step1:Import the packages

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

step-2:Read the data

In [3]:
```
path1=r"C:\\Users\\TATIREDDY\\Documents\\NareshIT\\visadataset.csv"
visa_df=pd.read_csv(path1)
visa_df
```

Out[3]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [9]:
```
path2=r"C:\\Users\\TATIREDDY\\Documents\\NareshIT\\bank.csv"
bank_df=pd.read_csv(path2,sep=';')
bank_df
```

Out[9]:

	age	job	marital	education	default	balance	housing	loan	contact
0	30	unemployed	married	primary	no	1787	no	no	cellular
1	33	services	married	secondary	no	4789	yes	yes	cellular
2	35	management	single	tertiary	no	1350	yes	no	cellular
3	30	management	married	tertiary	no	1476	yes	yes	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown
...	...	...	...	...	...	...	...	...	...
4516	33	services	married	secondary	no	-333	yes	no	cellular
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown
4518	57	technician	married	secondary	no	295	no	no	cellular
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular

4521 rows × 17 columns

In [11]:
```
path3=r"C:\\Users\\TATIREDDY\\Documents\\NareshIT\\telecom_churn_data.csv"
telecom_df=pd.read_csv(path3)
telecom_df
```

Out[11]:

	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

step-3:Understand the data

In [13]:
```
telecom_df.head()  # First 5 will come
```

Out[13]:

	year	customer_id	phone_no	gender	age	no_of_days_subscribed	multi_screen	ma
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	

In [15]:
```
telecom_df.shape
print(f"The number of rows are : {telecom_df.shape[0]}")
print(f"The number of columns are : {telecom_df.shape[1]}")
```

```
The number of rows are : 2000
The number of columns are : 16
```

