Data Wrangling I

1. Import all the required Python Libraries.

Why do we need pandas, why not our excel?

- **Advantage**

> quickly analyse data & gives you insight
>
> need not to be a programmer

- **Disadvantage**

> Can not handle large amount of data
>
> it may crashes while loading a data
>
> Missing value, cleanind data involves lots of process

- Pandas developed for data analysis
- support Multiple file format
- Time series analysis
- One Script can be used for similar operation again & again

In [127]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#so that we can view the graphs inside the notebook
```

In [128]:

```
#from google.colab import drive
#drive.mount('/content/gdrive')
```

Fundamental Data Types in Pandas

1. Series---- 1 D array with corresponding index
2. Data Frame ---- n D array In [129]:

```
s1 = pd.Series(range(1,10,1))
```

In [130]:

```
s1
```

Out[130]:

```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8 8    9 dtype: int64 In [131]:
```

```
s3 = pd.Series({1:21, 2:13,3:45})
```

In [132]:

```
s3
```

Out[132]:

```
1    21
2    13 3    45 dtype: int64
```

In [133]: s2 = pd.Series([1, 2, 3, 4], index=['p', 'q', 'r','s'],

```
name='one') In [134]:
```

```
s2
```

Out[134]:

```
p    1
q    2
r    3
s    4
Name: one, dtype: int64
```

In [135]:

```
df1 = pd.DataFrame(s2)
df1
```

```
```

Out[135]:

	one
p	1
q	2
r	3
s	4

2.Locate an open source data from the web (e.g. https://www.kaggle.com (https://www.kaggle.com)). Provide a clear description of the data and its source (i.e., URL of the web site).

3.Load the Dataset into pandas data frame

Real power- Import from different formats http://pandas.pydata.org/pandas-docs/version/0.20/io.html (http://pandas.pydata.org/pandas-docs/version/0.20/io.html)

In [136]:

```
df2 = pd.read_csv("/content/sample_data/california_housing_test.csv")
#dataframe_name = pd.read_<format>(filename) In
```

[137]:

```
df2.head(10)
```

Out[137]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	6
1	-118.30	34.26	43.0	1510.0	310.0	809.0	2

2	-117.81	33.78	27.0	3589.0	507.0	1484.0	4
3	-118.36	33.82	28.0	67.0	15.0	49.0	
4	-119.67	36.33	19.0	1241.0	244.0	850.0	2
5	-119.56	36.51	37.0	1018.0	213.0	663.0	2
6	-121.43	38.63	43.0	1009.0	225.0	604.0	2
7	-120.65	35.48	19.0	2310.0	471.0	1341.0	4
8	-122.84	38.40	15.0	3080.0	617.0	1446.0	5
9	-118.02	34.08	31.0	2402.0	632.0	2830.0	6

In [138]:

```
df2.tail(3)
```

Out[138]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
2997	-119.70	36.30	10.0	956.0	201.0	693.0	
2998	-117.12	34.10	40.0	96.0	14.0	46.0	
2999	-119.63	34.42	42.0	1765.0	263.0	753.0	

In [139]:

```
df2['median_house_value_new']=df2['median_house_value']+111
```

In [140]:

```
df2.tail(3)
```

Out[140]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
2997	-119.70	36.30	10.0	956.0	201.0	693.0	
2998	-117.12	34.10	40.0	96.0	14.0	46.0	
2999	-119.63	34.42	42.0	1765.0	263.0	753.0	

In [141]:

```
# write
# <dataframe's name>.to_<file_format>(<file_name>)
```

In [142]:

```

```

```
df2.to_json('data1.json')
```

In [143]:

```
#If our age dataset is an year old
#df[age_now]= df[age]+1
#df[salary_increment]=df[salary]+5000
```

In [144]:

```
#df1['value'] = df1['num']*2
# internally for each value in column num perform each_value*2 and save it as the corre
sponding
# result in the value column
#df1
```

In [145]:

```
len(df2['total_rooms'])
```

Out[145]:

```
3000
```

In [146]:

```
df2['total_rooms'].count()
```

Out[146]:

```
3000
```

In [147]:

```
df2['total_rooms'].mean()
```

Out[147]:

```
2599.578666666667
```

In [148]:

```
df2['total_rooms'].sum()
```

Out[148]:

```
7798736.0
```

In [149]:

```
df2['total_rooms'].median()
```

Out[149]:

```
2106.0
```

In

[150]:

```
df2['total_rooms'].std()
```

Out[150]:

```
2155.59333162558
```

In [151]:

```
df2['total_rooms'].min()
```

Out[151]:

```
6.0
```

In [152]:

```
df2['total_rooms'].max()
```

Out[152]:

```
30450.0
```

In [153]:

```
df2['total_rooms'].describe()
```

```
```

Out[153]:

```
count     3000.000000
mean      2599.578667 std
2155.593332 min
6.000000 25%
1401.000000
50%       2106.000000 75%
3129.000000 max      30450.000000
Name: total_rooms, dtype: float64
```

In [154]:

```
df2['total_rooms'].cumsum()
```

Out[154]:

```
0          3885.0
1          5395.0
2          8984.0
3          9051.0
4          10292.0
          ...
2995    7790662.0
2996    7795919.0
2997    7796875.0
2998    7796971.0
2999    7798736.0
Name: total_rooms, Length: 3000, dtype: float64
```

In [155]:

```
# When you give the whole dataframe, then all numerical columns will be analysis
df2.mean()
```

Out[155]:

```
longitude                  -119.589200
latitude                     35.635390
housing_median_age           28.845333
total_rooms                2599.578667
total_bedrooms              529.950667
population                 1402.798667
households                  489.912000
median_income                 3.807272
median_house_value       205846.275000
median_house_value_new   205957.275000
dtype: float64 In [156]:
```

```
df2.describe()
```

```
```

Out[156]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
count	3000.000000	3000.00000	3000.000000	3000.000000	3000.000000	3000.00
mean	-119.589200	35.63539	28.845333	2599.578667	529.950667	1402.79
std	1.994936	2.12967	12.555396	2155.593332	415.654368	1030.54
min	-124.180000	32.56000	1.000000	6.000000	2.000000	5.00
25%	-121.810000	33.93000	18.000000	1401.000000	291.000000	780.00
50%	-118.485000	34.27000	29.000000	2106.000000	437.000000	1155.00
75%	-118.020000	37.69000	37.000000	3129.000000	636.000000	1742.75
max	-114.490000	41.92000	52.000000	30450.000000	5419.000000	11935.00

In [157]: df =

```
pd.read_csv("/content/sample_data/california_housing_test.csv")
```

In [158]:

```
df.describe()
```

Out[158]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
count	3000.000000	3000.00000	3000.000000	3000.000000	3000.000000	3000.00
mean	-119.589200	35.63539	28.845333	2599.578667	529.950667	1402.79
std	1.994936	2.12967	12.555396	2155.593332	415.654368	1030.54
min	-124.180000	32.56000	1.000000	6.000000	2.000000	5.00
25%	-121.810000	33.93000	18.000000	1401.000000	291.000000	780.00
50%	-118.485000	34.27000	29.000000	2106.000000	437.000000	1155.00
75%	-118.020000	37.69000	37.000000	3129.000000	636.000000	1742.75
max	-114.490000	41.92000	52.000000	30450.000000	5419.000000	11935.00

In [159]:

```
df.columns
```

Out[159]:

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
 dtype='object')
```

```
In [160]:
```

```
df['longitude']
```

```
Out[160]:

0       -122.05
1       -118.30
2       -117.81
3       -118.36
4       -119.67
          ...
2995    -119.86
2996    -118.14
2997    -119.70
2998    -117.12
2999    -119.63
Name: longitude, Length: 3000, dtype: float64
```

```
In [161]:
```

```
df.longitude
```

```
Out[161]:

0       -122.05
1       -118.30
2       -117.81
3       -118.36
4       -119.67
          ...
2995    -119.86
2996    -118.14
2997    -119.70
2998    -117.12
2999    -119.63
Name: longitude, Length: 3000, dtype: float64
```

```
In [162]:
```

```
df.iloc[:,1:3]
```

```
Out[162]:
```

	latitude	housing_median_age
0	37.37	27.0
1	34.26	43.0
2	33.78	27.0
3	33.82	28.0
4	36.33	19.0

```
 ...     ...          ...
2995   34.42         23.0
2996   34.06         27.0
2997   36.30         10.0
2998   34.10         40.0
2999   34.42         42.0
3000 rows × 2 columns
```

4.Data Preprocessing:

check for missing values in the data using pandas , describe() function to get some initial statistics. Filling missing values using fillna(), replace() and interpolate()