2019115104 – Srikanth S 2019115107 - V.Suba Varshini

Exploratory data analysis

Steps to be followed:

- Description of data
- Handling missing data
- Handling outliers
- Understanding relationships and new insights through plots

Loading the data and reading through a .csv

```
In [1]: import pandas as ps
        import numpy as np
        import seaborn as sns
```

```
In [4]: data=ps.read_csv("F:/VT/TEAM 5.csv")
```

To read only first 5 rows

```
In [5]: data.head()
```

Out[5]:

	Area Code	Area	Months Code	Months	Y2001	Y2002	Y2003	Y2004	Y2005	Y2006	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2
0	10	Australia	7001	January	1.334	0.130	0.882	0.595	0.678	1.185	...	0.877	1.078	0.284	1.978	1.237	0.
1	10	Australia	7002	February	0.573	-0.400	1.025	0.957	0.787	0.661	...	1.069	0.068	-0.029	1.045	0.382	1.
2	10	Australia	7003	March	-0.451	0.384	-0.362	0.372	1.032	0.686	...	0.425	-0.679	-0.765	0.877	0.913	0.
3	10	Australia	7004	April	0.325	1.769	0.911	1.034	2.471	-0.486	...	1.491	-0.331	0.289	1.208	1.255	-0.
4	10	Australia	7005	May	-0.050	1.667	1.345	0.378	1.548	-0.565	...	0.865	-0.902	-0.057	1.437	1.956	0.

5 rows × 23 columns

Data.info() gives the types of dataset

```
In [6]: #how many columns are there and what are their datatypes
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 23 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area Code    78 non-null     int64
 1   Area         78 non-null     object
 2   Months Code  78 non-null     int64
 3   Months       78 non-null     object
 4   Y2001        78 non-null     float64
 5   Y2002        78 non-null     float64
 6   Y2003        78 non-null     float64
 7   Y2004        78 non-null     float64
 8   Y2005        78 non-null     float64
 9   Y2006        78 non-null     float64
 10  Y2007        78 non-null     float64
 11  Y2008        78 non-null     float64
 12  Y2009        78 non-null     float64
 13  Y2010        78 non-null     float64
 14  Y2011        78 non-null     float64
 15  Y2012        78 non-null     float64
 16  Y2013        78 non-null     float64
 17  Y2014        78 non-null     float64
 18  Y2015        78 non-null     float64
 19  Y2016        78 non-null     float64
 20  Y2017        78 non-null     float64
 21  Y2018        78 non-null     float64
 22  Y2019        78 non-null     float64
dtypes: float64(19), int64(2), object(2)
memory usage: 14.1+ KB
```

```
In [7]: #last 5 tables of datasets
        data.tail()
```

Out[7]:

	Area Code	Area	Months Code	Months	Y2001	Y2002	Y2003	Y2004	Y2005	Y2006	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018
73	231	United States of America	7009	September	0.582	1.180	0.021	0.185	1.425	-0.243	...	1.055	0.830	0.695	0.882	0.827	1.561	1.295	0.918	1.638
74	231	United States of America	7010	October	-0.245	-0.342	1.643	0.930	0.750	-0.052	...	1.065	0.716	-0.022	0.565	1.297	1.950	2.036	1.230	0.670
75	231	United States of America	7011	November	2.262	1.198	0.570	1.427	0.785	0.492	...	0.788	0.128	0.410	0.098	-0.389	1.784	2.895	1.797	-0.030
76	231	United States of America	7012	December	1.195	1.932	1.169	1.424	0.858	2.165	...	-0.643	1.775	1.070	-0.344	2.816	3.172	0.491	2.149	1.766
77	231	United States of America	7020	Meteorological year	0.776	0.946	0.986	0.855	1.143	1.013	...	0.679	0.530	1.437	0.597	0.463	1.508	2.197	1.408	1.244

5 rows × 23 columns

Comparing the Area as India

```
In [17]: data.query('Area=="India"')
```

Out[17]:

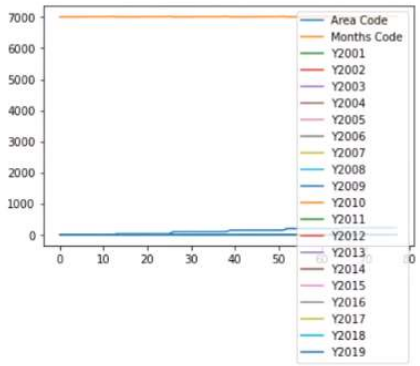
	Area Code	Area	Months Code	Months	Y2001	Y2002	Y2003	Y2004	Y2005	Y2006	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018	Y
26	100	India	7001	January	0.216	0.475	-0.017	0.246	0.453	0.738	...	0.209	-0.435	-0.252	0.009	0.190	-0.073	1.100	0.786	0.527	0
27	100	India	7002	February	0.999	0.562	0.895	0.468	0.599	2.533	...	0.982	0.354	0.023	0.496	-0.459	0.970	1.777	1.286	1.290	0
28	100	India	7003	March	0.354	0.849	0.183	1.866	0.747	0.045	...	2.192	0.570	0.269	0.560	-0.361	-0.186	1.607	0.592	1.395	0
29	100	India	7004	April	-0.253	0.868	0.866	0.923	-0.187	0.189	...	1.990	-0.491	0.203	0.085	0.261	-0.505	1.670	1.198	0.654	1
30	100	India	7005	May	0.224	0.846	0.598	-0.096	0.039	0.128	...	1.052	0.253	0.650	0.775	-0.224	0.640	0.581	0.842	0.610	0
31	100	India	7006	June	-0.973	0.239	0.889	-0.304	1.353	-0.023	...	0.810	-0.427	1.105	-0.610	1.424	0.040	0.663	0.173	0.429	1
32	100	India	7007	July	0.078	1.574	0.280	0.650	0.247	0.516	...	0.558	0.442	0.666	-0.014	0.935	0.844	0.275	0.363	0.428	0
33	100	India	7008	August	0.501	0.370	0.523	0.334	0.683	0.250	...	0.642	0.349	0.317	0.166	0.833	0.903	0.720	0.838	0.460	0
34	100	India	7009	September	1.053	0.450	0.276	0.700	0.491	0.512	...	0.301	0.287	0.324	0.778	0.385	1.255	0.555	1.023	0.519	0
35	100	India	7010	October	0.768	0.811	0.290	-0.296	0.220	0.860	...	0.916	0.707	0.105	0.391	0.696	1.368	0.816	1.260	0.803	0
36	100	India	7011	November	1.217	0.864	0.817	0.687	0.115	1.041	...	1.490	1.209	0.226	0.222	0.976	1.617	0.730	0.694	1.122	1
37	100	India	7012	December	0.917	1.398	0.517	0.991	0.145	1.072	...	0.009	0.729	0.827	0.375	0.035	1.231	1.342	0.987	0.308	0
38	100	India	7020	Meteorological year	0.398	0.735	0.583	0.475	0.479	0.578	...	1.017	0.235	0.364	0.307	0.419	0.576	0.977	0.866	0.769	0

13 rows × 23 columns

Plot a graph with all the header values from excel sheet

```
In [20]: data.plot()
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x23dc8d19670>
```



Pandas uses the `plot()` method to create diagrams.

We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

Scatter Plot

Specify that you want a scatter plot with the `kind` argument:

```
kind = 'scatter'
```

A scatter plot needs an x- and a y-axis.

In the example below we will use "Area" for the x-axis and "Y2001" for the y-axis.

Include the x and y arguments like this:

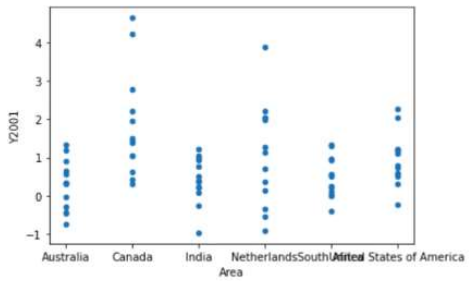
```
x = 'Area', y = 'Y2001'
```

```
In [21]: import matplotlib.pyplot as plt

In [23]: plt.show()

In [24]: data.plot(kind='scatter',x='Area',y='Y2001')
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x23dcaf33100>
```



Similarly using the bar graph representation:

```
In [27]: data.plot(kind='bar',x='Area',y='Y2001')
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x23dcc0dde20>
```