In [17]:
```
telecom_df.size
```

Out[17]:
```
32000
```

In [19]:
```
telecom_df.dtypes
```

Out[19]:
```
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 [21]:
```
telecom_df.columns
```

Out[21]:
```
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')
```

In [23]:
```
telecom_df.isnull().sum()
```

Out[23]:
```
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
```

In [25]:
```
telecom_df.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                  1976 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   1972 non-null   float64
 14  customer_support_calls  2000 non-null   int64  
 15  churn                   1965 non-null   float64
dtypes: float64(5), int64(7), object(4)
memory usage: 250.1+ KB
```

step-4:Drop unwanted columns

In [27]:
```
telecom_df['customer_id'].nunique()
```

Out[27]:
```
1999
```

In [29]:
```
telecom_df['phone_no'].nunique()
```

Out[29]:
```
2000
```

In [31]:
```
telecom_df['year'].nunique()
```

Out[31]:
```
1
```

In [33]:
```
telecom_df.drop(columns=['customer_id','phone_no','year'],inplace=True)
```

step-5::Fill the missing columns

maximum days inactive

In [35]:
```
telecom_df['maximum_days_inactive']
```

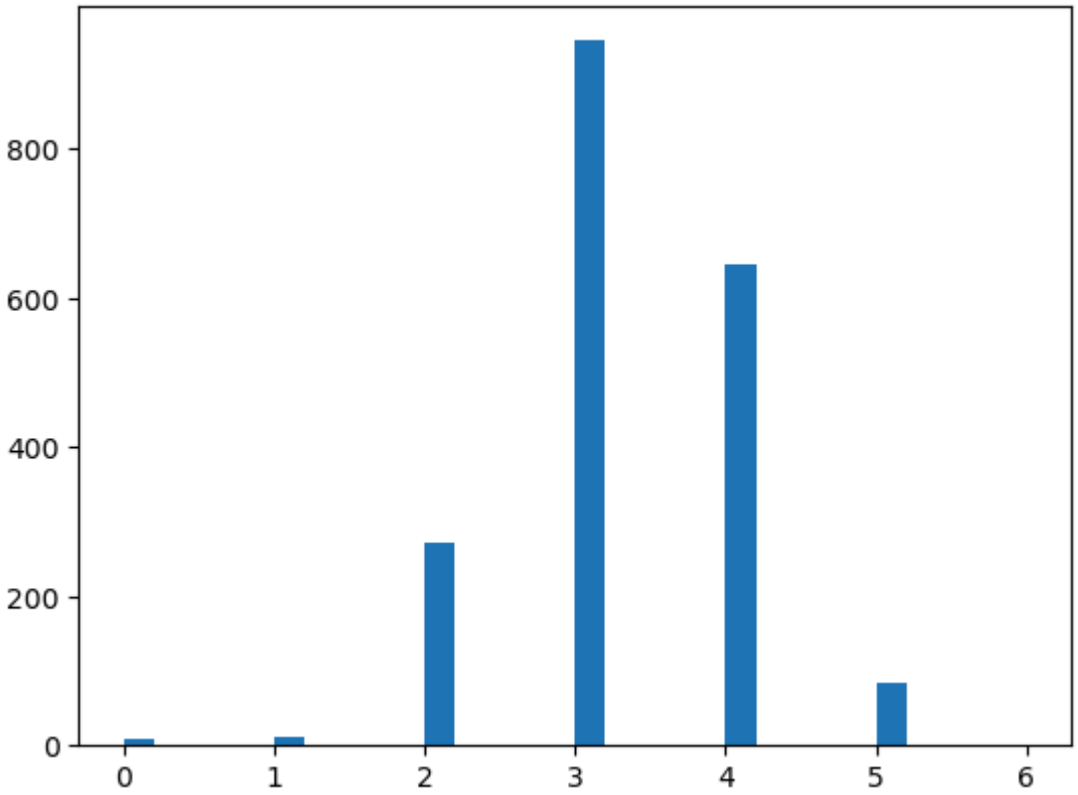
Out[35]:
```
0       4.0
1       3.0
2       4.0
3       3.0
4       3.0
       ...
1995    4.0
1996    3.0
1997    5.0
1998    NaN
1999    3.0
Name: maximum_days_inactive, Length: 2000, dtype: float64
```

In [37]:
```
telecom_df['maximum_days_inactive'].unique()
```

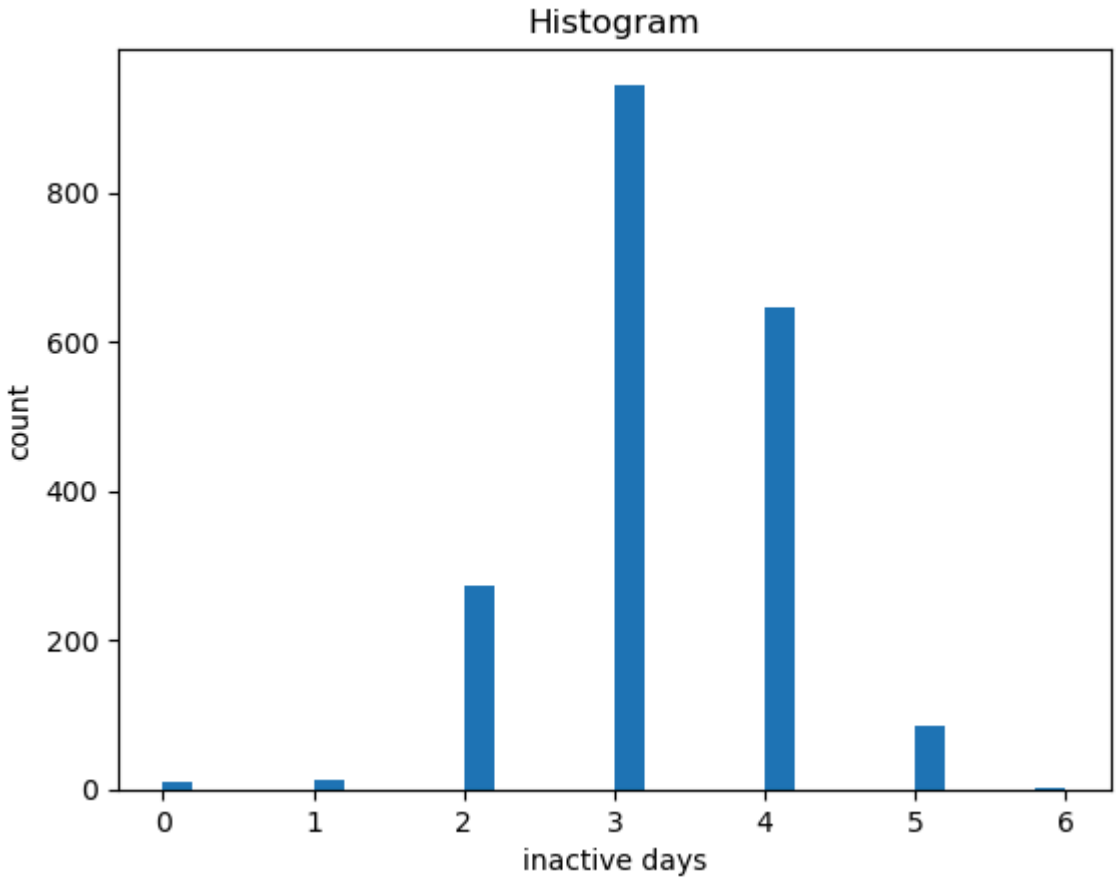
Out[37]:
```
array([ 4.,  3., nan,  2.,  5.,  1.,  0.,  6.])
```

In [39]:
```
plt.hist(telecom_df['maximum_days_inactive'],bins=30)
