Лабораторна робота №0.
Використання основних функцій бібліотек візуального аналізу

Повторіть лекцій матеріал слідуючи інструкціям

In [7]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Доступ до даних на google drive, якщо ви відкриваєте блокнот в google colab, а не на PC, можна отримати шляхом монтування google drive

In [1]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [2]:
```
!ls gdrive/'My Drive'/TEACHING/IntroDataScience/intro_to_data_science/Lab_3_4/da
```

```
howpop_train.csv  telecom_churn.csv  titanic_train.csv
```

In [5]:
```
# шлях до папки з даними на моєму google drive, відредагуйте згідно вашого випад
data_folder = "gdrive/My Drive/TEACHING/IntroDataScience/intro_to_data_science/La
```

In [8]:
```
#df = pd.read_csv('data/telecom_churn.csv')
df = pd.read_csv(data_folder+'/telecom_churn.csv')
```

In [9]:
```
df.head()
```

Out[9]:

	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
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122

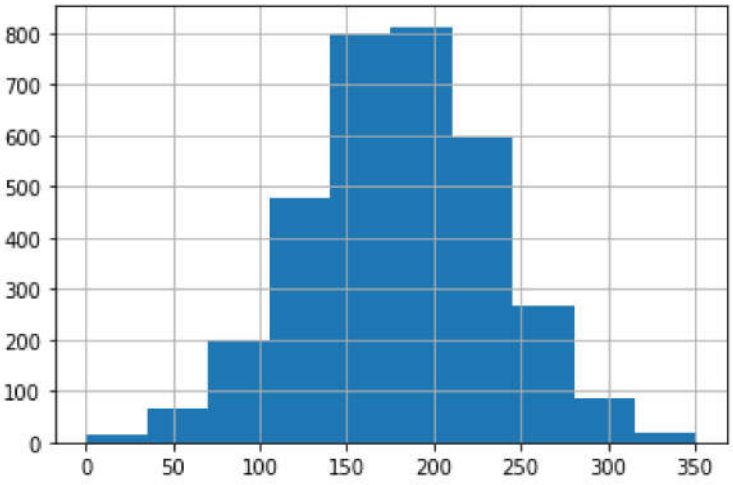
1. Візуалізація характеристик ознак по одній

1.1. Кількісні

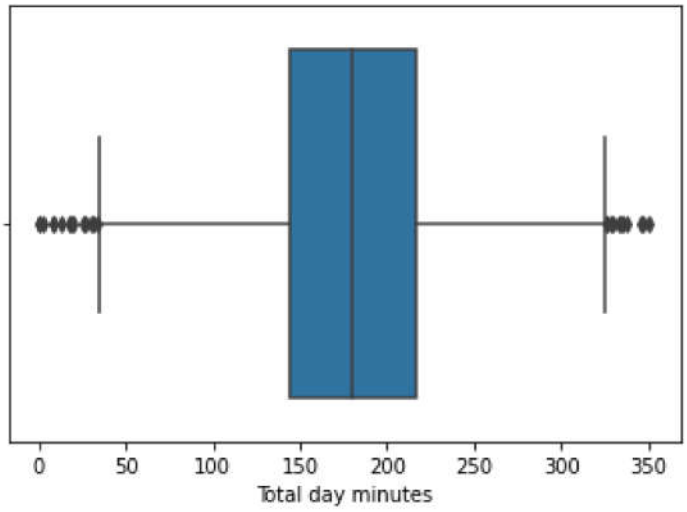
Гістограма і боксплот

In [10]:
```
df['Total day minutes'].hist()
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ca500b198>
```



In [11]:
```
sns.boxplot(df['Total day minutes']);
```



In [13]:
```
df.hist()
```

```
<string>:6: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'>
for compatibility.

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py in _unsigned_sub
tract(a, b)
    353     try:
--> 354         dt = signed_to_unsigned[dt.type]
    355     except KeyError:

KeyError: <class 'numpy.bool_'>

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
<ipython-input-13-f2dbbbaacc11> in <module>()
----> 1 df.hist(figsize=(20,12))

/usr/local/lib/python3.6/dist-packages/pandas/plotting/_core.py in hist_frame(d
ata, column, by, grid, xlabelsize, xrot, ylabelsize, yrot, ax, sharex, sharey,
 figsize, layout, bins, backend, **kwargs)
    206         layout=layout,
    207         bins=bins,
--> 208         **kwargs,
    209     )
    210

/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/hist.py in h
ist_frame(data, column, by, grid, xlabelsize, xrot, ylabelsize, yrot, ax, share
x, sharey, figsize, layout, bins, **kwds)
    402     for i, col in enumerate(com.try_sort(data.columns)):
    403         ax = _axes[i]
--> 404         ax.hist(data[col].dropna().values, bins=bins, **kwds)
    405         ax.set_title(col)
    406         ax.grid(grid)

/usr/local/lib/python3.6/dist-packages/matplotlib/__init__.py in inner(ax, dat
a, *args, **kwargs)
   1563     def inner(ax, *args, data=None, **kwargs):
   1564         if data is None:
-> 1565             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1566
   1567         bound = new_sig.bind(ax, *args, **kwargs)

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py in hist(self,
 x, bins, range, density, weights, cumulative, bottom, histtype, align, orienta
tion, rwidth, log, color, label, stacked, **kwargs)
   6658             # this will automatically overwrite bins,
   6659             # so that each histogram uses the same bins
-> 6660             m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwa
rgs)
   6661             tops.append(m)
   6662         tops = np.array(tops, float)  # causes problems later if it's a
```

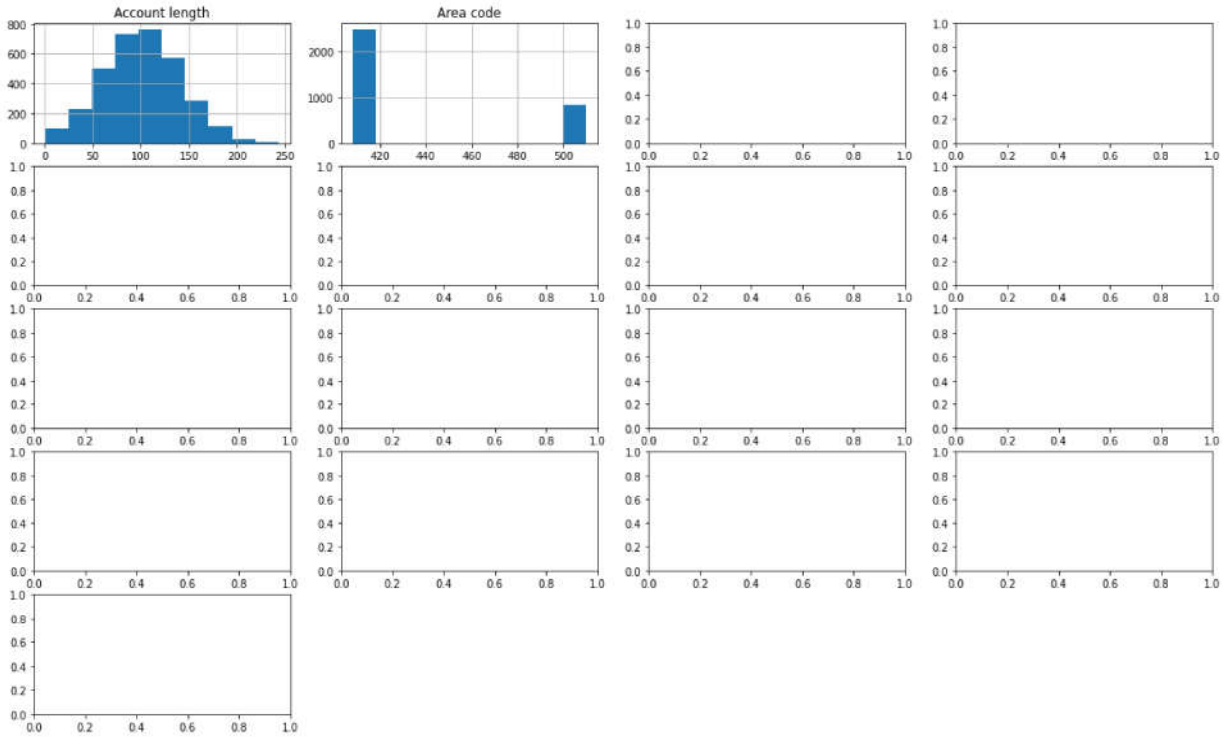
```
n int

<__array_function__ internals> in histogram(*args, **kwargs)

/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py in histogram(a,
 bins, range, normed, weights, density)
    823
    824         # Pre-compute histogram scaling factor
--> 825         norm = n_equal_bins / _unsigned_subtract(last_edge, first_edge)
    826
    827         # We iterate over blocks here for two reasons: the first is tha
t for

/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py in _unsigned_sub
tract(a, b)
    354         dt = signed_to_unsigned[dt.type]
    355     except KeyError:
--> 356         return np.subtract(a, b, dtype=dt)
    357     else:
    358         # we know the inputs are integers, and we are deliberately cast
ing

TypeError: numpy boolean subtract, the `-` operator, is not supported, use the
 bitwise_xor, the `^` operator, or the logical_xor function instead.
```



1.2. Категоріальні

countplot

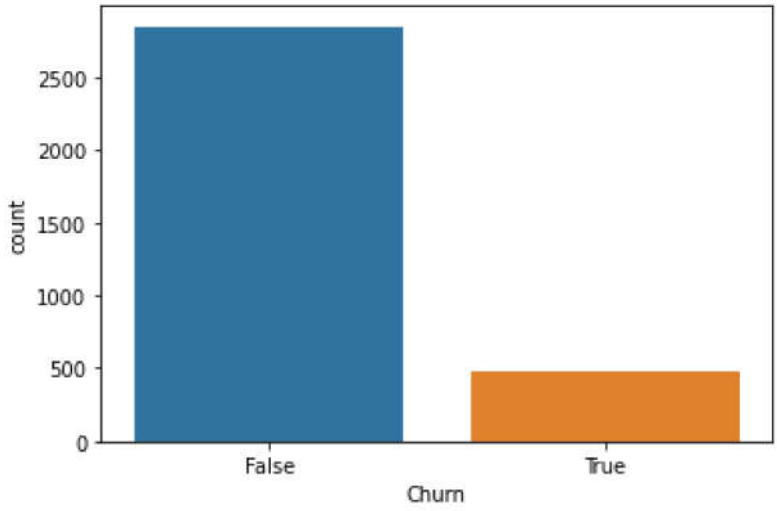
In [14]:
```
df['State'].value_counts().head()
```

Out[14]:
```
WV    106
MN     84
NY     83
AL     80
WI     78
Name: State, dtype: int64
```

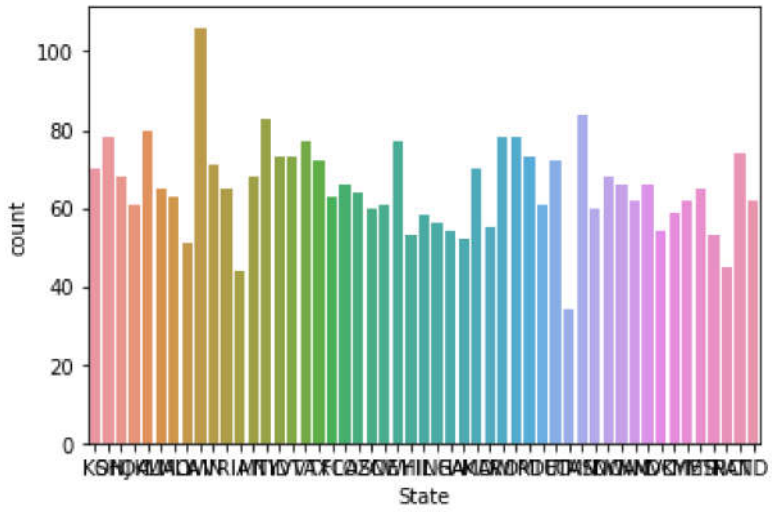
In [15]:
```
df['Churn'].value_counts()
```

Out[15]:
```
False    2850
True      483
Name: Churn, dtype: int64
```

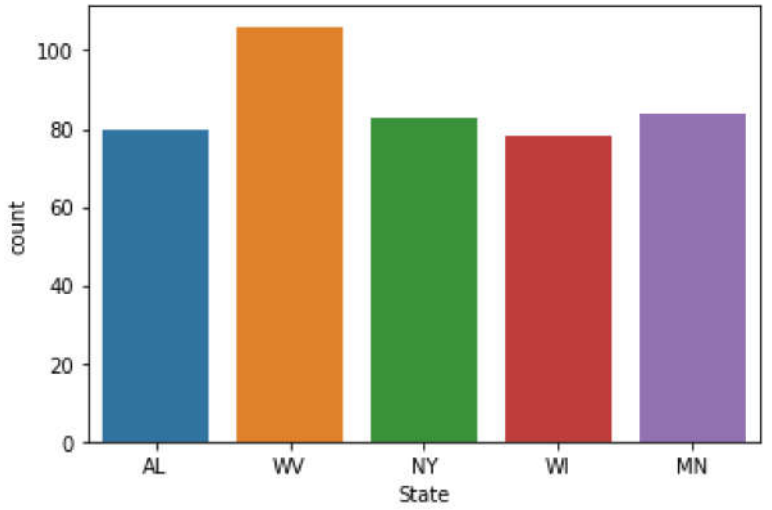
In [16]:
```
sns.countplot(df['Churn']);
```



In [17]:
```
sns.countplot(df['State']);
```



In [18]:
```
sns.countplot(df[df['State'].\
                 isin(df['State'].value_counts().head().index)]['State']);
```



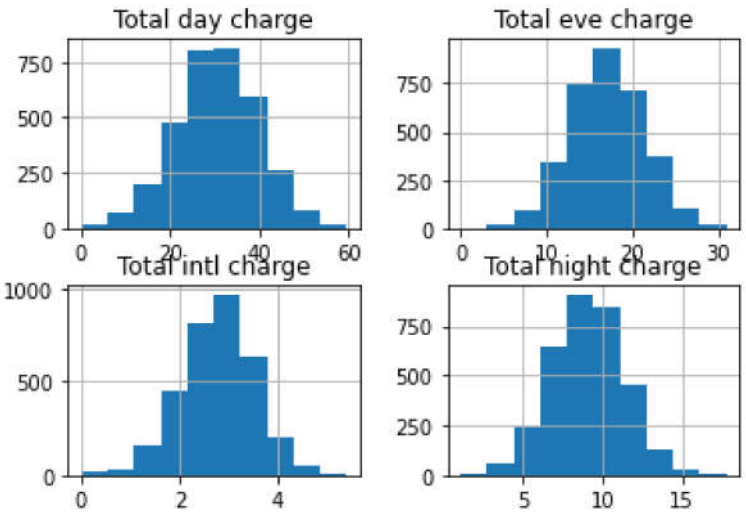
2. Взаємодія ознак

2.1. Кількісна з кількісною

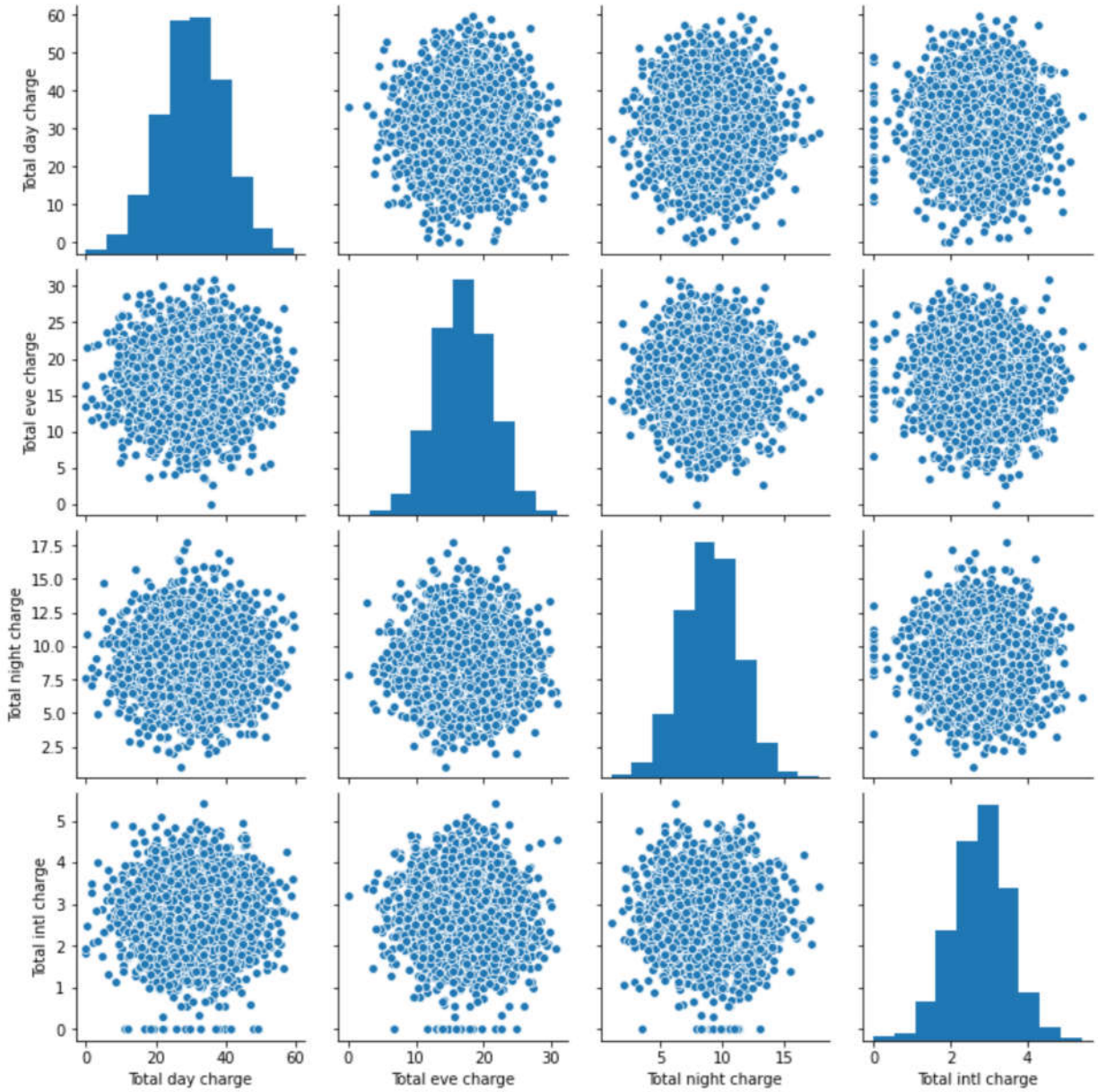
pairplot, scatterplot, кореляція, heatmap

In [19]:
```
feat = [f for f in df.columns if 'charge' in f]

df[feat].hist();
```



In [20]:
```
sns.pairplot(df[feat]);
```



In [21]:
```
df['Churn'].map({False: 'blue', True: 'orange'}).head()
```

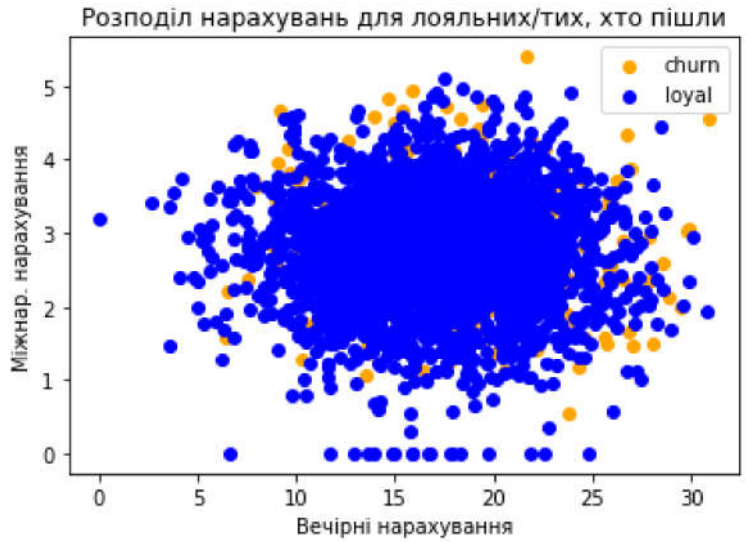
Out[21]:
```
0    blue
1    blue
2    blue
3    blue
4    blue
Name: Churn, dtype: object
```

In [22]:
```
df[~df['Churn']].head()
```

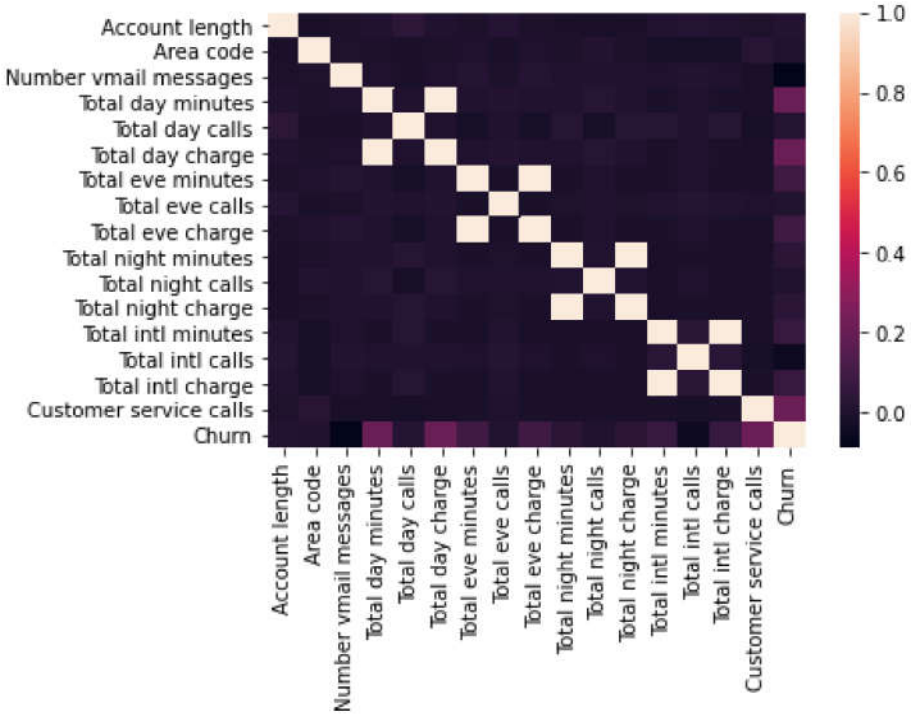
Out[22]:

	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
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122

In [23]:
```
plt.scatter(df[df['Churn']]['Total eve charge'],
            df[df['Churn']]['Total intl charge'],
            color='orange', label='churn');
plt.scatter(df[~df['Churn']]['Total eve charge'],
            df[~df['Churn']]['Total intl charge'],
            color='blue', label='loyal');
plt.xlabel('Вечірні нарахування');
plt.ylabel('Міжнар. нарахування');
plt.title('Розподіл нарахувань для лояльних/тих, хто пішли');
plt.legend();
```

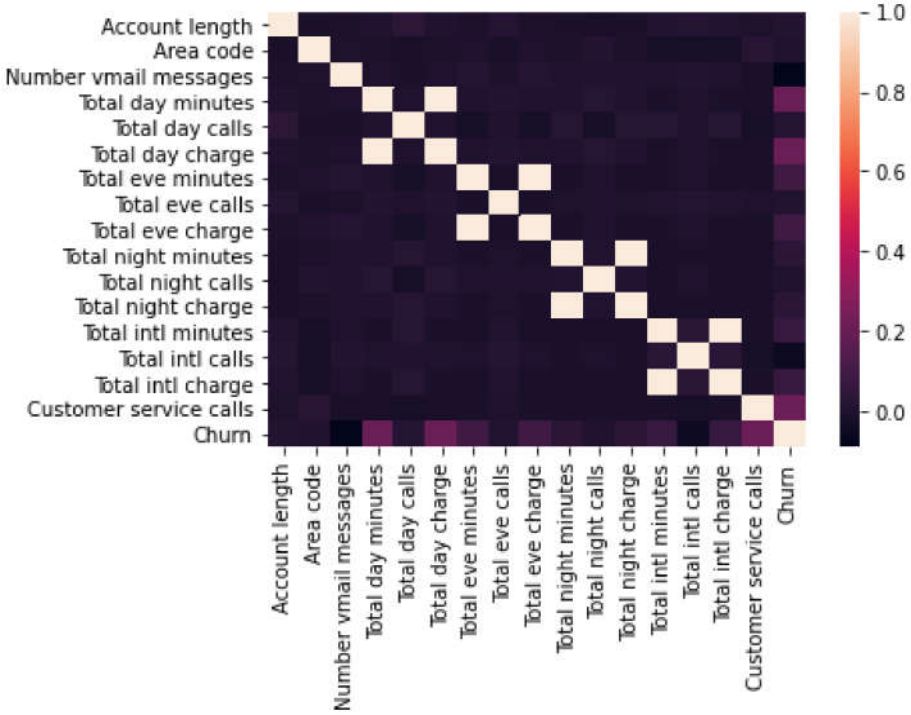


In [24]:
```
sns.heatmap(df.corr());
```



In [ ]:
```
df.drop(feat, axis=1, inplace=True)
```

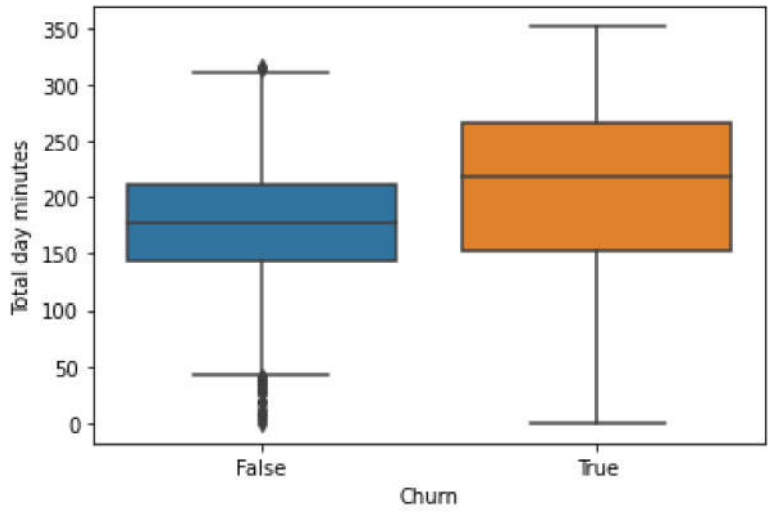
In [25]:
```
sns.heatmap(df.corr());
```



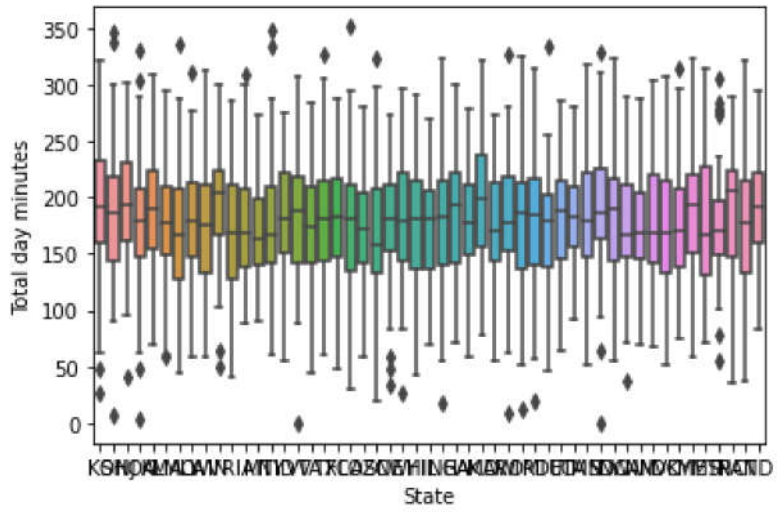
2.2. Кількісний з категоріальним

boxplot, violinplot

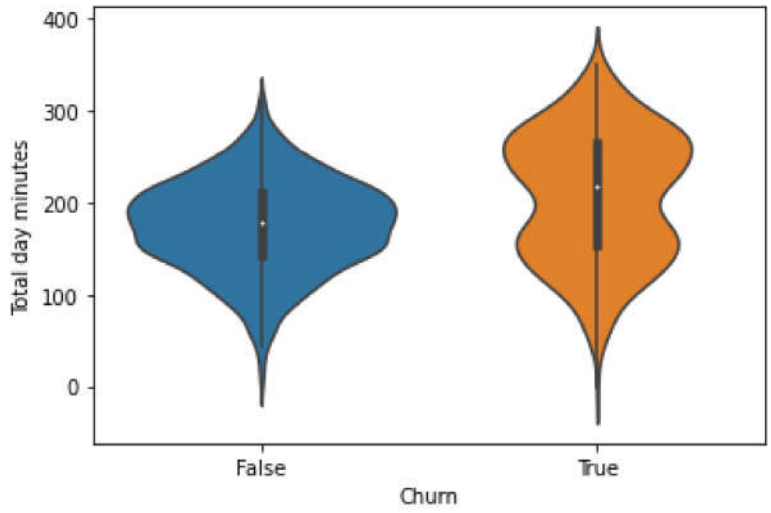
In [31]:
```
sns.boxplot(x='Churn', y='Total day minutes', data=df);
```



In [32]:
```
sns.boxplot(x='State', y='Total day minutes', data=df);
```



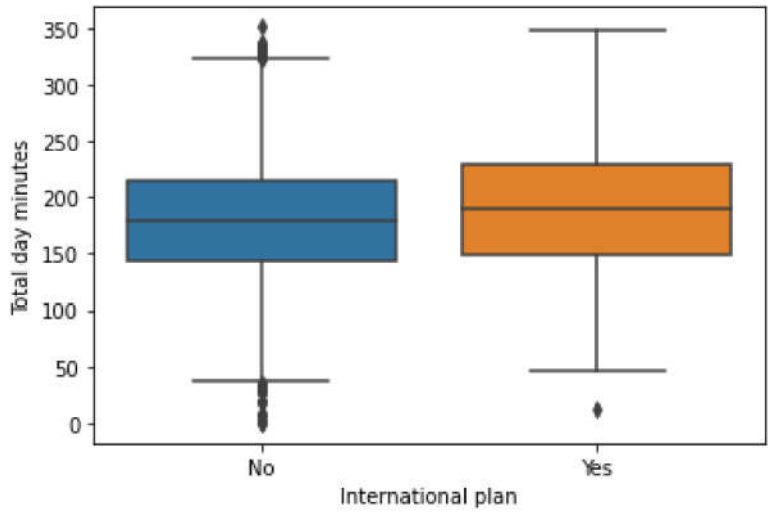
In [33]:
```
sns.violinplot(x='Churn', y='Total day minutes', data=df);
```



In [34]:
```
df.groupby('International plan')['Total day minutes'].mean()
```

Out[34]:
```
International plan
No     178.893887
Yes    187.986997
Name: Total day minutes, dtype: float64
```

In [35]:
```
sns.boxplot(x='International plan', y='Total day minutes', data=df);
```



2.3. Категоріальний з категоріальним

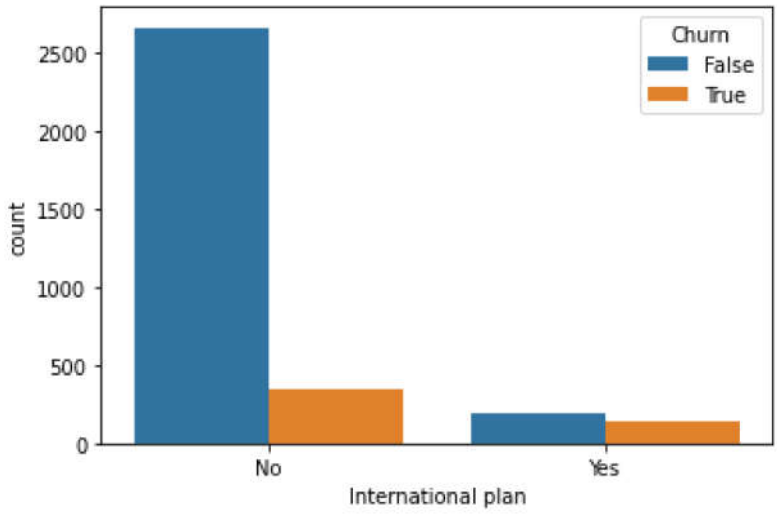
countplot

In [36]:
```
pd.crosstab(df['Churn'], df['International plan'])
```

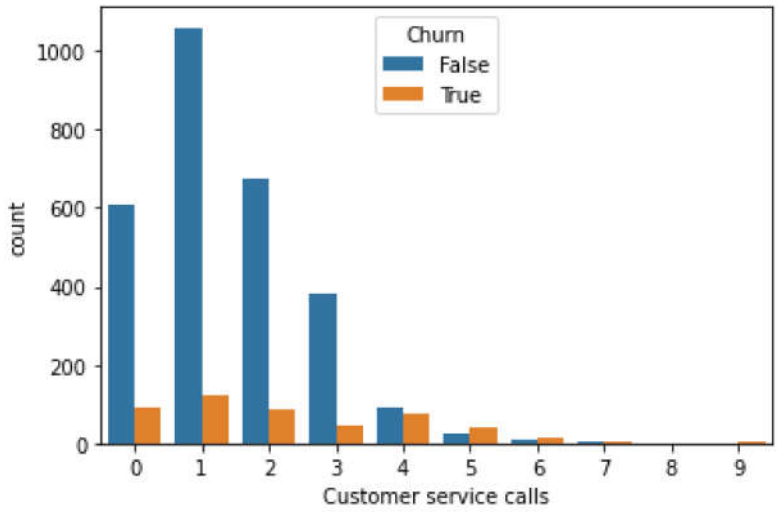
Out[36]:

International plan	No	Yes
Churn		
False	2664	186
True	346	137

In [37]:
```
sns.countplot(x='International plan', hue='Churn', data=df);
```



In [38]:
```
sns.countplot(x='Customer service calls', hue='Churn', data=df);
```



3. Інше

Manifold learning, один з представників – t-SNE

In [39]:
```
from sklearn.manifold import TSNE
```

In [40]:
```
tsne = TSNE(random_state=0)
```

In [41]:
```
df2 = df.drop(['State', 'Churn'], axis=1)
```

In [42]:
```
df2['International plan'] = df2['International plan'].map({'Yes': 1,
                                                           'No': 0})
df2['Voice mail plan'] = df2['Voice mail plan'].map({'Yes': 1,
                                                     'No': 0})
```

In [43]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Account length          3333 non-null   int64  
 1   Area code               3333 non-null   int64  
 2   International plan      3333 non-null   int64  
 3   Voice mail plan         3333 non-null   int64  
 4   Number vmail messages   3333 non-null   int64  
 5   Total day minutes       3333 non-null   float64
 6   Total day calls         3333 non-null   int64  
 7   Total day charge        3333 non-null   float64
 8   Total eve minutes       3333 non-null   float64
 9   Total eve calls         3333 non-null   int64  
 10  Total eve charge        3333 non-null   float64
 11  Total night minutes     3333 non-null   float64
 12  Total night calls       3333 non-null   int64  
 13  Total night charge      3333 non-null   float64
 14  Total intl minutes      3333 non-null   float64
 15  Total intl calls        3333 non-null   int64  
 16  Total intl charge       3333 non-null   float64
 17  Customer service calls  3333 non-null   int64  
dtypes: float64(8), int64(10)
memory usage: 468.8 KB
```

In [44]:
```
%%time
tsne.fit(df2)
```

```
CPU times: user 54.1 s, sys: 197 ms, total: 54.3 s
Wall time: 27.9 s
```

Out[44]:
```
TSNE(angle=0.5, early_exaggeration=12.0, init='random', learning_rate=200.0,
     method='barnes_hut', metric='euclidean', min_grad_norm=1e-07,
     n_components=2, n_iter=1000, n_iter_without_progress=300, n_jobs=None,
     perplexity=30.0, random_state=0, verbose=0)
```

In [45]:
```
plt.scatter(tsne.embedding_[df['Churn'].values, 0],
            tsne.embedding_[df['Churn'].values, 1],
            color='orange', alpha=.7);
plt.scatter(tsne.embedding_[~df['Churn'].values, 0],
            tsne.embedding_[~df['Churn'].values, 1],
            color='blue', alpha=.7);
```