Data Exploration using pandas

The World Series television ratings dataset contains data on the number of people who watched the World Series, a yearly baseball championship series, on television.

This dataset can be used to analyze the popularity and reach of the World Series.

It may be of interest to researchers, advertisers, and television networks seeking to understand the popularity and reach of major sporting events.

In [1]:

```
import pandas as pd
```

In [2]:

```
vig=pd.read_csv('C://Users//somas//Downloads//archive (1)//vig.csv')
```

In [3]:

```
vig
```

Out[3]:

	year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losingteam	losing team wins
0	2022	Fox	11762000	11475000.0	10789000.0	11162000.0	11809000.0	12786000.0	12549000.0	NaN	6	Houston Astros	Philadelphia Phillies	2
1	2021	Fox	11744000	10811000.0	10280000.0	11232000.0	10511000.0	13644000.0	13986000.0	NaN	6	Atlanta Braves	Houston Astros	2
2	2020	Fox	9785000	9195000.0	8950000.0	8156000.0	9332000.0	10059000.0	12627000.0	NaN	6	Los Angeles Dodgers	Tampa Bay Rays	2
3	2019	Fox	13912000	12194000.0	11925000.0	12220000.0	10219000.0	11390000.0	16425000.0	23013000.0	7	Washington Nationals	Houston Astros	3
4	2018	Fox	14125000	13761000.0	13458000.0	13251000.0	13563000.0	17634000.0	NaN	NaN	5	Boston Red Sox	Los Angeles Dodgers	1
5	2017	Fox	18926000	14968000.0	15483000.0	15676000.0	15400000.0	18940000.0	22229000.0	28240000.0	7	Houston Astros	Los Angeles Dodgers	3

In [4]:

```
vig.shape
```

Out[4]:

```
(54, 14)
```

From the Above World television ratings dataset its shows us there are 54 observation and 14 varaibles.

In [5]:

```
vig.head()
```

Out[5]:

	year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losingt
0	2022	Fox	11762000	11475000.0	10789000.0	11162000.0	11809000.0	12786000.0	12549000.0	NaN	6	Houston Astros	Philadel Ph
1	2021	Fox	11744000	10811000.0	10280000.0	11232000.0	10511000.0	13644000.0	13986000.0	NaN	6	Atlanta Braves	Hou As
2	2020	Fox	9785000	9195000.0	8950000.0	8156000.0	9332000.0	10059000.0	12627000.0	NaN	6	Los Angeles Dodgers	Tampa F
3	2019	Fox	13912000	12194000.0	11925000.0	12220000.0	10219000.0	11390000.0	16425000.0	23013000.0	7	Washington Nationals	Hou As
4	2018	Fox	14125000	13761000.0	13458000.0	13251000.0	13563000.0	17634000.0	NaN	NaN	5	Boston Red Sox	Los Ang Dod

In the above the World television ratings dataset, it shows first five data of the dataset , It starts in the year of 2022.

In [6]:

```
vig.tail()
```

Out[6]:

	year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losing
49	1973	NBC	34750000	27444000.0	35661000.0	38030000.0	36670000.0	37364000.0	28610000.0	39935000.0	7	Oakland Athletics	New
50	1972	NBC	31508714	22534000.0	27331000.0	37880000.0	38210000.0	23120000.0	30048000.0	41438000.0	7	Oakland Athletics	Cinc
51	1971	NBC	24298571	24778000.0	13633000.0	13633000.0	38898000.0	13633000.0	28095000.0	37420000.0	7	Pittsburgh Pirates	Balti O
52	1970	NBC	11708000	14120000.0	13400000.0	10340000.0	10340000.0	10340000.0	NaN	NaN	5	Baltimore Orioles	Cinc
53	1969	NBC	13174000	14920000.0	17080000.0	11290000.0	11290000.0	11290000.0	NaN	NaN	5	New York Mets	Balti O

In the above the World television ratings dataset, it shows Last five data of the dataset , It starts in the year of 1973.

In [7]:

```
vig.columns
```

Out[7]:

```
Index(['year', 'network', 'average audience', 'game 1 audience',
       'game 2 audience', 'game 3 audience', 'game 4 audience',
       'game 5 audience', 'game 6 audience', 'game 7 audience',
       'total games played', 'winningteam', 'losingteam', 'losing team wins'],
      dtype='object')
```

In the above Dataset we have selected there are column names consisting of Year,Network,average audience ,game 1 audience, game 2 audience, game 3 audience, game 4 audience,game 5 audience,game 6 audience,game 7 audience,total games played,winningteam,losingteam,losing team wins

In [8]:

```
vig.index
```

Out[8]:

```
RangeIndex(start=0, stop=54, step=1)
```

In this data set where the RangeIndex Row starts from 0 and ends at 54 where steps increasing at a rate of 1 per row

In [9]:

```
vig.describe()
```

Out[9]:

	year	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	tot game playe
count	54.000000	5.400000e+01	5.300000e+01	5.200000e+01	5.300000e+01	5.300000e+01	4.400000e+01	3.100000e+01	1.800000e+01	54.0000
mean	1995.518519	2.466182e+07	2.234287e+07	2.327375e+07	2.316519e+07	2.352834e+07	2.504327e+07	2.884103e+07	3.998883e+07	5.7777
std	15.730922	9.512589e+06	9.036234e+06	9.446336e+06	1.024255e+07	9.087722e+06	1.110832e+07	1.132817e+07	1.035137e+07	1.0931
min	1969.000000	9.785000e+06	9.195000e+06	8.950000e+06	8.156000e+06	9.332000e+06	1.005900e+07	1.254900e+07	2.301300e+07	4.0000
25%	1982.250000	1.598925e+07	1.494400e+07	1.424250e+07	1.325100e+07	1.553700e+07	1.557775e+07	2.164700e+07	3.246550e+07	5.0000
50%	1995.500000	2.453900e+07	2.099000e+07	2.150600e+07	2.219000e+07	2.276100e+07	2.337900e+07	2.861000e+07	3.999000e+07	6.0000
75%	2008.750000	3.374468e+07	2.744400e+07	3.107250e+07	3.280000e+07	3.146000e+07	3.270750e+07	3.493000e+07	4.992000e+07	7.0000
max	2022.000000	4.427900e+07	4.351000e+07	4.299000e+07	4.381000e+07	3.922000e+07	4.899000e+07	5.486000e+07	5.500000e+07	7.0000

To perform statistical functions for the dataset ,we use sample describe function, like count, mean, std, min, 25% of value, 50% of value, 75% of value and their maximum

In [10]:

```
vig.sort_values(["year"],ascending=True)
```

Out[10]:

	year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losingteam	losing team wins
53	1969	NBC	13174000	14920000.0	17080000.0	11290000.0	11290000.0	11290000.0	NaN	NaN	5	New York Mets	Baltimore Orioles	1
52	1970	NBC	11708000	14120000.0	13400000.0	10340000.0	10340000.0	10340000.0	NaN	NaN	5	Baltimore Orioles	Cincinnati Reds	1
51	1971	NBC	24298571	24778000.0	13633000.0	13633000.0	38898000.0	13633000.0	28095000.0	37420000.0	7	Pittsburgh Pirates	Baltimore Orioles	3
50	1972	NBC	31508714	22534000.0	27331000.0	37880000.0	38210000.0	23120000.0	30048000.0	41438000.0	7	Oakland Athletics	Cincinnati Reds	3
49	1973	NBC	34750000	27444000.0	35661000.0	38030000.0	36670000.0	37364000.0	28610000.0	39935000.0	7	Oakland Athletics	New York Mets	3
48	1974	NBC	29080000	23750000.0	30470000.0	29830000.0	29760000.0	31610000.0	NaN	NaN	5	Oakland Athletics	Los Angeles Dodgers	1

In the above data set we have sorted the values with state variable where it arranged the data in an ascending order in an alphabetical order.

In [11]:

```
vig.sort_values(["year"],ascending=False)
```

Out[11]:

	year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losingteam	losing team wins
0	2022	Fox	11762000	11475000.0	10789000.0	11162000.0	11809000.0	12786000.0	12549000.0	NaN	6	Houston Astros	Philadelphia Phillies	2
1	2021	Fox	11744000	10811000.0	10280000.0	11232000.0	10511000.0	13644000.0	13986000.0	NaN	6	Atlanta Braves	Houston Astros	2
2	2020	Fox	9785000	9195000.0	8950000.0	8156000.0	9332000.0	10059000.0	12627000.0	NaN	6	Los Angeles Dodgers	Tampa Bay Rays	2
3	2019	Fox	13912000	12194000.0	11925000.0	12220000.0	10219000.0	11390000.0	16425000.0	23013000.0	7	Washington Nationals	Houston Astros	3
4	2018	Fox	14125000	13761000.0	13458000.0	13251000.0	13563000.0	17634000.0	NaN	NaN	5	Boston Red Sox	Los Angeles Dodgers	1
5	2017	Fox	18926000	14968000.0	15483000.0	15676000.0	15400000.0	18940000.0	22229000.0	28240000.0	7	Houston Astros	Los Angeles Dodgers	3

In the above data set we have sorted the values with state variable where it arranged the data in an descending order.

In [12]:

```
vig['winningteam'].value_counts()
```

Out[12]:

```
New York Yankees          7
Boston Red Sox            4
Oakland Athletics         4
Los Angeles Dodgers       3
San Francisco Giants      3
St. Louis Cardinals       3
Cincinnati Reds           3
Atlanta Braves            3
Houston Astros            2
Toronto Blue Jays         2
Baltimore Orioles         2
New York Mets             2
Minnesota Twins           2
Pittsburgh Pirates        2
Florida Marlins           2
Philadelphia Phillies     2
Kansas City Royals        2
Arizona Diamondbacks      1
Anaheim Angels            1
Chicago Cubs              1
Detroit Tigers            1
Washington Nationals      1
Chicago White Sox         1
Name: winningteam, dtype: int64
```

In this data we have taken winnig team to use this function it show us variable which is data type is Int64.

In [13]:

```
vig['losingteam'].value_counts()
```

Out[13]:

```
Los Angeles Dodgers        5
New York Yankees           4
Cleveland Indians          4
St. Louis Cardinals        4
Atlanta Braves             4
Philadelphia Phillies      3
Baltimore Orioles          3
New York Mets              3
Boston Red Sox             2
Oakland Athletics          2
San Diego Padres           2
San Francisco Giants       2
Cincinnati Reds            2
Texas Rangers              2
Detroit Tigers             2
Kansas City Royals         2
Houston Astros             2
Tampa Bay Rays             2
Colorado Rockies           1
 Houston Astros            1
Milwaukee Brewers          1
 Philadelphia Phillies     1
Name: losingteam, dtype: int64
```

In this data we have taken losinteam to use this function it show us variable which is data type is Int64.

In [14]:

```
vig.drop_duplicates(inplace=True)
```

In [15]:

```
vig
```

Out[15]:

	year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losingteam	losing team wins
0	2022	Fox	11762000	11475000.0	10789000.0	11162000.0	11809000.0	12786000.0	12549000.0	NaN	6	Houston Astros	Philadelphia Phillies	2
1	2021	Fox	11744000	10811000.0	10280000.0	11232000.0	10511000.0	13644000.0	13986000.0	NaN	6	Atlanta Braves	Houston Astros	2
2	2020	Fox	9785000	9195000.0	8950000.0	8156000.0	9332000.0	10059000.0	12627000.0	NaN	6	Los Angeles Dodgers	Tampa Bay Rays	2
3	2019	Fox	13912000	12194000.0	11925000.0	12220000.0	10219000.0	11390000.0	16425000.0	23013000.0	7	Washington Nationals	Houston Astros	3
4	2018	Fox	14125000	13761000.0	13458000.0	13251000.0	13563000.0	17634000.0	NaN	NaN	5	Boston Red Sox	Los Angeles Dodgers	1
5	2017	Fox	18926000	14968000.0	15483000.0	15676000.0	15400000.0	18940000.0	22229000.0	28240000.0	7	Houston Astros	Los Angeles Dodgers	3

If there is any duplicates present in the data set it would shows us by having change in the observations and variables, After using this there was no change in the Dataset so there are no duplicates in the dataset

In [16]:

```
vig_subset=vig[['year','network']]
```

In [17]:

```
vig_subset
```

Out[17]:

	year	network
0	2022	Fox
1	2021	Fox
2	2020	Fox
3	2019	Fox
4	2018	Fox
5	2017	Fox
6	2016	Fox
7	2015	Fox
8	2014	Fox
9	2013	Fox
10	2012	Fox
11	2011	Fox
12	2010	Fox
13	2009	Fox
14	2008	Fox
15	2007	Fox
16	2006	Fox
17	2005	Fox
18	2004	Fox
19	2003	Fox
20	2002	Fox
21	2001	Fox
22	2000	Fox
23	1999	NBC
24	1998	Fox
25	1997	NBC
26	1996	Fox
27	1995	ABC
28	1995	NBC
29	1993	CBS
30	1992	CBS
31	1991	CBS
32	1990	CBS
33	1989	ABC
34	1988	NBC
35	1987	ABC
36	1986	NBC
37	1985	ABC
38	1984	NBC
39	1983	ABC
40	1982	NBC
41	1981	ABC
42	1980	NBC
43	1979	ABC
44	1978	NBC
45	1977	ABC
46	1976	NBC
47	1975	NBC
48	1974	NBC
49	1973	NBC
50	1972	NBC
51	1971	NBC
52	1970	NBC
53	1969	NBC

In the above dataset to display year and network as subset , To display a specific set of data

In [18]:

```
vig.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54 entries, 0 to 53
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   year                54 non-null     int64  
 1   network             54 non-null     object 
 2   average audience    54 non-null     int64  
 3   game 1 audience     53 non-null     float64
 4   game 2 audience     52 non-null     float64
 5   game 3 audience     53 non-null     float64
 6   game 4 audience     53 non-null     float64
 7   game 5 audience     44 non-null     float64
 8   game 6 audience     31 non-null     float64
 9   game 7 audience     18 non-null     float64
 10  total games played  54 non-null     int64  
 11  winningteam         54 non-null     object 
 12  losingteam          54 non-null     object 
 13  losing team wins    54 non-null     int64  
dtypes: float64(7), int64(4), object(3)
memory usage: 6.3+ KB
```

In the World television ratings dataset we have three datatypes as float64, int64 and object each of them consisting of 3,2,5 for each data type alomg with memory usage of 6.3 KB. Only variables that are having 53 Non - Null count are Game 1 audinece and Game 2 audience.

In [19]:

```
vig_index=vig.set_index('year')
```

In [20]:

```
vig_index
```

Out[20]:

year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losingteam	losing team wins
2022	Fox	11762000	11475000.0	10789000.0	11162000.0	11809000.0	12786000.0	12549000.0	NaN	6	Houston Astros	Philadelphia Phillies	2
2021	Fox	11744000	10811000.0	10280000.0	11232000.0	10511000.0	13644000.0	13986000.0	NaN	6	Atlanta Braves	Houston Astros	2
2020	Fox	9785000	9195000.0	8950000.0	8156000.0	9332000.0	10059000.0	12627000.0	NaN	6	Los Angeles Dodgers	Tampa Bay Rays	2
2019	Fox	13912000	12194000.0	11925000.0	12220000.0	10219000.0	11390000.0	16425000.0	23013000.0	7	Washington Nationals	Houston Astros	3
2018	Fox	14125000	13761000.0	13458000.0	13251000.0	13563000.0	17634000.0	NaN	NaN	5	Boston Red Sox	Los Angeles Dodgers	1
2017	Fox	18926000	14968000.0	15483000.0	15676000.0	15400000.0	18940000.0	22229000.0	28240000.0	7	Houston	Los Angeles	3

In this data set where the nothing RangeIndex

In [21]:

```
vig_index=vig.reset_index()
```

In [22]:

```
vig_index
```

44	44	1978	NBC	44279000	43510000.0	42720000.0	43810000.0	39220000.0	45870000.0	50600000.0	NaN	6	New York Yankees Los Angeles Dodgers
45	45	1977	ABC	37150000	36960000.0	35880000.0	37410000.0	31460000.0	37010000.0	44200000.0	NaN	6	New York Yankees Los Angeles Dodgers
46	46	1976	NBC	34720000	23730000.0	NaN	36250000.0	38790000.0	NaN	NaN	NaN	4	Cincinnati Reds New York Yankees
47	47	1975	NBC	35960000	20990000.0	24320000.0	37910000.0	34640000.0	40710000.0	41570000.0	51560000.0	7	Cincinnati Reds Boston Red Sox
48	48	1974	NBC	29080000	23750000.0	30470000.0	29830000.0	29760000.0	31610000.0	NaN	NaN	5	Oakland Athletics Los Angeles Dodgers
49	49	1973	NBC	34750000	27444000.0	35661000.0	38030000.0	36670000.0	37364000.0	28610000.0	39935000.0	7	Oakland Athletics New York Mets
50	50	1972	NBC	31508714	22534000.0	27331000.0	37880000.0	38210000.0	23120000.0	30048000.0	41438000.0	7	Oakland Athletics Cincinnati Reds
51	51	1971	NBC	24298571	24778000.0	13633000.0	13633000.0	38898000.0	13633000.0	28095000.0	37420000.0	7	Pittsburgh Pirates Baltimore Orioles

In this data set where the RangeIndex Row starts from 0 and ends at 53 where steps increasing at a rate of 1 per row.

In [23]:

```
vig_iloc=vig.iloc[0:3,4:6]
```

In [24]:

```
vig_iloc
```

Out[24]:

	game 2 audience	game 3 audience
0	10789000.0	11162000.0
1	10280000.0	11232000.0
2	8950000.0	8156000.0

we have displayed the specific rows and columns of [0:3 ,4:6] in order to display the specific data in the dataset

In [25]:

```
vig_row=vig[vig['average audience']>9785000]
```

In [26]:

```
vig_row
```

Out[26]:

	year	network	average audience	game 1 audience	game 2 audience	game 3 audience	game 4 audience	game 5 audience	game 6 audience	game 7 audience	total games played	winningteam	losingteam	losing team wins
0	2022	Fox	11762000	11475000.0	10789000.0	11162000.0	11809000.0	12786000.0	12549000.0	NaN	6	Houston Astros	Philadelphia Phillies	2
1	2021	Fox	11744000	10811000.0	10280000.0	11232000.0	10511000.0	13644000.0	13986000.0	NaN	6	Atlanta Braves	Houston Astros	2
3	2019	Fox	13912000	12194000.0	11925000.0	12220000.0	10219000.0	11390000.0	16425000.0	23013000.0	7	Washington Nationals	Houston Astros	3
4	2018	Fox	14125000	13761000.0	13458000.0	13251000.0	13563000.0	17634000.0	NaN	NaN	5	Boston Red Sox	Los Angeles Dodgers	1
5	2017	Fox	18926000	14968000.0	15483000.0	15676000.0	15400000.0	18940000.0	22229000.0	28240000.0	7	Houston Astros	Los Angeles Dodgers	3
6	2016	Fox	22847000	19368000.0	17395000.0	19384000.0	16705000.0	23638000.0	23396000.0	40045000.0	7	Chicago Cubs	Cleveland Indians	3

BIn the above data we have showed values which are greater than 9785000 in the metric tons variable.

In [27]:

```
vig['game 1 audience']=vig['game 1 audience'].astype('object')
```

In [28]:

```
vig.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54 entries, 0 to 53
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   year                54 non-null     int64  
 1   network             54 non-null     object 
 2   average audience    54 non-null     int64  
 3   game 1 audience     53 non-null     object 
 4   game 2 audience     52 non-null     float64
 5   game 3 audience     53 non-null     float64
 6   game 4 audience     53 non-null     float64
 7   game 5 audience     44 non-null     float64
 8   game 6 audience     31 non-null     float64
 9   game 7 audience     18 non-null     float64
 10  total games played  54 non-null     int64  
 11  winningteam         54 non-null     object 
 12  losingteam          54 non-null     object 
 13  losing team wins    54 non-null     int64  
dtypes: float64(6), int64(4), object(4)
memory usage: 6.3+ KB
```

we have use .astype function to change datatype of a variable. In this we have changed Game 1 audience as float64 type where it was a object before using this function

In [29]:

```
vig_agg=vig.groupby('game 1 audience')['game 2 audience'].mean()
```

In [30]:

```
vig_agg
```

```
23750000.0    30470000.0
24778000.0    13633000.0
25020000.0    29590000.0
25030000.0    25410000.0
25150000.0           NaN
26310000.0    30840000.0
27444000.0    35661000.0
28600000.0    30570000.0
30350000.0    34700000.0
30570000.0    33660000.0
31510000.0    31770000.0
33100000.0    32340000.0
33860000.0    32010000.0
36230000.0    34550000.0
36960000.0    35880000.0
37460000.0    37700000.0
38140000.0    36320000.0
39720000.0    39110000.0
42040000.0    42990000.0
43510000.0    42720000.0
```

In this we have used variable named dollars and Tsn which variables are grouped and calculated their mean value with the Game 1 audience value of data type as float64.