```

Out[39]:
```
(array([ 10.,   0.,   0.,   0.,   0.,  12.,   0.,   0.,   0.,   0., 273.,
          0.,   0.,   0.,   0., 945.,   0.,   0.,   0.,   0., 645.,   0.,
          0.,   0.,   0.,  85.,   0.,   0.,   0.,   2.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. , 2.2, 2.4,
        2.6, 2.8, 3. , 3.2, 3.4, 3.6, 3.8, 4. , 4.2, 4.4, 4.6, 4.8, 5. ,
        5.2, 5.4, 5.6, 5.8, 6. ]),
 <BarContainer object of 30 artists>)
```



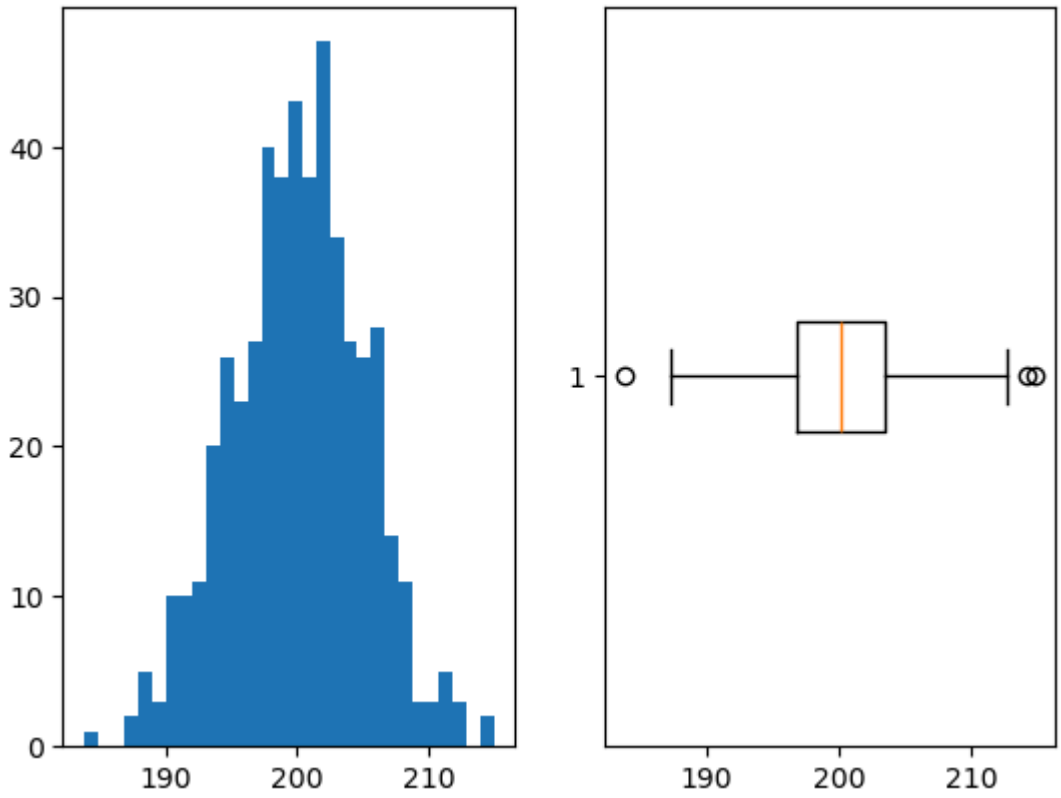
In [41]:
```
plt.hist(telecom_df['maximum_days_inactive'],bins=30)
plt.xlabel('inactive days')
plt.ylabel('count')
plt.title('Histogram')
plt.savefig('Max_days_inactive_histogram.jpg')
```



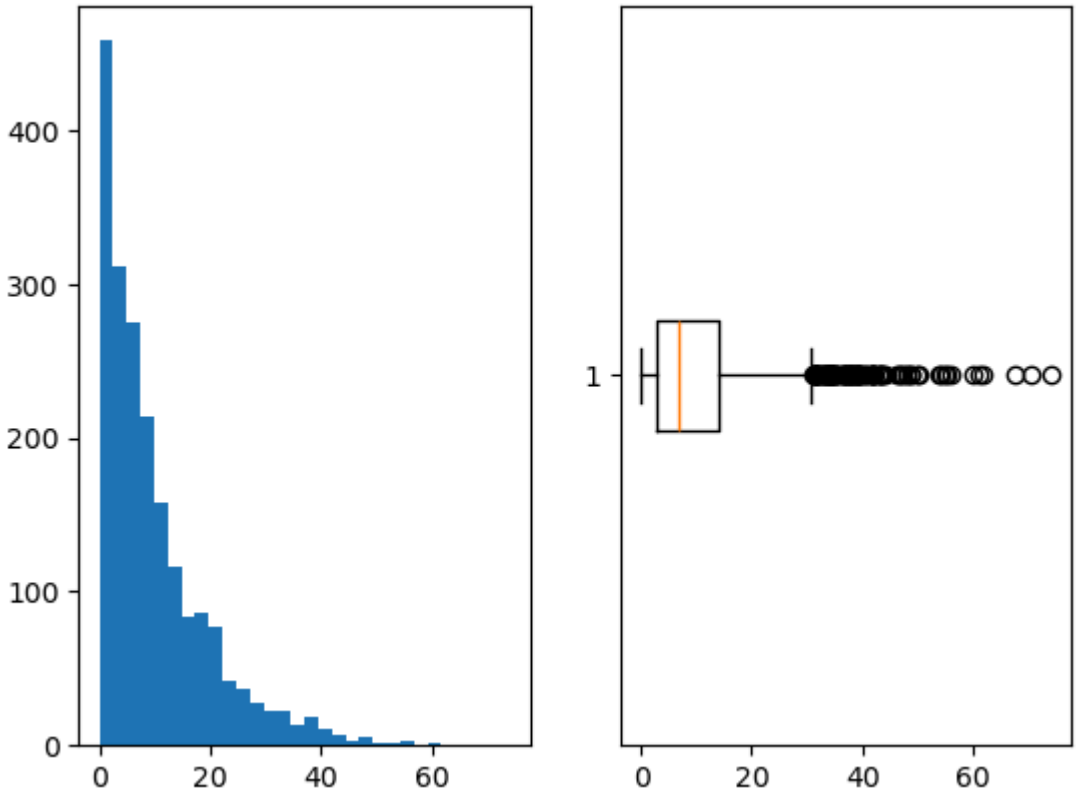
In [43]:
```
telecom_df['maximum_days_inactive'].value_counts()
```

Out[43]:
```
maximum_days_inactive
3.0    945
4.0    645
2.0    273
5.0     85
1.0     12
0.0     10
6.0      2
Name: count, dtype: int64
```

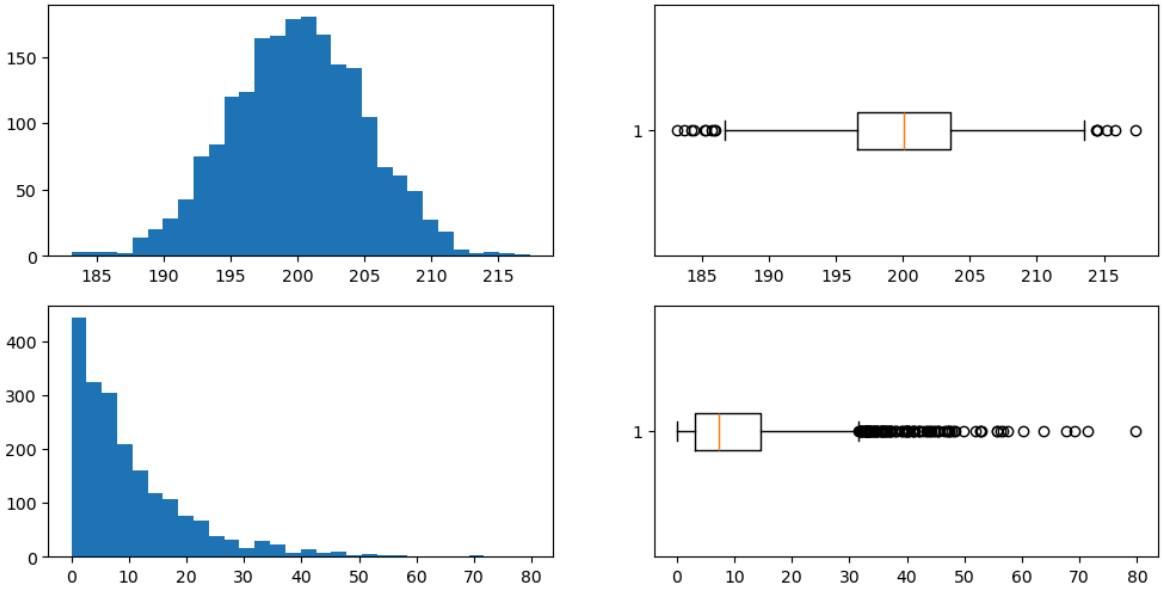
In [45]:
```
normal_data=np.random.normal(200,5,500)
plt.subplot(1,2,1).hist(normal_data,bins=30)
plt.subplot(1,2,2).boxplot(normal_data,vert=False)
plt.show()
```



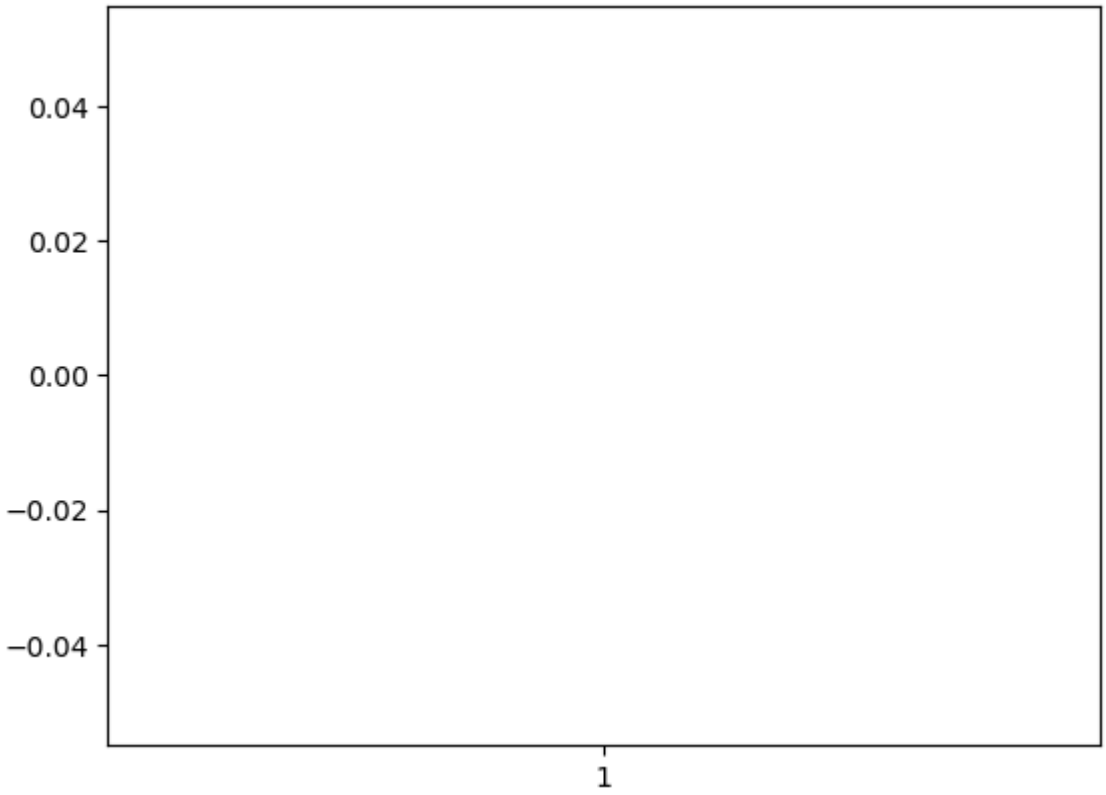
In [47]:
```
exp_data=np.random.exponential(10,2000)
plt.subplot(1,2,1).hist(exp_data,bins=30)
plt.subplot(1,2,2).boxplot(exp_data,vert=False)
plt.show()
```



In [49]:
```
normal_data=np.random.normal(200,5,2000)
exp_data=np.random.exponential(10,2000)
plt.figure(figsize=(12,6))
plt.subplot(2,2,1).hist(normal_data,bins=30)
plt.subplot(2,2,2).boxplot(normal_data,vert=False)
plt.subplot(2,2,3).hist(exp_data,bins=30)
plt.subplot(2,2,4).boxplot(exp_data,vert=False)
plt.show()
```



In [51]:
```
plt.boxplot(telecom_df['maximum_days_inactive'])
plt.show()
```



In [53]:
```
# step-1: get the median value
inactive_data=telecom_df['maximum_days_inactive']
median_inactive=inactive_data.median()
# Step-2: fill with median
telecom_df['maximum_days_inactive']=inactive_data.fillna(median_inactive)
```

In [55]:
```
# Step-3: Convert float to int
telecom_df['maximum_days_inactive']=telecom_df['maximum_days_inactive'].astype(i
```

In [57]:
```
telecom_df.isnull().sum()
```

Out[57]:
```
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      0
customer_support_calls     0
churn                     35
dtype: int64
```

churn

In [60]:
```
churn_data=telecom_df['churn']
median_churn=churn_data.median()
# Step-2: fill with median
telecom_df['churn']=churn_data.fillna(median_churn)
```

```
# Step-3: Convert float to int
telecom_df['churn']=telecom_df['churn'].astype(int)
```

In [62]:
```
telecom_df.isnull().sum()
```

Out[62]:
```
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      0
customer_support_calls     0
churn                      0
dtype: int64
```

In [64]:
```
telecom_df['churn'].value_counts()
```

Out[64]:
```
churn
0    1738
1     262
Name: count, dtype: int64
```

In [66]:
```
gender_data=telecom_df['gender']
gender_mode=gender_data.mode()
telecom_df['gender']=gender_data.fillna(gender_mode[0])
```

In [68]:
```
gender_mode[0]
```

Out[68]:
```
'Male'
```

In [70]:
```
telecom_df['gender'].value_counts()
```

Out[70]:
```
gender
Male      1077
Female     923
Name: count, dtype: int64
```

In [72]:
```
telecom_df['gender'].unique()
```

Out[72]:
```
array(['Female', 'Male'], dtype=object)
```

In [74]:
```
telecom_df.isnull().sum()
```

Out[74]:
```
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 [76]:
```
telecom_df
```

Out[76]:

	gender	age	no_of_days_subscribed	multi_screen	mail_subscribed	weekly_mins_
0	Female	36	62	no	no	
1	Female	39	149	no	no	
2	Female	65	126	no	no	
3	Female	24	131	no	yes	
4	Female	40	191	no	no	
...	...	...	...	...	...	
1995	Female	54	75	no	yes	
1996	Male	45	127	no	no	
1997	Male	53	94	no	no	
1998	Male	40	94	no	no	
1999	Male	37	73	no	no	

2000 rows × 13 columns

In [78]:
```
cat=telecom_df.select_dtypes(include='object').columns
num=telecom_df.select_dtypes(exclude='object').columns
cat,num
```

Out[78]:
```
(Index(['gender', 'multi_screen', 'mail_subscribed'], dtype='object'),
 Index(['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 [80]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
le.fit_transform(telecom_df['multi_screen'])
# fit means : d={'no':0,'yes':1}
```

Out[80]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [82]:
```
telecom_df['multi_screen'].values[:3]
```

Out[82]:
```
array(['no', 'no', 'no'], dtype=object)
```

In [84]:
```
# One hot encoder
pd.get_dummies(telecom_df['multi_screen'],
               dtype=int)
```

Out[84]:

	no	yes
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
1995	1	0
1996	1	0
1997	1	0
1998	1	0
1999	1	0

2000 rows × 2 columns

copy the update data and perform encoding

In [87]:
```
telecom_df1=telecom_df.copy()
```

In [89]:
```
cat
```

Out[89]:
```
Index(['gender', 'multi_screen', 'mail_subscribed'], dtype='object')
```

