Exploratory Data Analysis - Telecom Churn (Capstone Project 1)

Orange S.A., formerly France Télécom S.A., is a French multinational telecommunications corporation. The Orange Telecom's Churn Dataset, consists of cleaned customer activity data (features), along with a churn label specifying whether a customer canceled the subscription.

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

1. Reading Data from CSV file to Pandas DataFrame

In [2]:
```
TelecomChurnDF = pd.read_csv('./Telecom-Churn.csv')
TelecomChurnDF.head()
```

Out[2]:

	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	Tota ev cal
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	9
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	10
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	11
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	8
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	12

In [3]:
```
# Shape of the dataset
TelecomChurnDF.shape
```

Out[3]:
```
(3333, 20)
```

2. Finding if any missing values in any column

In [4]:
```
def FindNullValues(df):
```

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

FindNullValues(TelecomChurnDF)
```

```
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
```

Woo Hoo !! No missing Values

3. Finding Data types of Columns

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

```
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   3333 non-null   object
 1   Account length          3333 non-null   int64
 2   Area code               3333 non-null   int64
 3   International plan      3333 non-null   object
 4   Voice mail plan         3333 non-null   object
 5   Number vmail messages   3333 non-null   int64
 6   Total day minutes       3333 non-null   float64
 7   Total day calls         3333 non-null   int64
 8   Total day charge        3333 non-null   float64
 9   Total eve minutes       3333 non-null   float64
 10  Total eve calls         3333 non-null   int64
 11  Total eve charge        3333 non-null   float64
 12  Total night minutes     3333 non-null   float64
 13  Total night calls       3333 non-null   int64
 14  Total night charge      3333 non-null   float64
 15  Total intl minutes      3333 non-null   float64
 16  Total intl calls        3333 non-null   int64
 17  Total intl charge       3333 non-null   float64
 18  Customer service calls  3333 non-null   int64
 19  Churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
```

Excpet the columns State, International plan, Voice mail plan and Churn remaining all are Numerical (either int or float)

4. Statistical Description of Data (For Numerical Columns)

In [6]:
```
TelecomChurnDF.describe()
```

Out[6]:

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total e minut
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.0000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.9803
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.7138
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.6000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.4000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.3000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.7000

5. Dividing DataFrame into two parts(Churn = True and Churn = False), so we can compare the user behavior who churned with non churned

In [7]:
```
# Creating a new Data Frame with only Churned Users
ChurnedDF = TelecomChurnDF[TelecomChurnDF['Churn'] == True]
ChurnedDF
```

Out[7]:

	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
10	IN	65	415	No	No	0	129.1	137	21.95	228.5
15	NY	161	415	No	No	0	332.9	67	56.59	317.8
21	CO	77	408	No	No	0	62.4	89	10.61	169.9
33	AZ	12	408	No	No	0	249.6	118	42.43	252.4
41	MD	135	408	Yes	Yes	41	173.1	85	29.43	203.9
...	...	...	...	...	...	...	...	...	...	...
3301	CA	84	415	No	No	0	280.0	113	47.60	202.2
3304	IL	71	510	Yes	No	0	186.1	114	31.64	198.6
3320	GA	122	510	Yes	No	0	140.0	101	23.80	196.4
3322	MD	62	408	No	No	0	321.1	105	54.59	265.5
3323	IN	117	415	No	No	0	118.4	126	20.13	249.3

483 rows × 20 columns

In [8]:
```
# Creating a new Data Frame with only Non Churned Users
```

```
NonChurnedDF = TelecomChurnDF[TelecomChurnDF['Churn'] == False]
NonChurnedDF
```

Out[8]:

	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
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3
...	...	...	...	...	...	...	...	...	...	...
3328	AZ	192	415	No	Yes	36	156.2	77	26.55	215.5
3329	WV	68	415	No	No	0	231.1	57	39.29	153.4
3330	RI	28	510	No	No	0	180.8	109	30.74	288.8
3331	CT	184	510	Yes	No	0	213.8	105	36.35	159.6
3332	TN	74	415	No	Yes	25	234.4	113	39.85	265.9

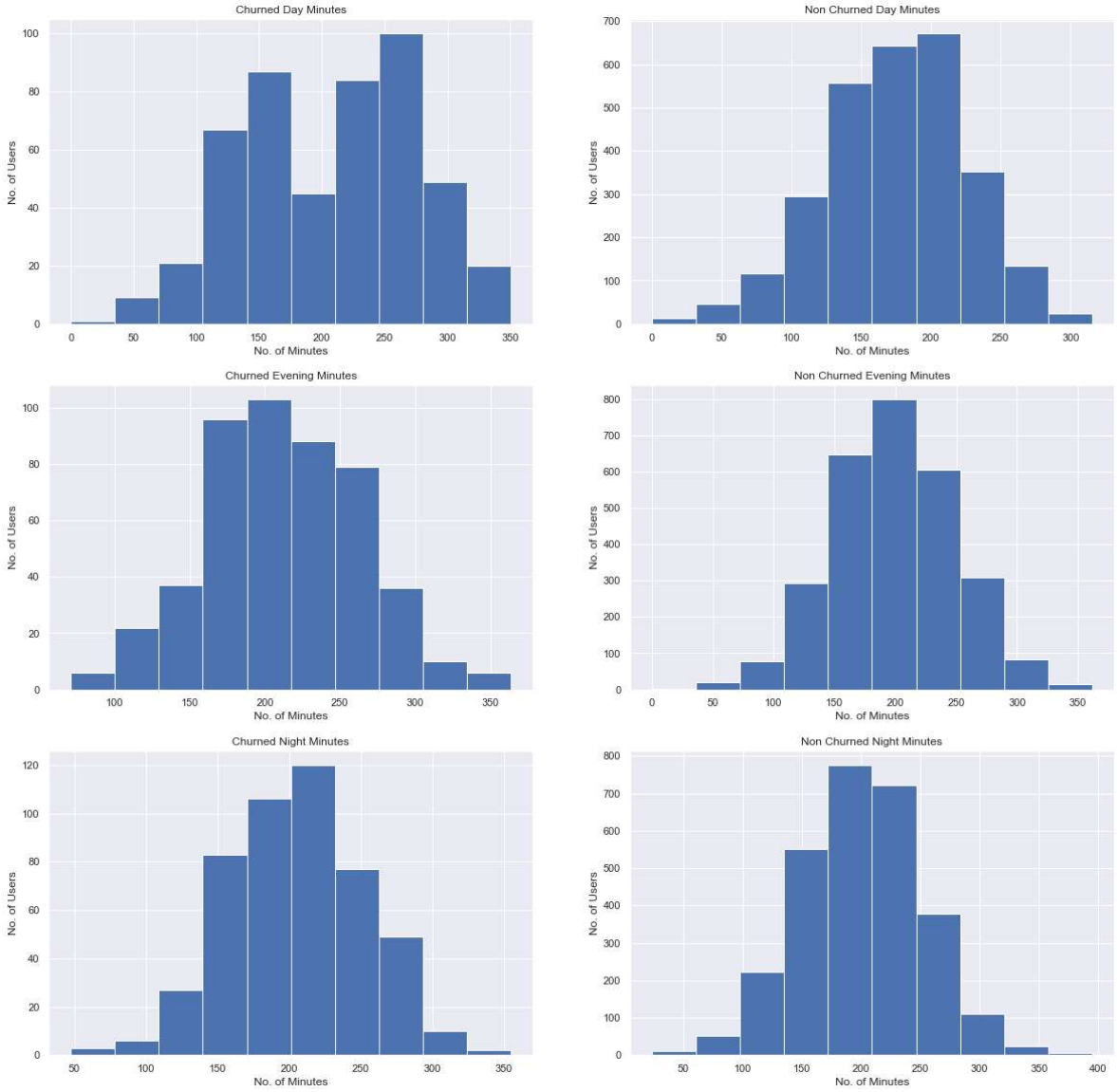
2850 rows × 20 columns

Out of Total 3333 users in data set 483 have churned and 2850 didn't

6. Frequency Comparison of Day, Evening and Night minutes in Both DataFrames

In [9]:

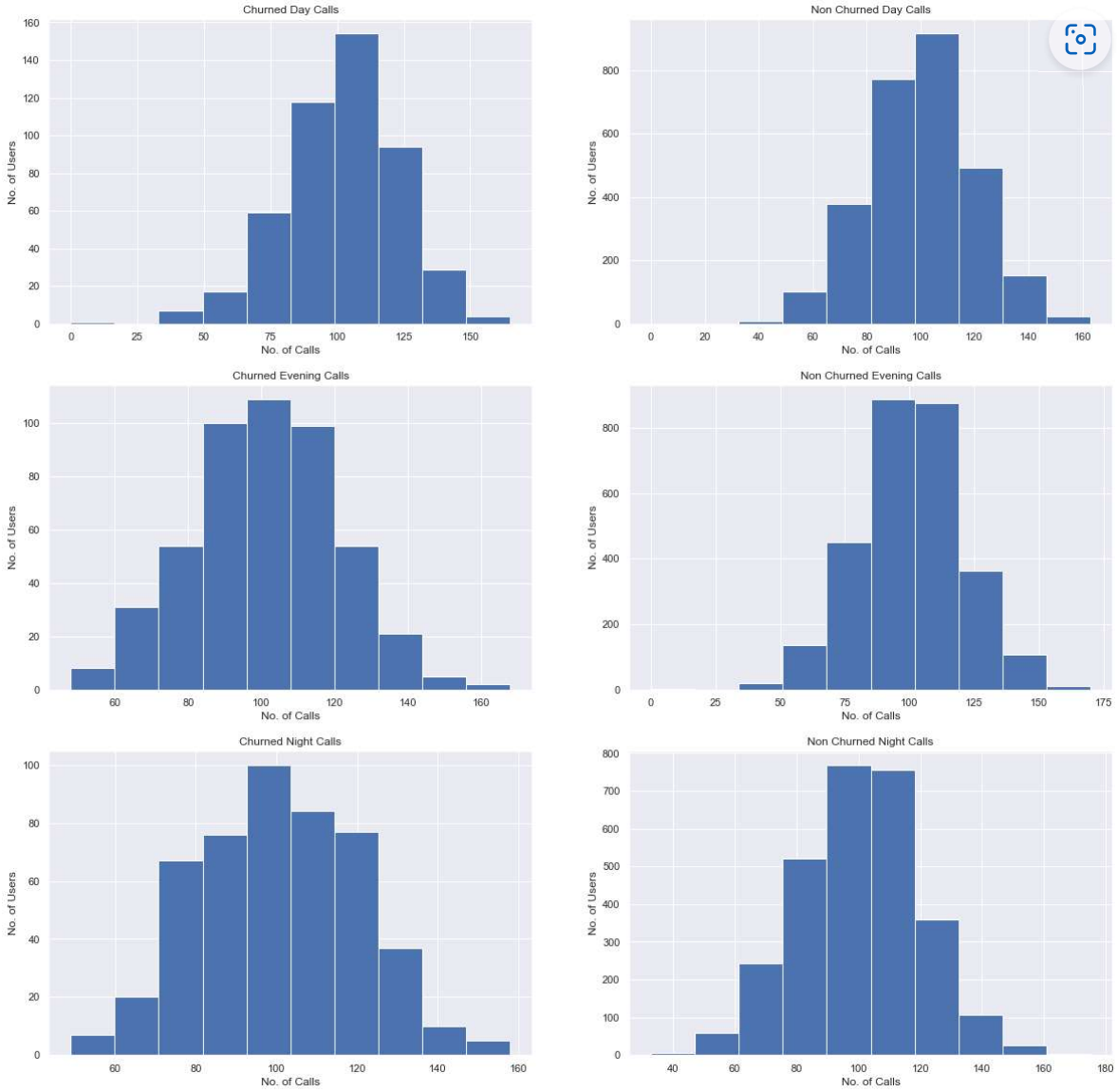
```
fig, axs = plt.subplots(3,2, figsize=(20,20))
axs[0, 0].hist(ChurnedDF['Total day minutes'])
axs[0, 0].set_title('Churned Day Minutes')
axs[0, 1].hist(NonChurnedDF['Total day minutes'])
axs[0, 1].set_title('Non Churned Day Minutes')
axs[1, 0].hist(ChurnedDF['Total eve minutes'])
axs[1, 0].set_title('Churned Evening Minutes')
axs[1, 1].hist(NonChurnedDF['Total eve minutes'])
axs[1, 1].set_title('Non Churned Evening Minutes')
axs[2, 0].hist(ChurnedDF['Total night minutes'])
axs[2, 0].set_title('Churned Night Minutes')
axs[2, 1].hist(NonChurnedDF['Total night minutes'])
axs[2, 1].set_title('Non Churned Night Minutes')

for ax in axs.flat:
    ax.set(xlabel='No. of Minutes', ylabel='No. of Users')
```



7. Frequency Comparison of Day, Evening and Night Calls in Both DataFrames

In [10]:

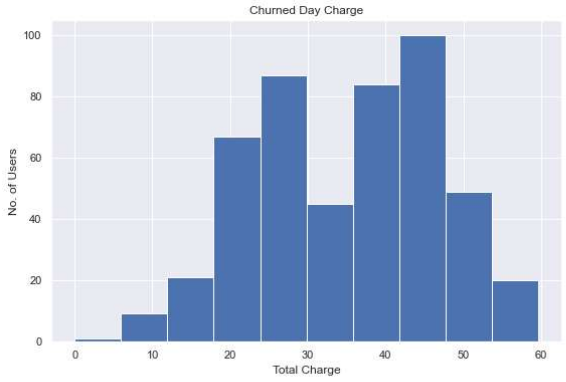
```
fig, axs = plt.subplots(3,2, figsize=(20,20))
axs[0, 0].hist(ChurnedDF['Total day calls'])
axs[0, 0].set_title('Churned Day Calls')
axs[0, 1].hist(NonChurnedDF['Total day calls'])
axs[0, 1].set_title('Non Churned Day Calls')
axs[1, 0].hist(ChurnedDF['Total eve calls'])
axs[1, 0].set_title('Churned Evening Calls')
axs[1, 1].hist(NonChurnedDF['Total eve calls'])
axs[1, 1].set_title('Non Churned Evening Calls')
axs[2, 0].hist(ChurnedDF['Total night calls'])
axs[2, 0].set_title('Churned Night Calls')
axs[2, 1].hist(NonChurnedDF['Total night calls'])
axs[2, 1].set_title('Non Churned Night Calls')

for ax in axs.flat:
    ax.set(xlabel='No. of Calls', ylabel='No. of Users')
```

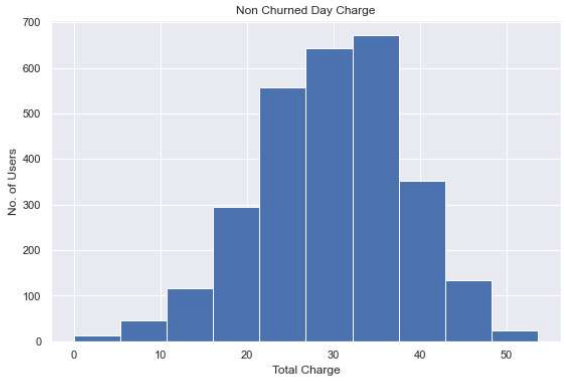


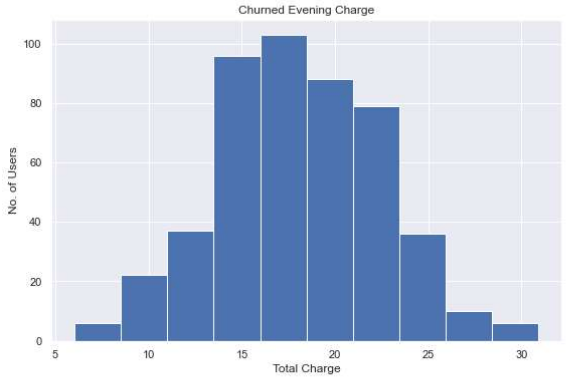
8. Frequency Comparison of Day, Evening and Night Charges in Both DataFrames

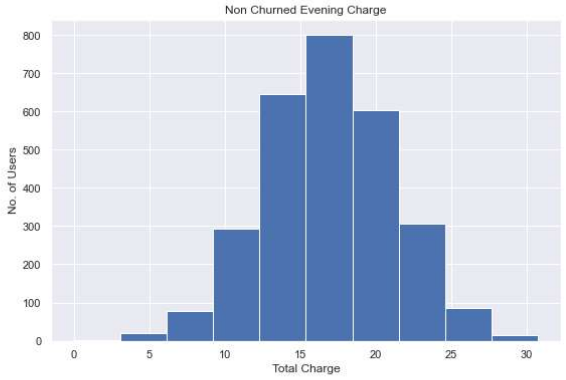
In [11]:

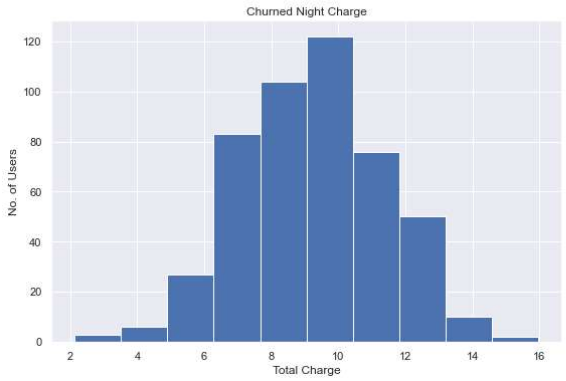
```
fig, axs = plt.subplots(3,2, figsize=(20,20))
axs[0, 0].hist(ChurnedDF['Total day charge'])
axs[0, 0].set_title('Churned Day Charge')
axs[0, 1].hist(NonChurnedDF['Total day charge'])
axs[0, 1].set_title('Non Churned Day Charge')
axs[1, 0].hist(ChurnedDF['Total eve charge'])
axs[1, 0].set_title('Churned Evening Charge')
axs[1, 1].hist(NonChurnedDF['Total eve charge'])
axs[1, 1].set_title('Non Churned Evening Charge')
axs[2, 0].hist(ChurnedDF['Total night charge'])
axs[2, 0].set_title('Churned Night Charge')
axs[2, 1].hist(NonChurnedDF['Total night charge'])
axs[2, 1].set_title('Non Churned Night Charge')

for ax in axs.flat:
    ax.set(xlabel='Total Charge', ylabel='No. of Users')
```

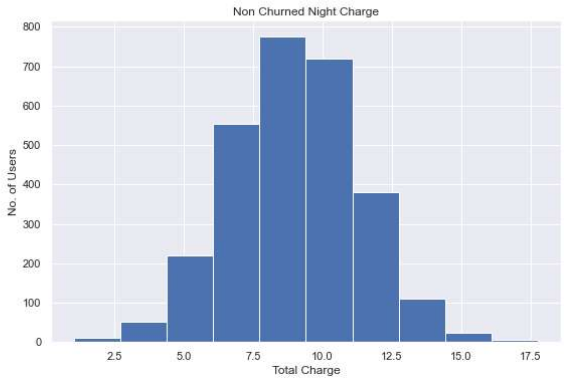












9. Finding Correlation among Morning,Evening and Night call minutes attributes and plotting Heat Map

In [12]:
```
# Extracting Total minutes of day, evening and night and finding correlc
TelecomChurnCallMinutesDF = TelecomChurnDF[['Total day minutes','Total eve minute
TelecomChurnCallMinutesCorrData = TelecomChurnCallMinutesDF.corr()
sns.heatmap(TelecomChurnCallMinutesCorrData, vmin=-1, vmax=1)
TelecomChurnCallMinutesCorrData
```

Out[12]:

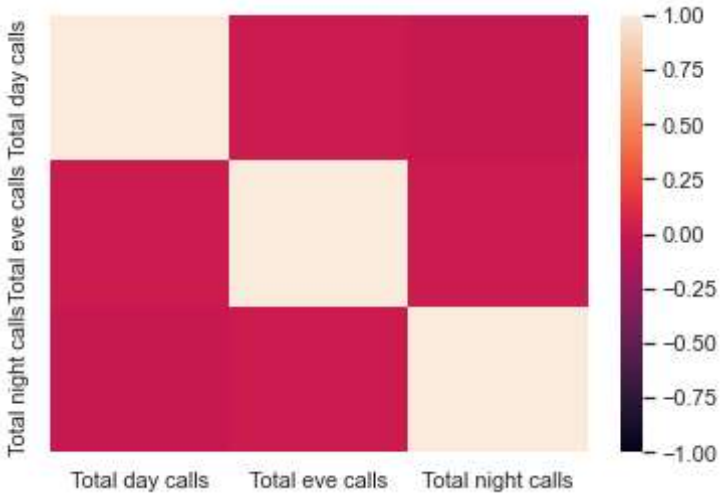
	Total day minutes	Total eve minutes	Total night minutes
Total day minutes	1.000000	0.007043	0.004323
Total eve minutes	0.007043	1.000000	-0.012584
Total night minutes	0.004323	-0.012584	1.000000

10. Finding Correlation among Morning,Evening and Night calls attributes and plotting Heat Map

In [13]:
```
# Extracting Total calls of day, evening and night and finding correlati
TelecomChurnCallsDF = TelecomChurnDF[['Total day calls','Total eve calls','Total
TelecomChurnCallsCorrData = TelecomChurnCallsDF.corr()
sns.heatmap(TelecomChurnCallsCorrData, vmin=-1, vmax=1)
TelecomChurnCallsCorrData
```

Out[13]:

	Total day calls	Total eve calls	Total night calls
Total day calls	1.000000	0.006462	-0.019557
Total eve calls	0.006462	1.000000	0.007710
Total night calls	-0.019557	0.007710	1.000000



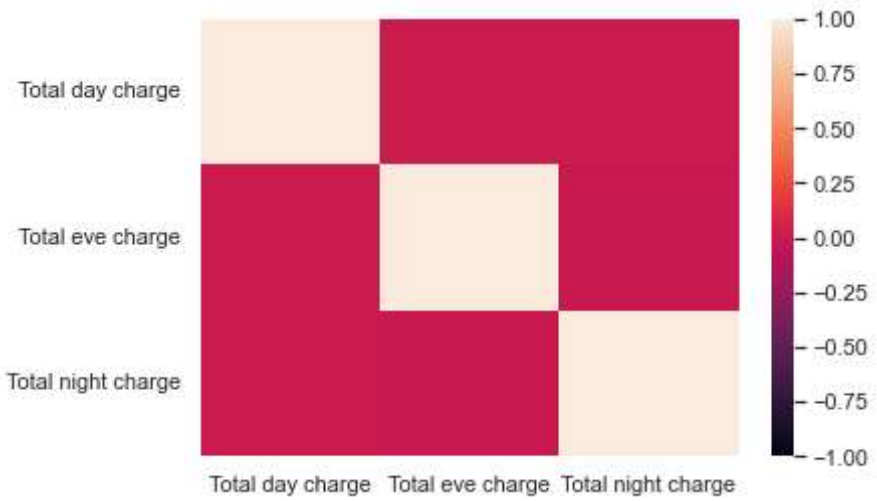
11. Finding Correlation among Morning,Evening and Night call charges attributes and plotting Heat Map

In [14]:
```
# Extracting Total charges of day, evening and night and finding correla
TelecomChurnCallChargesDF = TelecomChurnDF[['Total day charge','Total eve charge'
TelecomChurnCallChargesCorrData = TelecomChurnCallChargesDF.corr()
sns.heatmap(TelecomChurnCallChargesCorrData, vmin=-1, vmax=1)
TelecomChurnCallChargesCorrData
```

Out[14]:

	Total day charge	Total eve charge	Total night charge

	Total day charge	Total eve charge	Total night charge
Total day charge	1.000000	0.007036	0.004301
Total eve charge	0.007036	1.000000	-0.012601
Total night charge	0.004301	-0.012601	1.000000



Important insight from above 3 Correlation Data is that duration of the time either day, evening or night has almost 0 correlation among themselves

11. Seggregating Data based on Area Code for Analysis

In [15]:
```
# Finding  all unique  Area  Codes
uniqueAreaCodes = list(TelecomChurnDF['Area code'].unique())
print("Area Codes in Data are ", uniqueAreaCodes)
```

```
Area Codes in Data are  [415, 408, 510]
```

In [16]:
```
DictOfTelecomChurnDFBasedOnAreaCodes = {}
for i in uniqueAreaCodes:
    DictOfTelecomChurnDFBasedOnAreaCodes[i] = TelecomChurnDF[TelecomChurnDF['Area
```

So we have 3 area codes 415,408 and 510

In [17]:
```
print('-------------- Area Code',415,'--------------')
DictOfTelecomChurnDFBasedOnAreaCodes[415]
```

```
-------------- Area Code 415 --------------
```

Out[17]:

	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
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3
7	MO	147	415	Yes	No	0	157.0	79	26.69	103.1

	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
...	...	...	...	...	...	...	...	...	...	...
3326	OH	96	415	No	No	0	106.6	128	18.12	284.8
3327	SC	79	415	No	No	0	134.7	98	22.90	189.7
3328	AZ	192	415	No	Yes	36	156.2	77	26.55	215.5
3329	WV	68	415	No	No	0	231.1	57	39.29	153.4
3332	TN	74	415	No	Yes	25	234.4	113	39.85	265.9

1655 rows × 20 columns

In [18]:
```
print('-------------- Area Code',408,'--------------')
DictOfTelecomChurnDFBasedOnAreaCodes[408]
```

```
-------------- Area Code 408 --------------
```

Out[18]:

	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
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9
8	LA	117	408	No	No	0	184.5	97	31.37	351.6
12	IA	168	408	No	No	0	128.8	96	21.90	104.9
16	ID	85	408	No	Yes	27	196.4	139	33.39	280.9
21	CO	77	408	No	No	0	62.4	89	10.61	169.9
...	...	...	...	...	...	...	...	...	...	...
3309	VT	100	408	Yes	No	0	219.4	112	37.30	225.7
3312	SC	181	408	No	No	0	229.9	130	39.08	144.4
3313	ID	127	408	No	No	0	102.8	128	17.48	143.7
3322	MD	62	408	No	No	0	321.1	105	54.59	265.5
3325	OH	78	408	No	No	0	193.4	99	32.88	116.9

838 rows × 20 columns

In [19]:
```
print('-------------- Area Code',510,'--------------')
DictOfTelecomChurnDFBasedOnAreaCodes[510]
```

```
-------------- Area Code 510 --------------
```

Out[19]:

	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
5	AL	118	510	Yes	No	0	223.4	98	37.98	220.6
6	MA	121	510	No	Yes	24	218.2	88	37.09	348.5

	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
13	MT	95	510	No	No	0	156.6	88	26.62	247.6
17	VT	93	510	No	No	0	190.7	114	32.42	218.2
18	VA	76	510	No	Yes	33	189.7	66	32.25	212.8
...	...	...	...	...	...	...	...	...	...	...
3304	IL	71	510	Yes	No	0	186.1	114	31.64	198.6
3316	MS	103	510	No	Yes	29	164.1	111	27.90	219.1
3320	GA	122	510	Yes	No	0	140.0	101	23.80	196.4
3330	RI	28	510	No	No	0	180.8	109	30.74	288.8
3331	CT	184	510	Yes	No	0	213.8	105	36.35	159.6

840 rows × 20 columns

Comaprison and Analysis of Churn Vs Non Churn Area Code wise

In [20]:

```
AreaCodeWiseChurnAndNonChurn = {}
AreaCodeWiseChurnAndNonChurn['Churn'] = {} # Nested Dictionary
AreaCodeWiseChurnAndNonChurn['NonChurn'] = {} # Nested Dictionary
for i in uniqueAreaCodes:
    areaCodeData = DictOfTelecomChurnDFBasedOnAreaCodes[i]
    # Adding Churn Data to Churn Key (nested Dict)
    AreaCodeWiseChurnAndNonChurn['Churn'][i] = areaCodeData[areaCodeData['Churn']
    # Adding Non Churn Data to NonChurn Key (nested Dict)
    AreaCodeWiseChurnAndNonChurn['NonChurn'][i] = areaCodeData[areaCodeData['Chur
```

Explaining The Nested Dictionary Format that we formed Above

AreaCodeWiseChurnAndNonChurn = {

> 'Churn': {

```
            415: DataFrame of 415 Area Code with Churn equal to
    True,
            408: DataFrame of 408 Area Code with Churn equal to
    True,
            510: DataFrame of 510 Area Code with Churn equal to True
        },
        'NonChurn': {
            415: DataFrame of 415 Area Code with Churn equal to
    False,
            408: DataFrame of 408 Area Code with Churn equal to
    False,
            510: DataFrame of 510 Area Code with Churn equal to False
        }

}
```

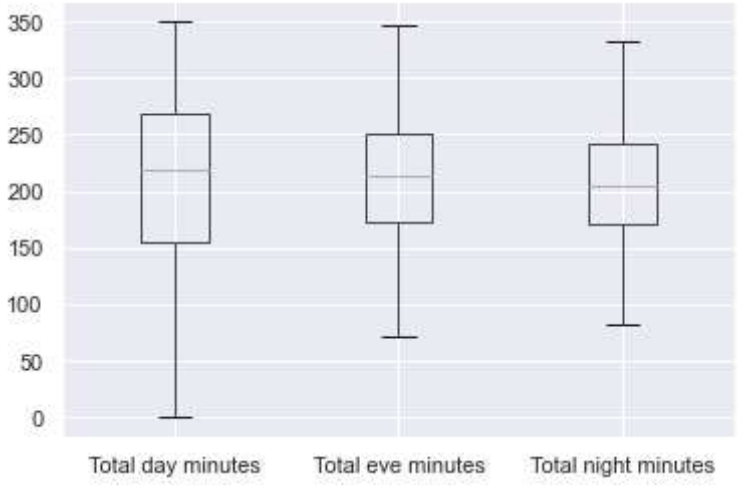
12. Data Comparison of Day, Evening and Night minutes of Both Churn and Non Churn with in Area Codes

12.1 Area Code - 415

In [21]:
```
print('Area Code 415 Churned Users Call Minutes')
AreaCodeWiseChurnAndNonChurn['Churn'][415].boxplot(['Total day minutes', 'Total e
```

```
Area Code 415 Churned Users Call Minutes
```

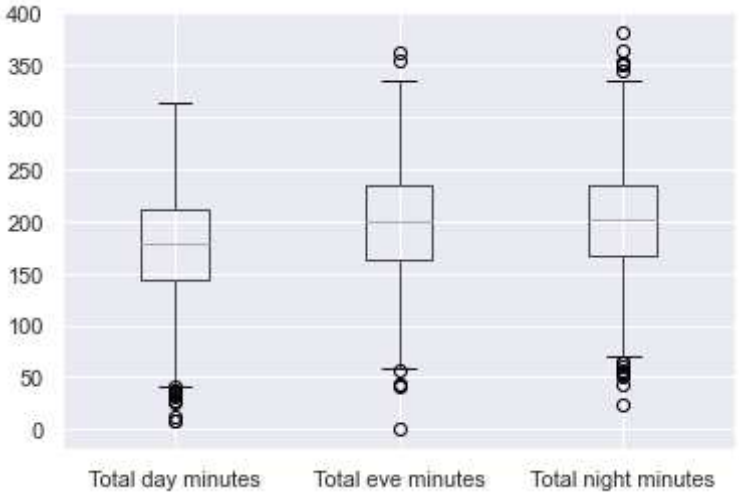
Out[21]:



In [22]:
```
print('Area Code 415 Non Churned Users Call Minutes')
AreaCodeWiseChurnAndNonChurn['NonChurn'][415].boxplot(['Total day minutes', 'Tota
```

```
Area Code 415 Non Churned Users Call Minutes
```

Out[22]:

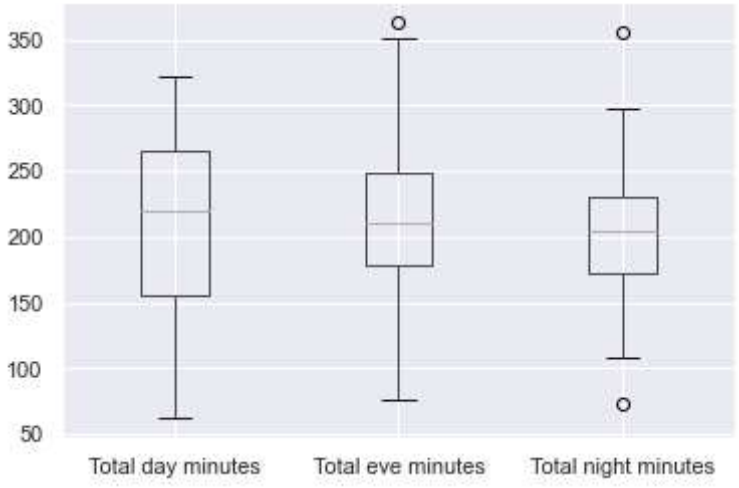


12.2 Area Code - 408

In [23]:
```
print('Area Code 408 Churned Users Call Minutes')
AreaCodeWiseChurnAndNonChurn['Churn'][408].boxplot(['Total day minutes', 'Total e
```

```
Area Code 408 Churned Users Call Minutes
```

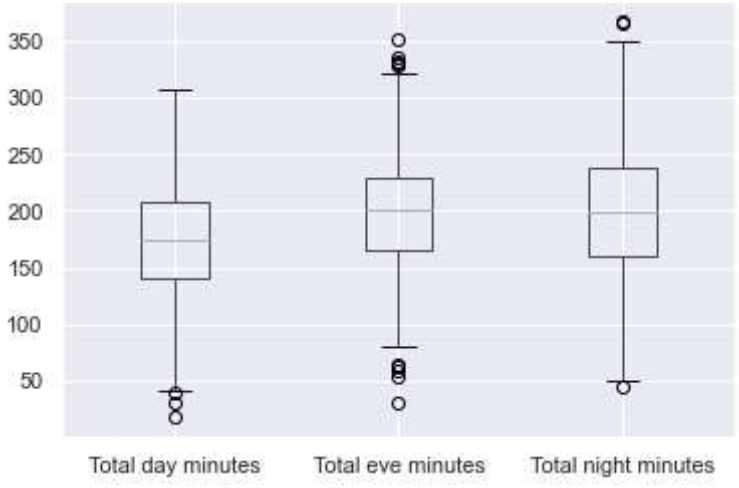
Out[23]:



In [24]:
```
print('Area Code 408 Non Churned Users Call Minutes')
AreaCodeWiseChurnAndNonChurn['NonChurn'][408].boxplot(['Total day minutes', 'Tota
```

```
Area Code 408 Non Churned Users Call Minutes
```

Out[24]:

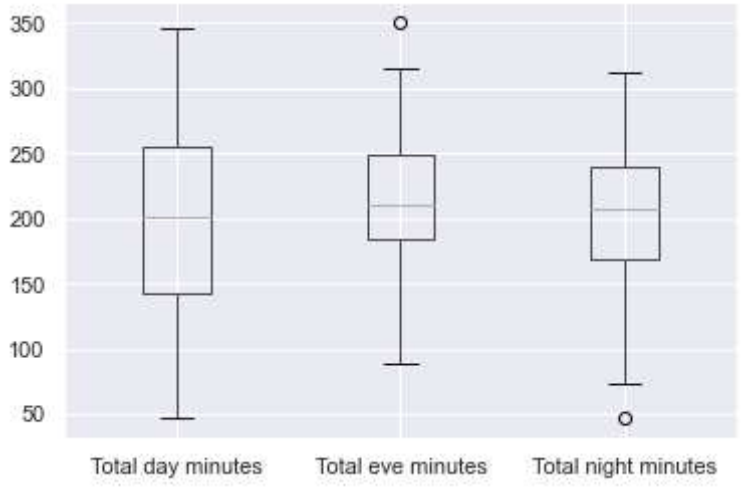


12.3 Area Code - 510

In [25]:
```
print('Area Code 510 Churned Users Call Minutes')
AreaCodeWiseChurnAndNonChurn['Churn'][510].boxplot(['Total day minutes', 'Total e
```

```
Area Code 510 Churned Users Call Minutes
```

Out[25]:

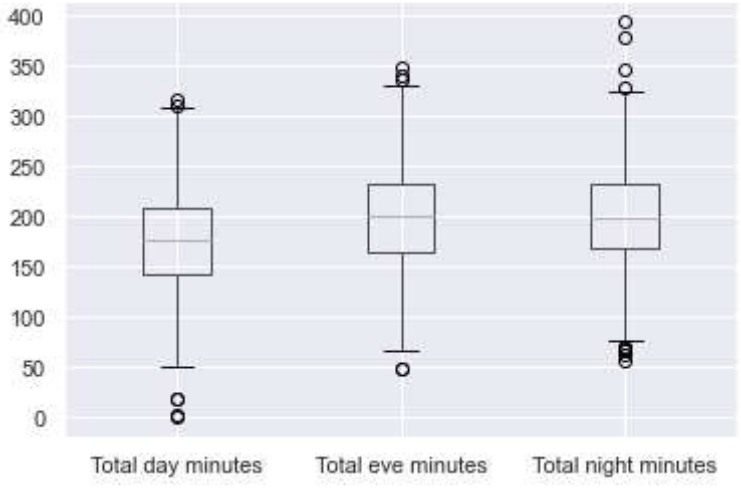


In [26]:

```
print('Area Code 510 Non Churned Users Call Minutes')
AreaCodeWiseChurnAndNonChurn['NonChurn'][510].boxplot(['Total day minutes', 'Tota
```

```
Area Code 510 Non Churned Users Call Minutes
```

Out[26]:



In all 3 Area Codes, the median of day and evening minutes is more in Churned users compared to Non Churned users, and for night it's comparable(Churned is slightly high)

And also if we see the IQR(Inter Quartile Range) is spread more for Churned Users

13. State Based Churn and Non Churned User Data Analysis

13.1 Counting Number of Users Churned and Non Churned State wise

In [27]:

```
# Finding  all unique  States
uniqueStates = list(TelecomChurnDF['State'].unique())
print("Unique States in Data are ", uniqueStates)
print('---------------------------------')
#  Counting  number  of  occurences for  each  state
StateWiseChurnedAndNonChurnedCounts = TelecomChurnDF.groupby(['State', 'Churn'])[
print("State wise Churn and Non Churn Count")
StateWiseChurnedAndNonChurnedCounts
```

```
Unique States in Data are  ['KS', 'OH', 'NJ', 'OK', 'AL', 'MA', 'MO', 'LA', 'WV',
'IN', 'RI', 'IA', 'MT', 'NY', 'ID', 'VT', 'VA', 'TX', 'FL', 'CO', 'AZ', 'SC', 'N
E', 'WY', 'HI', 'IL', 'NH', 'GA', 'AK', 'MD', 'AR', 'WI', 'OR', 'MI', 'DE', 'UT',
'CA', 'MN', 'SD', 'NC', 'WA', 'NM', 'NV', 'DC', 'KY', 'ME', 'MS', 'TN', 'PA', 'C
T', 'ND']
---------------------------------
State wise Churn and Non Churn Count
```

Out[27]:

```
State  Churn
AK     False    49
       True      3
AL     False    72
       True      8
AR     False    44
                ..
WI     True      7
WV     False    96
       True     10
WY     False    68
       True      9
Name: Churn, Length: 102, dtype: int64
```

13.2 Finding out states with high Churn rate (> 20%)

In [28]:

```
StateChurnedPercentage = {}
for i in uniqueStates:
    StateChurnedPercentage[i] = (StateWiseChurnedAndNonChurnedCounts.loc[i,True])
print('State wise Churn Rate')
print(StateChurnedPercentage)
print('----------------------')
print('States with more than 20% churn rate')
StateChurnedPercentageMoreThan20 = {k:v for k,v in StateChurnedPercentage.items()
print(StateChurnedPercentageMoreThan20)
```

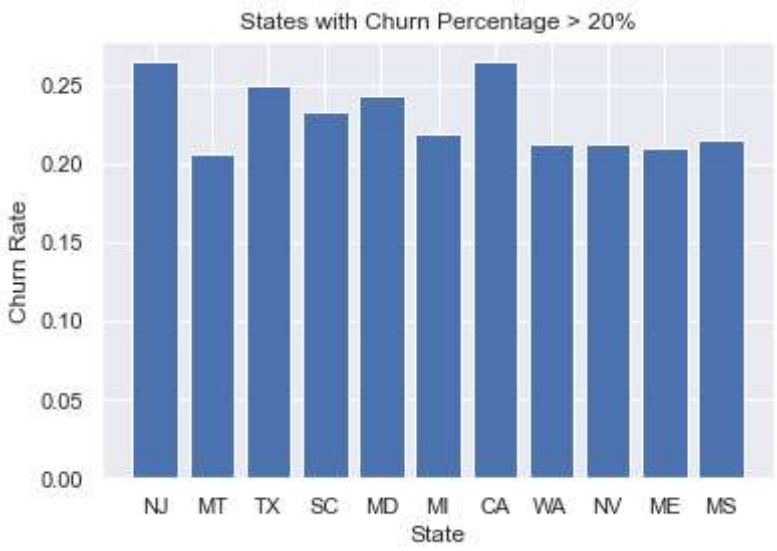
```
State wise Churn Rate
{'KS': 0.18571428571428572, 'OH': 0.1282051282051282, 'NJ': 0.2647058823529412,
'OK': 0.14754098360655737, 'AL': 0.1, 'MA': 0.16923076923076924, 'MO': 0.11111111
11111111, 'LA': 0.0784313725490196, 'WV': 0.09433962264150944, 'IN': 0.1267605633
802817, 'RI': 0.09230769230769231, 'IA': 0.06818181818181818, 'MT': 0.20588235294
117646, 'NY': 0.18072289156626506, 'ID': 0.1232876712328767, 'VT': 0.109589041095
8904, 'VA': 0.06493506493506493, 'TX': 0.25, 'FL': 0.12698412698412698, 'CO': 0.1
3636363636363635, 'AZ': 0.0625, 'SC': 0.23333333333333334, 'NE': 0.08196721311475
409, 'WY': 0.11688311688311688, 'HI': 0.05660377358490566, 'IL': 0.08620689655172
414, 'NH': 0.16071428571428573, 'GA': 0.14814814814814814, 'AK': 0.05769230769230
7696, 'MD': 0.24285714285714285, 'AR': 0.2, 'WI': 0.08974358974358974, 'OR': 0.14
102564102564102, 'MI': 0.2191780821917808, 'DE': 0.14754098360655737, 'UT': 0.138
8888888888889, 'CA': 0.2647058823529412, 'MN': 0.17857142857142858, 'SD': 0.13333
333333333333, 'NC': 0.16176470588235295, 'WA': 0.21212121212121213, 'NM': 0.09677
41935483871, 'NV': 0.21212121212121213, 'DC': 0.09259259259259259, 'KY': 0.135593
22033898305, 'ME': 0.20967741935483872, 'MS': 0.2153846153846154, 'TN': 0.0943396
2264150944, 'PA': 0.17777777777777778, 'CT': 0.16216216216216217, 'ND': 0.0967741
935483871}
----------------------
States with more than 20% churn rate
{'NJ': 0.2647058823529412, 'MT': 0.20588235294117646, 'TX': 0.25, 'SC': 0.2333333
3333333334, 'MD': 0.24285714285714285, 'MI': 0.2191780821917808, 'CA': 0.26470588
23529412, 'WA': 0.21212121212121213, 'NV': 0.21212121212121213, 'ME': 0.209677419
35483872, 'MS': 0.2153846153846154}
```

In [29]:

```
plt.bar(*zip(*StateChurnedPercentageMoreThan20.items()))
plt.title('States with Churn Percentage > 20%')
```

```
plt.xlabel('State')
plt.ylabel('Churn Rate')
```

Out[29]:
```
Text(0, 0.5, 'Churn Rate')
```



Out of 51 states 11 states have more than 20% Churned Users

States NJ and CA have highest Churn rate of 26.47%

14. International Plan Based Churn and Non Churned User Data Analysis

14.1 Counting Number of Users Churned and Non Churned Based on wheather opted International Plan or not

In [30]:
```
ChurnedAndNonChurnedCountsBasedOnInternationalPlan = TelecomChurnDF.groupby(['Int
print("Churn and Non Churn Count Based on Opting for International Plan")
ChurnedAndNonChurnedCountsBasedOnInternationalPlan
```

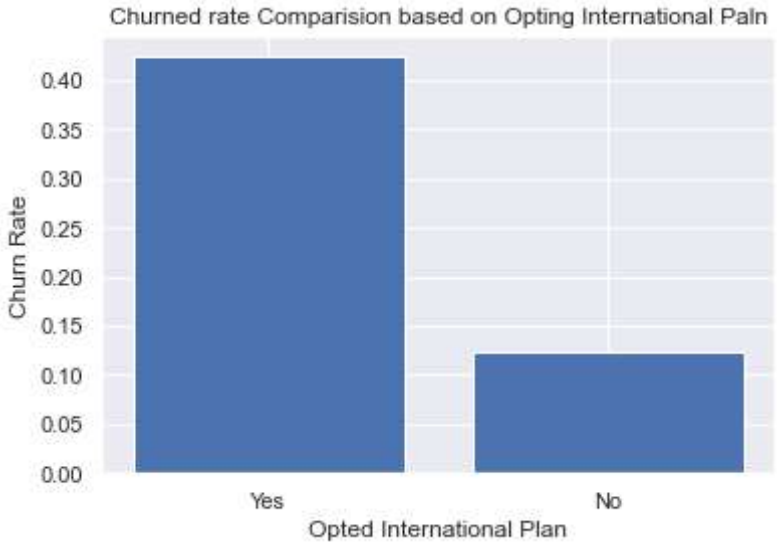
```
Churn and Non Churn Count Based on Opting for International Plan
```

Out[30]:
```
International plan  Churn
No                  False    2664
                    True      346
Yes                 False     186
                    True      137
Name: Churn, dtype: int64
```

14.2 Comparison plot between who opted International Plan and Not

In [31]:
```
ChurnedPercentBasedOnInternationalPlan = {}
ChurnedPercentBasedOnInternationalPlan['Yes'] = (ChurnedAndNonChurnedCountsBasedO
ChurnedPercentBasedOnInternationalPlan['No'] = (ChurnedAndNonChurnedCountsBasedOn
plt.bar(*zip(*ChurnedPercentBasedOnInternationalPlan.items()))
plt.title('Churned rate Comparision based on Opting International Paln')
plt.xlabel('Opted International Plan')
plt.ylabel('Churn Rate')
```

Out[31]:
```
Text(0, 0.5, 'Churn Rate')
```



We can clearly see users who opted for International Plan has churned almost 4 times(more than 40%) when compared to users who didn't opt

15.Voice Mail Plan Based Churn and Non Churned User Data Analysis

15.1 Counting Number of Users Churned and Non Churned Based on wheather opted Voice mail plan or not

In [32]:
```
ChurnedAndNonChurnedCountsBasedOnVoiceMailPlan = TelecomChurnDF.groupby(['Voice m
print("Churn and Non Churn Count Based on Opting for Voice mail Plan")
ChurnedAndNonChurnedCountsBasedOnVoiceMailPlan
```

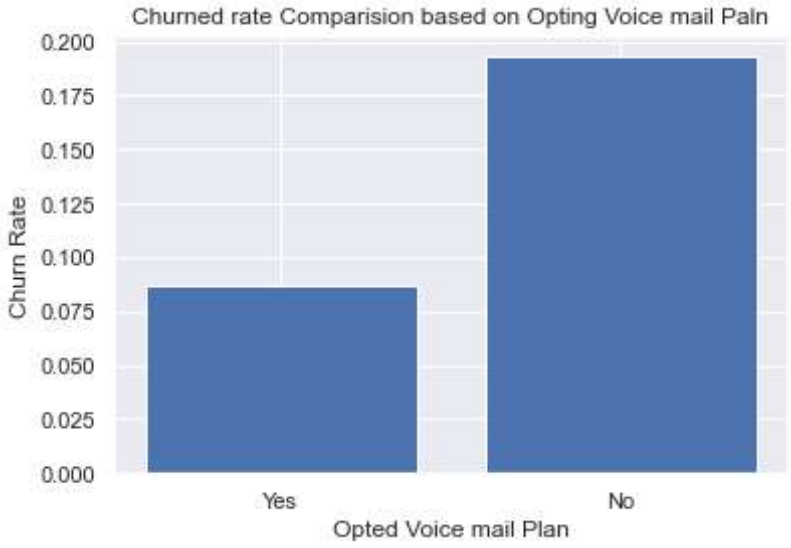
```
Churn and Non Churn Count Based on Opting for Voice mail Plan
```

Out[32]:
```
Voice mail plan  Churn
No               False    2008
                 True      403
Yes              False     842
                 True       80
Name: Churn, dtype: int64
```

15.2 Comparison plot between who opted Voice Plan and Not

In [33]:
```
ChurnedPercentBasedOnVoiceMailPlan = {}
ChurnedPercentBasedOnVoiceMailPlan['Yes'] = (ChurnedAndNonChurnedCountsBasedOnVoi
ChurnedPercentBasedOnVoiceMailPlan['No'] = (ChurnedAndNonChurnedCountsBasedOnVoic
plt.bar(*zip(*ChurnedPercentBasedOnVoiceMailPlan.items()))
plt.title('Churned rate Comparision based on Opting Voice mail Paln')
plt.xlabel('Opted Voice mail Plan')
plt.ylabel('Churn Rate')
```

Out[33]:
```
Text(0, 0.5, 'Churn Rate')
```



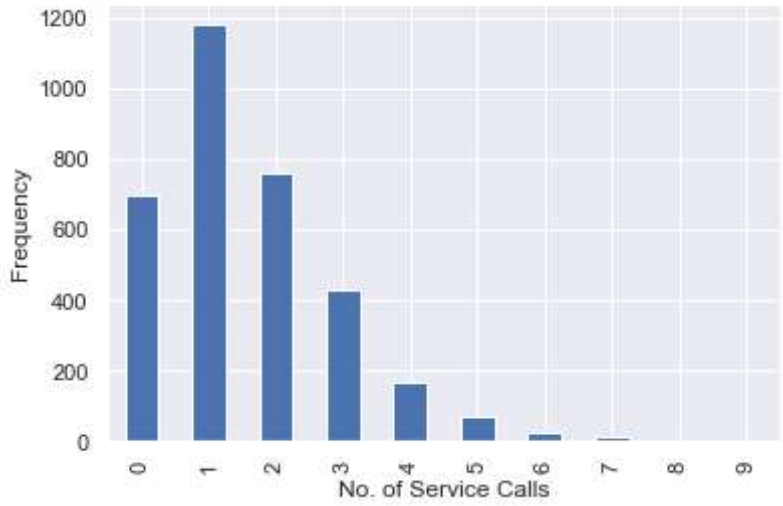
This gives conclusion that voice mail has not impacted in high churn rate but in a way users who opted for Voice mail plan are satisfied with the service

16. Analysis based on Number of Customer Service Calls made

In [34]:
```
print('Percentage Distribution for number of Service Calls made')
print('-----------------------------------------------')
print(TelecomChurnDF['Customer service calls'].value_counts(normalize=True).sort_
TelecomChurnDF['Customer service calls'].value_counts().sort_index().plot(kind='b
```

```
Percentage Distribution for number of Service Calls made
-----------------------------------------------
0    0.209121
1    0.354335
2    0.227723
3    0.128713
4    0.049805
5    0.019802
6    0.006601
7    0.002700
8    0.000600
9    0.000600
Name: Customer service calls, dtype: float64
```

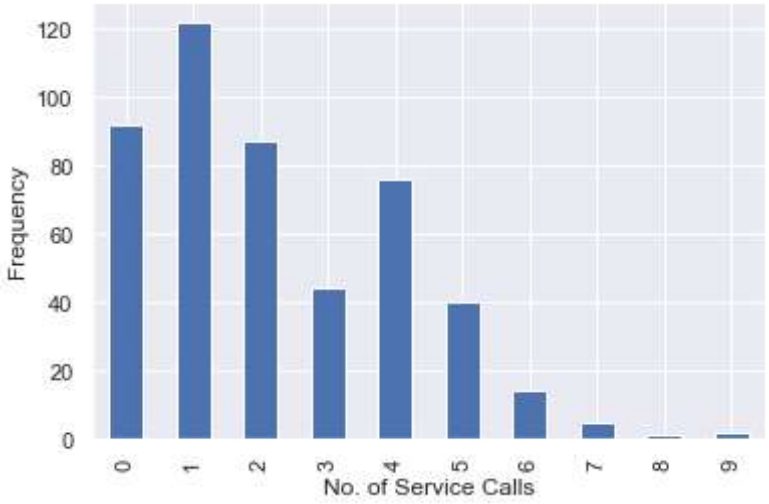
Out[34]:



In [35]:
```
print('Percentage Distribution of Churned Users for number of Service Calls made'
print('-------------------------------------------')
print(ChurnedDF['Customer service calls'].value_counts(normalize=True).sort_index
ChurnedDF['Customer service calls'].value_counts().sort_index().plot(kind='bar',
```

```
Percentage Distribution of Churned Users for number of Service Calls made
-------------------------------------------
0    0.190476
1    0.252588
2    0.180124
3    0.091097
4    0.157350
5    0.082816
6    0.028986
7    0.010352
8    0.002070
9    0.004141
Name: Customer service calls, dtype: float64
```

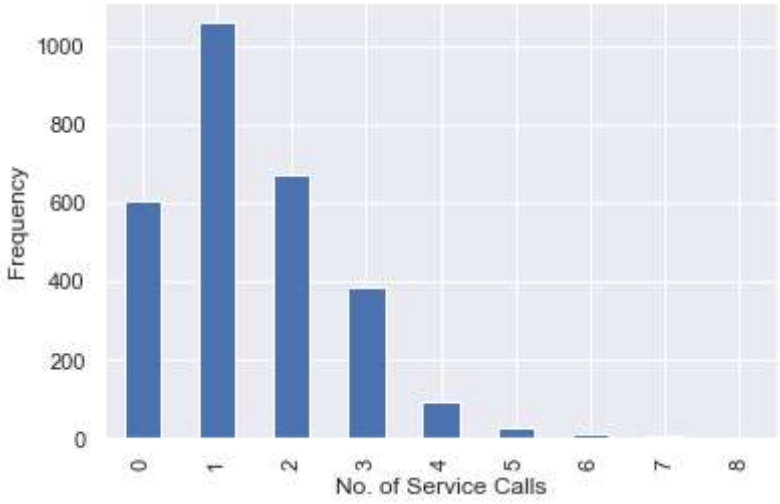
Out[35]:



In [36]:
```
print('Percentage Distribution of Non Churned Users for number of Service Calls m
print('-------------------------------------------')
print(NonChurnedDF['Customer service calls'].value_counts(normalize=True).sort_ir
NonChurnedDF['Customer service calls'].value_counts().sort_index().plot(kind='bar
```

```
Percentage Distribution of Non Churned Users for number of Service Calls made
-------------------------------------------
0    0.212281
1    0.371579
2    0.235789
3    0.135088
4    0.031579
5    0.009123
6    0.002807
7    0.001404
8    0.000351
Name: Customer service calls, dtype: float64
```

Out[36]:



From the plot it is evident that those users who are making more than or equal to 4 serivice call are most likely to Churn

Conclusions

1. Users with International Plan tend to churn more frequently.

2. 11 states out of 51 states have more than 20% churn rate.

3. Inter Quartile Range(IQR) is more spread for churned users compared to non churned users.

4. Users opting for voice mail plan are more likely to continue with their telecom service.

5. There's almost no(near to zero) correlation among day, evening and night time calls.

6. Users with more than or equal to 4 customer service calls are more likely to churn

In [ ]: