```
# EDA with Pandas
'''
> an approach to analyzing data sets to summarize their main characteristics
> way of story-telling : explore data and find patterns
> approach for data analysis that empploys a variety of techniques to :-
  - maximize insight into data set
  - uncover underlying structure
  - extract important variables
  - detect outliers and anomalies
  - test underlying assumptions
> a process of organizing, plotting and summarizing a data set

'''
print('\n')

```

```
# Import the Libraries
import numpy as np
import pandas as pd
```

```
# Load The Dataset
df = pd.read_csv('https://trello-attachments.s3.amazonaws.com/600d1f10d700af20b1924b3c/600d1f700fcd0
df.head()
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	c
0	KS	128	415	382-4657	no	yes	25	265.1	110	
1	OH	107	415	371-7191	no	yes	26	161.6	123	
2	NJ	137	415	358-1921	no	no	0	243.4	114	
3	OH	84	408	375-9999	yes	no	0	299.4	71	
4	OK	75	415	330-6626	yes	no	0	166.7	113	

```
# Retrieve the columns names
df.columns              # churn : false means active user
```

```
Index(['state', 'account length', 'area code', 'phone number',
       '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')
```

```
# Check dimensions of data shape
df.shape                # 3333 = rows, 21 = columns
```

```
(3333, 21)
```

```
# Info method - generate full summary of dataset
df.info()     # help analyse data in easy manner
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 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   phone number            3333 non-null   object 
 4   international plan      3333 non-null   object 
 5   voice mail plan         3333 non-null   object 
 6   number vmail messages   3333 non-null   int64  
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64  
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64  
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64  
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64  
 18  total intl charge       3333 non-null   float64
 19  customer service calls  3333 non-null   int64  
 20  churn                   3333 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
```

```
# Approach : astype(), Changing the dtype from a column
df['churn'] = df['churn'].astype('int64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 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   phone number            3333 non-null   object 
 4   international plan      3333 non-null   object 
 5   voice mail plan         3333 non-null   object 
 6   number vmail messages   3333 non-null   int64  
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64  
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64  
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64  
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64  
 18  total intl charge       3333 non-null   float64
 19  customer service calls  3333 non-null   int64  
 20  churn                   3333 non-null   int64  
dtypes: float64(8), int64(9), object(4)
memory usage: 546.9+ KB
```

```
# describe()
'''
> generate statistical summary of dataset
> by default only works on int and float val
'''
df.describe()
```

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435
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
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000

```
# To generate statistical of Object(non numerical)
'''
> need to specify some Parameters
> in describe(?)

'''
df.describe(include=['object','bool'])
```

	state	phone number	international plan	voice mail plan
count	3333	3333	3333	3333

```
# What is unique, top, frequency?
'''
>·unique······:·total·no.·of·unique·val
>·top·········:·most·repeated·character
>·frequency···:·the·frequency·of·repeated·character
'''
data = pd.Series(['A','B','C','D','E','C','B','C','B','A'])
data.describe()
```

```
count     10
unique     5
top        B
freq       3
dtype: object
```

```
# Check how many Churn user using values_counts()
df['churn'].value_counts()
```

```
0    2850
1     483
Name: churn, dtype: int64
```

```
# Print the top 3
df.head(3)
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	c
0	KS	128	415	382-4657	no	yes	25	265.1	110	
1	OH	107	415	371-7191	no	yes	26	161.6	123	
2	NJ	137	415	358-1921	no	no	0	243.4	114	

```
# Sorting the database
'''
> sort_values(by='specify column')
> by default in ascending order
'''
df.sort_values(by='total day charge').head(10)
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls
1345	SD	98	415	392-2555	no	no	0	0.0	0
1397	VT	101	510	413-7655	no	no	0	0.0	0
2736	OK	127	510	403-1128	no	yes	27	2.6	113
2753	OH	134	415	406-4158	no	no	0	7.8	86
1986	WI	70	415	405-9233	no	no	0	7.9	100
1052	OR	98	415	378-6772	yes	no	0	12.5	67
2252	NH	148	408	333-7449	no	no	0	17.6	121
3046	MI	110	510	357-5784	no	no	0	18.9	92
1621	SC	138	510	370-9533	no	yes	21	19.5	149

```
# Sorting in descending order
df.sort_values(by='total day charge',ascending=False).head(10)
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls
				343-					

```
# Sorting multiple columns
df.sort_values(by=['total day minutes','total day calls']).head(10)
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls
1345	SD	98	415	392-2555	no	no	0	0.0	0
1397	VT	101	510	413-7655	no	no	0	0.0	0
2736	OK	127	510	403-1128	no	yes	27	2.6	113
2753	OH	134	415	406-4158	no	no	0	7.8	86
1986	WI	70	415	405-9233	no	no	0	7.9	100
1052	OR	98	415	378-6772	yes	no	0	12.5	67
2252	NH	148	408	333-7449	no	no	0	17.6	121
3046	MI	110	510	357-5784	no	no	0	18.9	92
1621	SC	138	510	370-9533	no	yes	21	19.5	149
1076	WY	53	415	337-4339	no	yes	27	25.9	119

```
# Ask from the dataset
'''
> eg  : max value of total day charge?
'''
df['total day charge'].max()
```

```
59.64
```

```
# More advance queries
'''
> how much time on an avg do churn users spend on phone during
  daytime?
> 1st print only churn user, [churn]==1
> then print their avg time, ['total day minutes']
> for avg value use, .mean()
```

```
'''
df[df['churn']==1]['total day minutes'].mean()
```

```
206.91407867494814
```

```
# From queries above churn user spend more time, why?
df[df['churn']==0]['total day minutes'].mean()
```

```
175.17575438596492
```

```
# What is the max length of international calls among loyal users
df[df['churn']==0]['total intl calls'].max()
```

```
19
```

```
# Retrieve all the max val from every col
df.max()
```

```
state                          WY
account length                243
area code                     510
phone number             422-9964
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                           1
dtype: object
```

```
# Retrieve only those rec where state starts with letter 'W'
'''
using lambda function
'''
df[df['state'].apply(lambda state: state[0]=='W')].head()
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes
9	WV	141	415	330-8173	yes	yes	37	258.6	84	43.96	222.0
26	WY	57	408	357-3817	no	yes	39	213.0	115	36.21	191.1
44	WI	64	510	352-1237	no	no	0	154.0	67	26.18	225.8

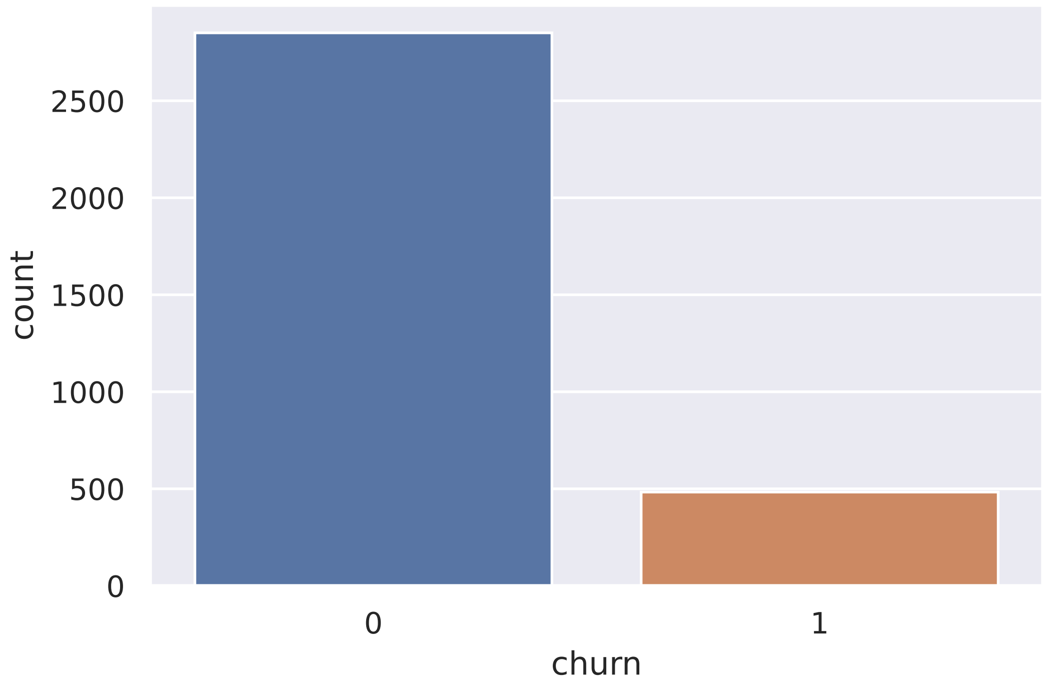
```
# Check the missing values
df.isnull().sum()         # no missing data
```

```
state                     0
account length            0
area code                 0
phone number              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
```

```
# Generate graph out of the dataset
'''
> using the seaborn data library
> more presentable
'''
import seaborn as sns
# Set the configuration as seaborn, not matplotlib (default)
sns.set()
# Set seaborn graphs as svg (sharper image)
%config InlineBackend.figure_format = 'svg'
```

```
# Generate graph of active and churn users
sns.countplot(x='churn',data=df)          #data variable are from df var above
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbfdfdc6dd0>
```


count
2500
2000
1500
1000
500
0
0
1
churn