In [91]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in cat:
    telecom_df1[i]=le.fit_transform(telecom_df1[i])
```

In [93]:
```
telecom_df1
```

Out[93]:

	gender	age	no_of_days_subscribed	multi_screen	mail_subscribed	weekly_mins_
0	0	36	62	0	0	
1	0	39	149	0	0	
2	0	65	126	0	0	
3	0	24	131	0	1	
4	0	40	191	0	0	
...	...	...	...	...	...	
1995	0	54	75	0	1	
1996	1	45	127	0	0	
1997	1	53	94	0	0	
1998	1	40	94	0	0	
1999	1	37	73	0	0	

2000 rows × 13 columns

Step-7: Scaling

In [96]:
```
num
```

Out[96]:
```
Index(['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 [98]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
v1=ss.fit_transform(telecom_df1[['no_of_days_subscribed']])
```

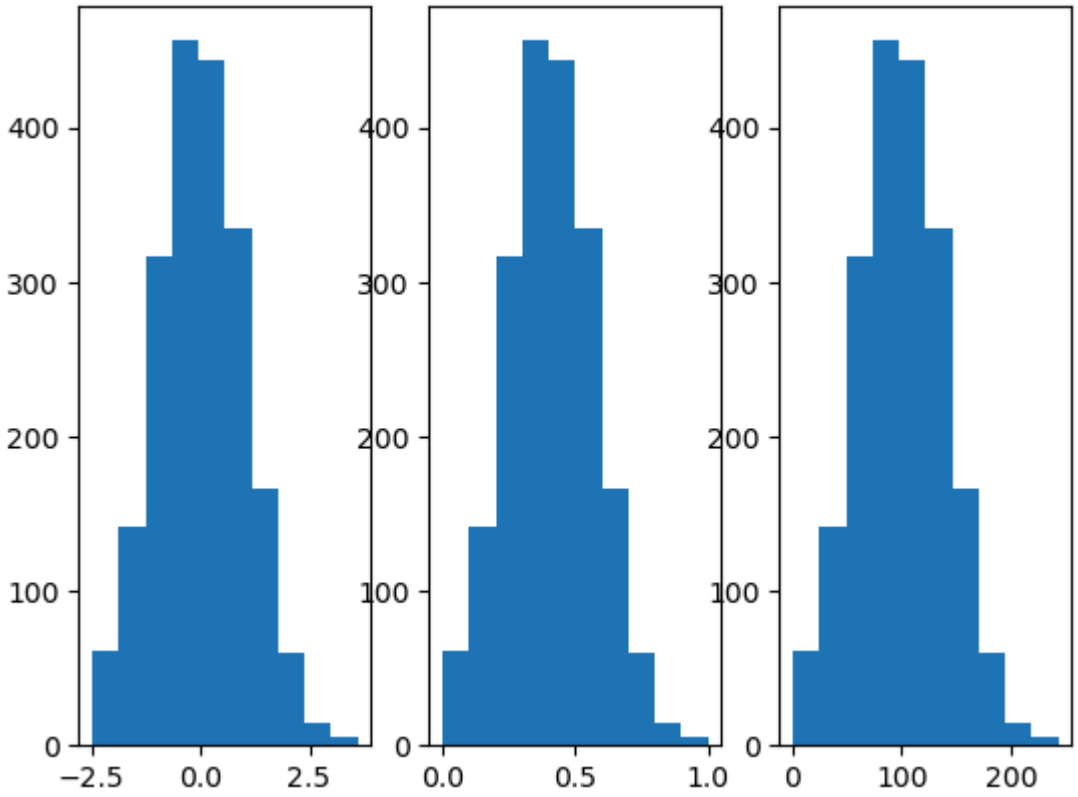
In [100…
```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
v2=mms.fit_transform(telecom_df1[['no_of_days_subscribed']])
```

In [102…
```
plt.subplot(1,3,1).hist(v1)

plt.subplot(1,3,2).hist(v2)

plt.subplot(1,3,3).hist(telecom_df1['no_of_days_subscribed'])
```

Out[102…
```
(array([ 61., 141., 317., 456., 444., 334., 166.,  60.,  15.,   6.]),
 array([  1. ,  25.2,  49.4,  73.6,  97.8, 122. , 146.2, 170.4, 194.6,
        218.8, 243. ]),
 <BarContainer object of 10 artists>)
```



In [104…
```
telecom_df2=telecom_df1.copy()
```

In [108…
```
cols=num[:len(num)-1]
```

In [110…
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
telecom_df2[cols]=ss.fit_transform(telecom_df2[cols])
```

In [112…
```
telecom_df2
```

Out[112...

	gender	age	no_of_days_subscribed	multi_screen	mail_subscribed	weekly_
0	0	-0.263675	-0.949794	0	0	
1	0	0.030332	1.239136	0	0	
2	0	2.578388	0.660453	0	0	
3	0	-1.439701	0.786254	0	1	
4	0	0.128334	2.295860	0	0	
...	...	...	...	...	...	
1995	0	1.500364	-0.622713	0	1	
1996	1	0.618345	0.685613	0	0	
1997	1	1.402362	-0.144671	0	0	
1998	1	0.128334	-0.144671	0	0	
1999	1	-0.165673	-0.673033	0	0	

2000 rows × 13 columns

In [ ]: