



National Institute of Technology Rourkela, Odisha, India, 769008

Department of Computer Science Engineering

Laboratory-2

(Data Science Laboratory)
Nikhil Jaiswal (121CS1133)

In [1]:

```
import numpy as np #Numpy stands for Numerical Python. used for numerical Operations
import pandas as pd #Pandas is used for DataFrame/Database Operations
import warnings
warnings.filterwarnings("ignore")

pd.set_option("display.precision", 2)
```

C:\Users\ Laptop\AppData\Local\Temp\ipykernel_5880\2695976249.py:2: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas
(pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interopera
rability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at <https://github.com/pandas-dev/pandas/issues/54466>

import pandas as pd #Pandas is used for DataFrame/Database Operations

Read the telecom_churn csv file.

In [2]:

```
#Your answer here
dataSet = pd.read_csv(r"C:\Users\ Laptop\OneDrive - National Institute of Technology, Ro
```

Show the first 10 rows of the dataset

In [3]:

```
#Your answer here
dataSet.head(10)
```

Out[3]:

	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
0	KS	128	415	No	Yes	25.0	265.1	110	45.07	197.4	99	16.78
1	OH	107	415	No	Yes	26.0	161.6	123	27.47	195.5	103	16.62
2	NJ	137	415	No	No	NaN	243.4	114	41.38	121.2	110	10.30
3	OH	84	408	Yes	No	NaN	299.4	71	50.90	61.9	88	5.26
4	OK	75	415	Yes	No	0.0	166.7	113	28.34	148.3	122	12.61

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
5	AL	118	510	Yes	No	0.0	Nan	98	37.98	220.6	101
6	MA	121	510	No	Yes	24.0	218.2	88	37.09	348.5	108
7	MO	147	415	Yes	No	0.0	157.0	79	26.69	103.1	94
8	LA	117	408	No	No	0.0	184.5	97	31.37	351.6	80
9	WV	141	415	Yes	Yes	37.0	258.6	84	43.96	222.0	111

Show the last 10 rows of the dataset

In [4]:

```
#Your answer here
dataSet.tail(10)
```

Out[4]:

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
3323	IN	117	415	No	No	0.0	118.4	126	20.13	249.3	97
3324	WV	159	415	No	No	0.0	169.8	114	28.87	197.7	105
3325	OH	78	408	No	No	0.0	193.4	99	32.88	116.9	88
3326	OH	96	415	No	No	0.0	106.6	128	18.12	284.8	87
3327	SC	79	415	No	No	0.0	134.7	98	22.90	189.7	68
3328	AZ	192	415	No	Yes	36.0	156.2	77	26.55	215.5	126
3329	WV	68	415	No	No	0.0	231.1	57	39.29	153.4	55
3330	RI	28	510	No	No	0.0	180.8	109	30.74	288.8	58
3331	CT	184	510	Yes	No	0.0	213.8	105	36.35	159.6	84
3332	TN	74	415	No	Yes	25.0	234.4	113	39.85	265.9	82

Show the dimensions (No. of rows and columns) of the dataset

In [5]:

```
#Your answer here
print(dataSet.shape)
```

(3333, 20)

Print all the column names of the dataset

In [6]:

```
#Your answer here
print(dataSet.columns)
```

```
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')
```

Print general information of the dataset like column, and datatype.

In [7]:

```
#Your answer here
print(dataSet.info())
```

```
<class 'pandas.core.frame.DataFrame'>
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 3328 non-null    float64 
 6   Total day minutes 3329 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(9), int64(7), object(3)
memory usage: 498.1+ KB
None
```

After getting the datatype of every feature, you can see that there is a feature "**churn**" having Dtype as **bool**. Change that Dtype to **int64**.

In [8]:

```
#Your answer here
dataSet['Churn'] = dataSet['Churn'].astype('int64')
```

Which method shows basic statistical characteristics of each numerical feature (`int64` and `float64` types): number of non-missing values, mean, standard deviation, range, median, 0.25 and 0.75 quartiles.

In [9]:

```
#Your answer here
dataSet.describe()
```

Out[9]:

	Account length	Area code	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
count	3333.00	3333.00	3328.00	3329.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00
mean	101.06	437.18	8.11	179.78	100.44	30.56	200.98	100.11	17.08	200.87	100.00
std	39.82	42.37	13.70	54.48	20.07	9.26	50.71	19.92	4.31	50.57	19.92
min	1.00	408.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	23.20	3.00
25%	74.00	408.00	0.00	143.70	87.00	24.43	166.60	87.00	14.16	167.00	8.00
50%	101.00	415.00	0.00	179.40	101.00	30.50	201.40	100.00	17.12	201.20	100.00
75%	127.00	510.00	20.00	216.40	114.00	36.79	235.30	114.00	20.00	235.30	11.00
max	243.00	510.00	51.00	350.80	165.00	59.64	363.70	170.00	30.91	395.00	17.00

◀ ▶

Now show basic statistics on non-numerical features.

In [10]:

```
#Your answer here
dataSet.describe(include=['object', 'bool'])
```

Out[10]:

	State	International plan	Voice mail plan
count	3333	3333	3333
unique	51	2	2
top	WV	No	No
freq	106	3010	2411

How to show the basic statistics for categorical (type `object`) and boolean (type `bool`) features?

In [11]:

```
#Your answer here
dataSet.describe(include=['object', 'bool'])
```

Out[11]:

	State	International plan	Voice mail plan
count	3333	3333	3333
unique	51	2	2
top	WV	No	No
freq	106	3010	2411

Check for Missing Values

How to check for missing values in each column?

In [12]:

```
#Your answer here
dataSet.isnull().sum()
```

```
Out[12]: State          0
          Account length    0
          Area code         0
          International plan 0
          Voice mail plan   0
          Number vmail messages 5
          Total day minutes 4
          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
```

Sorting

How to sort a DataFrame using one of the variables (i.e columns) in ascending order and also in descending order?

In [13]:

```
#Your answer here
dataSet.sort_values(by='Total day calls', ascending=True)
# for descending order we do ascending=False
```

Out[13]:

	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 ch
1397	VT	101	510	No	No	0.0	0.0	0	0.00	192.1	119	1
1345	SD	98	415	No	No	0.0	0.0	0	0.00	159.6	130	1
1144	NH	155	408	No	No	0.0	216.7	30	36.84	144.3	125	1
1989	MT	124	415	No	Yes	30.0	144.5	35	24.57	262.3	101	2
692	NE	82	408	No	No	0.0	185.8	36	31.59	276.5	134	2
...
315	MA	39	408	No	No	0.0	60.4	158	10.27	306.2	120	2
1057	WV	86	415	No	Yes	38.0	123.0	158	20.91	133.9	119	1
1460	MT	80	415	No	No	0.0	198.1	160	33.68	156.7	87	1
468	AZ	86	415	No	Yes	32.0	70.9	163	12.05	166.7	121	1
1121	MI	91	415	No	No	0.0	154.4	165	26.25	168.3	121	1

3333 rows × 20 columns

How to sort a DataFrame using multiple variables (i.e columns) in ascending order and also in descending order?

In [14]:

```
#Your answer here
dataSet.sort_values(by=['Churn', 'Total day calls'], ascending=[True, False])
```

Out[14]:

	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	1 ch
468	AZ	86	415	No	Yes	32.0	70.9	163	12.05	166.7	121	1
1460	MT	80	415	No	No	0.0	198.1	160	33.68	156.7	87	1
315	MA	39	408	No	No	0.0	60.4	158	10.27	306.2	120	2
1057	WV	86	415	No	Yes	38.0	123.0	158	20.91	133.9	119	1
2392	WY	90	510	No	No	0.0	125.4	158	21.32	269.1	83	2
...
1346	PA	106	408	Yes	No	0.0	133.7	45	22.73	187.8	107	1
2884	UT	170	415	No	No	0.0	285.7	44	48.57	167.5	144	1
2964	OR	99	408	No	No	0.0	256.4	44	43.59	214.5	105	1
1322	DE	2	415	Yes	No	0.0	132.1	42	22.46	138.9	88	1
1345	SD	98	415	No	No	0.0	0.0	0	0.00	159.6	130	1

3333 rows × 20 columns

Indexing and retrieving data

What is the proportion of churned users in our dataframe?

In [15]:

```
#Your answer here
no_of_churn = dataSet[dataSet['Churn']==1].shape[0]
print("No of churn users:", no_of_churn)
percentage_of_churn = (no_of_churn / dataSet.shape[0]) * 100
print("Percentage of churn users:", round(percentage_of_churn, 2), "%")
dataSet[dataSet['Churn']==1]
```

No of churn users: 483

Percentage of churn users: 14.49 %

Out[15]:

	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	1 ch
10	IN	65	415	No	No	0.0	129.1	137	21.95	228.5	83	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	Total eve calls	1 ch
15	NY	161	415	No	No	0.0	332.9	67	56.59	317.8	97	2
21	CO	77	408	No	No	0.0	62.4	89	10.61	169.9	121	1
33	AZ	12	408	No	No	0.0	249.6	118	42.43	252.4	119	2
41	MD	135	408	Yes	Yes	41.0	173.1	85	29.43	203.9	107	1
...
3301	CA	84	415	No	No	0.0	280.0	113	47.60	202.2	90	1
3304	IL	71	510	Yes	No	0.0	186.1	114	31.64	198.6	140	1
3320	GA	122	510	Yes	No	0.0	140.0	101	23.80	196.4	77	1
3322	MD	62	408	No	No	0.0	321.1	105	54.59	265.5	122	2
3323	IN	117	415	No	No	0.0	118.4	126	20.13	249.3	97	2

483 rows × 20 columns

What are average values of numerical features for churned users?

In [16]:

```
#Your answer here
for i in dataSet.columns:
    if dataSet[i].dtype in ['int64', 'float64']:
        print("Mean for the column '",i,"' is:",dataSet[dataSet['Churn']==1][i].mean())
```

```
Mean for the column ' Account length ' is: 102.66459627329192
Mean for the column ' Area code ' is: 437.8178053830228
Mean for the column ' Number vmail messages ' is: 5.115942028985507
Mean for the column ' Total day minutes ' is: 206.91407867494823
Mean for the column ' Total day calls ' is: 101.33540372670808
Mean for the column ' Total day charge ' is: 35.17592132505176
Mean for the column ' Total eve minutes ' is: 212.41014492753624
Mean for the column ' Total eve calls ' is: 100.56107660455487
Mean for the column ' Total eve charge ' is: 18.05496894409938
Mean for the column ' Total night minutes ' is: 205.23167701863352
Mean for the column ' Total night calls ' is: 100.39958592132506
Mean for the column ' Total night charge ' is: 9.23552795031056
Mean for the column ' Total intl minutes ' is: 10.700000000000001
Mean for the column ' Total intl calls ' is: 4.163561076604555
Mean for the column ' Total intl charge ' is: 2.8895445134575573
Mean for the column ' Customer service calls ' is: 2.229813664596273
Mean for the column ' Churn ' is: 1.0
```

How much time (on average) do churned users spend on the phone during daytime?

In [17]:

```
#Your answer here
print("Total avg Min Spend by churned users on the phone during the daytime:",dataSet[d
```

```
Total avg Min Spend by churned users on the phone during the daytime: 206.91407867494823
```

What is the maximum length of international calls among loyal users (Churn == 0) who do not have an international plan?

In [18]:

```
#Your answer here
dataSet[(dataSet['Churn']==0) & (dataSet['International plan']=='No')]['Total intl minutes'].max()
```

Out[18]:

18.9

How to get the first or the last line of the data frame?

In [19]:

```
#Your answer here
print(dataSet.iloc[0]) # First Line of the data frame
print(dataSet.iloc[-1]) # Last Line of the data frame
```

State	KS
Account length	128
Area code	415
International plan	No
Voice mail plan	Yes
Number vmail messages	25.0
Total day minutes	265.1
Total day calls	110
Total day charge	45.07
Total eve minutes	197.4
Total eve calls	99
Total eve charge	16.78
Total night minutes	244.7
Total night calls	91
Total night charge	11.01
Total intl minutes	10.0
Total intl calls	3
Total intl charge	2.7
Customer service calls	1
Churn	0
Name:	0, dtype: object
State	TN
Account length	74
Area code	415
International plan	No
Voice mail plan	Yes
Number vmail messages	25.0
Total day minutes	234.4
Total day calls	113
Total day charge	39.85
Total eve minutes	265.9
Total eve calls	82
Total eve charge	22.6
Total night minutes	241.4
Total night calls	77
Total night charge	10.86
Total intl minutes	13.7
Total intl calls	4
Total intl charge	3.7
Customer service calls	0
Churn	0
Name:	3332, dtype: object

Applying Functions to Cells, Columns and Rows

How can you apply a function to each column of a DataFrame to find the maximum value in each column?

In [20]:

```
#Your answer here
dataSet.max()
```

Out[20]:

State	WY
Account length	243
Area code	510
International plan	Yes
Voice mail plan	Yes
Number vmail messages	51.0
Total day minutes	350.8
Total day calls	165
Total day charge	59.64
Total eve minutes	363.7
Total eve calls	170
Total eve charge	30.91
Total night minutes	395.0
Total night calls	175
Total night charge	17.77
Total intl minutes	20.0
Total intl calls	20
Total intl charge	5.4
Customer service calls	9
Churn	1
dtype:	object

How can you filter rows in a DataFrame to select all states that start with the letter "W"?

In [21]:

```
#Your answer here
dataSet[dataSet['State'].apply(lambda state: state[0]=='W')]
```

Out[21]:

	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	1
9	WV	141	415	Yes	Yes	37.0	258.6	84	43.96	222.0	111	1
26	WY	57	408	No	Yes	39.0	213.0	115	36.21	191.1	112	1
44	WI	64	510	No	No	0.0	154.0	67	26.18	225.8	118	1
49	WY	97	415	No	Yes	24.0	133.2	135	22.64	217.2	58	1
54	WY	87	415	No	No	0.0	151.0	83	25.67	219.7	116	1
...
3278	WI	87	415	No	No	0.0	238.0	97	40.46	164.5	97	1
3303	WI	114	415	No	Yes	26.0	137.1	88	23.31	155.7	125	1
3319	WY	89	415	No	No	0.0	115.4	99	19.62	209.9	115	1
3324	WV	159	415	No	No	0.0	169.8	114	28.87	197.7	105	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	Total eve calls	Total eve ch
3329	WV	68	415	No	No	0.0	231.1	57	39.29	153.4	55 1

327 rows × 20 columns

How can you replace values in a DataFrame column, such as changing the values of "International plan" and "Voice mail plan" from "Yes" or "No" to True or False?

In [22]:

```
#Your answer here

dataSet['International plan'] = dataSet['International plan'].map({'Yes': 'True', 'No':
```

Grouping

In general, grouping data in Pandas works as follows:

```
df.groupby(by=grouping_columns)[columns_to_show].function()
```

1. First, the `groupby` method divides the `grouping_columns` by their values. They become a new index in the resulting dataframe.
2. Then, columns of interest are selected (`columns_to_show`). If `columns_to_show` is not included, all non groupby clauses will be included.
3. Finally, one or several functions are applied to the obtained groups per selected columns.

How can you group data by a specific column (e.g., "Churn") and then display statistical summaries (e.g., mean, standard deviation, etc.) for selected columns such as "Total day minutes," "Total eve minutes," and "Total night minutes"?

In [23]:

```
#Your answer here
cols = ['Total day minutes', 'Total eve minutes', 'Total night minutes']
for i in cols:
    print("For column '", i, "'")
    print("Mean: ")
    print(dataSet.groupby(by='Churn')[i].mean())

    print("Standard Deviation: ")
    print(dataSet.groupby(by='Churn')[i].std())

    print("Total: ")
    print(dataSet.groupby(by='Churn')[i].sum())
    print()
```

```
For column ' Total day minutes '
Mean:
Churn
0    175.18
1    206.91
Name: Total day minutes, dtype: float64
Standard Deviation:
```

```

Churn
0    50.2
1    69.0
Name: Total day minutes, dtype: float64
Total:
Churn
0    498556.3
1    99939.5
Name: Total day minutes, dtype: float64

For column ' Total eve minutes '
Mean:
Churn
0    199.04
1    212.41
Name: Total eve minutes, dtype: float64
Standard Deviation:
Churn
0    50.29
1    51.73
Name: Total eve minutes, dtype: float64
Total:
Churn
0    567273.4
1    102594.1
Name: Total eve minutes, dtype: float64

For column ' Total night minutes '
Mean:
Churn
0    200.13
1    205.23
Name: Total night minutes, dtype: float64
Standard Deviation:
Churn
0    51.11
1    47.13
Name: Total night minutes, dtype: float64
Total:
Churn
0    570379.6
1    99126.9
Name: Total night minutes, dtype: float64

```

How can you group data by a specific column (e.g., "Churn") and then apply multiple aggregation functions (e.g., mean, standard deviation, min, max) to selected columns such as "Total day minutes," "Total eve minutes," and "Total night minutes"?

In [24]:

```

#Your answer here
cols = ['Total day minutes', 'Total eve minutes', 'Total night minutes']
for i in cols:
    print("For column '",i,"''")
    print("Mean: ")
    print(dataSet.groupby(by='Churn')[i].mean())

    print("Standard Deviation: ")
    print(dataSet.groupby(by='Churn')[i].std())

```

```
print("Min: ")
print(dataSet.groupby(by='Churn')[i].min())

print("Max: ")
print(dataSet.groupby(by='Churn')[i].max())
print()
```

For column ' Total day minutes '

Mean:

Churn

```
0    175.18
1    206.91
```

Name: Total day minutes, dtype: float64

Standard Deviation:

Churn

```
0    50.2
1    69.0
```

Name: Total day minutes, dtype: float64

Min:

Churn

```
0    0.0
1    0.0
```

Name: Total day minutes, dtype: float64

Max:

Churn

```
0    315.6
1    350.8
```

Name: Total day minutes, dtype: float64

For column ' Total eve minutes '

Mean:

Churn

```
0    199.04
1    212.41
```

Name: Total eve minutes, dtype: float64

Standard Deviation:

Churn

```
0    50.29
1    51.73
```

Name: Total eve minutes, dtype: float64

Min:

Churn

```
0    0.0
1    70.9
```

Name: Total eve minutes, dtype: float64

Max:

Churn

```
0    361.8
1    363.7
```

Name: Total eve minutes, dtype: float64

For column ' Total night minutes '

Mean:

Churn

```
0    200.13
1    205.23
```

Name: Total night minutes, dtype: float64

Standard Deviation:

Churn

```
0    51.11
```

```

1    47.13
Name: Total night minutes, dtype: float64
Min:
Churn
0    23.2
1    47.4
Name: Total night minutes, dtype: float64
Max:
Churn
0    395.0
1    354.9
Name: Total night minutes, dtype: float64

```

Summary tables

How can you create a contingency table to observe the distribution of data across two variables, such as "Churn" and "International plan"?

In [25]:

```
#Your answer here
pd.crosstab(dataSet['Churn'], dataSet['International plan'])
```

Out[25]:

	International plan	False	True
Churn			
0	2664	186	
1	346	137	

How can you create a contingency table to observe the distribution of data across two variables, such as "Churn" and "Voice mail plan"?

In [26]:

```
#Your answer here
pd.crosstab(dataSet['Churn'], dataSet['Voice mail plan'])
```

Out[26]:

	Voice mail plan	No	Yes
Churn			
0	2008	842	
1	403	80	

How can you create a pivot table to calculate the average number of day, evening, and night calls grouped by "Area code"?

In [27]:

```
#Your answer here
pd.pivot_table(dataSet, ['Total day calls', 'Total eve calls', 'Total night calls'], 'A
```

Out[27]:

	Total day calls	Total eve calls	Total night calls
Area code			
408	100.50	99.79	99.04

	Total day calls	Total eve calls	Total night calls
Area code			
415	100.58	100.50	100.40
510	100.10	99.67	100.60

DataFrame transformations

How can you calculate the total number of calls ("Total day calls", "Total eve calls", "Total night calls", and "Total intl calls") for all users and add this information as a new column named "Total calls" to the DataFrame?

In [28]:

```
#Your answer here
dataSet['Total calls'] = dataSet['Total day calls'] + dataSet['Total eve calls'] + data
```

How can you delete specific columns or rows from a DataFrame? **Note:** It should alter the current DataFrame and should not return a new one.

In [29]:

```
#Your answer here
dataSet.drop(['Total calls'], axis=1, inplace=True)
```

Visualizing the Data

We'll do this using visual analysis with `matplotlib` and `Seaborn`.

In [30]:

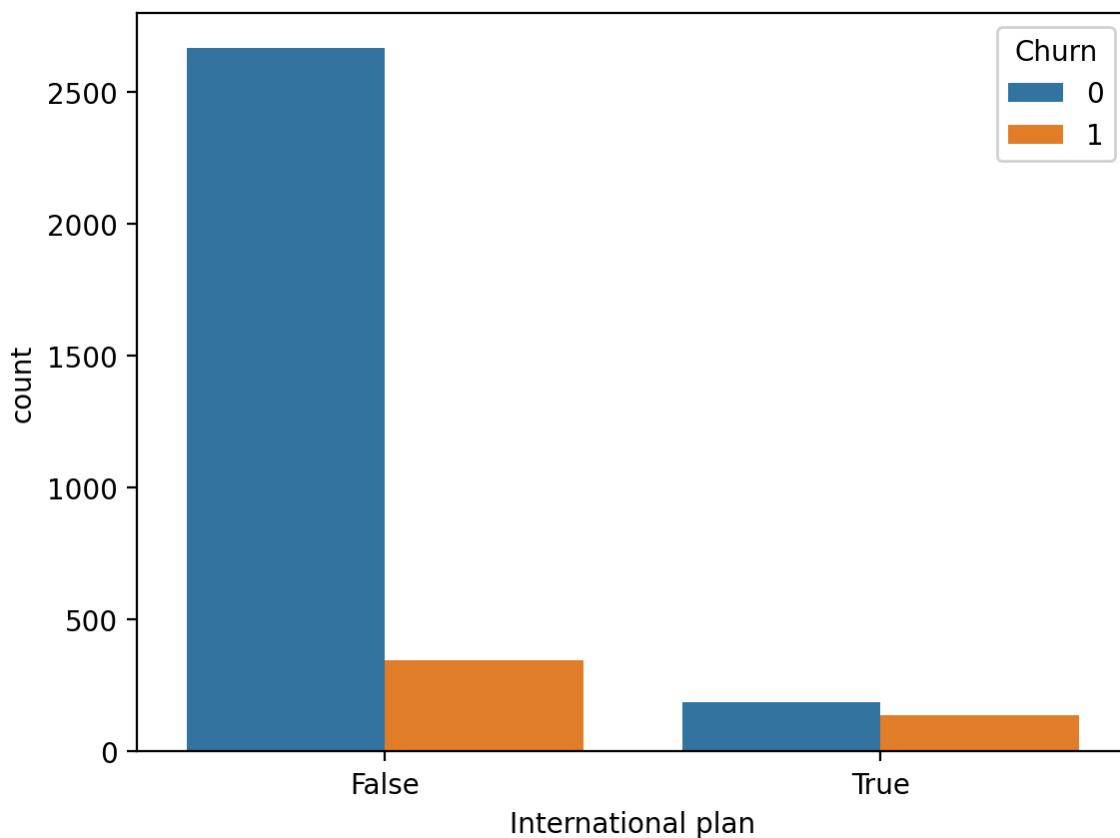
```
# some imports to set up plotting
import matplotlib.pyplot as plt
# !pip install seaborn
import seaborn as sns

# Graphics in retina format are more sharp and Legible
%config InlineBackend.figure_format = 'retina'
```

How can you create a count plot to visualize the distribution of the "International plan" variable, with separate bars for each "Churn" category,

In [31]:

```
#Your answer here
sns.countplot(x='International plan', hue='Churn', data=dataSet);
```



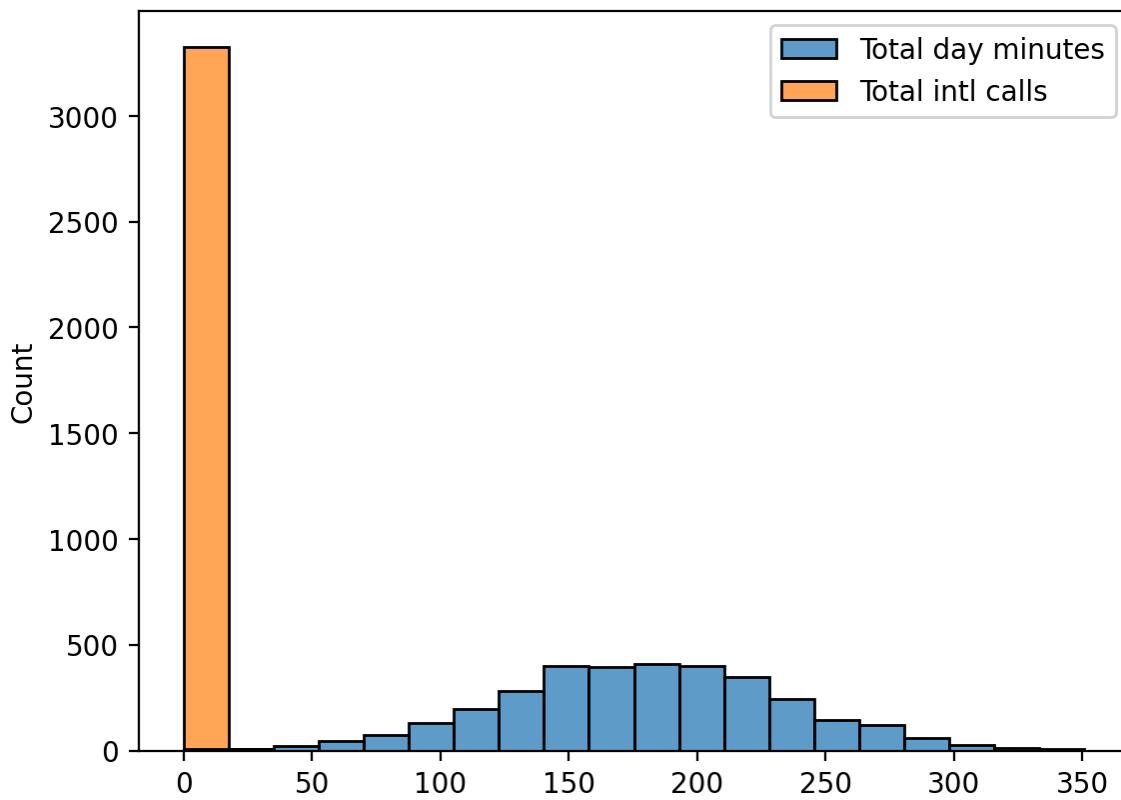
Univariate visualization

Univariate analysis looks at one feature at a time. When we analyze a feature independently, we are usually mostly interested in the *distribution of its values* and ignore other features in the dataset.

Create histograms to visualize the distribution of values for the features "Total day minutes" and "Total intl calls"?

In [32]:

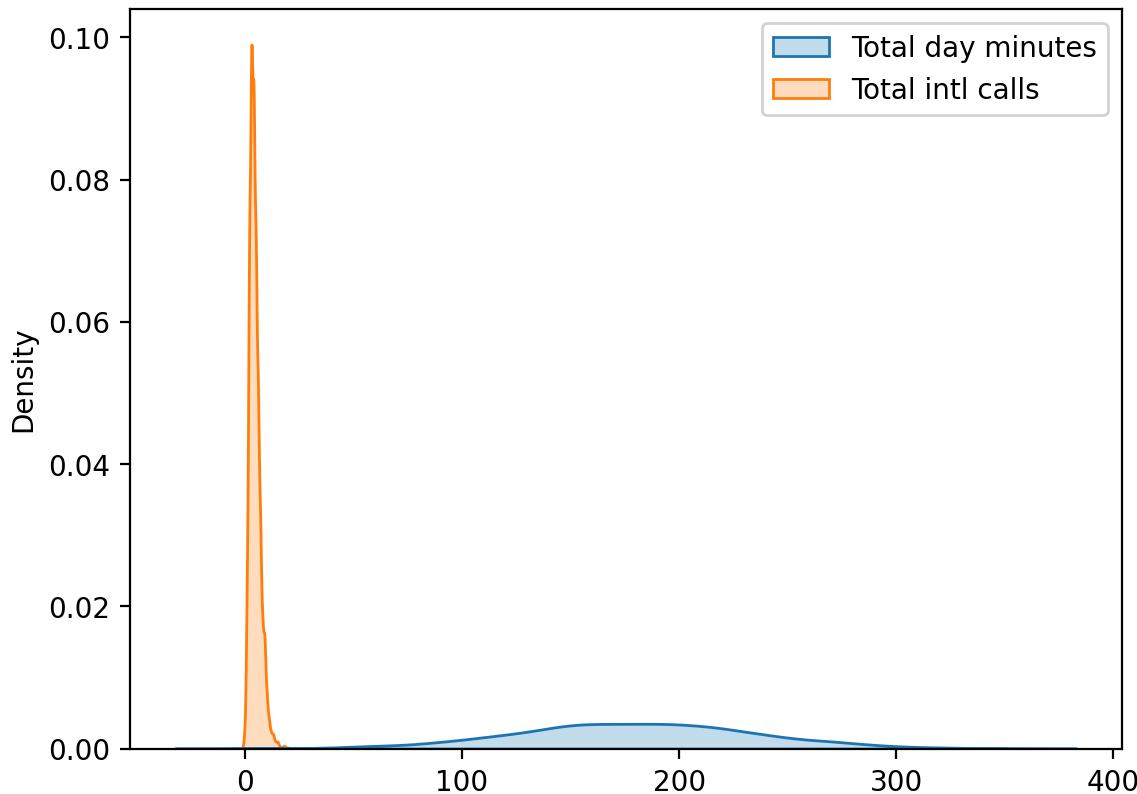
```
#Your answer here
sns.histplot(dataSet[['Total day minutes', 'Total intl calls']], bins=20, color='red',
```



Create density plots to visualize the distribution of values for the features "Total day minutes" and "Total intl calls"?

In [33]:

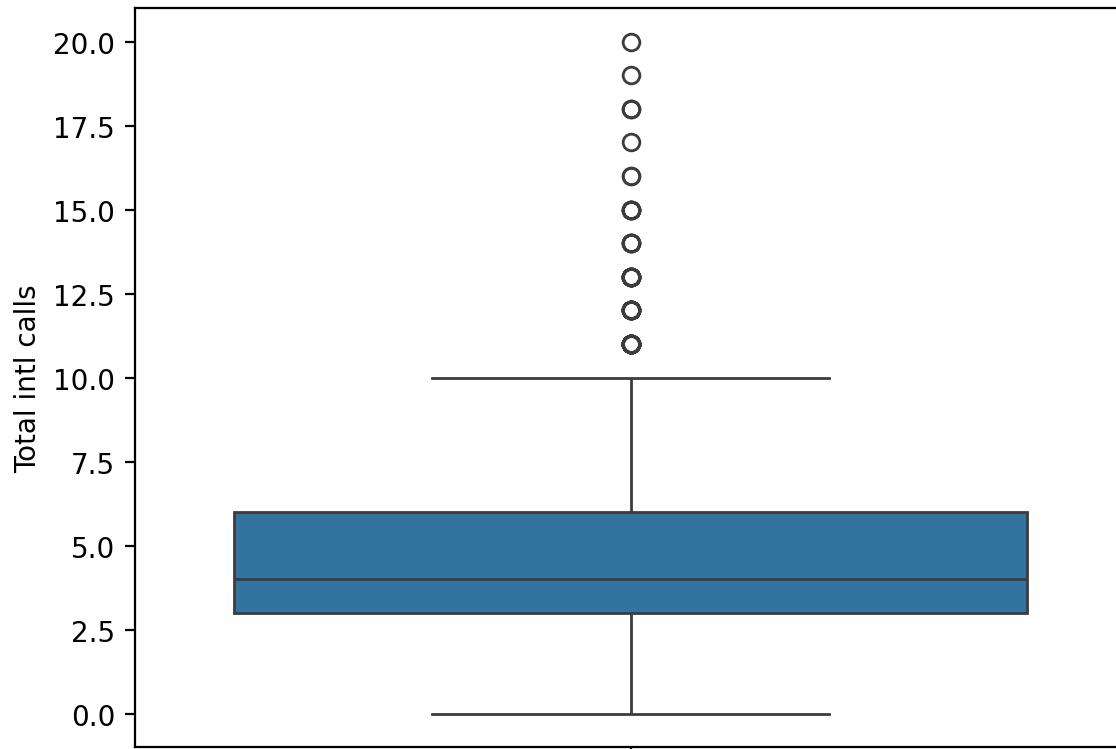
```
#Your answer here
sns.kdeplot(dataSet[['Total day minutes', 'Total intl calls']], shade=True, label='Total day minutes', color='blue')
sns.kdeplot(dataSet[['Total day minutes', 'Total intl calls']], shade=True, label='Total intl calls', color='orange')
```



Create a box plot to visualize the distribution of the "Total intl calls" feature

In [34]:

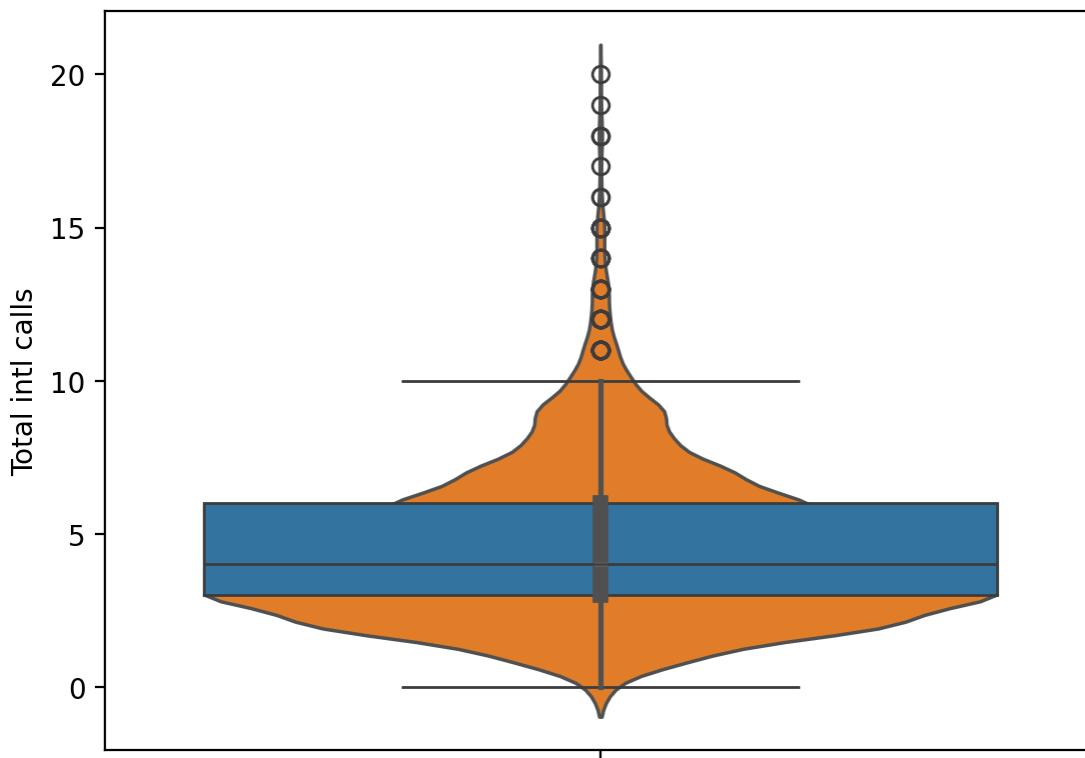
```
#Your answer here  
sns.boxplot(y='Total intl calls', data=dataSet);
```



Make a comparison between a box plot and a violin plot for visualizing the distribution of the "Total intl calls" feature.

In [35]:

```
#Your answer here  
sns.boxplot(y='Total intl calls', data=dataSet);  
sns.violinplot(y='Total intl calls', data=dataSet);
```

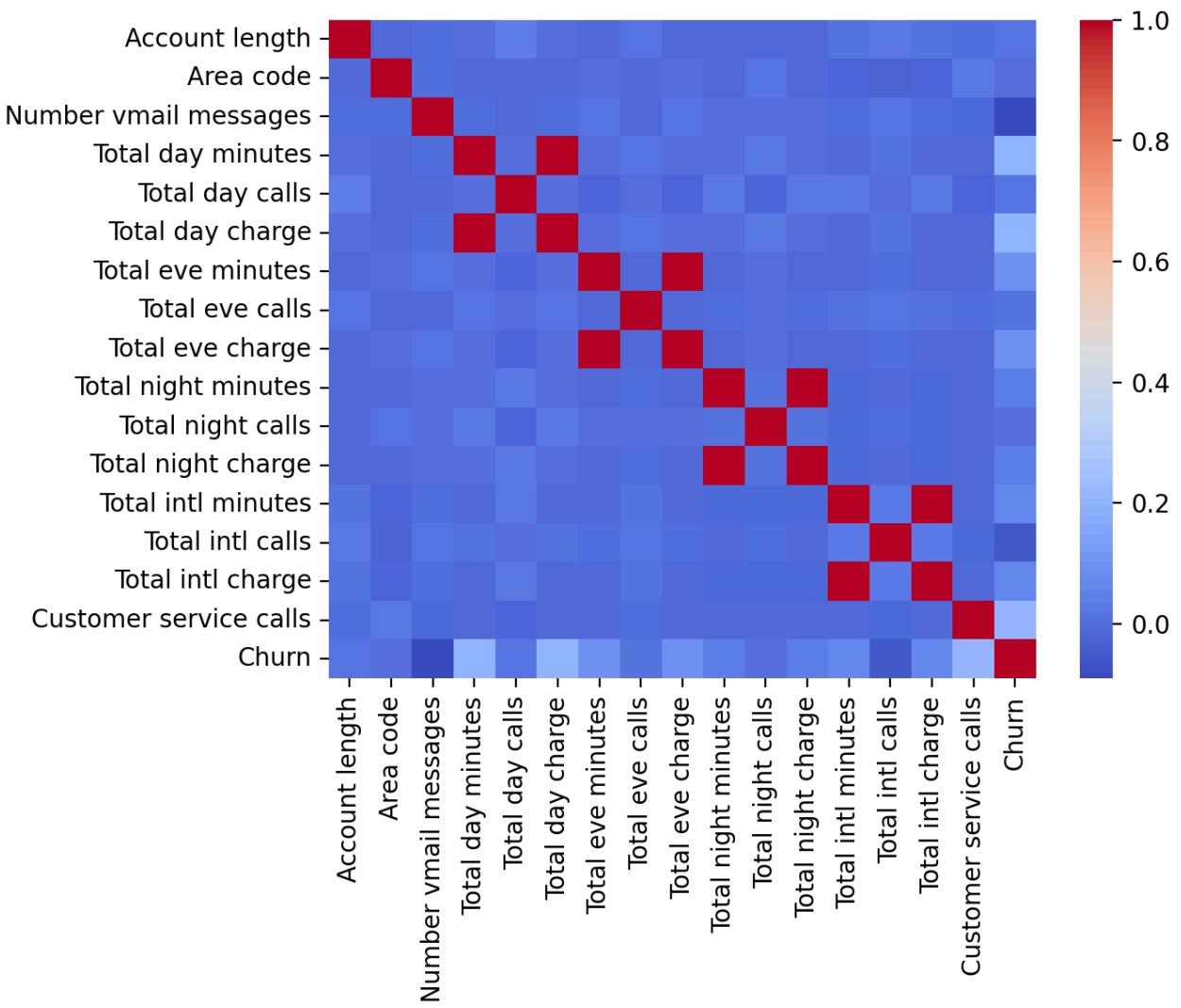


Multivariate visualization

Create a correlation matrix to visualize the relationships among numerical variables in a DataFrame, and display it using a heatmap?

In [36]:

```
#Your answer here
sns.heatmap(dataSet.select_dtypes(include=['int64', 'float64', 'bool']).corr(), cmap="co
```

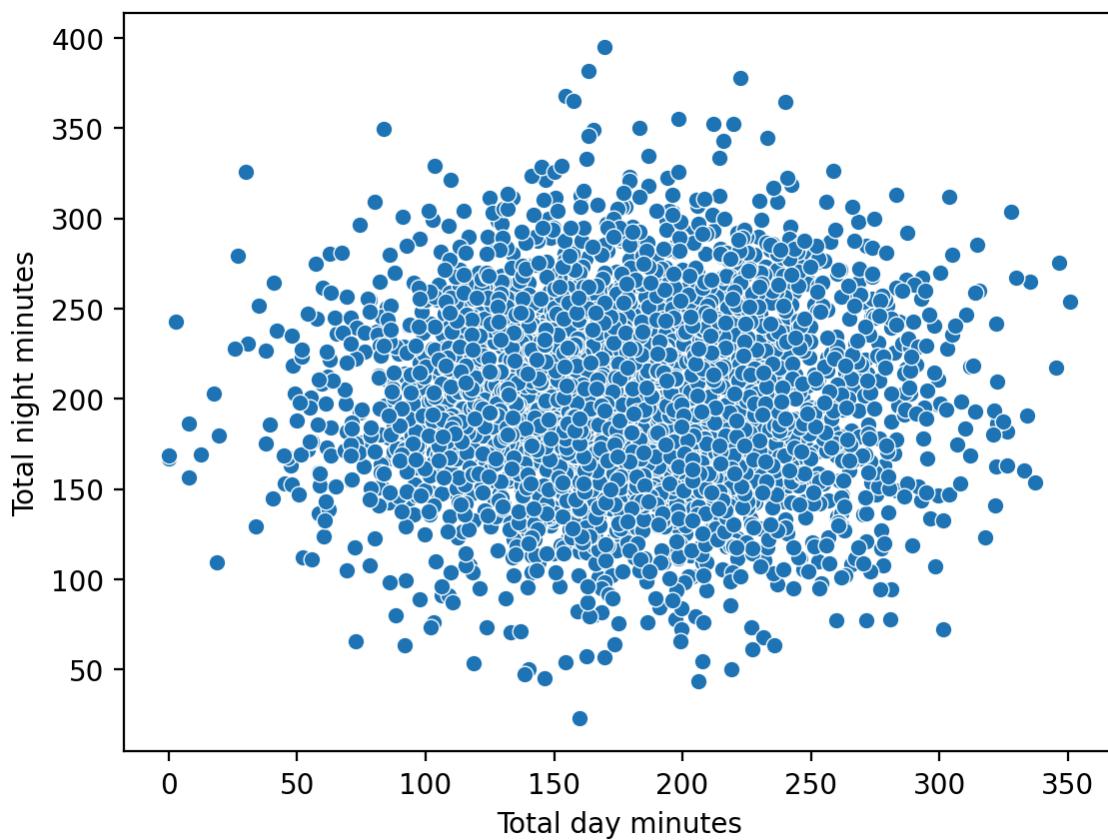


Scatter plot

Create a scatter plot to visualize the relationship between two numerical variables, such as "Total day minutes" and "Total night minutes,"?

In [37]:

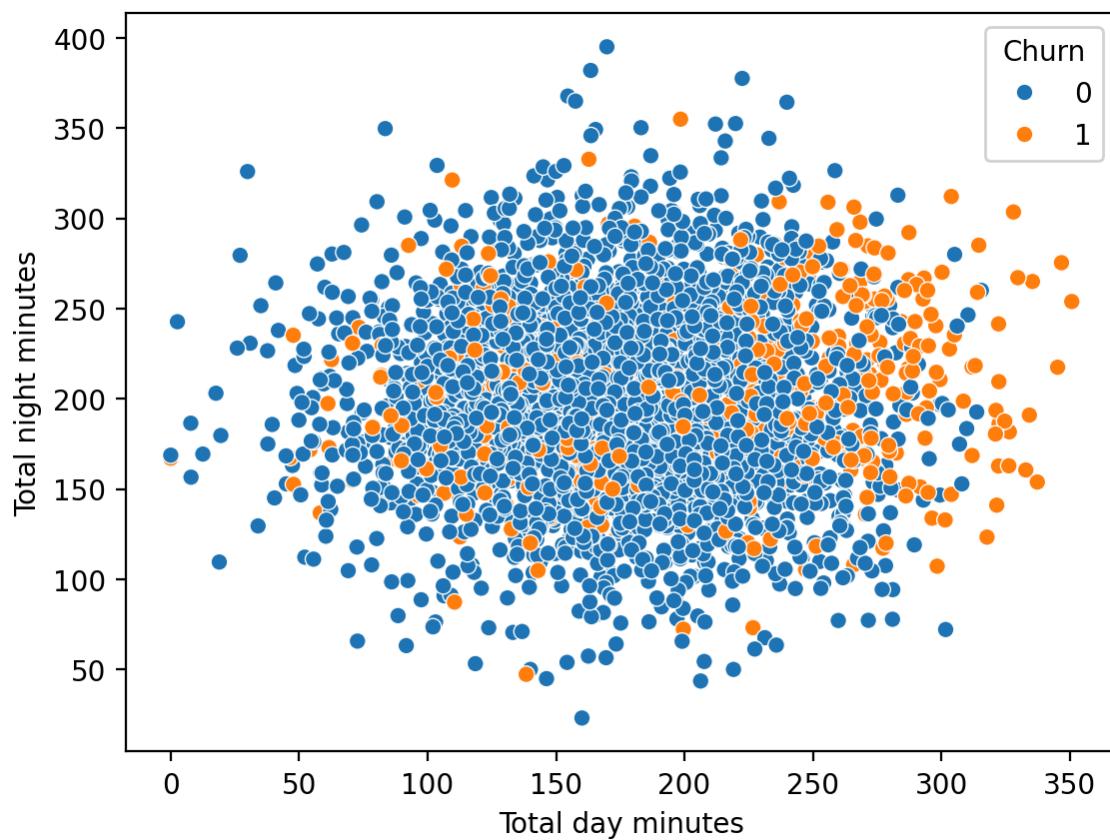
```
#Your answer here
sns.scatterplot(x='Total day minutes', y='Total night minutes', data=dataSet);
```



Create a scatter plot with different colors for the "Churn" categories to visualize the relationship between "Total day minutes" and "Total night minutes"

In [38]:

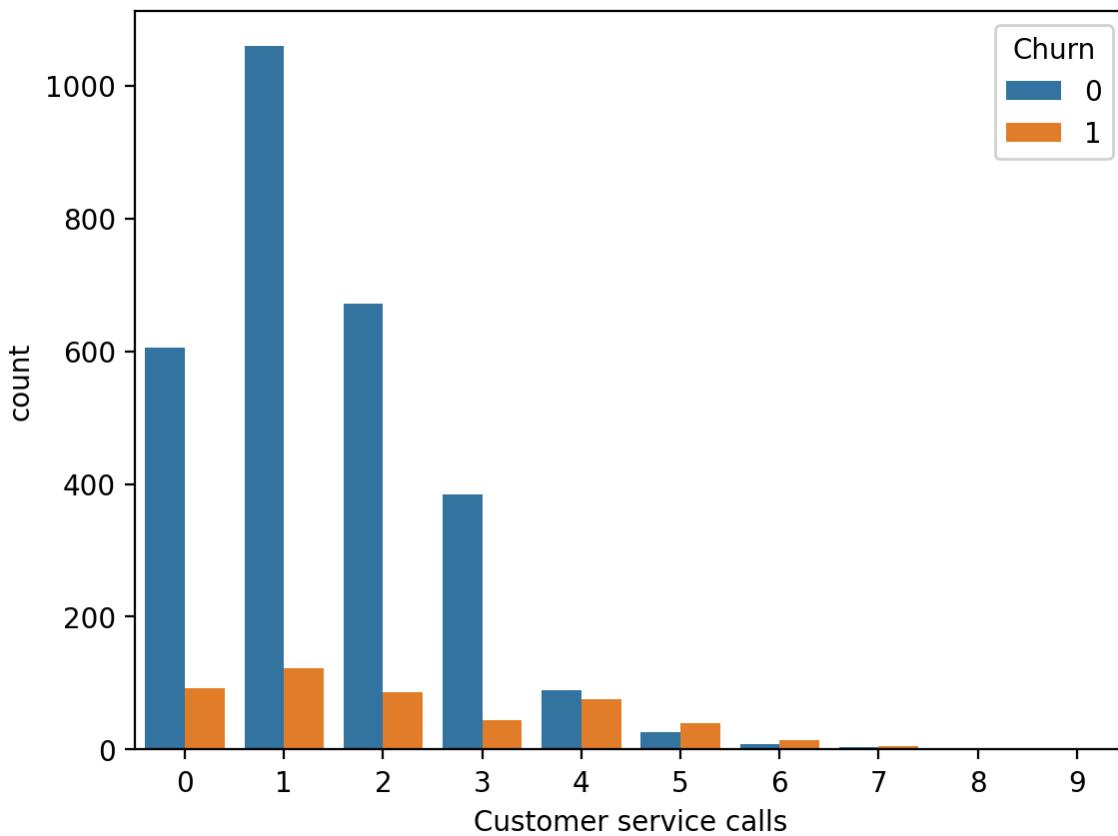
```
#Your answer here
sns.scatterplot(x='Total day minutes', y='Total night minutes', data=dataSet, hue='Chur
```



Create a count plot to visualize the distribution of the "Customer service calls" variable, with different colors representing the "Churn" categories.

In [39]:

```
#Your answer here
sns.countplot(x='Customer service calls', data=dataSet, hue='Churn');
```



Data Transformation

In [40]:

```
from sklearn.preprocessing import StandardScaler
```

Delete specific columns ("Churn", "State") from a DataFrame, convert binary categorical values ("Yes"/"No") into numerical values

In [41]:

```
#Your answer here
dataSet.drop(['State', 'Churn'], axis=1, inplace=True)
dataSet['International plan'] = dataSet['International plan'].map({'True': '1', 'False': '0'})
dataSet['Voice mail plan'] = dataSet['Voice mail plan'].map({'Yes': '1', 'No': '0'})
scaler = StandardScaler()
dataSet_scaled = scaler.fit_transform(dataSet)
dataSet_scaled
```

Out[41]:

```
array([[ 0.67648946, -0.52360328, -0.32758048, ..., -0.60119509,
       -0.0856905 , -0.42793202],
       [ 0.14906505, -0.52360328, -0.32758048, ..., -0.60119509,
       1.2411686 , -0.42793202],
       [ 0.9025285 , -0.52360328, -0.32758048, ...,  0.21153386,
       0.69715637, -1.1882185 ],
       ...,
       [-1.83505538,  1.71881732, -0.32758048, ...,  0.61789834,
        1.3871231 ,  0.33235445],
       [ 2.08295458,  1.71881732,  3.05268496, ...,  2.24335625,
       -1.87695028,  0.33235445],
       [-0.67974475, -0.52360328, -0.32758048, ..., -0.19483061,
       1.2411686 , -1.1882185 ]])
```

Now normalize the feature data by subtracting the mean and dividing by the standard deviation, and then convert the scaled data into a DataFrame

In [42]:

```
#Your answer here
dataSet_scaled = (dataSet_scaled - dataSet_scaled.mean(axis = 0))/dataSet_scaled.std(axis = 0)
dataSet_scaled = pd.DataFrame(dataSet_scaled, columns=dataSet.columns)
dataSet_scaled.head()
```

Out[42]:

	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 min
0	0.68	-0.52	-0.33	1.62	NaN	NaN	0.48	1.57	-0.07	-0.06	-0.07	1
1	0.15	-0.52	-0.33	1.62	NaN	NaN	1.12	-0.33	-0.11	0.14	-0.11	1
2	0.90	-0.52	-0.33	-0.62	NaN	NaN	0.68	1.17	-1.57	0.50	-1.57	-1
3	-0.43	-0.69	3.05	-0.62	NaN	NaN	-1.47	2.20	-2.74	-0.61	-2.74	-1
4	-0.65	-0.52	3.05	-0.62	NaN	NaN	0.63	-0.24	-1.04	1.10	-1.04	-1

Splitting a Dataset into Train & Test Set

How can you split a dataset into training and testing sets, with 25% of the data reserved for testing?

In [43]:

```
dataSet = pd.read_csv(r"C:\Users\Laptop\OneDrive - National Institute of Technology, Rourkela\7th-sem\ds\week2\Churn.csv")
dataSet['International plan'] = dataSet['International plan'].map({'Yes': '1', 'No': '0'})
dataSet['Voice mail plan'] = dataSet['Voice mail plan'].map({'Yes': '1', 'No': '0'})

print(dataSet.shape)
```

(3333, 20)

In [44]:

```
#Your answer here
train_index = int(dataSet.shape[0] * 0.75)
print(train_index)
# predict churn
X_train = dataSet.drop(['Churn', 'State'], axis=1).iloc[:train_index, :]
X_train = X_train
print(X_train.shape)
y_train = dataSet['Churn'].iloc[:train_index]

X_test = dataSet.drop(['Churn', 'State'], axis=1).iloc[train_index:, :]
y_test = dataSet['Churn'].iloc[train_index:]

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
2499  
(2499, 18)  
(2499, 18) (2499,)  
(834, 18) (834,)
```

Classification using Machine Learning

In [45]:

```
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import accuracy_score
```

Train a Decision Tree classifier on the training data and make predictions on the test data

In [46]:

```
#Your answer here  
tree = DecisionTreeClassifier(random_state=17)  
tree.fit(X_train, y_train)  
tree_predictions = tree.predict(X_test)  
  
score = accuracy_score(y_test, tree_predictions)  
print("Accuracy: ", score)
```

Accuracy: 0.9268585131894485

Some additional questions

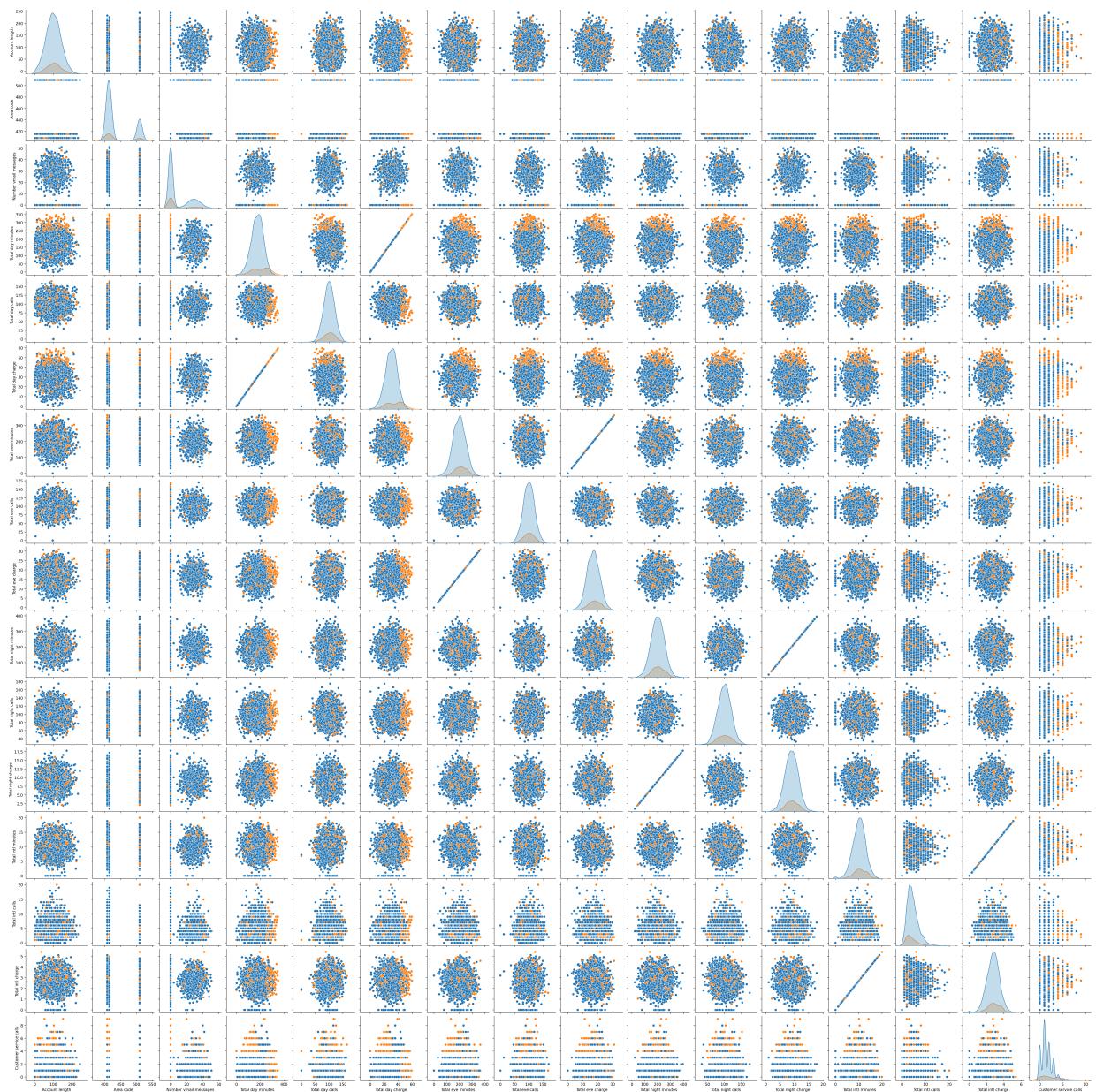
- 1) If you will be asked to choose a single feature for classification which one you will choose. Justify with analysis.

In [47]:

```
# data = dataSet.drop(['State'], axis=1)  
data = dataSet.drop(['International plan', 'Voice mail plan', 'State'], axis=1)
```

In [48]:

```
#Your answer here  
sns.pairplot(data, hue='Churn');
```



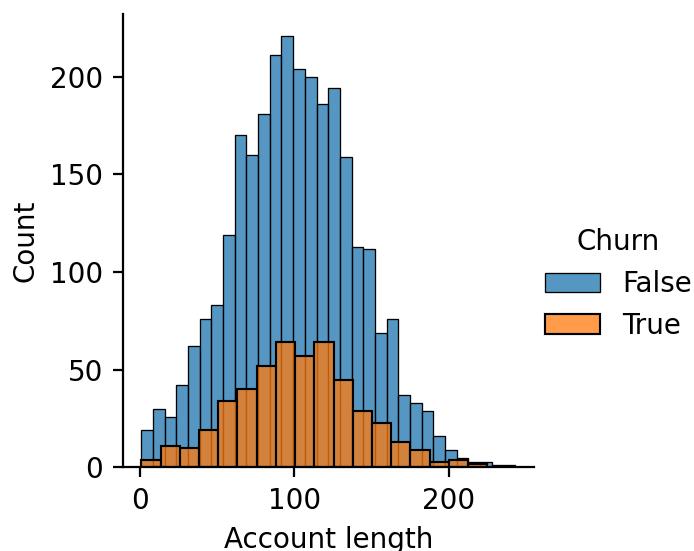
In [49]:

```

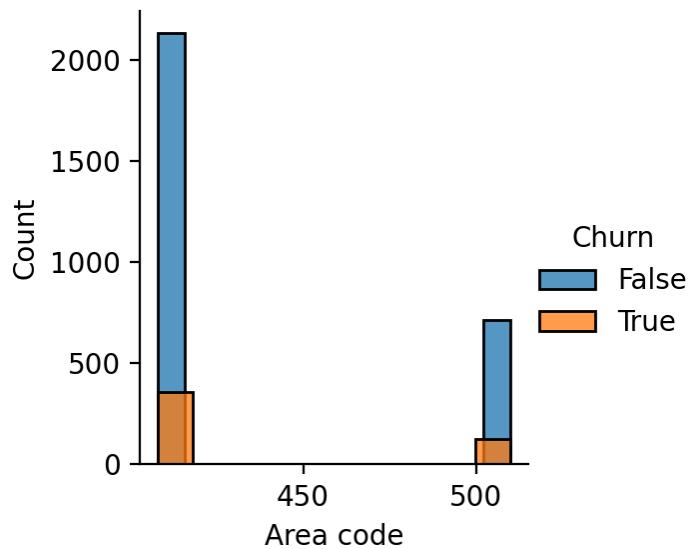
for col in data.columns:
    if col == 'Churn':
        continue
    plt.figure()
    sns.FacetGrid(data, hue='Churn') \
        .map(sns.histplot, col) \
        .add_legend()

```

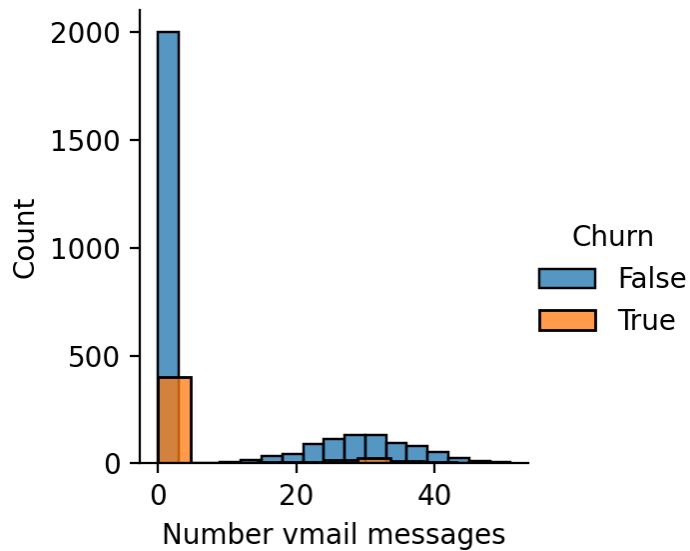
<Figure size 640x480 with 0 Axes>



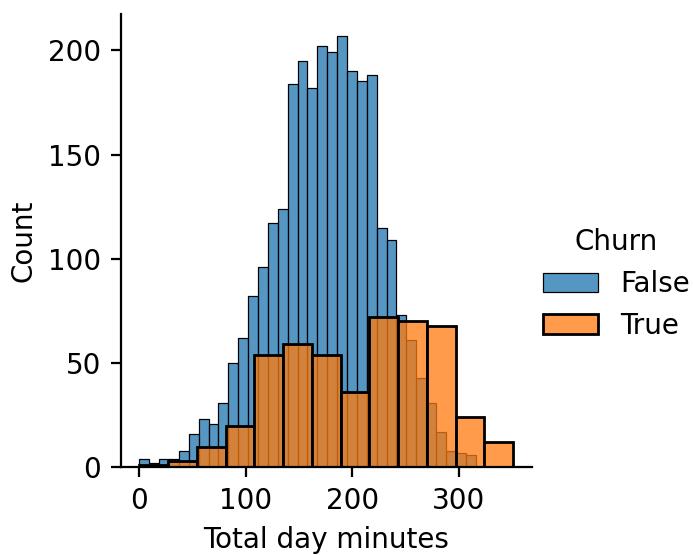
<Figure size 640x480 with 0 Axes>



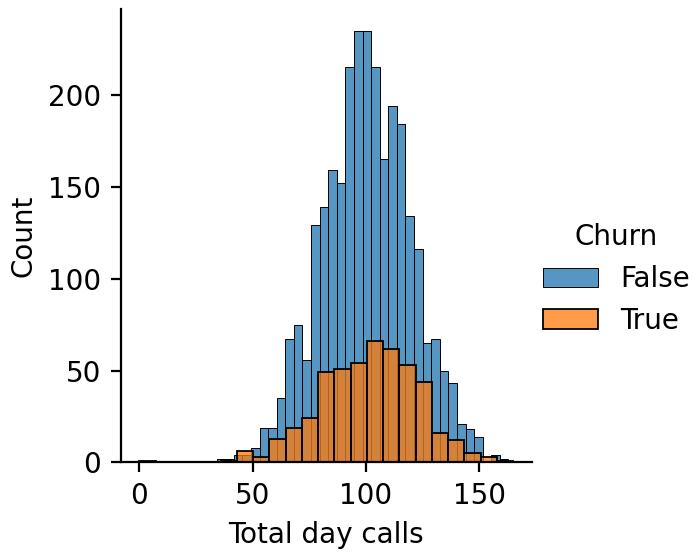
<Figure size 640x480 with 0 Axes>



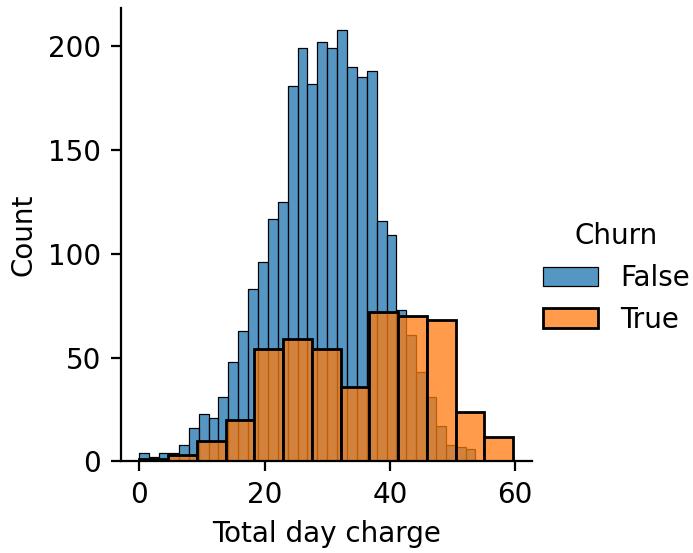
<Figure size 640x480 with 0 Axes>



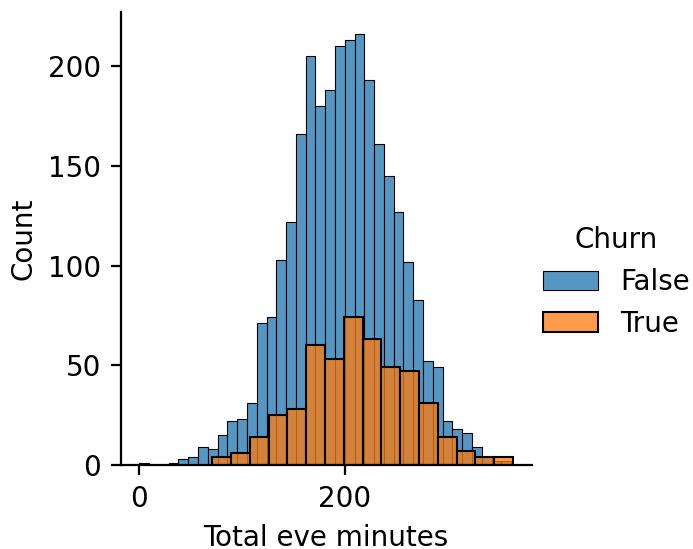
<Figure size 640x480 with 0 Axes>



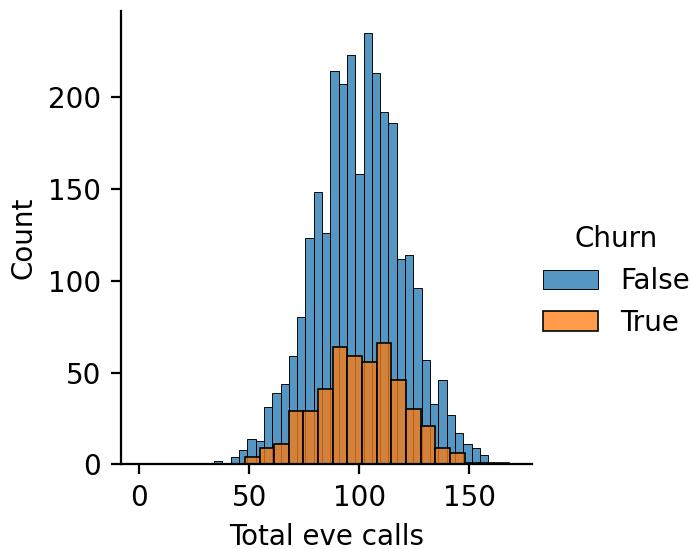
<Figure size 640x480 with 0 Axes>



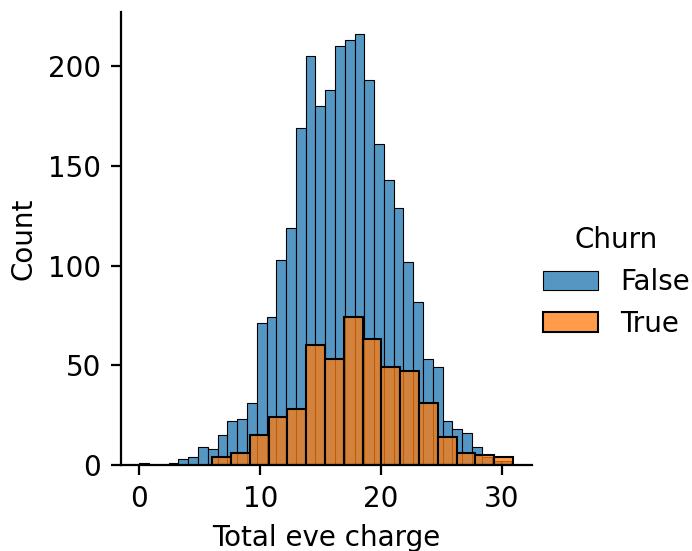
<Figure size 640x480 with 0 Axes>



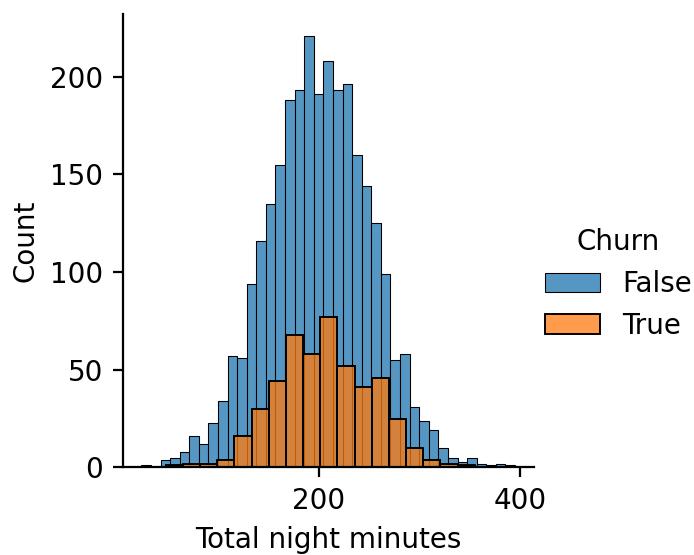
<Figure size 640x480 with 0 Axes>



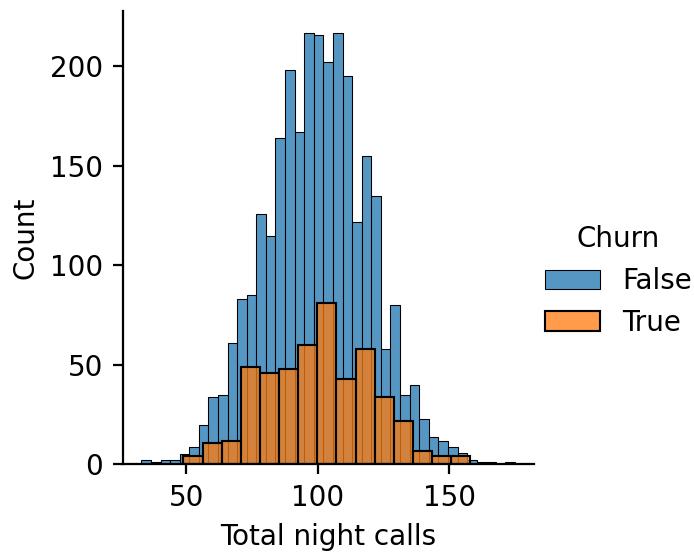
<Figure size 640x480 with 0 Axes>



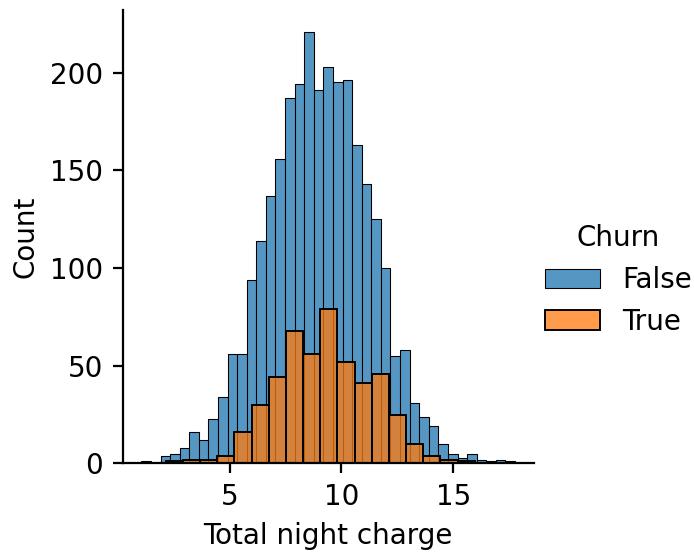
<Figure size 640x480 with 0 Axes>



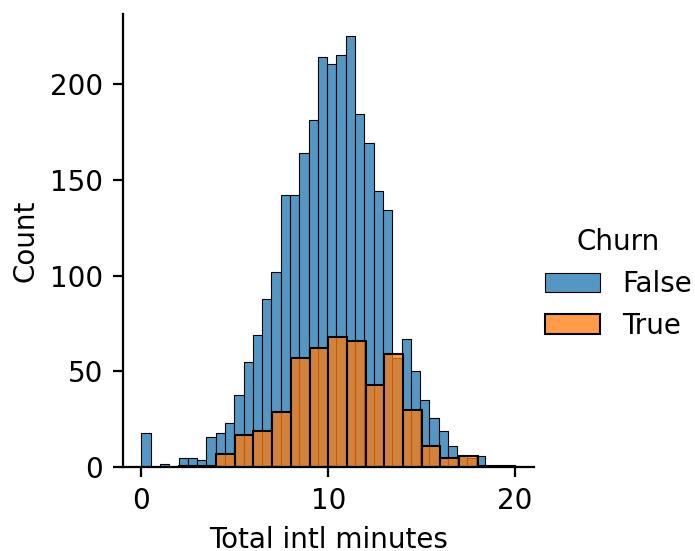
<Figure size 640x480 with 0 Axes>



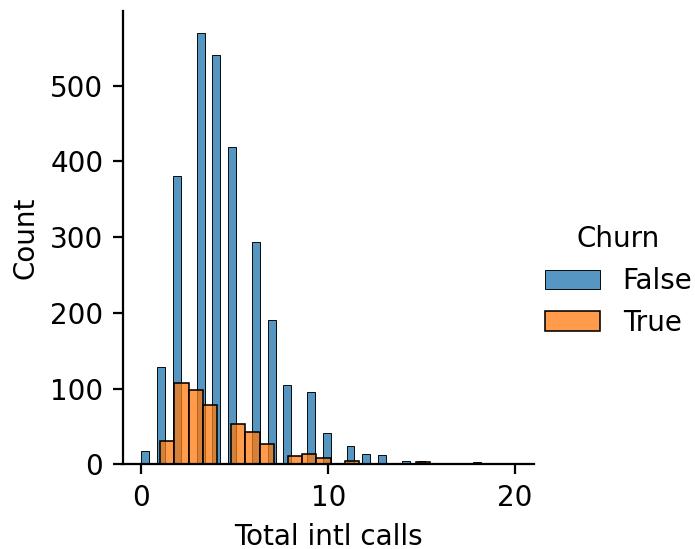
<Figure size 640x480 with 0 Axes>



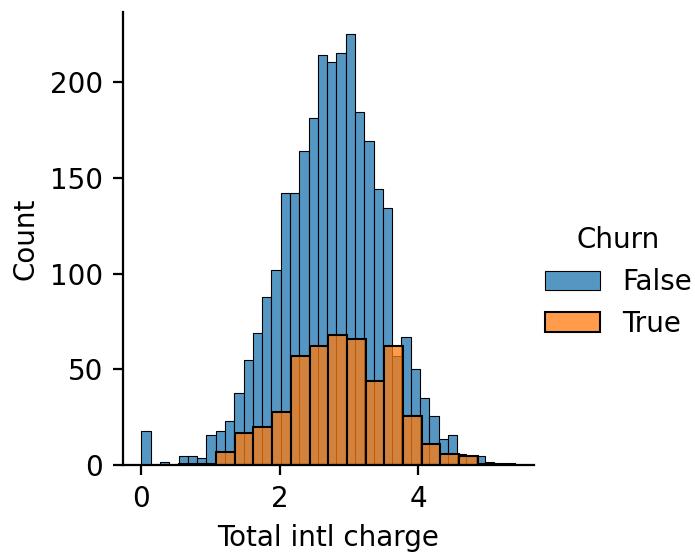
<Figure size 640x480 with 0 Axes>



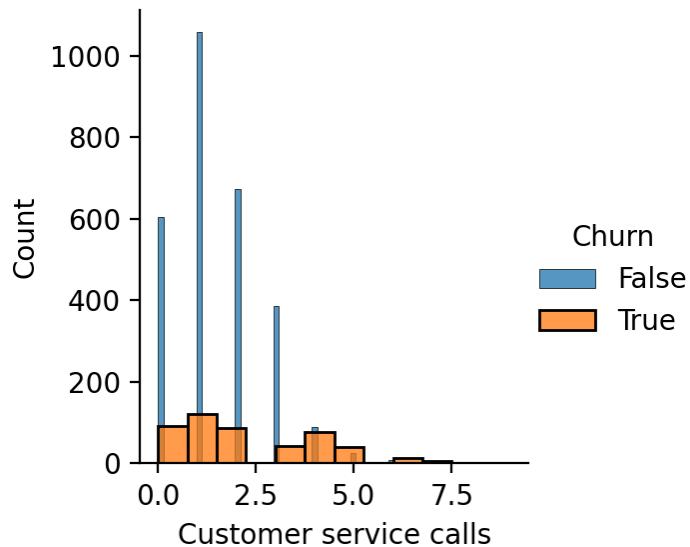
<Figure size 640x480 with 0 Axes>



<Figure size 640x480 with 0 Axes>



<Figure size 640x480 with 0 Axes>

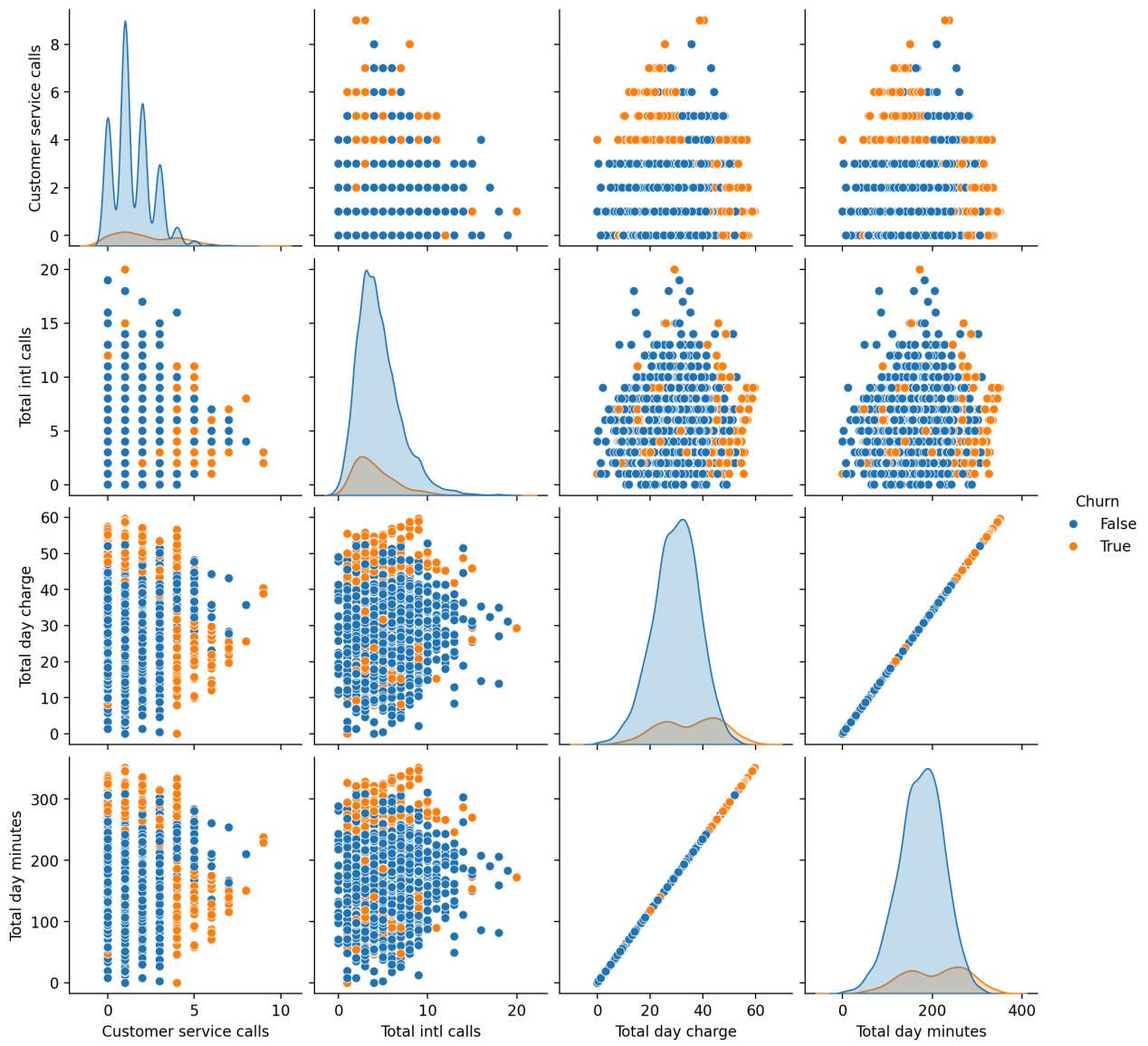


From the histograms we can see total day charge and customer service calls have the least overlapping so if i had to select 1 feature then i would select Customer Service calls

2) If you will be asked to choose two features for classification which two you will choose. Justify with analysis.

In [50]:

```
#Your answer here  
sns.pairplot(data[['Customer service calls', 'Churn', 'Total intl calls', 'Total day cha
```



From the pair plot we can see total day charge and customer service calls have the least overlapping so if i had to select 2 feature then i would select these two