Name : Ankur Damke **Date:** 30/01/2024

Reg. ID : 201090082 **Group No :** 11

DATA SCIENCE AND ANALYSIS (DSA)

Lab

Experiment No. 2

Implement Data Visualization and Wrangling (ggplot)

Aim : To perform data wrangling on a given dataset.

Software Used : Python 3 with Jupyter Notebook.

Theory :

Data Wrangling in Python

Data Wrangling is the process of gathering, collecting, and transforming Raw data into another format for better understanding, decision-making, accessing, and analysis in less time. Data Wrangling is also known as Data Munging.

Importance Of Data Wrangling

Data Wrangling is a very important step in a Data science project. The below example will explain its importance:

Books selling Website want to show top-selling books of different domains, according to user preference. For example, if a new user searches for motivational books, then they want to show those motivational books which sell the most or have a high rating, etc.

But on their website, there are plenty of raw data from different users. Here the concept of Data Munging or Data Wrangling is used. As we know Data wrangling is not by the System itself. This process is done by Data Scientists. So, the data Scientist will wrangle data in such a way that they will sort the motivational books that are sold more or have high ratings or user buy this book with these package of Books, etc. On the basis of that, the new user will make a choice. This will explain the importance of Data wrangling.

Data Wrangling in Python

Data Wrangling is a crucial topic for Data Science and Data Analysis. Pandas Framework of Python is used for Data Wrangling. **Pandas** is an open-source library in **Python** specifically developed for Data Analysis and Data Science. It is used for processes like data sorting or filtration, Data grouping, etc.

Data wrangling in Python deals with the below functionalities:

1. **Data exploration:** In this process, the data is studied, analyzed, and understood by visualizing representations of data.
2. **Dealing with missing values:** Most of the datasets having a vast amount of data contain missing values of *NaN, they are needed to be taken* care of by replacing them with mean, mode, the most frequent value of the column, or simply by dropping the row having a *NaN* value.
3. **Reshaping data:** In this process, data is manipulated according to the requirements, where new data can be added or pre-existing data can be modified.
4. **Filtering data:** Some times datasets are comprised of unwanted rows or columns which are required to be removed or filtered
5. **Other:** After dealing with the raw dataset with the above functionalities we get an efficient dataset as per our requirements and then it can be used for a required purpose like **data analyzing, machine learning, data visualization, model training** etc.

Code and Output :

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import seaborn as sns
```

```
In [7]: books=pd.read_csv(r"C:\Users\shubh\Downloads\Books.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_bad_lines=
        users=pd.read_csv(r"C:\Users\shubh\Downloads\Users.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_bad_lines=
        ratings=pd.read_csv(r"C:\Users\shubh\Downloads\Book-Ratings.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_b
```

```
C:\Users\shubh\AppData\Local\Temp\ipykernel_18860\1315193708.py:1: FutureWarning: The error_bad_lines argument has been depreca
ted and will be removed in a future version. Use on_bad_lines in the future.


  books=pd.read_csv(r"C:\Users\shubh\Downloads\Books.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_bad_li
nes=False)
C:\Users\shubh\AppData\Local\Temp\ipykernel_18860\1315193708.py:1: FutureWarning: The warn_bad_lines argument has been deprecat
ed and will be removed in a future version. Use on_bad_lines in the future.


  books=pd.read_csv(r"C:\Users\shubh\Downloads\Books.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_bad_li
nes=False)
C:\Users\shubh\AppData\Local\Temp\ipykernel_18860\1315193708.py:1: DtypeWarning: Columns (3) have mixed types. Specify dtype op
tion on import or set low_memory=False.
  books=pd.read_csv(r"C:\Users\shubh\Downloads\Books.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_bad_li
nes=False)
C:\Users\shubh\AppData\Local\Temp\ipykernel_18860\1315193708.py:2: FutureWarning: The error_bad_lines argument has been depreca
ted and will be removed in a future version. Use on_bad_lines in the future.


  users=pd.read_csv(r"C:\Users\shubh\Downloads\Users.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_bad_li
nes=False)
C:\Users\shubh\AppData\Local\Temp\ipykernel_18860\1315193708.py:2: FutureWarning: The warn_bad_lines argument has been deprecat
ed and will be removed in a future version. Use on_bad_lines in the future.


  users=pd.read_csv(r"C:\Users\shubh\Downloads\Users.csv",delimiter=';',error_bad_lines=False,encoding='ISO-8859-1',warn_bad_li
nes=False)
C:\Users\shubh\AppData\Local\Temp\ipykernel_18860\1315193708.py:3: FutureWarning: The error_bad_lines argument has been depreca
ted and will be removed in a future version. Use on_bad_lines in the future.
```

```
In [17]: books.drop(['Image-URL-S','Image-URL-M','Image-URL-L'],axis=1,inplace=True)
         books.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_18860\2017871901.py in <module>
----> 1 books.drop(['Image-URL-S','Image-URL-M','Image-URL-L'],axis=1,inplace=True)
      2 books.head()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312 
    313         return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, inplace, err
ors)
   4955                 weight  1.0     0.8
   4956         """
-> 4957         return super().drop(
   4958             labels=labels,
   4959             axis=axis,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level, inplace, e
rrors)
   4265         for axis, labels in axes.items():
   4266             if labels is not None:
-> 4267                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4268 
   4269         if inplace:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors, consolidate,
only_slice)
   4309                 new_axis = axis.drop(labels, level=level, errors=errors)
   4310             else:
-> 4311                 new_axis = axis.drop(labels, errors=errors)
   4312             indexer = axis.get_indexer(new_axis)
   4313 
```

In [14]:
```
books.isnull().sum()
```

Out[14]:
```
ISBN                   0
Book-Title             0
Book-Author            1
Year-Of-Publication    0
Publisher              2
dtype: int64
```

In [15]:
```
books.loc[books['Publisher'].isnull(),:]
```

Out[15]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
128890	193169656X	Tyrant Moon	Elaine Corvidae	2002	NaN
129037	1931696993	Finders Keepers	Linnea Sinclair	2001	NaN

In [19]:
```
books.loc[books['Book-Author'].isnull(),:]
```

Out[19]:

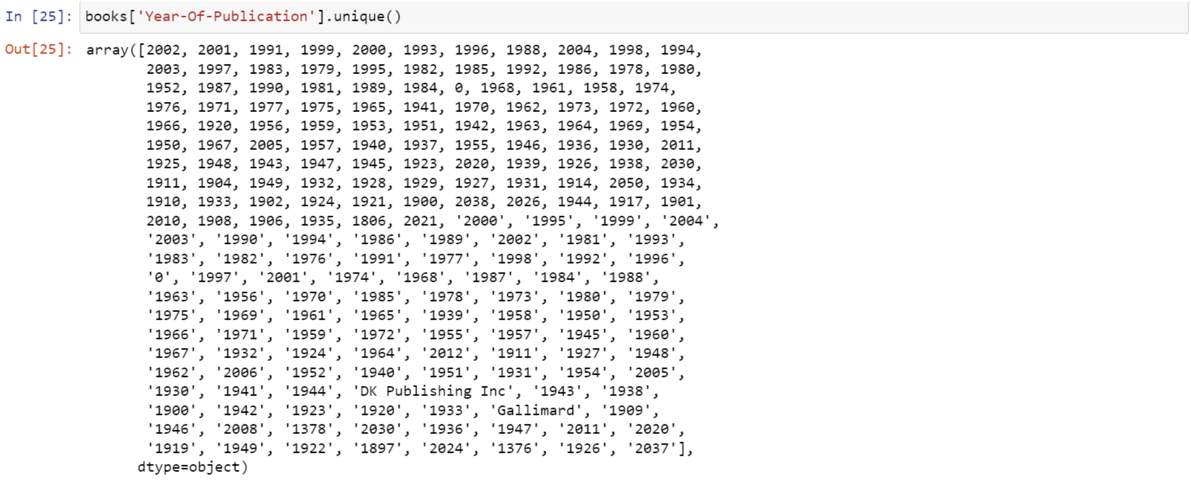
	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
187689	9627982032	The Credit Suisse Guide to Managing Your Perso...	NaN	1995	Edinburgh Financial Publishing

In [20]:
```
books.loc[books['Publisher'].isnull(),:]
```

Out[20]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
128890	193169656X	Tyrant Moon	Elaine Corvidae	2002	NaN
129037	1931696993	Finders Keepers	Linnea Sinclair	2001	NaN

In [24]:
```
books.at[187689,'Book-Author']='Other'
books.at[128890,'Publisher']='Other'
books.at[129037,'Publisher']='Other'
```



In [25]:
```
books['Year-Of-Publication'].unique()
```

Out[25]:
```
array([2002, 2001, 1991, 1999, 2000, 1993, 1996, 1988, 2004, 1998, 1994,
       2003, 1997, 1983, 1979, 1995, 1982, 1985, 1992, 1986, 1978, 1980,
       1952, 1987, 1990, 1981, 1989, 1984, 0, 1968, 1961, 1958, 1974,
       1976, 1971, 1977, 1975, 1965, 1941, 1970, 1962, 1973, 1972, 1960,
       1966, 1920, 1956, 1959, 1953, 1951, 1942, 1963, 1964, 1969, 1954,
       1950, 1967, 2005, 1957, 1940, 1937, 1955, 1946, 1936, 1930, 2011,
       1925, 1948, 1943, 1947, 1945, 1923, 2020, 1939, 1926, 1938, 2030,
       1911, 1904, 1949, 1932, 1928, 1929, 1927, 1931, 1914, 2050, 1934,
       1910, 1933, 1902, 1924, 1921, 1900, 2038, 2026, 1944, 1917, 1901,
       2010, 1908, 1906, 1935, 1806, 2021, '2000', '1995', '1999', '2004',
       '2003', '1990', '1994', '1986', '1989', '2002', '1981', '1993',
       '1983', '1982', '1976', '1991', '1977', '1998', '1992', '1996',
       '0', '1997', '2001', '1974', '1968', '1987', '1984', '1988',
       '1963', '1956', '1970', '1985', '1978', '1973', '1980', '1979',
       '1975', '1969', '1961', '1965', '1939', '1958', '1950', '1953',
       '1966', '1971', '1959', '1972', '1955', '1957', '1945', '1960',
       '1967', '1932', '1924', '1964', '2012', '1911', '1927', '1948',
       '1962', '2006', '1952', '1940', '1951', '1931', '1954', '2005',
       '1930', '1941', '1944', 'DK Publishing Inc', '1943', '1938',
       '1900', '1942', '1923', '1920', '1933', 'Gallimard', '1909',
       '1946', '2008', '1378', '2030', '1936', '1947', '2011', '2020',
       '1919', '1949', '1922', '1897', '2024', '1376', '1926', '2037'],
      dtype=object)
```

In [32]:
```
books.loc[books['Year-Of-Publication']=='DK Publishing Inc',:]
```

Out[32]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
209538	078946697X	DK Readers: Creating the X-Men, How It All Beg...	2000	DK Publishing Inc	http://images.amazon.com/images/P/078946697X.0...
221678	0789466953	DK Readers: Creating the X-Men, How Comic Book...	2000	DK Publishing Inc	http://images.amazon.com/images/P/0789466953.0...

In [49]:
```
books.at[209538,'Publisher']='DK Publishing Inc'
books.at[221678,'Publisher']='DK Publishing Inc'
books.at[209538,'Year-Of-Publication']=2000
books.at[221678,'Year-Of-Publication']=2000
```

In [56]:
```
books.loc[books['Year-Of-Publication']=='Gallimard',:]
```

Out[56]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
220731	2070426769	Peuple du ciel, suivi de 'Les Bergers\";Jean-M...	2003	Gallimard	Gallimard

In [57]:
```
books.at[220731,'Publisher']='Gallimard'
books.at[220731,'Year-Of-Publication']=2003
```

In [58]:
```
books['Year-Of-Publication']=books['Year-Of-Publication'].astype(int)
```

In [59]:
```
print(sorted(list(books['Year-Of-Publication'].unique())))
```

```
[0, 1376, 1378, 1806, 1897, 1900, 1901, 1902, 1904, 1906, 1908, 1909, 1910, 1911, 1914, 1917, 1919, 1920, 1921, 1922, 1923, 192
4, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 194
5, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 196
6, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 198
7, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 201
0, 2011, 2012, 2020, 2021, 2024, 2026, 2030, 2037, 2038, 2050]
```

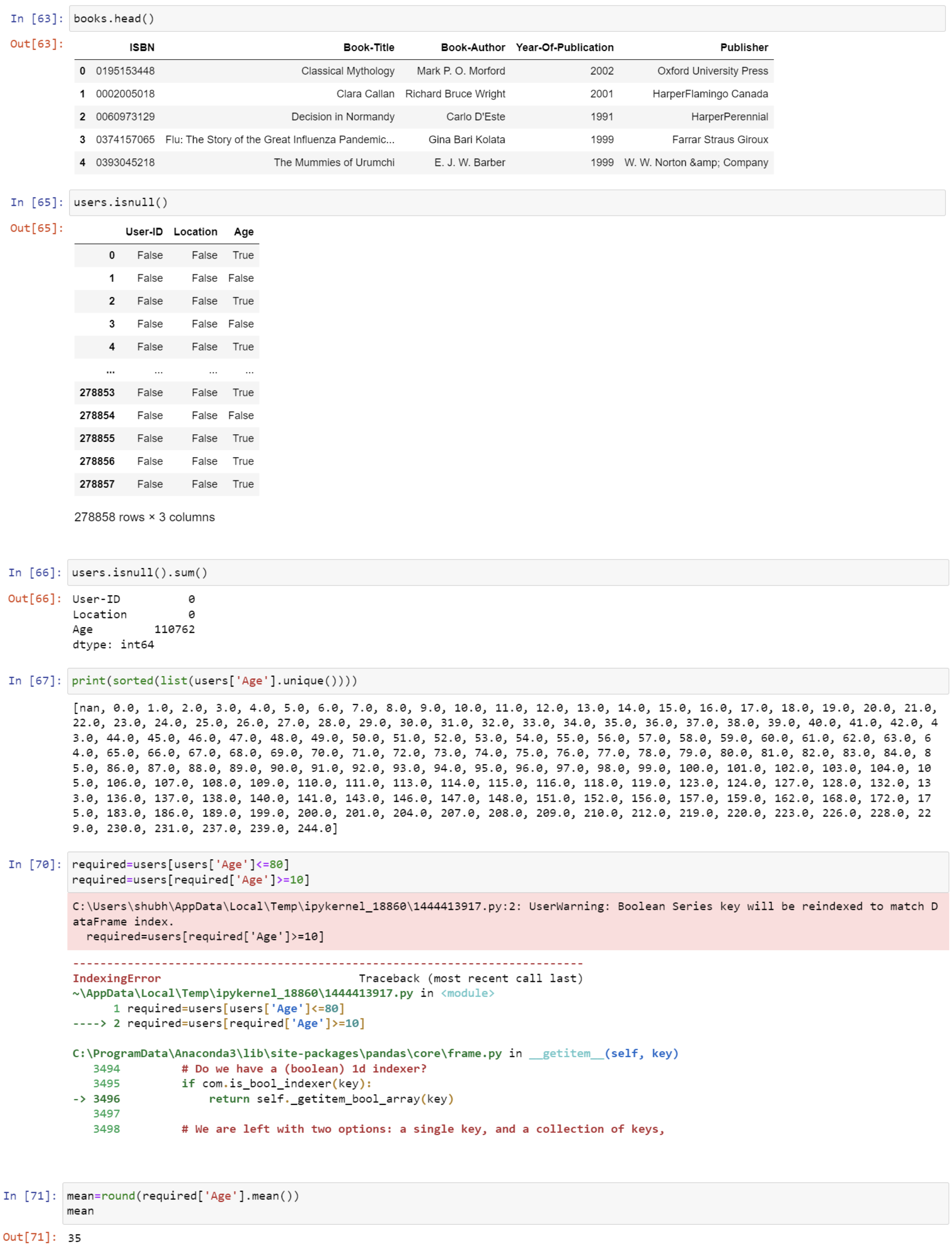
In [60]:
```
from collections import Counter
count=Counter(books['Year-Of-Publication'])
[k for k,v in count.items() if v==max(count.values())]
```

Out[60]:
```
[2002]
```

In [61]:
```
books.loc[books['Year-Of-Publication']>2001,'Year-Of-Publication']=2002
books.loc[books['Year-Of-Publication']==0,'Year-Of-Publication']=2002
```

In [62]:
```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271360 entries, 0 to 271359
Data columns (total 5 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   ISBN                 271360 non-null  object
 1   Book-Title           271360 non-null  object
 2   Book-Author          271360 non-null  object
 3   Year-Of-Publication  271360 non-null  int32 
 4   Publisher            271360 non-null  object
dtypes: int32(1), object(4)
memory usage: 9.3+ MB
```



```
In [63]: books.head()
Out[63]:
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &amp; Company

```
In [65]: users.isnull()
Out[65]:
```

	User-ID	Location	Age
0	False	False	True
1	False	False	False
2	False	False	True
3	False	False	False
4	False	False	True
...	...	...	...
278853	False	False	True
278854	False	False	False
278855	False	False	True
278856	False	False	True
278857	False	False	True

278858 rows × 3 columns

```
In [66]: users.isnull().sum()
Out[66]: User-ID          0
         Location         0
         Age         110762
         dtype: int64
```

```
In [67]: print(sorted(list(users['Age'].unique())))

         [nan, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0,
         22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 4
         3.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 59.0, 60.0, 61.0, 62.0, 63.0, 6
         4.0, 65.0, 66.0, 67.0, 68.0, 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 84.0, 8
         5.0, 86.0, 87.0, 88.0, 89.0, 90.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0, 99.0, 100.0, 101.0, 102.0, 103.0, 104.0, 10
         5.0, 106.0, 107.0, 108.0, 109.0, 110.0, 111.0, 113.0, 114.0, 115.0, 116.0, 118.0, 119.0, 123.0, 124.0, 127.0, 128.0, 132.0, 13
         3.0, 136.0, 137.0, 138.0, 140.0, 141.0, 143.0, 146.0, 147.0, 148.0, 151.0, 152.0, 156.0, 157.0, 159.0, 162.0, 168.0, 172.0, 17
         5.0, 183.0, 186.0, 189.0, 199.0, 200.0, 201.0, 204.0, 207.0, 208.0, 209.0, 210.0, 212.0, 219.0, 220.0, 223.0, 226.0, 228.0, 22
         9.0, 230.0, 231.0, 237.0, 239.0, 244.0]
```

```
In [70]: required=users[users['Age']<=80]
         required=users[required['Age']>=10]

         C:\Users\shubh\AppData\Local\Temp\ipykernel_18860\1444413917.py:2: UserWarning: Boolean Series key will be reindexed to match D
         ataFrame index.
           required=users[required['Age']>=10]

         ---------------------------------------------------------------------------
         IndexingError                             Traceback (most recent call last)
         ~\AppData\Local\Temp\ipykernel_18860\1444413917.py in <module>
               1 required=users[users['Age']<=80]
         ----> 2 required=users[required['Age']>=10]

         C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
            3494         # Do we have a (boolean) 1d indexer?
            3495         if com.is_bool_indexer(key):
         -> 3496             return self._getitem_bool_array(key)
            3497
            3498         # We are left with two options: a single key, and a collection of keys,
```

```
In [71]: mean=round(required['Age'].mean())
         mean
Out[71]: 35
```

Conclusion : We have successfully implemented data wrangling(gg plot) using jupyter notebook.