In [109]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
%matplotlib inline
from sklearn.model_selection import train_test_split as tts
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
```

In [2]:

```
import warnings
warnings.simplefilter("ignore")
```

In [3]:

```
df = pd.read_csv('Python_Case_Studytop.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	State	Account_length	Area_code	International_plan	Voice_mail_plan	Number_vmail_messag
0	KS	128	415	No	Yes	
1	OH	107	415	No	Yes	
2	NJ	137	415	No	No	
3	OH	84	408	Yes	No	
4	OK	75	415	Yes	No	

In [5]:

```
df.columns
```

Out[5]:

```
Index(['State', 'Account_length', 'Area_code', 'International_plan',
       'Voice_mail_plan', 'Number_vmail_messages', 'Total_day_minutes',
       'Total_day_calls', 'Total_day_charge', 'Total_eve_minutes',
       'Total_eve_calls', 'Total_eve_charge', 'Total_night_minutes',
       'Total_night_calls', 'Total_night_charge', 'Total_intl_minutes',
       'Total_intl_calls', 'Total_intl_charge', 'Customer_service_calls',
       'Churn'],
      dtype='object')
```

In [6]:

```
df.describe()
```

Out[6]:

	Account_length	Area_code	Number_vmail_messages	Total_day_minutes	Total_day_call
count	2666.000000	2666.000000	2666.000000	2666.00000	2666.00000
mean	100.620405	437.438860	8.021755	179.48162	100.31020
std	39.563974	42.521018	13.612277	54.21035	19.98816
min	1.000000	408.000000	0.000000	0.00000	0.00000
25%	73.000000	408.000000	0.000000	143.40000	87.00000
50%	100.000000	415.000000	0.000000	179.95000	101.00000
75%	127.000000	510.000000	19.000000	215.90000	114.00000
max	243.000000	510.000000	50.000000	350.80000	160.00000

In [7]:

```
df.isnull().any()
```

Out[7]:

```
State                   False
Account_length          False
Area_code               False
International_plan      False
Voice_mail_plan         False
Number_vmail_messages   False
Total_day_minutes       False
Total_day_calls         False
Total_day_charge        False
Total_eve_minutes       False
Total_eve_calls         False
Total_eve_charge        False
Total_night_minutes     False
Total_night_calls       False
Total_night_charge      False
Total_intl_minutes      False
Total_intl_calls        False
Total_intl_charge       False
Customer_service_calls  False
Churn                   False
dtype: bool
```

In [8]:

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

Out[8]:

```
State                   0
Account_length          0
Area_code               0
International_plan      0
Voice_mail_plan         0
Number_vmail_messages   0
Total_day_minutes       0
Total_day_calls         0
Total_day_charge        0
Total_eve_minutes       0
Total_eve_calls         0
Total_eve_charge        0
Total_night_minutes     0
Total_night_calls       0
Total_night_charge      0
Total_intl_minutes      0
Total_intl_calls        0
Total_intl_charge       0
Customer_service_calls  0
Churn                   0
dtype: int64
```

In [9]:

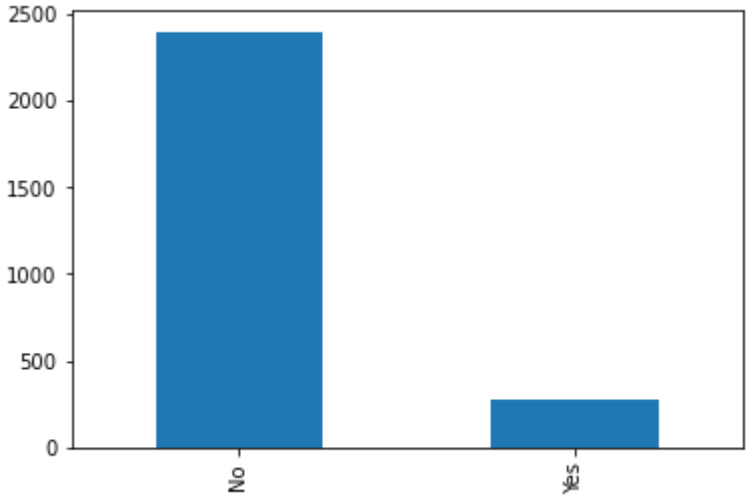
```
df['International_plan'].value_counts()
```

Out[9]:

```
No     2396
Yes     270
Name: International_plan, dtype: int64
```

In [10]:

```
df['International_plan'].value_counts().plot(kind='bar');
```



In [11]:

```
df['Voice_mail_plan'].value_counts()
```

Out[11]:

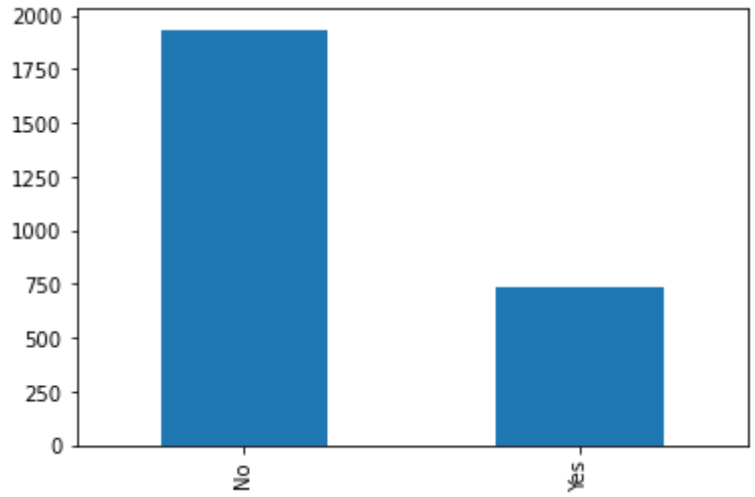
```
No     1933
Yes     733
Name: Voice_mail_plan, dtype: int64
```

In [12]:

```
df['Voice_mail_plan'].value_counts().plot(kind='bar')
```

Out[12]:

<AxesSubplot:>



In [13]:

```
c0_50 = df[(df.Total_day_calls < 50 )]
c51_75 = df[(df.Total_day_calls > 50 ) & (df.Total_day_calls <= 75)]
c76_100 = df[(df.Total_day_calls > 75 ) & (df.Total_day_calls <= 100)]
c101_125 = df[(df.Total_day_calls > 100 ) & (df.Total_day_calls <= 125)]
c126_150 = df[(df.Total_day_calls > 125 ) & (df.Total_day_calls <= 150)]
c151_175 = df[(df.Total_day_calls > 150 )]
```

In [14]:

```
print(c0_50.shape)
print(c51_75.shape)
print(c76_100.shape)
print(c101_125.shape)
print(c126_150.shape)
print(c151_175.shape)
```

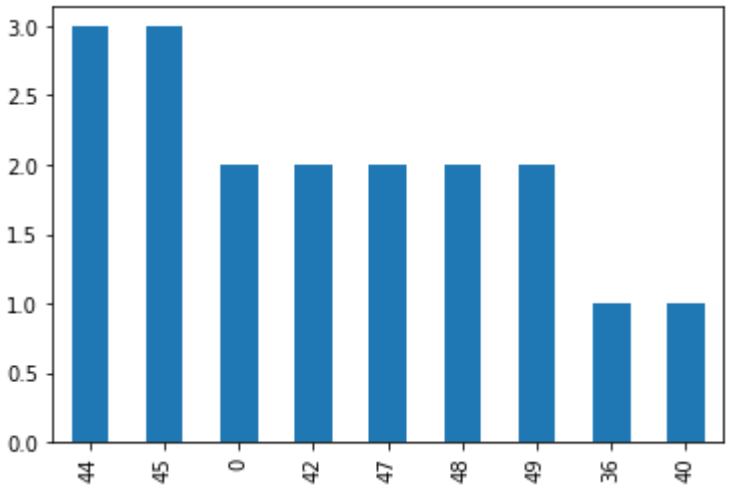
```
(18, 20)
(260, 20)
(1048, 20)
(1076, 20)
(252, 20)
(12, 20)
```

In [15]:

```
c0_50['Total_day_calls'].value_counts().plot(kind='bar')
```

Out[15]:

<AxesSubplot:>

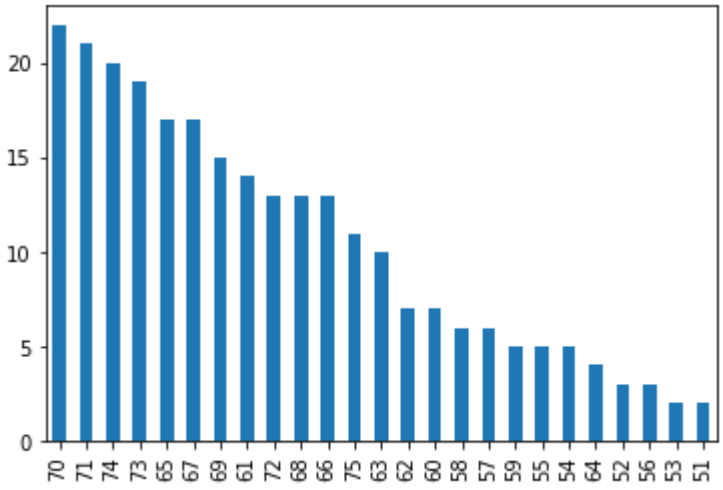


In [16]:

```
c51_75['Total_day_calls'].value_counts().plot(kind='bar')
```

Out[16]:

<AxesSubplot:>

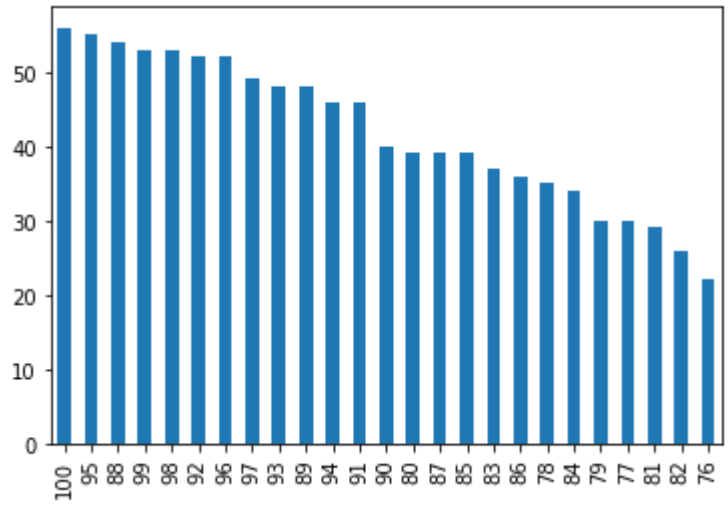


In [17]:

```
c76_100['Total_day_calls'].value_counts().plot(kind='bar')
```

Out[17]:

<AxesSubplot:>

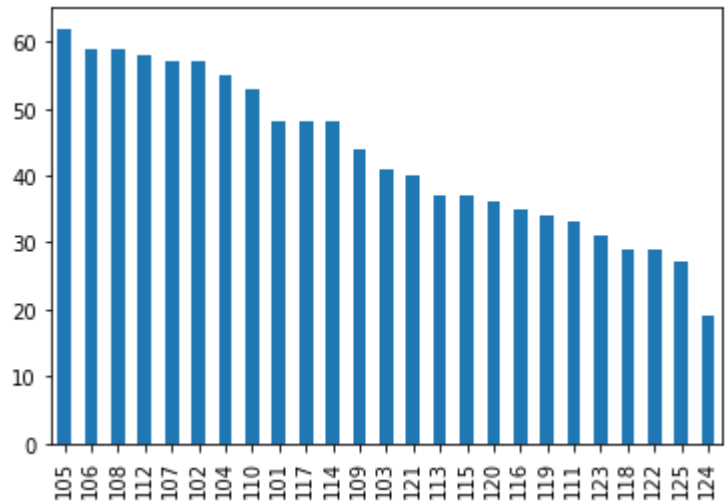


In [18]:

```
c101_125['Total_day_calls'].value_counts().plot(kind='bar')
```

Out[18]:

<AxesSubplot:>

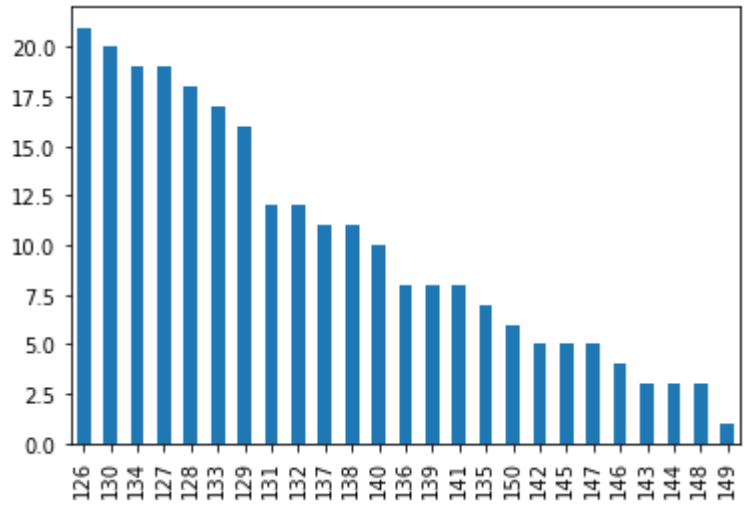


In [19]:

```
c126_150['Total_day_calls'].value_counts().plot(kind='bar')
```

Out[19]:

<AxesSubplot:>

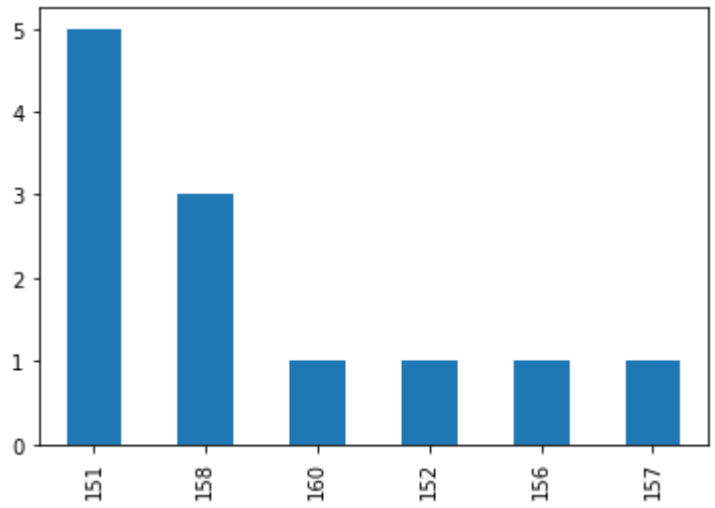


In [20]:

```
c151_175['Total_day_calls'].value_counts().plot(kind='bar')
```

Out[20]:

<AxesSubplot:>



In [21]:

```
c0_5 = df[(df.Total_day_charge <= 5 )]
c6_10 = df[(df.Total_day_charge > 5 ) & (df.Total_day_charge <= 10)]
c11_15 = df[(df.Total_day_charge > 10 ) & (df.Total_day_charge <= 15)]
c16_20 = df[(df.Total_day_charge > 15 ) & (df.Total_day_charge <= 20)]
c21_25 = df[(df.Total_day_charge > 20 ) & (df.Total_day_charge <= 25)]
c26_30 = df[(df.Total_day_charge > 25 ) & (df.Total_day_charge <= 30)]
c31_35 = df[(df.Total_day_charge > 30 ) & (df.Total_day_charge <= 35)]
c36_40 = df[(df.Total_day_charge > 35 ) & (df.Total_day_charge <= 40)]
c41_45 = df[(df.Total_day_charge > 40 ) & (df.Total_day_charge <= 45)]
c46_50 = df[(df.Total_day_charge > 45 ) & (df.Total_day_charge <= 50)]
c51_55 = df[(df.Total_day_charge > 50 ) & (df.Total_day_charge <= 55)]
c56_60 = df[(df.Total_day_charge > 55 ) & (df.Total_day_charge <= 60)]
```

In [22]:

```
print(c0_5.shape)
print(c6_10.shape)
print(c11_15.shape)
print(c16_20.shape)
print(c21_25.shape)
print(c26_30.shape)
print(c31_35.shape)
print(c36_40.shape)
print(c41_45.shape)
print(c46_50.shape)
print(c51_55.shape)
print(c56_60.shape)
```

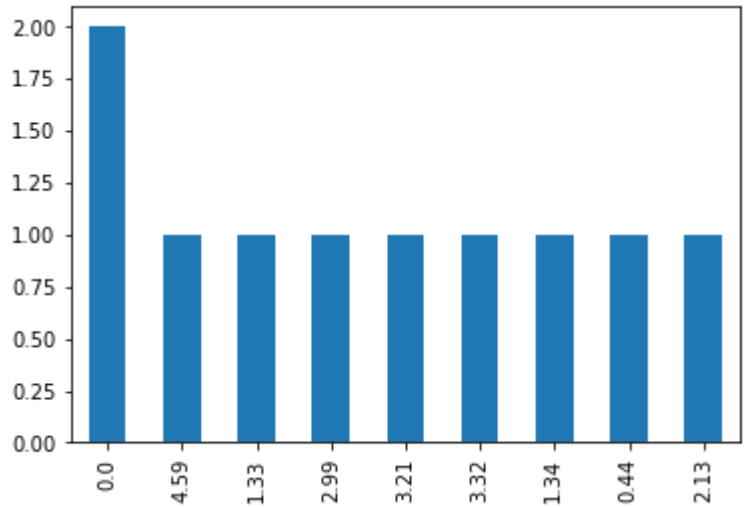
```
(10, 20)
(32, 20)
(80, 20)
(216, 20)
(400, 20)
(533, 20)
(551, 20)
(443, 20)
(243, 20)
(114, 20)
(36, 20)
(8, 20)
```

