days_subscribed	weekly_mins_watched	minimum_
0	2015	100198	36	62	148.35	
1	2015	100643	39	149	294.45	
2	2015	100756	65	126	87.30	
3	2015	101595	24	131	321.30	
4	2015	101653	40	191	243.00	
...	...	...	...	...	...	
1995	2015	997132	54	75	182.25	
1996	2015	998086	45	127	273.45	
1997	2015	998474	53	94	128.85	
1998	2015	998934	40	94	178.05	
1999	2015	999961	37	73	326.70	

2000 rows × 2018 columns

In [105...

```
count=0
for i in cat_cols[1:]:
    count= count+churn_data[i].nunique()

print(count)
print(count+len(num_cols))
```

```
6
18
```

Scaling Methods

Scaling methods Z minmax scalar

idmax-idmin

In [107...

```
num_cols
```

Out[107...

```
Index(['year', 'customer_id', 'age', 'no_of_days_subscribed',
       'weekly_mins_watched', 'minimum_daily_mins', 'maximum_daily_mins',
       'weekly_max_night_mins', 'videos_watched', 'maximum_days_inactive',
       'customer_support_calls', 'churn'],
      dtype='object')
```

In [109...

```
min_id=churn_data[['age']].idxmin().values[0]
max_id=churn_data[['age']].idxmax().values[0]
print(f"The maximun id age is : {max_id}" )
print(f"The minimun id age is : {min_id}" )
```

```
The maximun id age is : 682
The minimun id age is : 921
```

In [111…
```
churn_data['age'].max(),churn_data['age'].min()
```

Out[111…
```
(82, 18)
```

In [116…
```
churn_data[['age']].loc[[682]]
```

Out[116…

	age
682	82

In [115…
```
churn_data[['age']].loc[[921]]
```

Out[115…

	age
921	18

In [117…
```
churn_data[['age']].loc[[max_id,min_id]]
```

Out[117…

	age
682	82
921	18

Scalar

In [119…
```
x=churn_data['age']
x_min=churn_data['age'].min()
x_max=churn_data['age'].min()
Nr=x-x_min
Dr=x_max/x_min
scalar=Nr/Dr
churn_data['scalar']=scalar
churn_data[['scalar']]
```

Out[119…

	scalar
0	18.0
1	21.0
2	47.0
3	6.0
4	22.0
…	…
1995	36.0
1996	27.0
1997	35.0
1998	22.0
1999	19.0

2000 rows × 1 columns

Standard scalar

package

In [120…

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
churn_data['ss_age']=ss.fit_transform(churn_data[['age']])
churn_data[['ss_age']]
```

Out[120…

	ss_age
0	-0.263675
1	0.030332
2	2.578388
3	-1.439701
4	0.128334
…	…
1995	1.500364
1996	0.618345
1997	1.402362
1998	0.128334
1999	-0.165673

2000 rows × 1 columns

Z-formula

In [124…

```
df=churn_data['age']
mean=churn_data['age'].mean()
std=churn_data['age'].std()
Nr=df-mean
Z_Formula=Nr/std
Z_Formula
churn_data['Z_Formula']=Z_Formula
churn_data[['Z_Formula']]
```

Out[124…

	Z_Formula
0	-0.263609
1	0.030324
2	2.577743
3	-1.439341
4	0.128302
...	...
1995	1.499989
1996	0.618190
1997	1.402011
1998	0.128302
1999	-0.165631

2000 rows × 1 columns

comparing

In [128…

```
churn_data[['age','scalar','Z_Formula']]
```

Out[128...

	age	scalar	Z_Formula
0	36	18.0	-0.263609
1	39	21.0	0.030324
2	65	47.0	2.577743
3	24	6.0	-1.439341
4	40	22.0	0.128302
...	...	...	...
1995	54	36.0	1.499989
1996	45	27.0	0.618190
1997	53	35.0	1.402011
1998	40	22.0	0.128302
1999	37	19.0	-0.165631

2000 rows × 3 columns

In [ ]: