In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("telecom_churn.csv")
```

In [3]:
```
# head() : display first 5 rows of data with all columns

df.head()
```

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
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

In [4]:
```
df.head(15)
```

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
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
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
7	MO	147	415	Yes	No	0	157.0	79	26.69	103.1
8	LA	117	408	No	No	0	184.5	97	31.37	351.6
9	WV	141	415	Yes	Yes	37	258.6	84	43.96	222.0
10	IN	65	415	No	No	0	129.1	137	21.95	228.5
11	RI	74	415	No	No	0	187.7	127	31.91	163.4
12	IA	168	408	No	No	0	128.8	96	21.90	104.9
13	MT	95	510	No	No	0	156.6	88	26.62	247.6
14	IA	62	415	No	No	0	120.7	70	20.52	307.2

In [5]:
```
# shape : gives count of rows and columns

df.shape
```

Out[5]: (3333, 20)

In [6]:
```
# size : number of data points (row * columns)

df.size
```

Out[6]: 66660

In [7]:
```
# tail() : display last 5 rows of data with all columns
df.tail()
```

Out[7]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	To e minut
3328	AZ	192	415	No	Yes	36	156.2	77	26.55	215
3329	WV	68	415	No	No	0	231.1	57	39.29	153
3330	RI	28	510	No	No	0	180.8	109	30.74	288
3331	CT	184	510	Yes	No	0	213.8	105	36.35	159
3332	TN	74	415	No	Yes	25	234.4	113	39.85	265

In [8]:
```
df.tail(7) # display last 7 rows of my table
```

Out[8]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	To e minut
3326	OH	96	415	No	No	0	106.6	128	18.12	284
3327	SC	79	415	No	No	0	134.7	98	22.90	189
3328	AZ	192	415	No	Yes	36	156.2	77	26.55	215
3329	WV	68	415	No	No	0	231.1	57	39.29	153
3330	RI	28	510	No	No	0	180.8	109	30.74	288
3331	CT	184	510	Yes	No	0	213.8	105	36.35	159
3332	TN	74	415	No	Yes	25	234.4	113	39.85	265

In [9]:
```
# columns : to view all columns

df.columns
```

Out[9]:
```
Index(['State', 'Account length', 'Area code', 'International plan
',
       'Voice mail plan', 'Number vmail messages', 'Total day minu
tes',
       '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 minu
tes',
       'Total intl calls', 'Total intl charge', 'Customer service
calls',
       'Churn'],
      dtype='object')
```

In [10]:
```
# info : return information related to individual columns

df.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   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
```

In [11]:
```
# describe : it gibes statistical information present in the da

df.describe()
```

Out[11]:

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	T
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	333
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	20(
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	5(
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	(
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	16(
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	20
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	23
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	36

In [12]:
```
# describe on categorical values
df.describe(include = ["object","bool"])
```

Out[12]:

	State	International plan	Voice mail plan	Churn
count	3333	3333	3333	3333
unique	51	2	2	2
top	WV	No	No	False
freq	106	3010	2411	2850

In [13]:
```
# value_counts() : it gives unique data points and its count in

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

Out[13]:
```
False    2850
True      483
Name: Churn, dtype: int64
```

In [14]:
```
df['International plan'].value_counts()
```

Out[14]:
```
No     3010
Yes     323
Name: International plan, dtype: int64
```

In [15]:
```
# number of rows
df.shape[0]
```

Out[15]:
```
3333
```

In [16]:
```
# value_counts in precentage

(df['Churn'].value_counts()/df.shape[0])*100
```

Out[16]:
```
False    85.508551
True     14.491449
Name: Churn, dtype: float64
```

In [17]:
```
# value_counts in normalized form
df['Churn'].value_counts(normalize = True)
```

Out[17]:
```
False    0.855086
True     0.144914
Name: Churn, dtype: float64
```

In [18]:
```
# indexing using loc : provide row index and column names as an

df.loc[0:10,"State":"International plan"]
```

Out[18]:

	State	Account length	Area code	International plan
0	KS	128	415	No
1	OH	107	415	No
2	NJ	137	415	No
3	OH	84	408	Yes
4	OK	75	415	Yes
5	AL	118	510	Yes
6	MA	121	510	No
7	MO	147	415	Yes
8	LA	117	408	No
9	WV	141	415	Yes
10	IN	65	415	No

In [19]:
```
df.loc[0:10,["State","International plan"]]
```

Out[19]:

	State	International plan
0	KS	No
1	OH	No
2	NJ	No
3	OH	Yes
4	OK	Yes
5	AL	Yes
6	MA	No
7	MO	Yes
8	LA	No
9	WV	Yes
10	IN	No

In [20]:
```
# indexing using iloc : slicing based on index
df.iloc[0:10,0:4]
```

Out[20]:

	State	Account length	Area code	International plan
0	KS	128	415	No
1	OH	107	415	No
2	NJ	137	415	No
3	OH	84	408	Yes
4	OK	75	415	Yes
5	AL	118	510	Yes
6	MA	121	510	No
7	MO	147	415	Yes
8	LA	117	408	No
9	WV	141	415	Yes

In [21]:
```
# sort_values : sort dataframe by column name

df.sort_values(by = 'State',ascending = False).head()
```

Out[21]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	To e minut
273	WY	139	415	No	No	0	192.8	104	32.78	234
731	WY	90	415	No	No	0	207.2	121	35.22	292
2912	WY	151	415	No	No	0	170.2	89	28.93	187
1628	WY	131	510	No	No	0	110.9	74	18.85	115
2915	WY	58	510	No	No	0	210.1	126	35.72	248

In [22]:
```
df.sort_values(by = 'Area code').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	To e minut
2536	CT	119	408	No	No	0	294.2	100	50.01	232
887	IA	128	408	No	No	0	158.8	75	27.00	264
2486	MS	76	408	No	No	0	173.2	93	29.44	131
2482	MT	157	408	No	No	0	240.2	67	40.83	153
2476	WV	84	408	No	No	0	146.8	133	24.96	171

In [23]:
```
# sort on multiple column
df.sort_values(by = ['Area code','State']).head()
```

Out[23]:

ount ngth	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
36	408	No	Yes	30	146.3	128	24.87	162.5	80	13.81
104	408	No	No	0	278.4	106	47.33	81.0	113	6.89
78	408	No	No	0	225.1	67	38.27	199.2	127	16.93
110	408	No	No	0	100.1	90	17.02	233.3	93	19.83
127	408	No	No	0	182.3	124	30.99	169.9	110	14.44

In [24]:
```
# isnull().sum() : it gives sum of missing values of an individ

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

Out[24]:
```
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 [25]:
```
df_t = pd.read_csv("titanic_train.csv")
df_t.head()
```

Out[25]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [26]:
```
df_t.shape
```

Out[26]:
```
(891, 12)
```

In [27]:
```
df_t.isnull().sum()
```

Out[27]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [28]:
```
(df_t.isnull().sum() / df_t.shape[0])*100
```

Out[28]:
```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [29]:
```
sum(df_t.isnull().sum())
```

Out[29]:
```
866
```

In [30]:
```
import numpy as np
```

In [31]:
```
# apply() : useful when we wants to perform some operation over

df.apply(np.max)
```

Out[31]:
```
State                        WY
Account length              243
Area code                   510
International plan          Yes
Voice mail plan             Yes
Number vmail messages        51
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
Total night calls           175
Total night charge        17.77
Total intl minutes           20
Total intl calls             20
Total intl charge           5.4
Customer service calls        9
Churn                      True
dtype: object
```

In [32]:
```
df.head()
```

Out[32]:

	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

In [33]:
```
new = df['Area code'].apply(lambda x : x + 1000)
```

In [34]:
```
new['area_changed'] = df['Area code'].apply(lambda x : x + 1000
```

In [35]:
```
#new.head()
```

In [36]:
```
#new_df = pd.DataFrame(new,columns = ['Index','Area_changed'])
```

In [ ]:
```

```

In [37]:
```
type(new)
```

Out[37]:
```
pandas.core.series.Series
```

In [38]:
```
# subset of data using conditions ( one condition)

df_filter = df[df['Churn'] == True]
```

In [39]:
```
df_filter.head()
```

Out[39]:

	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

In [40]:
```
df_filter['Churn'].value_counts()
```

Out[40]:
```
True    483
Name: Churn, dtype: int64
```

In [41]:
```
# subset of data using conditions (multiple condition)

df_filter_multiple = df[(df['Churn'] == True) & (df['Area code'
df_filter_multiple.head()
```

Out[41]:

	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
48	ID	119	415	No	No	0	159.1	114	27.05	231.3
54	WY	87	415	No	No	0	151.0	83	25.67	219.7
76	DC	82	415	No	No	0	300.3	109	51.05	181.0

In [42]:
```
df_filter_multiple.shape[0]
```

Out[42]:
```
236
```

In [43]:
```
df_filter_multiple['Area code'].value_counts()
```

Out[43]:
```
415    236
Name: Area code, dtype: int64
```

In [44]:
```
# crosstab() : it gives calculation between two or more columns
pd.crosstab(df['Churn'], df['International plan'])
```

Out[44]:

International plan	No	Yes
Churn		
False	2664	186
True	346	137

In [45]:
```
pd.crosstab(df['Churn'], df['Area code'])
```

Out[45]:

Area code	408	415	510
Churn			
False	716	1419	715
True	122	236	125

In [46]:
```
pd.crosstab(df['Churn'], [df['Area code'],df['International pla
```

Out[46]:

Area code	408		415		510	
International plan	No	Yes	No	Yes	No	Yes
Churn						
False	677	39	1331	88	656	59
True	90	32	174	62	82	43

In [ ]:
```

```

In [47]:
```
# groupby

# df.groupby(grouping columns)[columns_to_show].function()
```

In [51]:
```
columns_to_show = ['Total day minutes',"Total eve minutes"]
df.groupby(['Churn'])[columns_to_show].mean()
```

Out[51]:

	Total day minutes	Total eve minutes
Churn		
False	175.175754	199.043298
True	206.914079	212.410145

In [ ]:
```
# data  - churn_false - total day min col - take maximum of all
# data  - churn_true - total day min col - take maximum of all
```

In [59]:
```
columns_to_show = ['Total day minutes',"Total eve minutes"]
df.groupby(['Area code'])[columns_to_show].mean()
```

Out[59]:

Area code	Total day minutes	Total eve minutes
408	177.175418	201.284248
415	181.592628	200.652085
510	178.787619	201.323929

In [55]:
```
columns_to_show = ['Total intl calls']
df.groupby(['Churn','Area code'])[columns_to_show].sum()
```

Out[55]:

Churn	Area code	Total intl calls
False	**408**	3208
	415	6537
	510	3174
True	**408**	501
	415	1013
	510	497

In [ ]:
```
# pivot table : similar to groupby
```

In [58]:
```
df.pivot_table(['Total day minutes',"Total eve minutes"],['Area
```

Out[58]:

Area code	Total day minutes	Total eve minutes
408	177.175418	201.284248
415	181.592628	200.652085
510	178.787619	201.323929

In [ ]:
```

```

In [ ]:
```
# concat  : joining two dataframes
```

In [61]:
```
series1 = pd.Series([1,2,3])
series2 = pd.Series(['A','B','C'])
```

In [62]:
```
series1
```

Out[62]:
```
0    1
1    2
2    3
dtype: int64
```

In [63]:
```
series2
```

Out[63]:
```
0    A
1    B
2    C
dtype: object
```

In [64]:
```
# normal concat
pd.concat([series1,series2])
```

Out[64]:
```
0    1
1    2
2    3
0    A
1    B
2    C
dtype: object
```

In [65]:
```
# Horizonal concat
pd.concat([series1,series2],axis = 1)
```

Out[65]:

	0	1
0	1	A
1	2	B
2	3	C

In [ ]:
```
# concatenation on dataframe
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
# merge
```

In [66]:

```
import pandas as pd

data1 = {
  "name": ["Sally", "Mary", "John"],
  "age": [50, 40, 30]
}

data2 = {
  "name": ["Sally", "Peter", "Micky"],
  "age": [77, 44, 22]
}

df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

```

In [67]:

```
df1
```

Out[67]:

	name	age
0	Sally	50
1	Mary	40
2	John	30

In [68]:

```
df2
```

Out[68]:

	name	age
0	Sally	77
1	Peter	44
2	Micky	22

In [69]:

```

newdf = df1.merge(df2, how='right')

print(newdf)
```

```
    name  age
0  Sally   77
1  Peter   44
2  Micky   22
```

In [70]:

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'key': ['K0', 'K1', 'K2', 'K3'],
         'key1': ['K0', 'K1', 'K0', 'K1'],
         'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],}

# Define a dictionary containing employee data
data2 = {'key': ['K0', 'K1', 'K2', 'K3'],
         'key1': ['K0', 'K0', 'K0', 'K0'],
         'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1)

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2)


print(df, "\n\n", df1)
```

```
  key key1    Name  Age
0  K0   K0     Jai   27
1  K1   K1  Princi   24
2  K2   K0  Gaurav   22
3  K3   K1    Anuj   32

   key key1    Address Qualification
0  K0   K0     Nagpur         Btech
1  K1   K0     Kanpur           B.A
2  K2   K0  Allahabad          Bcom
3  K3   K0    Kannuaj        B.hons
```

In [71]:

```
# diff merge methods :

# left : use keys from left dataframe only
# right : use keys from right dataframe only
# outer : used union from both dataframes
# inner : used intersection from both dataframes
```

In [72]:

```
# merging using "left"

output = pd.merge(df, df1, how = 'left',on = ["key","key1"])
```

In [73]:
```
output
```

Out[73]:

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27	Nagpur	Btech
1	K1	K1	Princi	24	NaN	NaN
2	K2	K0	Gaurav	22	Allahabad	Bcom
3	K3	K1	Anuj	32	NaN	NaN

In [74]:
```
# merging using "right"

output_left = pd.merge(df, df1, how = 'right',on = ["key","key1
output_left
```

Out[74]:

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27.0	Nagpur	Btech
1	K2	K0	Gaurav	22.0	Allahabad	Bcom
2	K1	K0	NaN	NaN	Kanpur	B.A
3	K3	K0	NaN	NaN	Kannuaj	B.hons

In [75]:
```
# merging using "outer"

output_outer = pd.merge(df, df1, how = 'outer',on = ["key","key
output_outer
```

Out[75]:

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27.0	Nagpur	Btech
1	K1	K1	Princi	24.0	NaN	NaN
2	K2	K0	Gaurav	22.0	Allahabad	Bcom
3	K3	K1	Anuj	32.0	NaN	NaN
4	K1	K0	NaN	NaN	Kanpur	B.A
5	K3	K0	NaN	NaN	Kannuaj	B.hons

In [76]:
```
# merging using "inner"

output_inner = pd.merge(df, df1, how = 'inner',on = ["key","key
output_inner
```

Out[76]:

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27	Nagpur	Btech
1	K2	K0	Gaurav	22	Allahabad	Bcom

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```