In [23]:

```
c0_5['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[23]:

<AxesSubplot:>

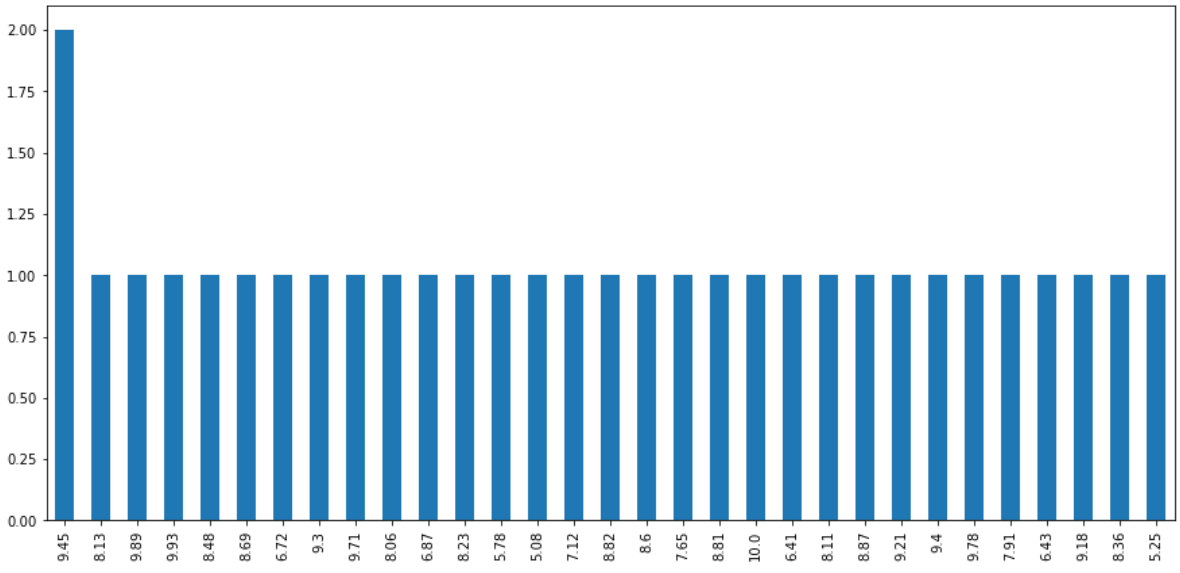


In [24]:

```
plt.figure(figsize=(15,7))
c6_10['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[24]:

<AxesSubplot:>

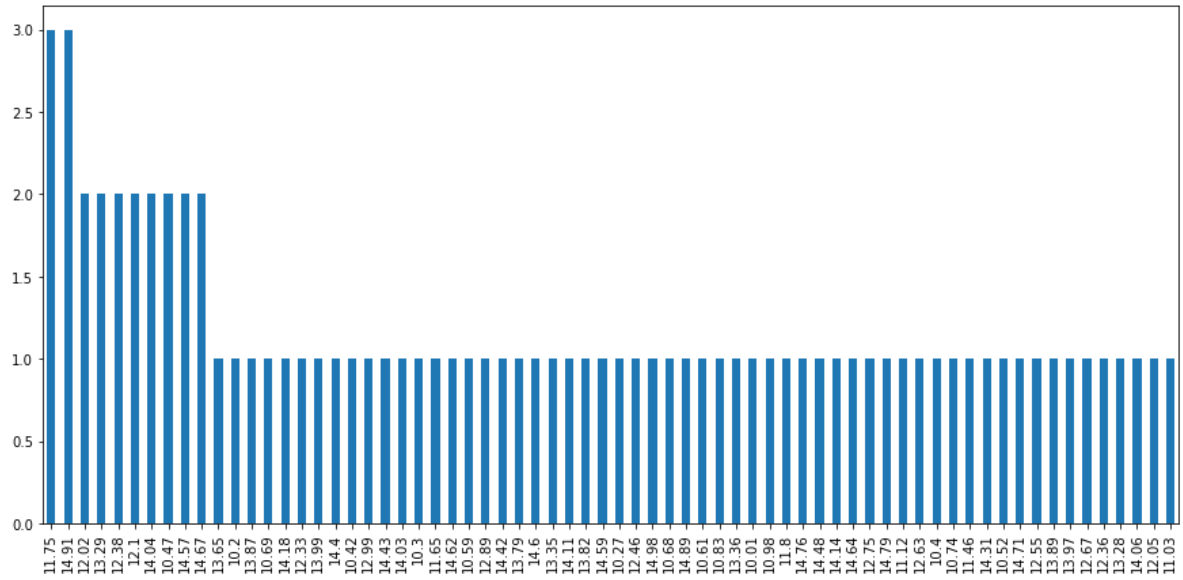


In [25]:

```
plt.figure(figsize=(15,7))
c11_15['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[25]:

<AxesSubplot:>

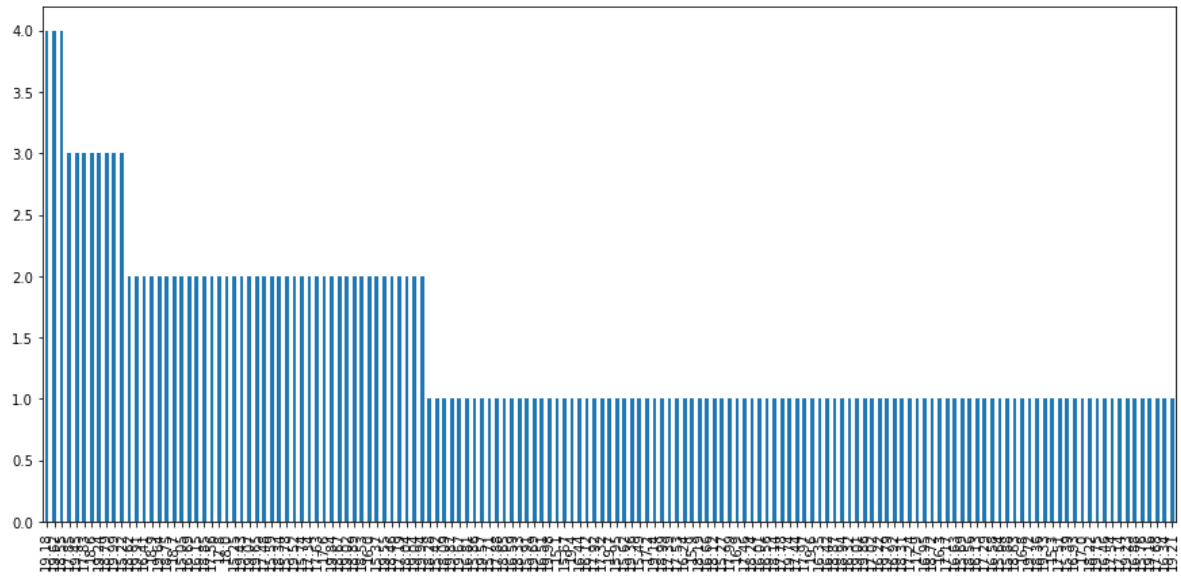


In [26]:

```
plt.figure(figsize=(15,7))
c16_20['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[26]:

<AxesSubplot:>

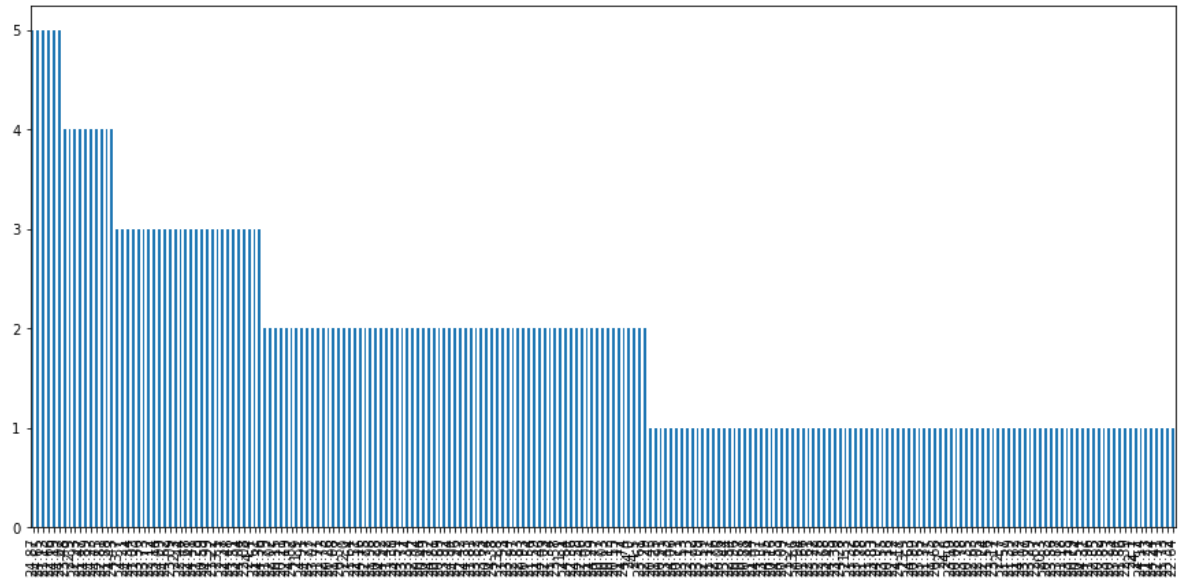


In [27]:

```
plt.figure(figsize=(15,7))
c21_25['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[27]:

<AxesSubplot:>

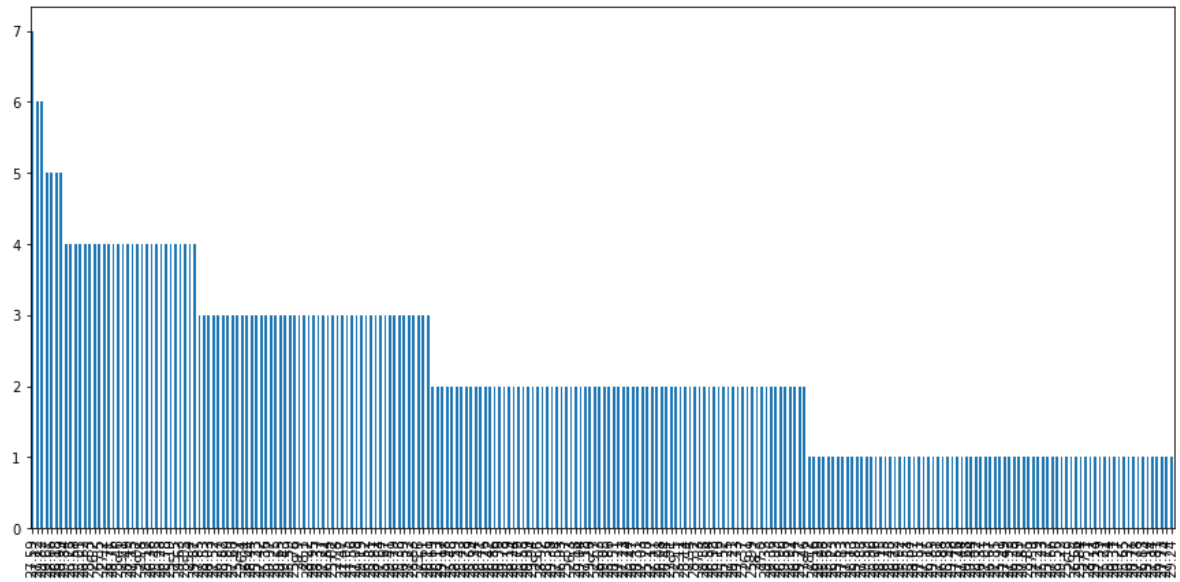


In [28]:

```
plt.figure(figsize=(15,7))
c26_30['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[28]:

<AxesSubplot:>

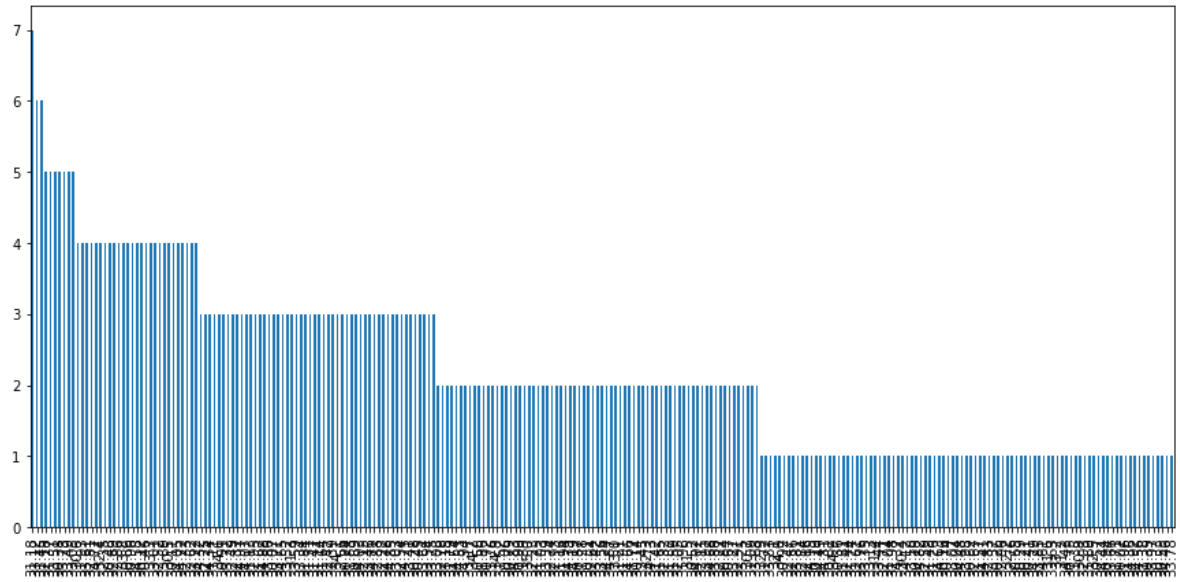


In [29]:

```
plt.figure(figsize=(15,7))
c31_35['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[29]:

<AxesSubplot:>

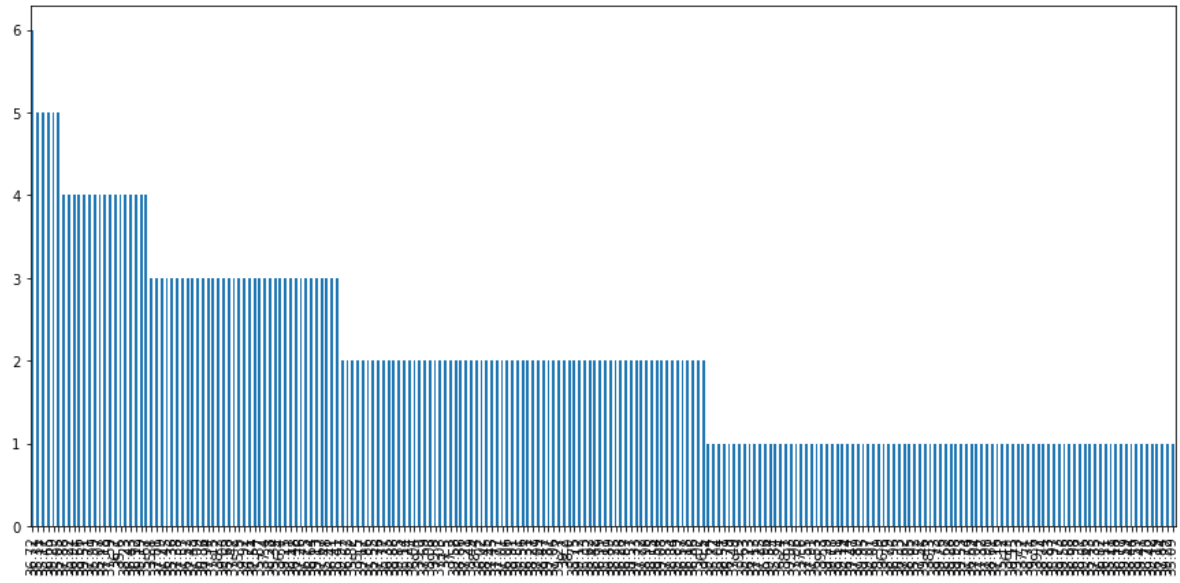


In [30]:

```
plt.figure(figsize=(15,7))
c36_40['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[30]:

<AxesSubplot:>

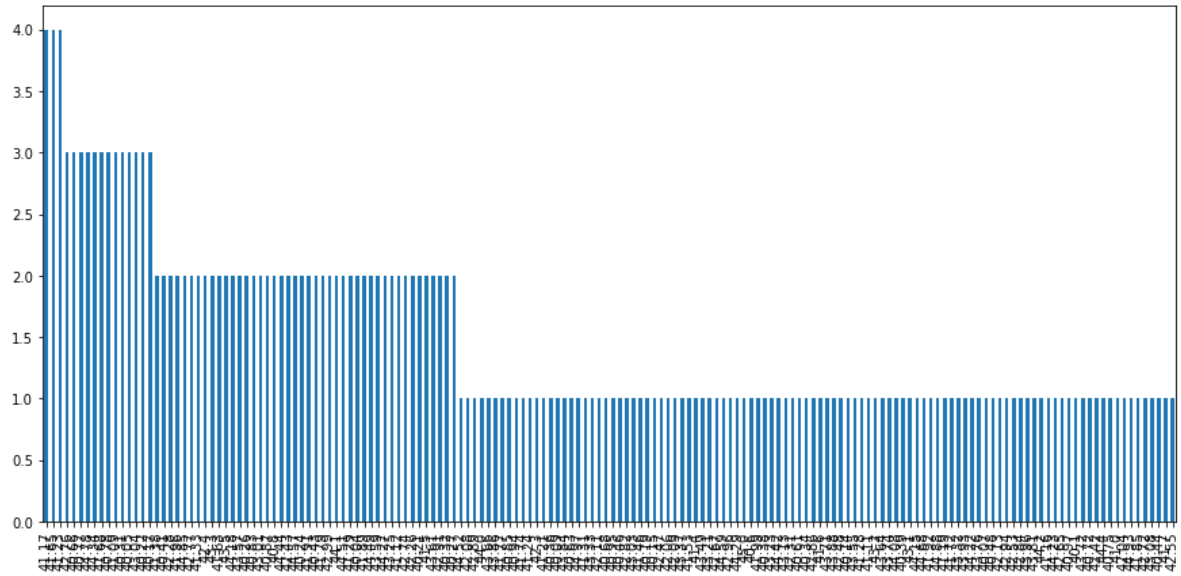


In [31]:

```
plt.figure(figsize=(15,7))
c41_45['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[31]:

<AxesSubplot:>

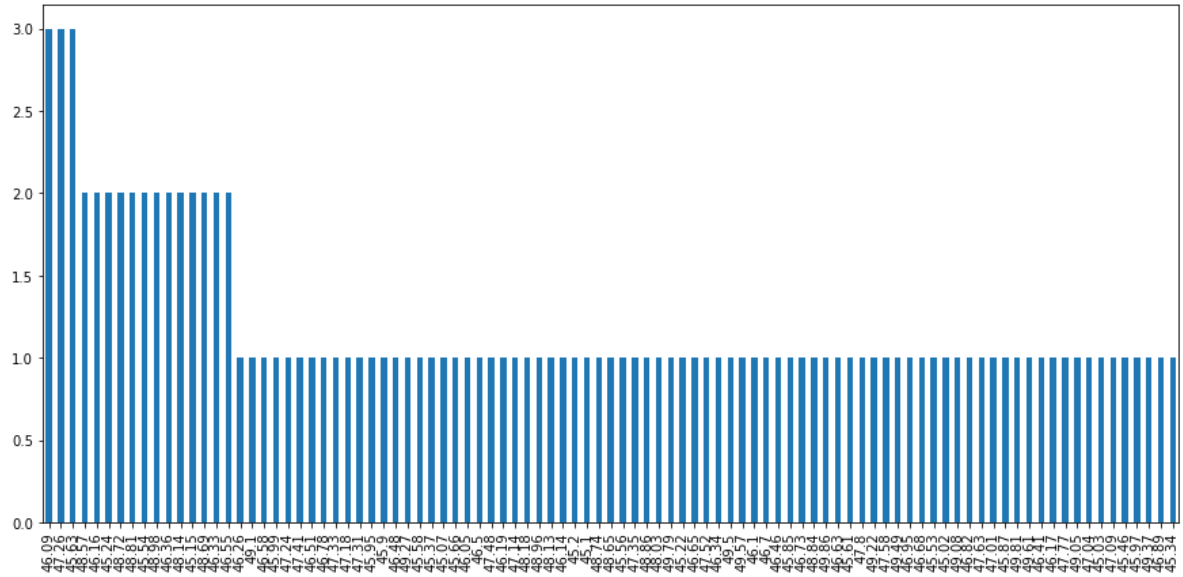


In [32]:

```
plt.figure(figsize=(15,7))
c46_50['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[32]:

<AxesSubplot:>

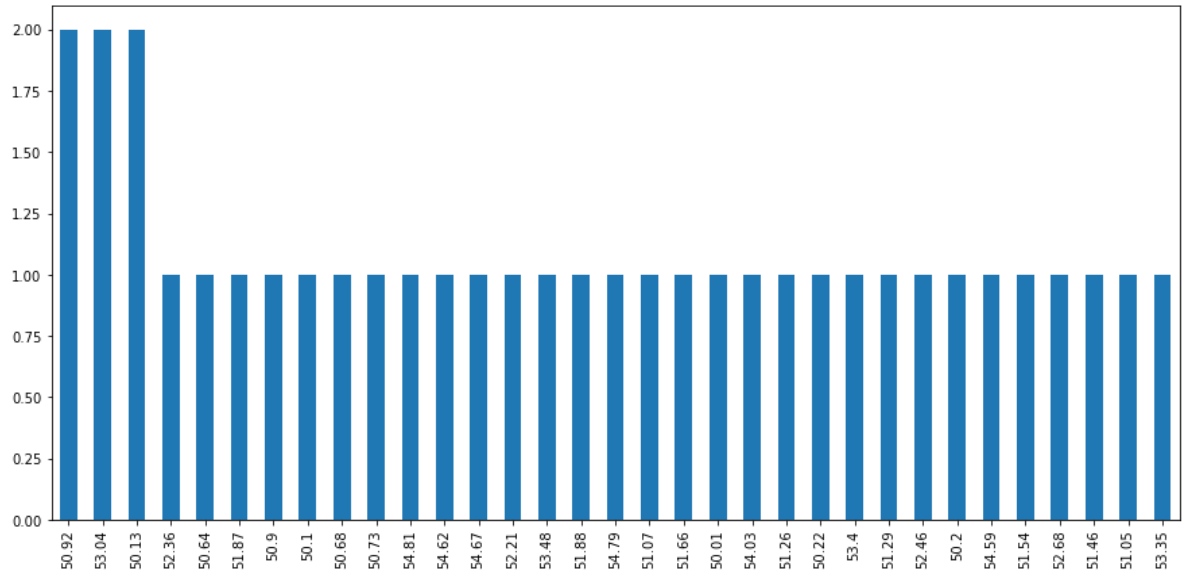


In [33]:

```
plt.figure(figsize=(15,7))
c51_55['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[33]:

<AxesSubplot:>

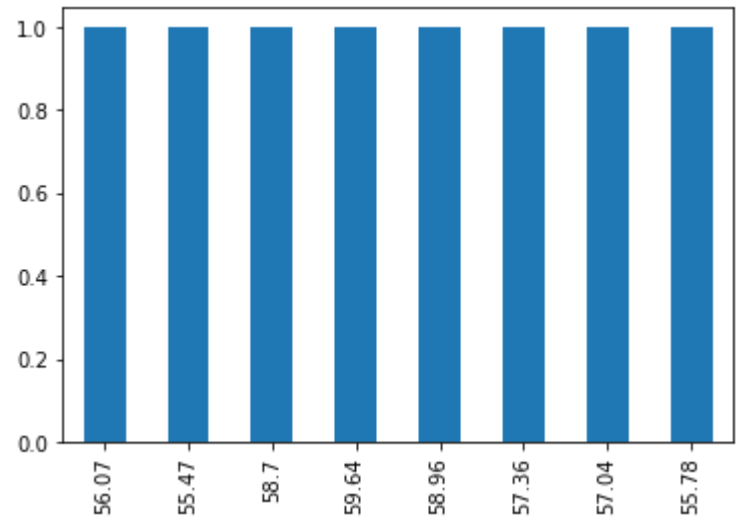


In [34]:

```
c56_60['Total_day_charge'].value_counts().plot(kind='bar')
```

Out[34]:

<AxesSubplot:>

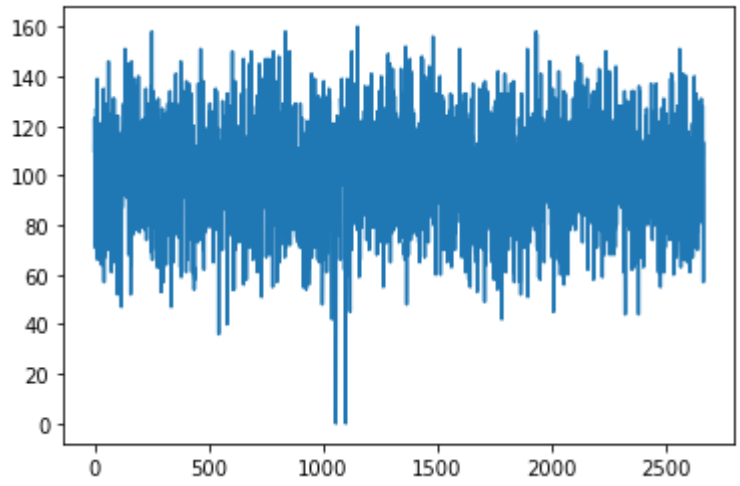


In [35]:

```
plt.plot(df['Total_day_calls'])
```

Out[35]:

```
[<matplotlib.lines.Line2D at 0xe787aa8>]
```

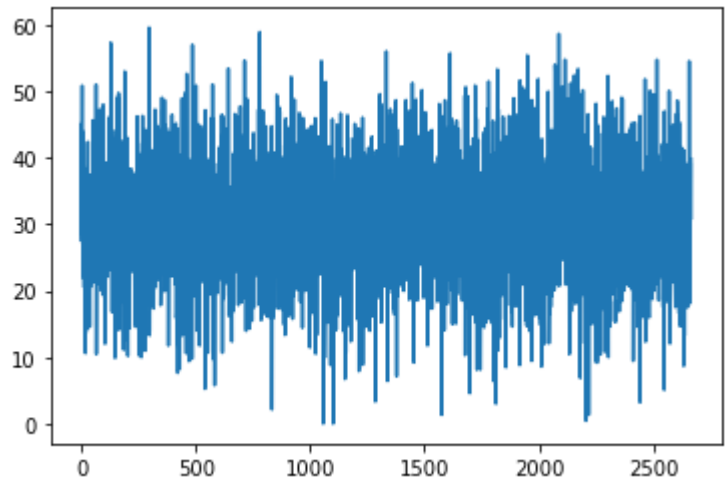


In [36]:

```
plt.plot(df['Total_day_charge'])
```

Out[36]:

```
[<matplotlib.lines.Line2D at 0xea16778>]
```



In [37]:

```
a = df['Churn'].value_counts()
a
```

Out[37]:

```
False    2278
True      388
Name: Churn, dtype: int64
```

In [38]:

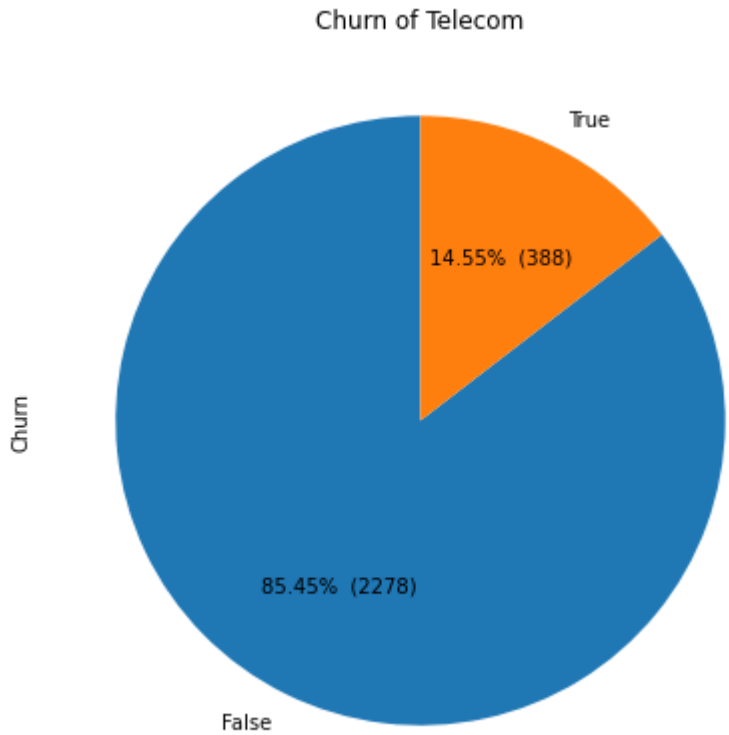
```
def make_autopct(a):
    def my_autopct(pct):
        total = sum(a)
        val = int(round(pct*total/100.0))
        return '{p:.2f}%  ({v:d})'.format(p=pct,v=val)
    return my_autopct
```

In [39]:

```
plt.figure(figsize=(15,7))
plt.title('Churn of Telecom')
df['Churn'].value_counts().plot(kind='pie',autopct=make_autopct(a),startangle = 90)
```

Out[39]:

```
<AxesSubplot:title={'center':'Churn of Telecom'}, ylabel='Churn'>
```

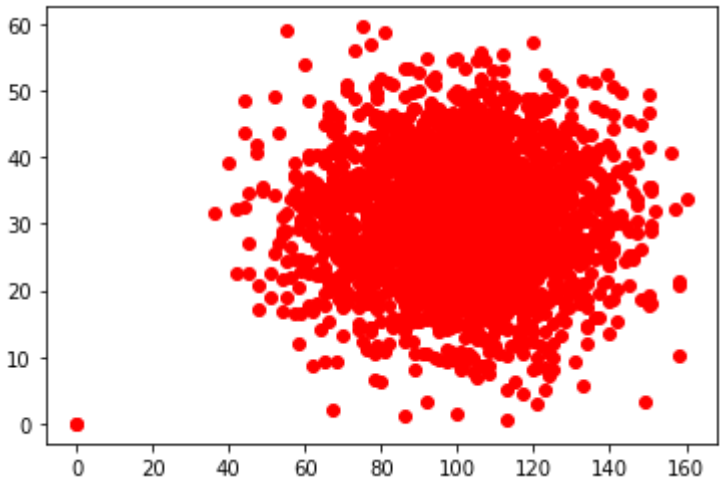


In [40]:

```
plt.scatter(x='Total_day_calls',y='Total_day_charge',data=df,color='red')
```

Out[40]:

```
<matplotlib.collections.PathCollection at 0xeb4c928>
```

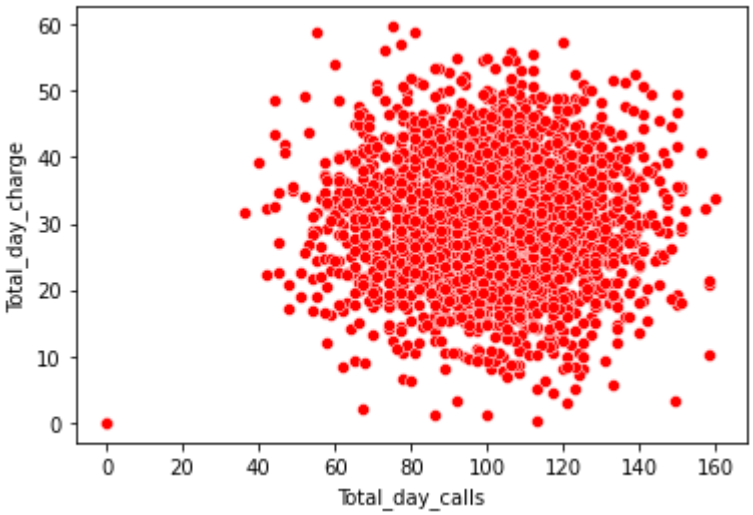


In [41]:

```
sb.scatterplot(x='Total_day_calls',y='Total_day_charge',data=df,color='red')
```

Out[41]:

```
<AxesSubplot:xlabel='Total_day_calls', ylabel='Total_day_charge'>
```

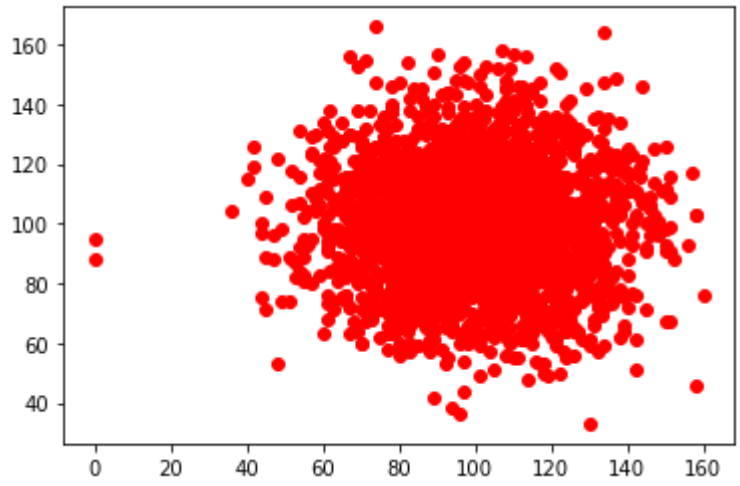


In [42]:

```
plt.scatter(x='Total_day_calls',y='Total_night_calls',data=df,color='red')
```

Out[42]:

```
<matplotlib.collections.PathCollection at 0xeed4ca0>
```

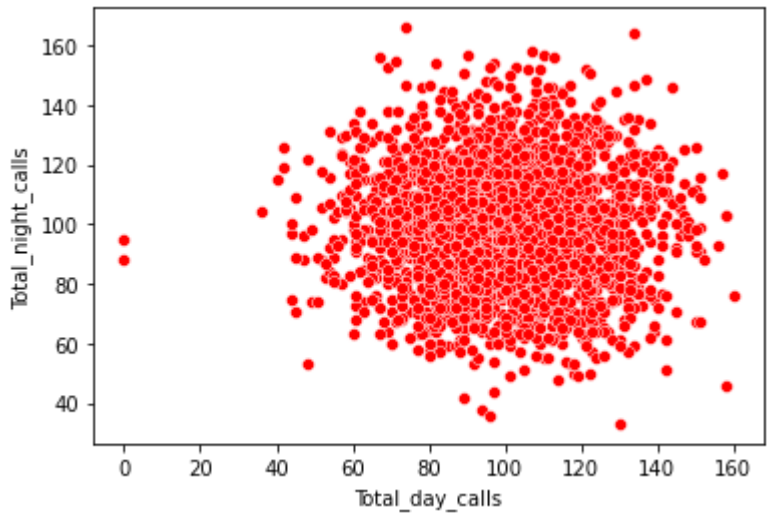


In [43]:

```
sb.scatterplot(x='Total_day_calls',y='Total_night_calls',data=df,color='red')
```

Out[43]:

```
<AxesSubplot:xlabel='Total_day_calls', ylabel='Total_night_calls'>
```

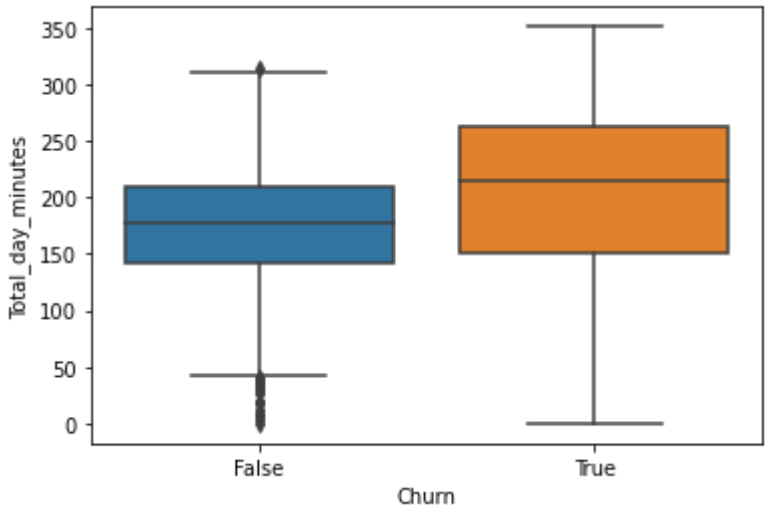


In [44]:

```
sb.boxplot(x = 'Churn',y = 'Total_day_minutes',data = df)
```

Out[44]:

<AxesSubplot:xlabel='Churn', ylabel='Total_day_minutes'>

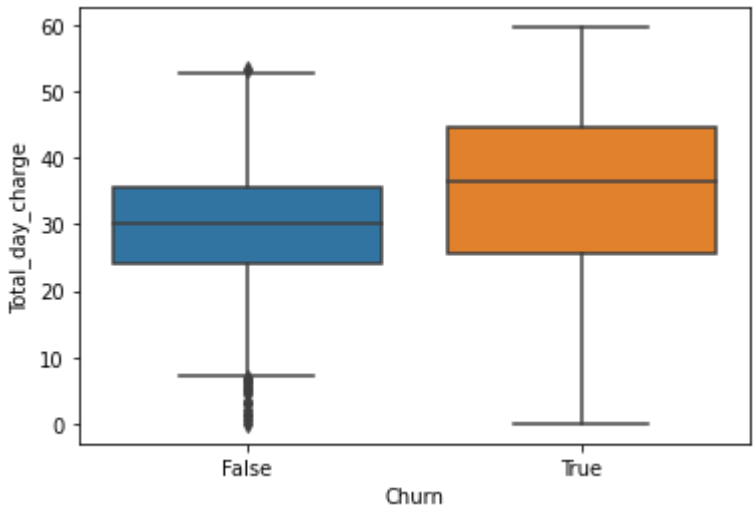


In [45]:

```
sb.boxplot(x = 'Churn',y = 'Total_day_charge',data = df)
```

Out[45]:

<AxesSubplot:xlabel='Churn', ylabel='Total_day_charge'>



In [46]:

```
LE = LabelEncoder()
```

In [47]:

```
df['Churn'] = LE.fit_transform(df['Churn'])
```

In [48]:

```
df.head()
```

Out[48]:

	State	Account_length	Area_code	International_plan	Voice_mail_plan	Number_vmail_messag
0	KS	128	415	No	Yes	
1	OH	107	415	No	Yes	
2	NJ	137	415	No	No	
3	OH	84	408	Yes	No	
4	OK	75	415	Yes	No	

In [49]:

```
df['International_plan'] = LE.fit_transform(df['International_plan'])
```

In [50]:

```
df.head()
```

Out[50]:

	State	Account_length	Area_code	International_plan	Voice_mail_plan	Number_vmail_messag
0	KS	128	415	0	Yes	
1	OH	107	415	0	Yes	
2	NJ	137	415	0	No	
3	OH	84	408	1	No	
4	OK	75	415	1	No	

In [51]:

```
df['Voice_mail_plan'] = LE.fit_transform(df['Voice_mail_plan'])
```

In [52]:

```
df.head()
```

Out[52]:

	State	Account_length	Area_code	International_plan	Voice_mail_plan	Number_vmail_messag
0	KS	128	415	0	1	
1	OH	107	415	0	1	
2	NJ	137	415	0	0	
3	OH	84	408	1	0	
4	OK	75	415	1	0	

In [53]:

```
df.corr()
```

Out[53]:

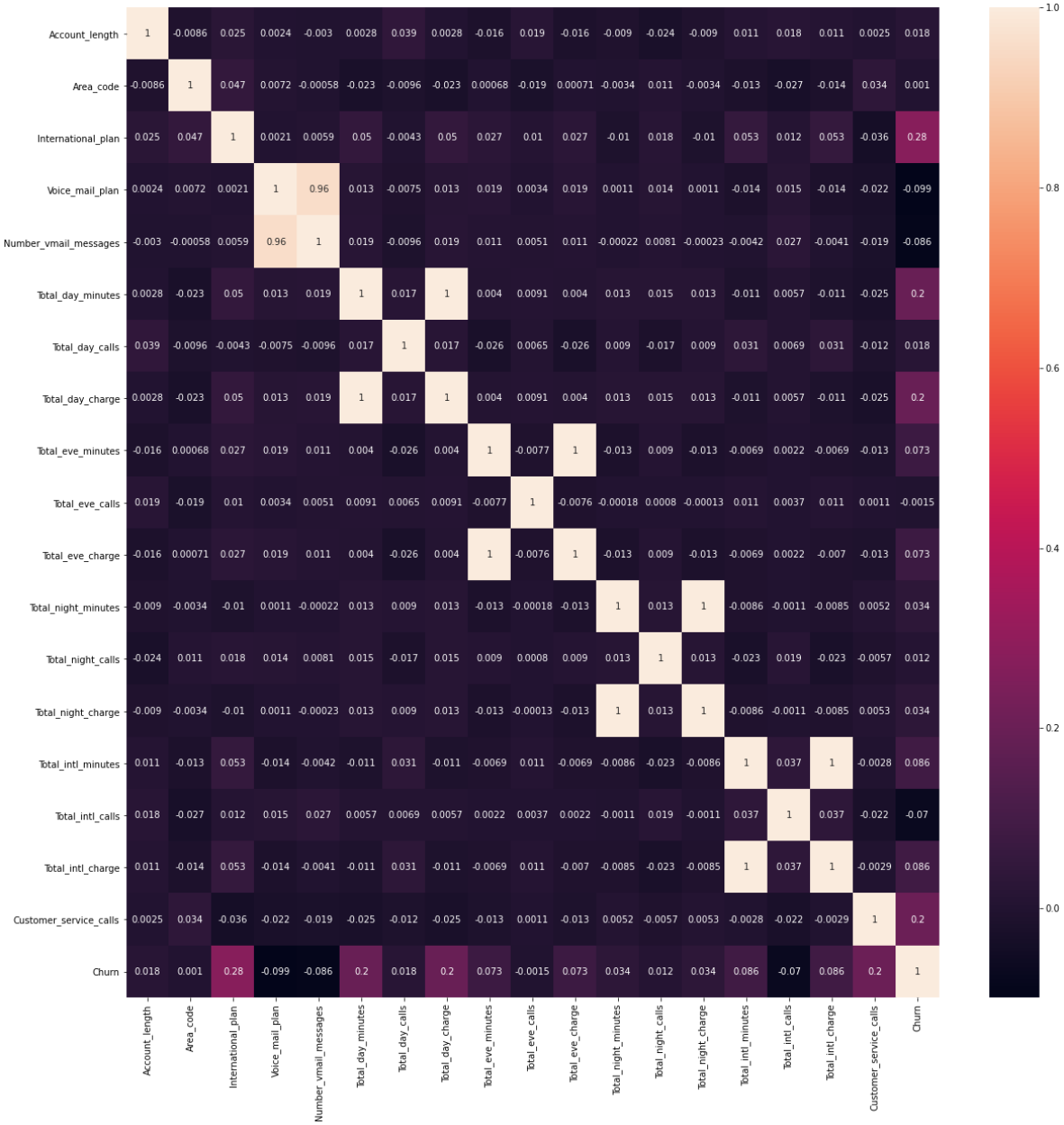
	Account_length	Area_code	International_plan	Voice_mail_plan	Numb
Account_length	1.000000	-0.008620	0.024500	0.002448	
Area_code	-0.008620	1.000000	0.047099	0.007180	
International_plan	0.024500	0.047099	1.000000	0.002131	
Voice_mail_plan	0.002448	0.007180	0.002131	1.000000	
Number_vmail_messages	-0.002996	-0.000584	0.005858	0.957159	
Total_day_minutes	0.002847	-0.023134	0.049550	0.013438	
Total_day_calls	0.038862	-0.009629	-0.004277	-0.007541	
Total_day_charge	0.002843	-0.023130	0.049555	0.013439	
Total_eve_minutes	-0.015923	0.000679	0.026616	0.019132	
Total_eve_calls	0.018552	-0.018602	0.010277	0.003404	
Total_eve_charge	-0.015909	0.000707	0.026623	0.019147	
Total_night_minutes	-0.008994	-0.003353	-0.010310	0.001065	
Total_night_calls	-0.024007	0.011455	0.018081	0.013985	
Total_night_charge	-0.008999	-0.003382	-0.010316	0.001066	
Total_intl_minutes	0.011369	-0.013418	0.053162	-0.013963	
Total_intl_calls	0.017627	-0.027423	0.011549	0.015196	
Total_intl_charge	0.011383	-0.013534	0.053037	-0.013931	
Customer_service_calls	0.002455	0.034442	-0.035955	-0.022054	
Churn	0.017728	0.001019	0.277489	-0.099291	

In [54]:

```
plt.figure(figsize=(20,20))
sb.heatmap(df.corr(),annot=True)
```

Out[54]:

<AxesSubplot:>



In [62]:

```
x = df[['International_plan','Total_day_minutes','Total_day_charge','Customer_service_calls
y = df['Churn']
```

In [64]:

```
x.head()
```

Out[64]:

	International_plan	Total_day_minutes	Total_day_charge	Customer_service_calls	Total_eve_m
0	0	265.1	45.07	1	
1	0	161.6	27.47	1	
2	0	243.4	41.38	0	
3	1	299.4	50.90	2	
4	1	166.7	28.34	3	

In [66]:

```
y.head()
```

Out[66]:

```
0    0
1    0
2    0
3    0
4    0
Name: Churn, dtype: int32
```

In [68]:

```
x_train,x_test,y_train,y_test = tts(x,y,train_size = 0.7, random_state = 20)
```

In [69]:

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(1866, 6)
(1866,)
(800, 6)
(800,)
```

In [84]:

```
LR = LogisticRegression(solver='liblinear')
LR.fit(x_train,y_train)
y_pred = LR.predict(x_test)
print("Train Score :",LR.score(x_train,y_train))
print("Test Score :",LR.score(x_test,y_test))
print("Prediction :", y_pred)
```

```
Train Score : 0.8590568060021436
Test Score : 0.85
Prediction : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0
 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
```

Multiple model in one code

In [98]:

```
cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'), x_train, y_train,
```

Out[98]:

```
array([0.86012862, 0.86012862, 0.85209003])
```

In [99]:

```
cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'), x_test, y_test,cv
```

Out[99]:

```
array([0.84269663, 0.87640449, 0.86090226])
```

In [100]:

```
cross_val_score(SVC(gamma='auto'), x_train, y_train,cv=3)
```

Out[100]:

```
array([0.85369775, 0.85369775, 0.85369775])
```

In [101]:

```
cross_val_score(SVC(gamma='auto'), x_test, y_test,cv=3)
```

Out[101]:

```
array([0.8576779 , 0.85393258, 0.85338346])
```

In [102]:

```
cross_val_score(RandomForestClassifier(n_estimators=40),x_train, y_train,cv=3)
```

Out[102]:

```
array([0.89228296, 0.90032154, 0.90353698])
```

In [103]:

```
cross_val_score(RandomForestClassifier(n_estimators=40),x_test, y_test,cv=3)
```

Out[103]:

```
array([0.87265918, 0.8988764 , 0.89097744])
```

In [107]:

```
def get_score(model, x_train, x_test, y_train, y_test):
    model.fit(x_train, y_train)
    return model.score(x_test, y_test)

print('Logistic Regration :',get_score(LogisticRegression(solver='liblinear',multi_class='o
          x_train, x_test, y_train, y_test))

print('SVM :',get_score(SVC(gamma='auto'),x_train, x_test, y_train, y_test))

print("RandomForest :",get_score(RandomForestClassifier(n_estimators=45),x_train, x_test, y
```

```
Logistic Regration : 0.85
SVM : 0.85875
RandomForest : 0.89125
```

In [87]:

```
Confusion_matrix = confusion_matrix(y_test,y_pred)
Confusion_matrix
```

Out[87]:

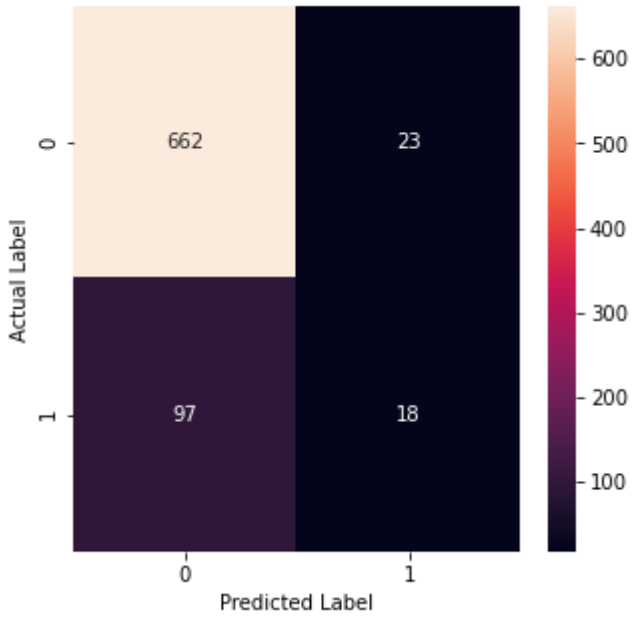
```
array([[662,  23],
       [ 97,  18]], dtype=int64)
```

In [89]:

```
plt.figure(figsize=(5,5))
sb.heatmap(Confusion_matrix,annot=True,fmt='g')
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
```

Out[89]:

```
Text(24.0, 0.5, 'Actual Label')
```



Perform Hyper parameter tuning

In [111]:

```
def get_score(model, x_train, x_test, y_train, y_test):
    model.fit(x_train, y_train)
    return model.score(x_test, y_test)

print('Logistic Regration :',get_score(LogisticRegression(solver='liblinear',multi_class='o
          x_train, x_test, y_train, y_test))

print('SVM :',get_score(SVC(gamma='auto'),x_train, x_test, y_train, y_test))

print("RandomForest :",get_score(RandomForestClassifier(n_estimators=45),x_train, x_test, y
```

```
Logistic Regration : 0.85
SVM : 0.85875
RandomForest : 0.88875
```

In [ ]: