ADV-C191

DATA SET

What is our GOAL for this MODULE?

We learned how to analyse 2 data sets and based on that we plotted the bar graphs.

What did we ACHIEVE in the class TODAY?

- We have completed plotting 2 bar graphs using a data file.

Which CONCEPTS/ CODING did we cover today?

- Plot bar graphs for the countries who have the heaviest satellites.
- Plot bar graphs of the countries satellites which consume the least power in space.

How did we DO the activities?

Activity 1-

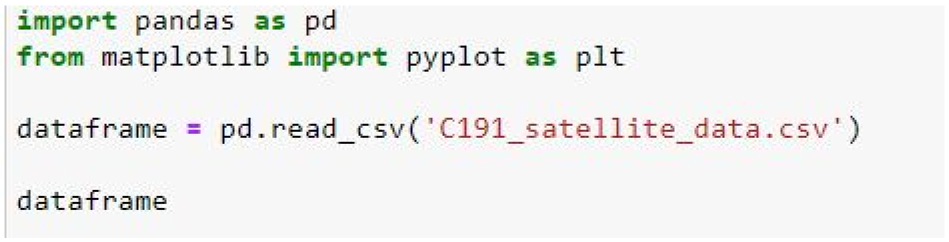
Objective - Sort the data and find out the which Countries has the heaviest satellites, and plot a bar graph out of it

```
#Activity-1
#Q - Sort the data and find out the which Countries has the heaviest satellites, and plot a bar graph out of it
import pandas as pd
from matplotlib import pyplot as plt

dataframe = pd.read_csv('C191_satellite_data.csv')

dataframe
```

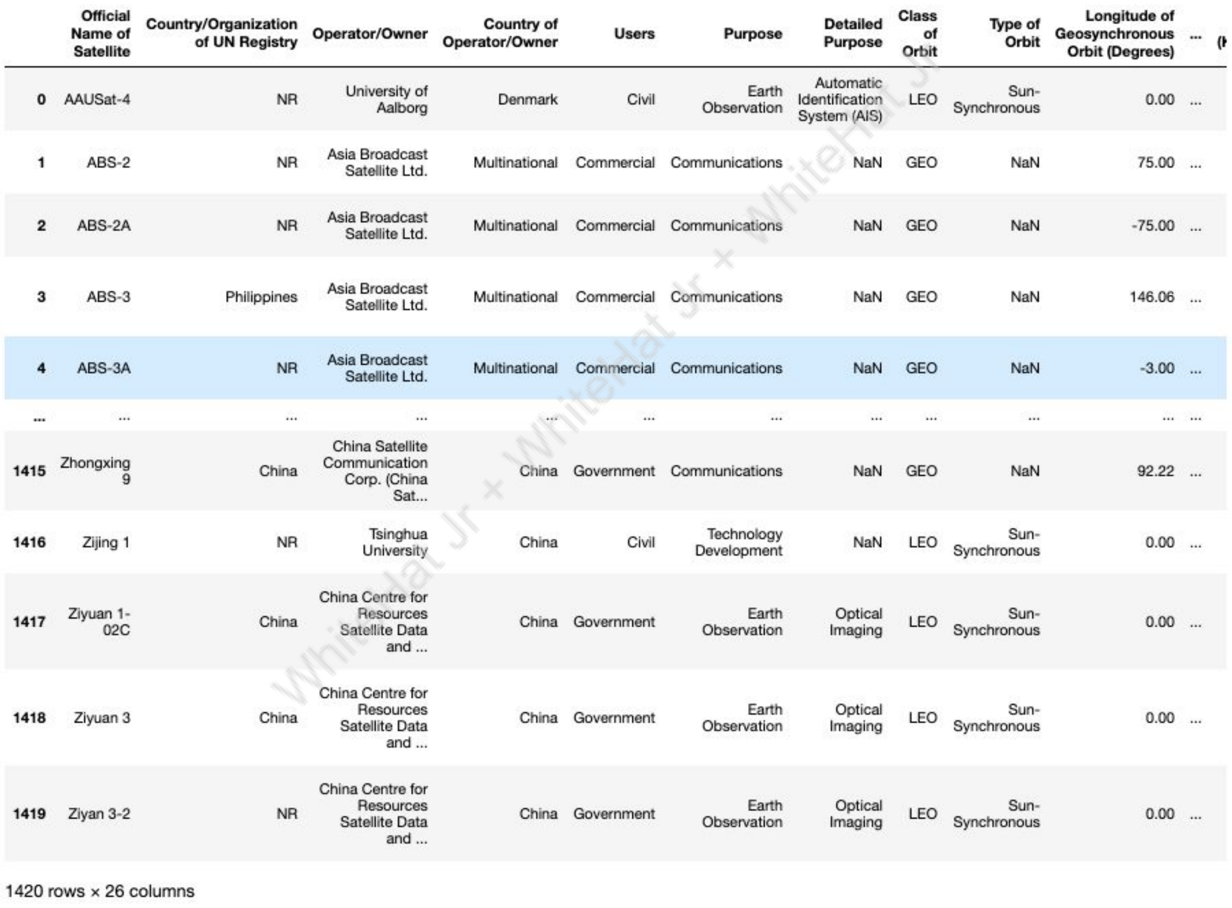
First we will import all the required packages. As we know the commonly used packages for data visualization are **pandas** and **matplotlib:**

- Pandas - Library used to read and manipulate data. We will import pandas and create an alias or short name for pandas that is **pd** using **as** keyword
  ```
  import pandas as pd
  ```
- Matplotlib - Library used to draw graphs and help visualize the data in graphical format. We will import pyplot from matplotlib and create an alias or short name for pyplot that is **plt** using **as** keyword
  ```
  import pandas as pd
  from matplotlib import pyplot as plt
  ```

1. We will **plot a bar graph for 5 Satellites who are the heaviest.** For this we have to first get the data. We have the data required for this activity stored in **C191_satellite_data.csv**. So we read the data from this file and store it in a variable, the function to read this file is **read_csv()**
   Syntax:
   pd.read_csv('file_name')
   'file_name'- should be the name of the file that is **C191_satellite_data.csv**
   And store it in a variable
   ```
   dataframe = pd.read_csv('C191_satellite_data.csv')
   ```
   If you try to print the variable data frame you will get the following -



```
import pandas as pd
from matplotlib import pyplot as plt

dataframe = pd.read_csv('C191_satellite_data.csv')

dataframe
```



	Official Name of Satellite	Country/Organization of UN Registry	Operator/Owner	Country of Operator/Owner	Users	Purpose	Detailed Purpose	Class of Orbit	Type of Orbit	Longitude of Geosynchronous Orbit (Degrees)	...
0	AAUSat-4	NR	University of Aalborg	Denmark	Civil	Earth Observation	Automatic Identification System (AIS)	LEO	Sun-Synchronous	0.00	...
1	ABS-2	NR	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	75.00	...
2	ABS-2A	NR	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	-75.00	...
3	ABS-3	Philippines	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	146.06	...
4	ABS-3A	NR	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	-3.00	...
...	...	...	...	...	...	...	...	...	...	...	...
1415	Zhongxing 9	China	China Satellite Communication Corp. (China Sat...	China	Government	Communications	NaN	GEO	NaN	92.22	...
1416	Zijing 1	NR	Tsinghua University	China	Civil	Technology Development	NaN	LEO	Sun-Synchronous	0.00	...
1417	Ziyuan 1-02C	China	China Centre for Resources Satellite Data and ...	China	Government	Earth Observation	Optical Imaging	LEO	Sun-Synchronous	0.00	...
1418	Ziyuan 3	China	China Centre for Resources Satellite Data and ...	China	Government	Earth Observation	Optical Imaging	LEO	Sun-Synchronous	0.00	...
1419	Ziyan 3-2	NR	China Centre for Resources Satellite Data and ...	China	Government	Earth Observation	Optical Imaging	LEO	Sun-Synchronous	0.00	...

1420 rows × 26 columns

Thus this table has **1420 rows and 26 columns.**
Just see the data and keep in mind that there are some rows which have NaN values and empty values.
We don't require the entire 26 columns as the objective of this activity is to find out the 5 name of the Satellite which are the most heaviest satellites in the space
Therefore we will need the following columns from the table:

- **'Dry Mass (Kilograms)'** - dry mass means - The **mass** means the weight of an object/satellite. Therefore we will use this column as it gives the weight of the satellite
- And **'Official Name of Satellite'** - That is the name of the satellite

These 2 columns are required to satisfy the objective of this activity
Therefore we will create a new table which has only these 2 columns.
We will use a class from pandas library named **DataFrame()** as this class
Using this class we can create a new table with just these 2 columns
Syntax:
pd.DataFrame()
This function takes following parameter-

- Data - a variable holding the data, **dataframe** variable holds the data
- Columns - this parameter takes column names which you would like to have in this table. **'Dry Mass (Kilograms)' , 'Official Name of Satellite'** we will give these column names in list as these are the 2 columns that we want in our table.

And store it in a variable.

```
df = pd.DataFrame(dataframe, columns =['Dry Mass (Kilograms)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)
```

2. As you saw that the table has multiple NaN values and empty values in the table cell. These values usually clutter the data leading to unclear visualization and wrong information. Therefore we are supposed to clean the data, that is remove the Nan and empty values from the table.
There is no function to remove the empty values rows from the table, so we will replace empty values with Nan. As we have **dropna()** function which removes the rows from the table which contain Nan values. This was we can get rid of all Nan and empty values
   - First we will replace the empty values with Nan, for this we will be using the **replace()** function on the table. This function is from pandas library.
     Syntax:
     df.replace(to_replace , value, inplace=True/False)
     Where **to_replace** - is the value that we want to replace, we want to replace empty value so we will give empty string (" ")
     value - The value with which we want to replace, we want to replace empty string with Nan. Nan is not a string is it a value, **float("NaN")** gives value Nan
     Inplace - if this parameter is set to True then the replace() will be applied on

the current working table that is the variable df which is holding the table.
And if this parameter is set to False then a copy of this data frame is created in which the empty values will be replaced with Nan values.
We will be setting this parameter to **True** as we don't want to create another copy of this table instead we want to replace the empty value with NaN value in the same table.

```
df = pd.DataFrame(dataframe, columns =['Dry Mass (Kilograms)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)
```

- Then we will use **dropna()** function on the table to remove all the NaN values from the table and store it in a variable

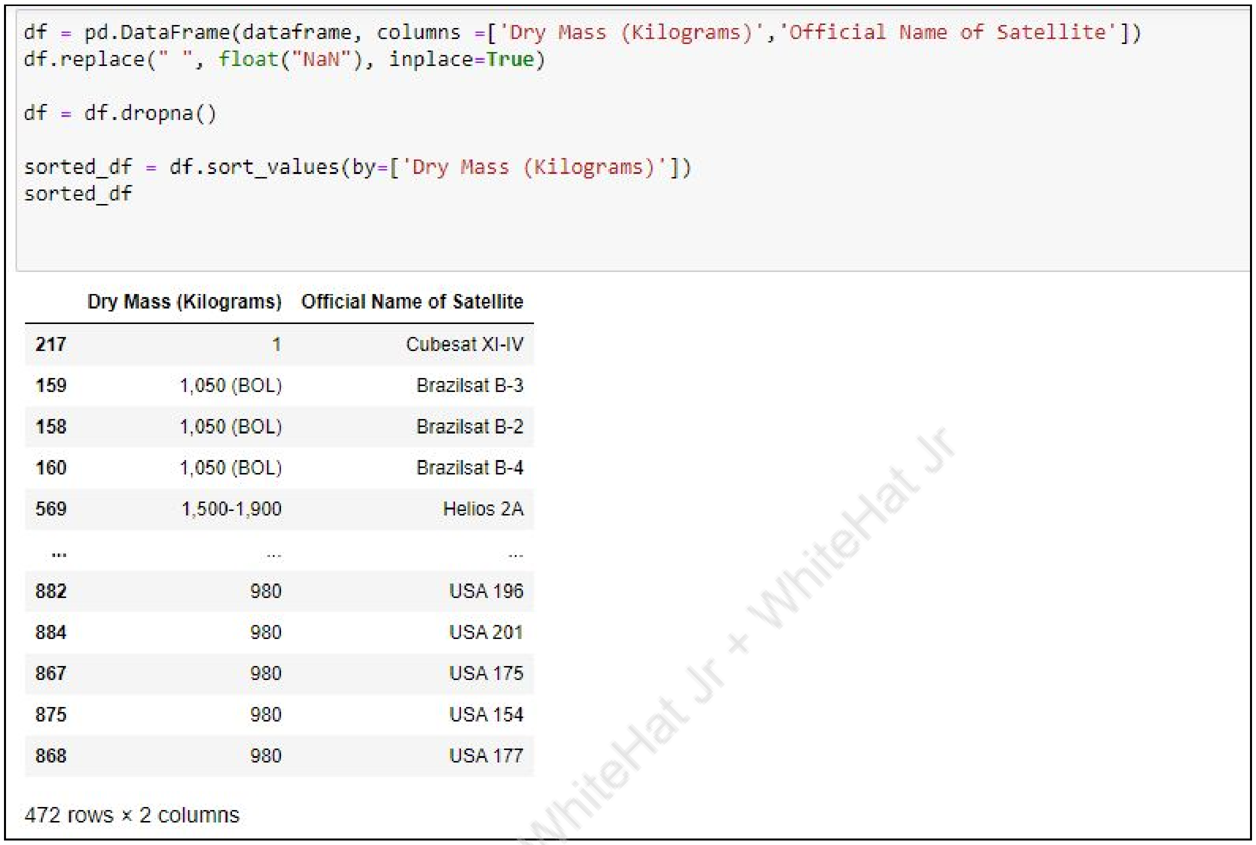
```
df = pd.DataFrame(dataframe, columns =['Dry Mass (Kilograms)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)

df = df.dropna()
```

3. As the table contains values randomly stored we have to sort the table based on 'Dry Mass (Kilograms)' column as this is the column that holds the weight of the satellite. Therefore we will sort the column by using **sort_values()** function on the table. This is a function of pandas which helps us to sort the data.
   Syntax:
   df.sort_values(by=['column_name'])
   Where **column_name** - is the name of the column based on which we want to sort the table. 'Dry Mass (Kilograms)' based on which we want to sort the table.

```
df = pd.DataFrame(dataframe, columns =['Dry Mass (Kilograms)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)

df = df.dropna()

sorted_df = df.sort_values(by=['Dry Mass (Kilograms)'])
```

   sorted_df variable will hold the table which is sorted based on 'Dry Mass (Kilograms)'

4. Then just write the variable name and execute the cell to view the table.



```
df = pd.DataFrame(dataframe, columns =['Dry Mass (Kilograms)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)

df = df.dropna()

sorted_df = df.sort_values(by=['Dry Mass (Kilograms)'])
sorted_df
```

	Dry Mass (Kilograms)	Official Name of Satellite
217	1	Cubesat XI-IV
159	1,050 (BOL)	Brazilsat B-3
158	1,050 (BOL)	Brazilsat B-2
160	1,050 (BOL)	Brazilsat B-4
569	1,500-1,900	Helios 2A
...	...	...
882	980	USA 196
884	980	USA 201
867	980	USA 175
875	980	USA 154
868	980	USA 177

472 rows × 2 columns

Write the following code in next cell of jupyter notebook

5. Now that the data is sorted, to get the top 5 heavy satellites we just have to get the 1st 5 rows of the table. To get the 1st 5 rows we will use **head()** function from pandas library
   sorted_df.head(number)
   Where **sorted_df** - is the variable holding sorted data in the tabular format.
   number - is integer, that is the initial number of rows to be read/get from the table. 5 as we just need 1st 5 rows

```
heavy_satellite_5 = sorted_df.head(5)
```

6. Print the variable **heavy_satellite_5** which holds the 1st 5 rows of the table which have the 5 most heavy satellites in space.

```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)
```

7. To plot the bar graph which displays the 5 heaviest satellites in space.
   We need to have 2 array:
   - 1 holding the 1st 5 most heavy satellite names and
   - 2nd array to hold the weight of these satellites.

```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']
```

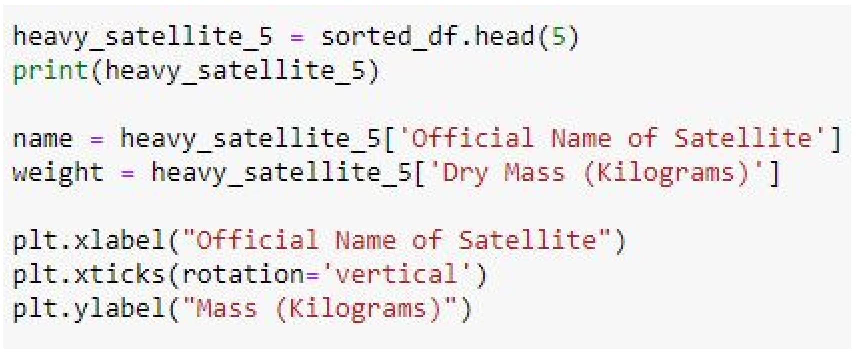
   - First we will create an 1D array holding satellite names and store it in a variable. **heavy_satellite_5** variable holds the 1st 5 rows which are the 5 most heavy satellites. So we will use this variable and pass the column name 'Official Name of Satellite' in a square bracket `= heavy_satellite_5['Official Name of Satellite']`. This will give 1D array holding the 5 most heavy satellites names. Therefore we will store this in a variable.

```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
```

   - Then we will create an array holding the weight of these satellites and store it in a variable. **heavy_satellite_5** variable holds the 1st 5 rows which are the 5 most heavy satellites. So we will use this variable and pass the column name 'Dry Mass (Kilograms)' as it contains the weight of each satellites in a square bracket `heavy_satellite_5['Dry Mass (Kilograms)']`. This will give 1D array holding the weight of satellites. Therefore we will store this list in a variable

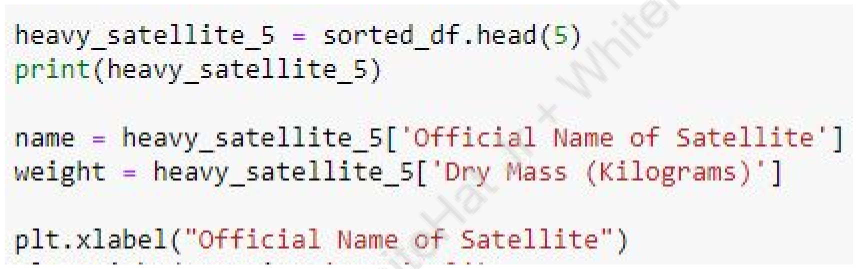
```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']
```

8. Then we will setup the plot on which we want to draw the bar graph, by naming the x axis and y axis



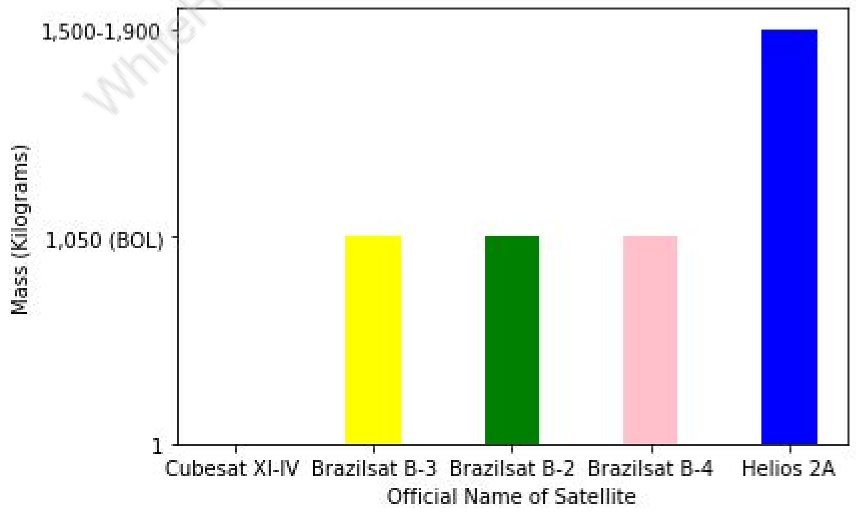
```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Mass (Kilograms)")
```

- Then we will set the label of x axis as "Official Name of Satellite" as the satellite names will be placed on the x axis. To set the label for the x axis we use **xlabel()** function. This is a function of pyplot module, therefore we use the module name **plt** (alias of pyplot module)



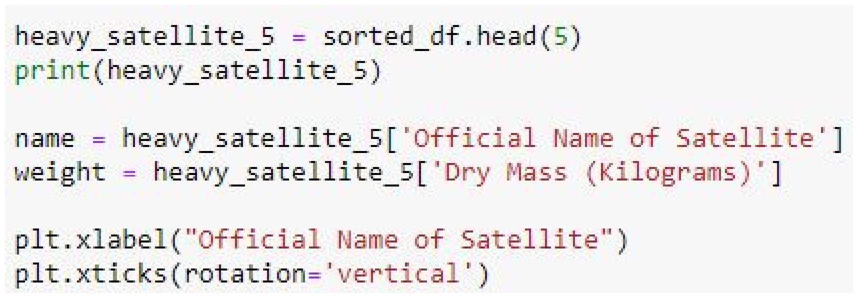
```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']

plt.xlabel("Official Name of Satellite")
```

- Now the names of the satellites on the x axis will be displayed horizontal by default and as the plot is small and the country name and long the country names overlap on each other and might give output something like this -



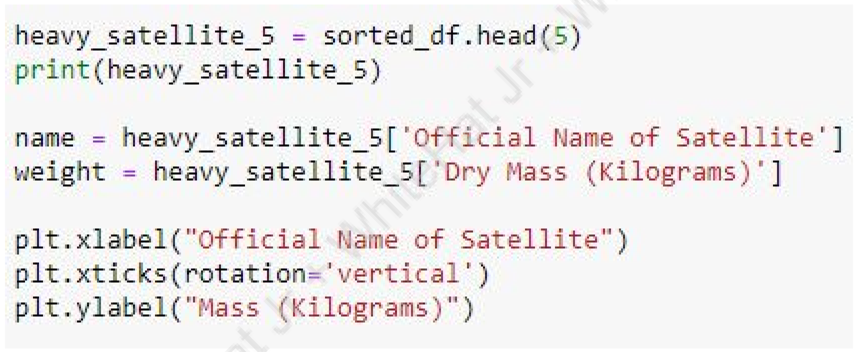
This leads to the satellite name not clear to understand. Therefore we set the values on the x axis vertically. For this we use the **xticks()** function and set the

rotation parameter as **'vertical'.** This is a function of pyplot module, therefore we use the module name **plt** (alias of pyplot module)



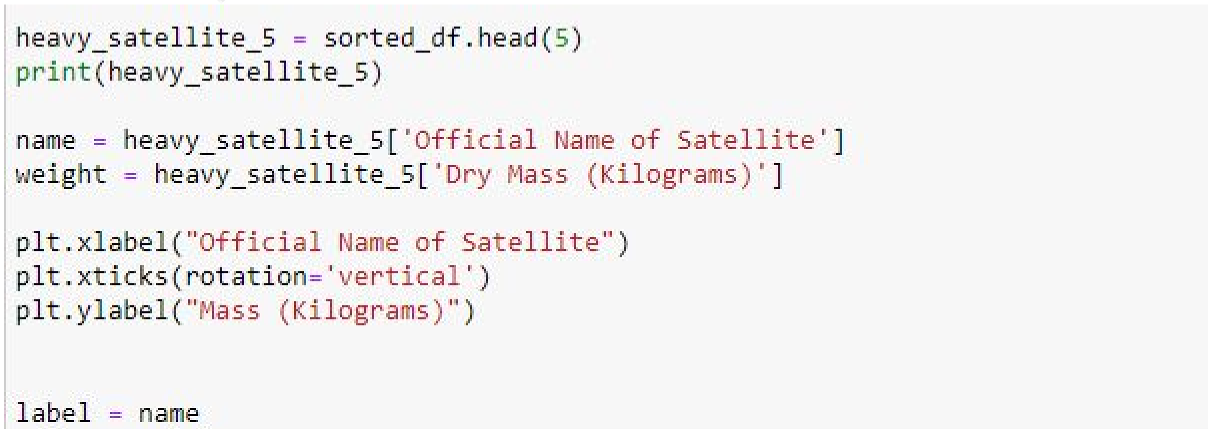
```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
```

- Then we will set the label of y axis as "Mass (Kilograms)" as the number of satellites will be placed on the y axis. To set the label for the y axis we use **ylabel()** function. This is a function of pyplot module, therefore we use the module name **plt** (alias of pyplot module)



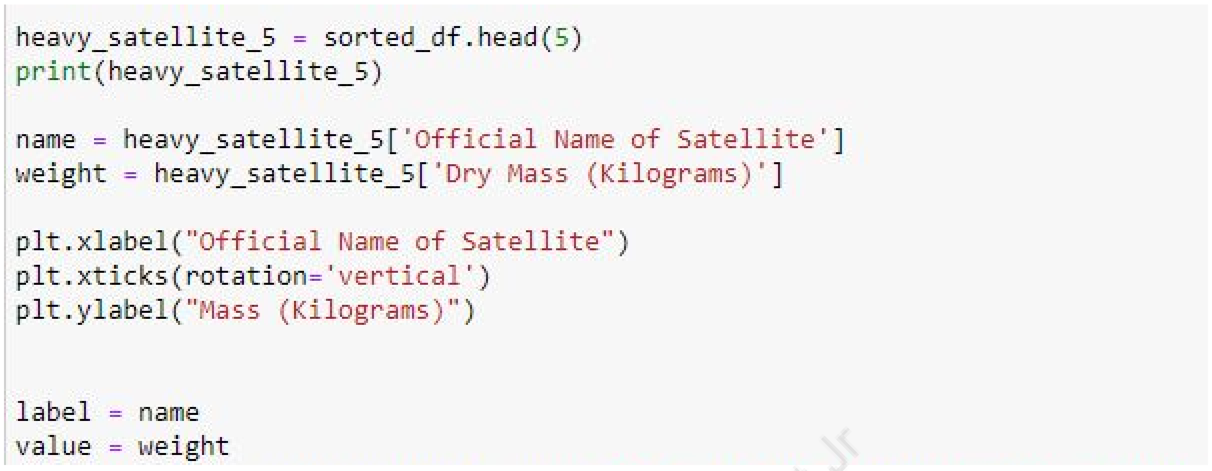
```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Mass (Kilograms)")
```

9. Now we will store the list of all satellite name which was stored in 'name' variable to **label** variable



```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Mass (Kilograms)")


label = name
```

10. Then we will store the list of weights of satellites in space which was stored in 'weight' variable to **value** variable



```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Mass (Kilograms)")


label = name
value = weight
```

11. Now we will plot the bar graph. To plot the bar graph we will use the **bar()** function from the pyplot module.
    Syntax.
    plt.bar(x_value, y_values, width=width_value, colors=tuple_of_colors)
    Where **x_value** - sequence of scalars , list of values for x coordinates. **label** variable holds the list of all country name which will be our x coordinate values
    Y_value = sequence of scalars representing the height(s) of the bars, list of values for y coordinates representing the height of the bars. **value** variable holds the list of number of satellites for respective country which will be our y coordinate values
    Width - this parameter set the width of the bar on the bar graph, this parameter takes value in decimal. We are setting the width of the bar as **0.4**
    Colors - take a set of colors in a tuple. **('red','blue','green','pink','yellow')** these colors will give 5 respective bar on the plot

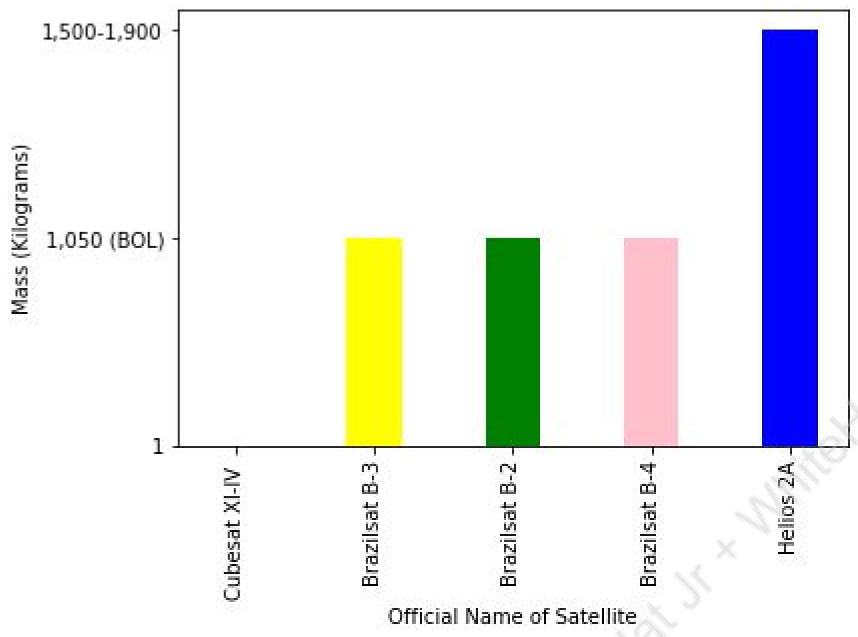
```
heavy_satellite_5 = sorted_df.head(5)
print(heavy_satellite_5)

name = heavy_satellite_5['Official Name of Satellite']
weight = heavy_satellite_5['Dry Mass (Kilograms)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Mass (Kilograms)")


label = name
value = weight
plt.bar(label, value,width=0.4, color=('red','yellow','green','pink','blue'))
```

Output of this code is:

```
    Dry Mass (Kilograms) Official Name of Satellite
217                    1             Cubesat XI-IV
159          1,050 (BOL)             Brazilsat B-3
158          1,050 (BOL)             Brazilsat B-2
160          1,050 (BOL)             Brazilsat B-4
569          1,500-1,900                Helios 2A
```

This is the output of print(heavy_satellite_5) as it holds the 1st 5 rows of the table

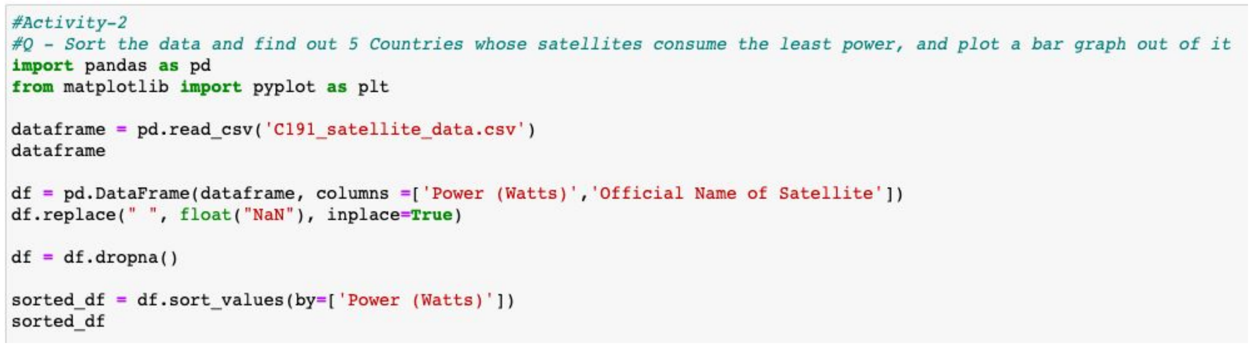
```
<BarContainer object of 5 artists>
```



And a plot holding the 5 most heavy satellites. Here 1 means 1000 kilograms.

Activity 2-

Objective - Sort the data and find out 5 Countries whose satellites consume the least power, and plot a bar graph out of it



```
#Activity-2
#Q - Sort the data and find out 5 Countries whose satellites consume the least power, and plot a bar graph out of it
import pandas as pd
from matplotlib import pyplot as plt

dataframe = pd.read_csv('C191_satellite_data.csv')
dataframe

df = pd.DataFrame(dataframe, columns =['Power (Watts)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)

df = df.dropna()

sorted_df = df.sort_values(by=['Power (Watts)'])
sorted_df
```

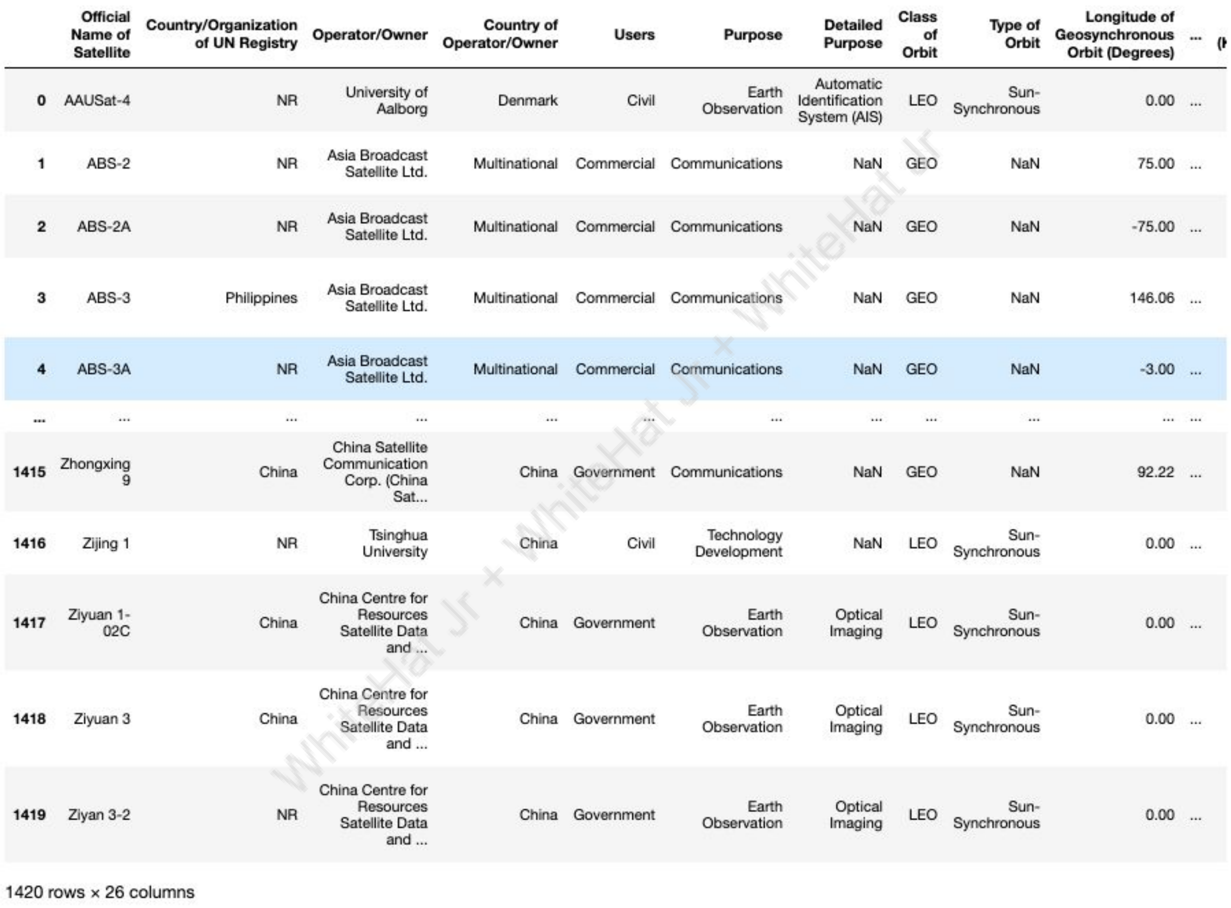
First we will import all the required packages. As we know the commonly used packages for data visualization are **pandas** and **matplotlib:**

Now we will use the same csv file

12. We will **plot a bar graph for 5 countries whose satellites consume the least power.** For this we have to first get the data. We have the data required for this activity stored in **C191_satellite_data.csv.** So we read the data from this file and store it in a variable **dataframe** using the function **read_csv().**

```
dataframe = pd.read_csv('C191_satellite_data.csv')
```

If you try to print the variable data frame you will get the following -



	Official Name of Satellite	Country/Organization of UN Registry	Operator/Owner	Country of Operator/Owner	Users	Purpose	Detailed Purpose	Class of Orbit	Type of Orbit	Longitude of Geosynchronous Orbit (Degrees)	...
0	AAUSat-4	NR	University of Aalborg	Denmark	Civil	Earth Observation	Automatic Identification System (AIS)	LEO	Sun-Synchronous	0.00	...
1	ABS-2	NR	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	75.00	...
2	ABS-2A	NR	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	-75.00	...
3	ABS-3	Philippines	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	146.06	...
4	ABS-3A	NR	Asia Broadcast Satellite Ltd.	Multinational	Commercial	Communications	NaN	GEO	NaN	-3.00	...
...	...	...	...	...	...	...	...	...	...	...	...
1415	Zhongxing 9	China	China Satellite Communication Corp. (China Sat...	China	Government	Communications	NaN	GEO	NaN	92.22	...
1416	Zijing 1	NR	Tsinghua University	China	Civil	Technology Development	NaN	LEO	Sun-Synchronous	0.00	...
1417	Ziyuan 1-02C	China	China Centre for Resources Satellite Data and ...	China	Government	Earth Observation	Optical Imaging	LEO	Sun-Synchronous	0.00	...
1418	Ziyuan 3	China	China Centre for Resources Satellite Data and ...	China	Government	Earth Observation	Optical Imaging	LEO	Sun-Synchronous	0.00	...
1419	Ziyan 3-2	NR	China Centre for Resources Satellite Data and ...	China	Government	Earth Observation	Optical Imaging	LEO	Sun-Synchronous	0.00	...

1420 rows × 26 columns

Now to satisfy our objective we need two columns, which are the **'Power (Watts)'** and **'Official Name of Satellite'.**

Therefore we will need the following columns from the table:

- **'Power (Watts)'** - The **Power** means the consumption of power by the object/satellite. Therefore we will use this column as it gives the power of the satellite in watts.
- And **'Official Name of Satellite'** - That is the name of the satellite

These 2 columns are required to satisfy the objective of this activity
Therefore we will create a new table which has only these 2 columns.
We will use a class from pandas library named **DataFrame()** as this class
Using this class we can create a new table with just these 2 columns
Syntax:
pd.DataFrame()
This function takes following parameter-

- Data - a variable holding the data, **dataframe** variable holds the data
- Columns - this parameter takes column names which you would like to have in this table. **'Power (Watts)'** , **'Official Name of Satellite'** we will give these column names in list as these are the 2 columns that we want in our table.

And store it in a variable **df.**

```
df = pd.DataFrame(dataframe, columns =['Power (Watts)','Official Name of Satellite'])
```

13. As you saw that the table has multiple Nan values and empty values in the table cell. We should remove these cells having Nan values.
dropna() function will remove the rows from the table which contain Nan values. We will use this function to get rid of all NaN and empty values
    - First we will replace the empty values with NaN, for this we will be using the **replace()** function on the table. This function is from pandas library.
    Syntax:
    df.replace(to_replace , value, inplace=True/False)

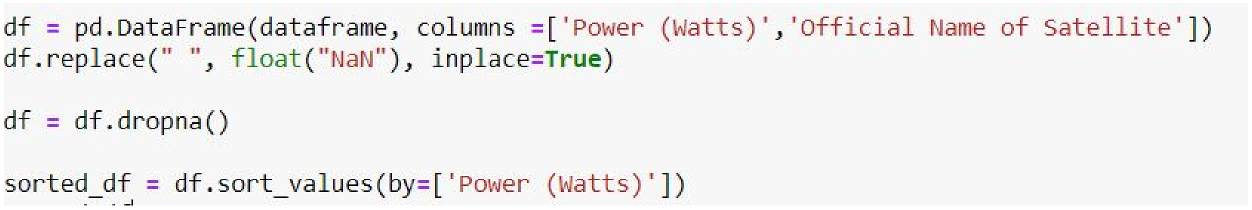
```
df = pd.DataFrame(dataframe, columns =['Power (Watts)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)
```

- Then we will use **dropna()** function on the table to remove all the Nan values from the table and store it in a variable

```
df = pd.DataFrame(dataframe, columns =['Power (Watts)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)

df = df.dropna()
```

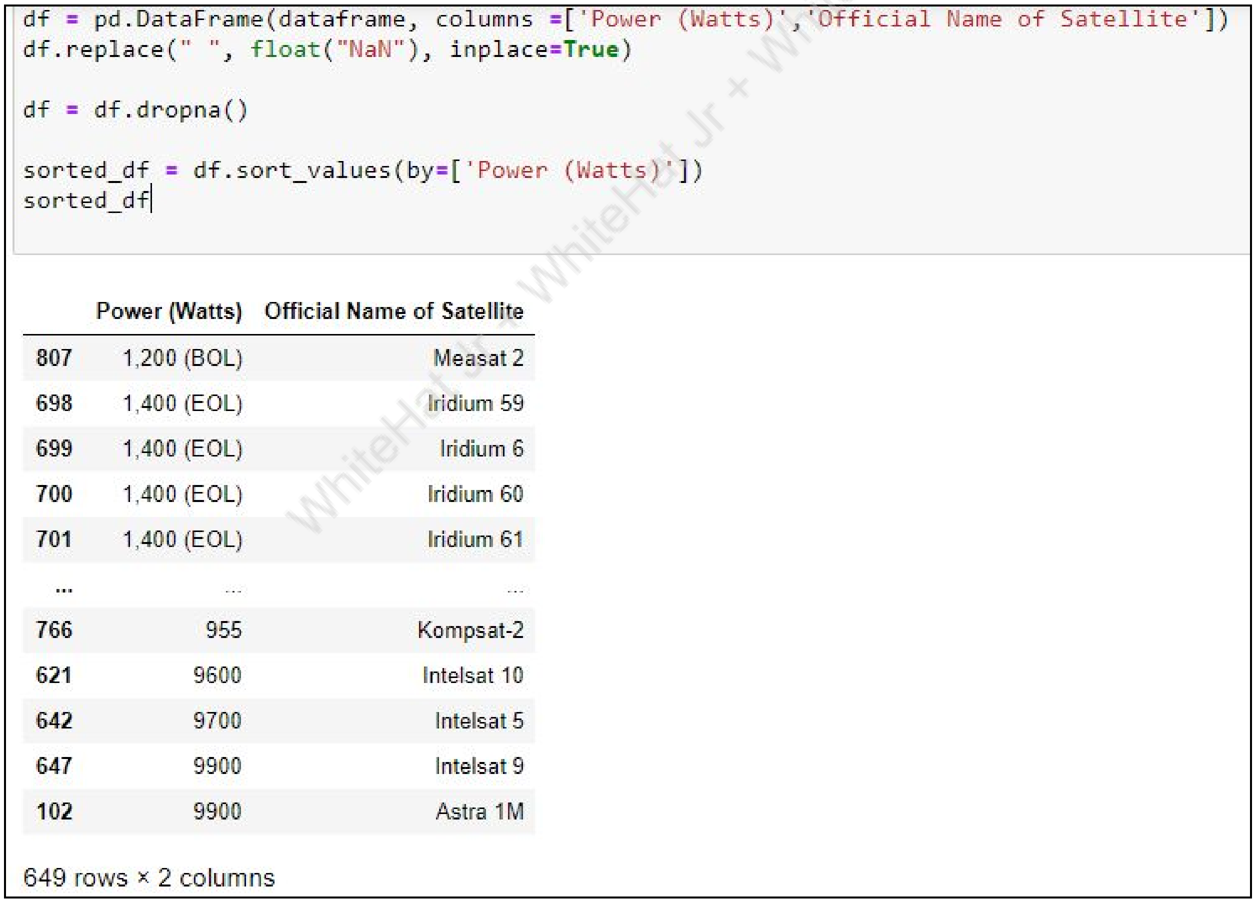
14. As the table contains values randomly stored we have to sort the table on the basis of the "Power (Watts)' column as this is the column that holds the power of the satellite. Therefore we will sort the column by using **sort_values()** function on the table. As we have seen in previous activity, **sort_values()** is a function of pandas which helps us to sort the data.

Syntax:
df.sort_values(by=[‘column_name’])
Where **column_name** - is the name of the column based on which we want to sort the table. ''Power (Watts)' based on which we want to sort the table.

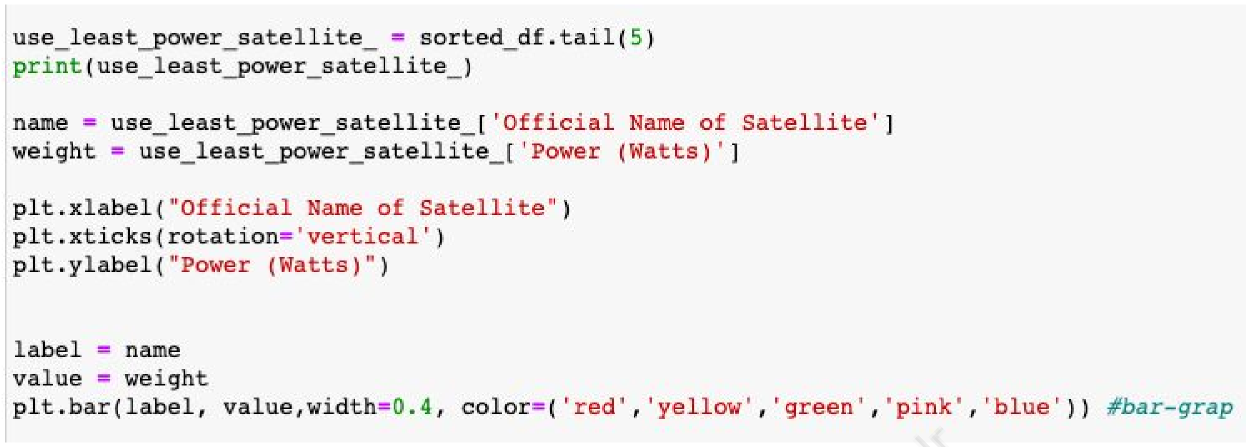

```
df = pd.DataFrame(dataframe, columns =['Power (Watts)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)

df = df.dropna()

sorted_df = df.sort_values(by=['Power (Watts)'])
```

sorted_df variable will hold the table which is sorted based on 'Power(Watts)'

15. Then just write the variable name and execute the cell to view the table.

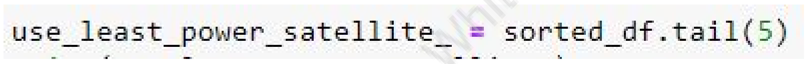

```
df = pd.DataFrame(dataframe, columns =['Power (Watts)','Official Name of Satellite'])
df.replace(" ", float("NaN"), inplace=True)

df = df.dropna()

sorted_df = df.sort_values(by=['Power (Watts)'])
sorted_df
```

```
     Power (Watts)  Official Name of Satellite
807    1,200 (BOL)                    Measat 2
698    1,400 (EOL)                  Iridium 59
699    1,400 (EOL)                   Iridium 6
700    1,400 (EOL)                  Iridium 60
701    1,400 (EOL)                  Iridium 61
...            ...                         ...
766            955                   Kompsat-2
621           9600                 Intelsat 10
642           9700                  Intelsat 5
647           9900                  Intelsat 9
102           9900                    Astra 1M

649 rows × 2 columns
```



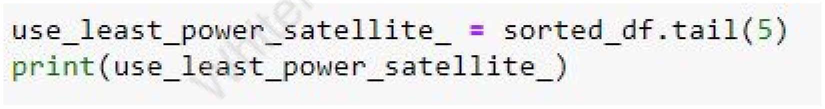
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
weight = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Power (Watts)")


label = name
value = weight
plt.bar(label, value,width=0.4, color=('red','yellow','green','pink','blue')) #bar-grap
```

16. Now that the data is sorted, to get the 5 Countries whose satellites consume the least power, we just have to get the last 5 rows of the table
    To get the 1st 5 rows we will use **tail()** function from pandas library
    sorted_df.tail(number)
    Where sorted_df - is the variable holding sorted data in the tabular format.
    number - is integer, that is the last number of rows to be read/get from the table. 5 as we just need last 5 rows



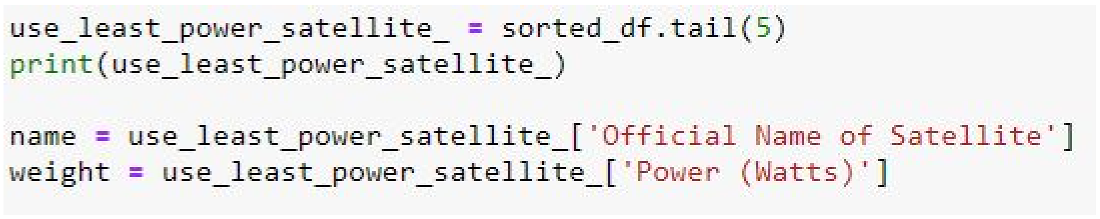
```
use_least_power_satellite_ = sorted_df.tail(5)
```

17. Print the variable **use_least_power_satellite_** which holds the last 5 rows of the table which have the 5 Countries whose satellites consume the least power.



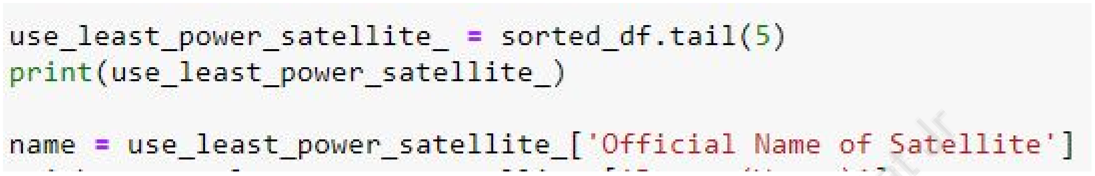
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)
```

18. To plot the bar graph which displays the 5 Countries whose satellites consume the least power.
    We need to have 2 array:
    - 1 holding the least power having satellite names and
    - 2nd array to hold the power of these satellites in watts.



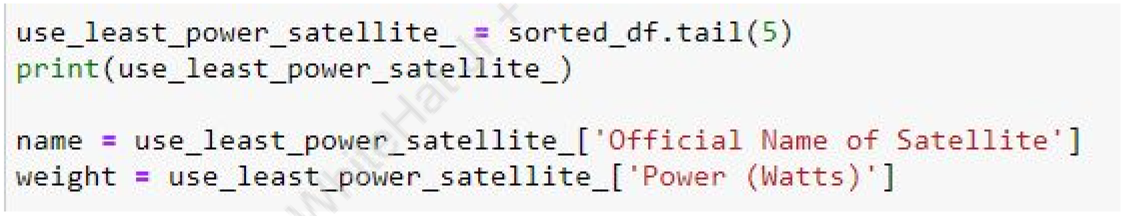
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
weight = use_least_power_satellite_['Power (Watts)']
```

- First we will create an 1D array holding satellite names and store it in a variable. The variable **use_least_power_satellite_** holds the last 5 rows which have 5 countries satellites having least power. So we will use this variable and pass the column name 'Official Name of Satellite' in a square bracket `use_least_power_satellite_['Official Name of Satellite']`. This will give 1D array holding the 5 countries having least power. Therefore we will store this in a variable.



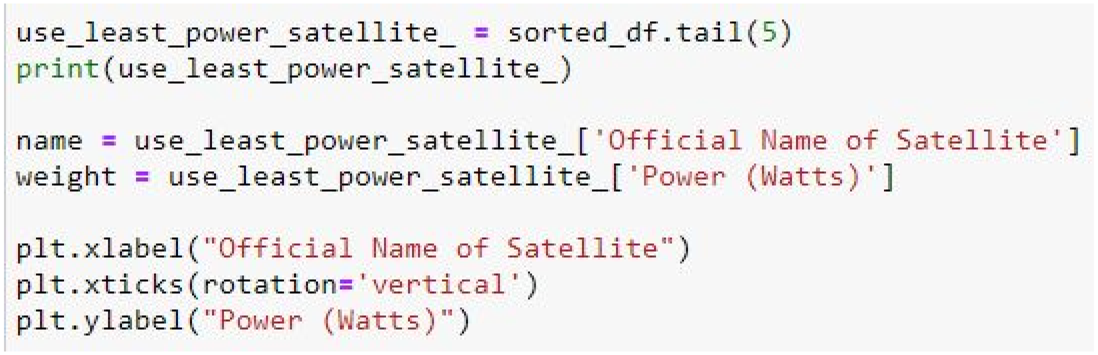
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
```

- Then we will create an array holding the power of these satellites and store it in a variable. The variable **use_least_power_satellite_** variable holds the last 5 rows which have the 5 countries with least power. So we will use this variable and pass the column name "Power (Watts)" as it contains the Power of each satellites in a square bracket `use_least_power_satellite_['Power (Watts)']`. This will give 1D array holding the Power of satellites. Therefore we will store this list in a variable



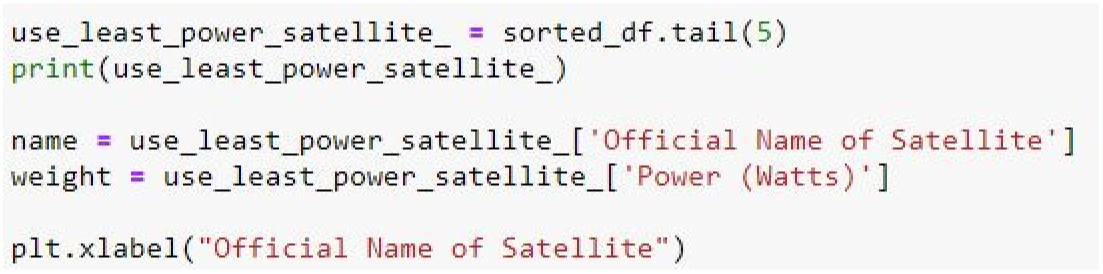
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
weight = use_least_power_satellite_['Power (Watts)']
```

19. Then we will setup the plot on which we want to draw the bar graph, by naming the x axis and y axis



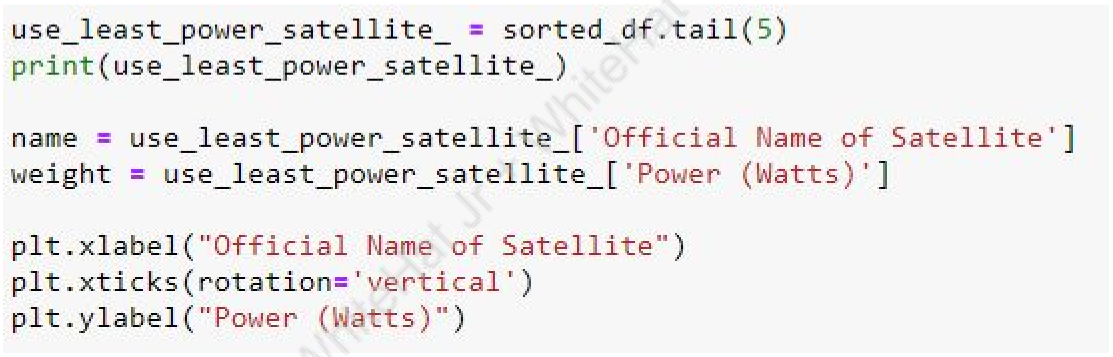
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
weight = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Power (Watts)")
```

- Then we will set the label of x axis as "Official Name of Satellite" as the satellite names will be placed on the x axis. To set the label for the x axis we use **xlabel()** function. This is a function of pyplot module, therefore we use the module name **plt** (alias of pyplot module)



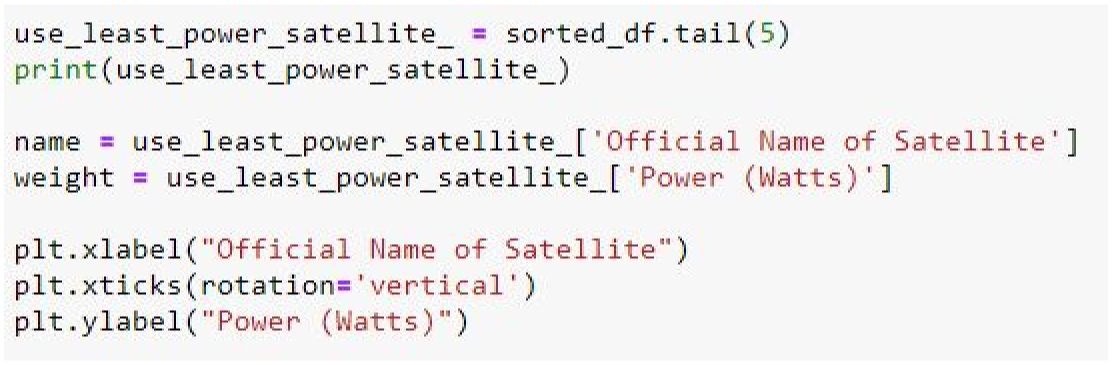
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
weight = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
```

- Now the names of the satellites on the x axis will be displayed horizontal by default and as the plot is small and the satellite names may be long and long the satellite names overlap on each other.
  This leads to the satellite name not clear to understand. Therefore we set the values on the x axis vertically. For this we use the **xticks()** function and set the **rotation** parameter as **'vertical'**. This is a function of pyplot module, therefore we use the module name **plt** (alias of pyplot module)



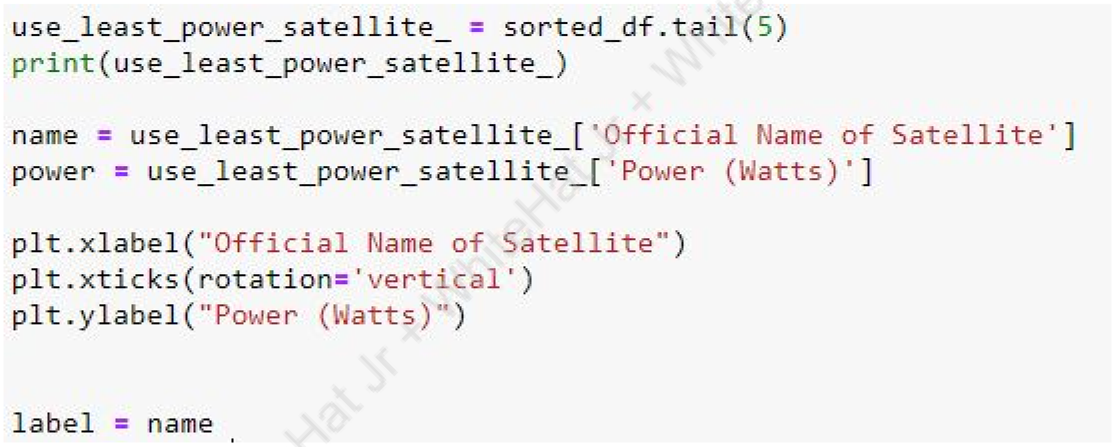
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
weight = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Power (Watts)")
```

- Then we will set the label of y axis as "Power(Watts)" as the number of satellites will be placed on the y axis. To set the label for the y axis we use **ylabel()** function. This is a function of pyplot module, therefore we use the module name **plt** (alias of pyplot module)



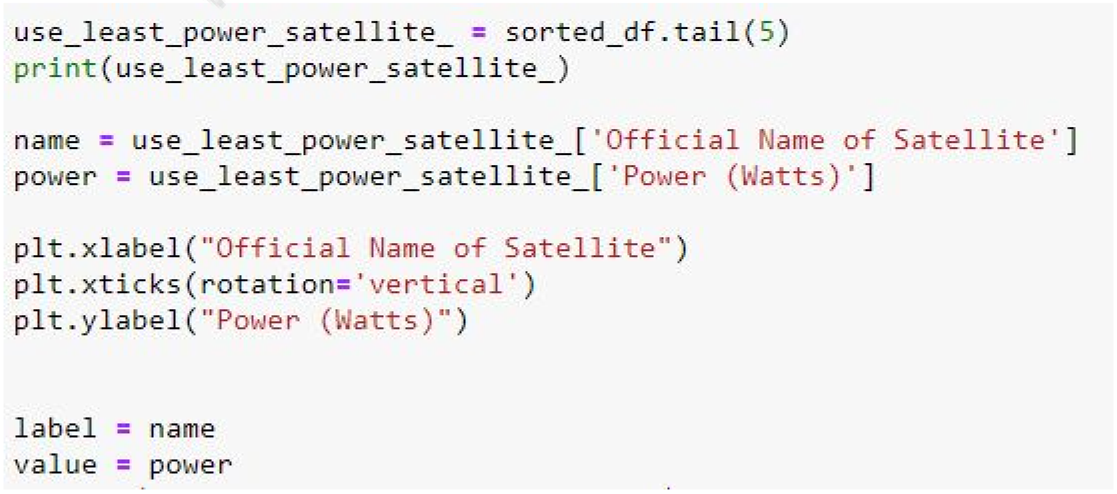
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
weight = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Power (Watts)")
```

20. Now we will store the list of all satellite name which was stored in 'name' variable to **label** variable



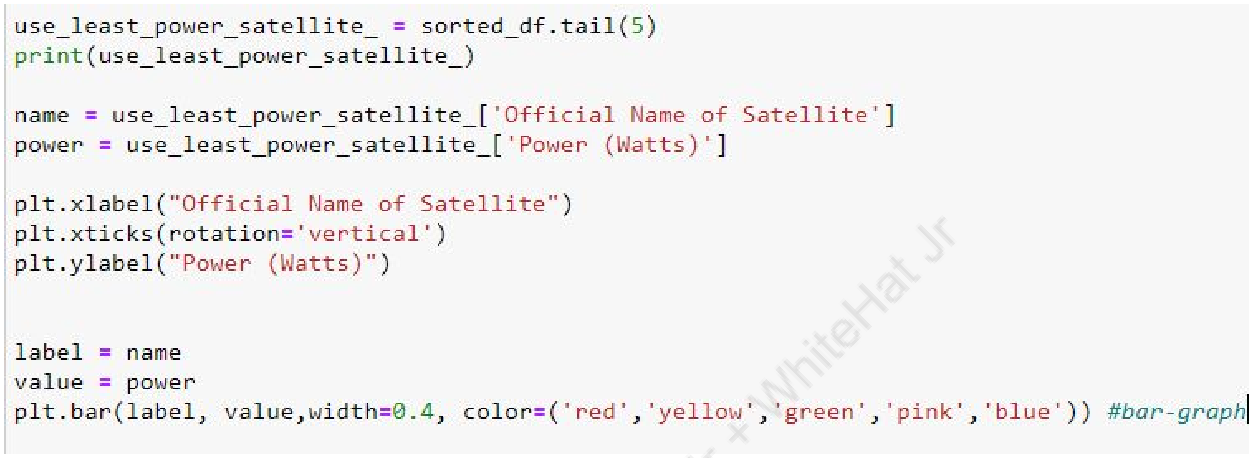
```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
power = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Power (Watts)")


label = name
```

21. Then we will store the list of Power of satellites which was stored in 'power' variable to **value** variable



```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
power = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Power (Watts)")


label = name
value = power
```

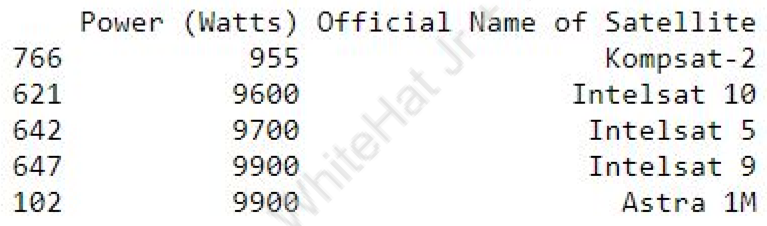
22. Now we will plot the bar graph. To plot the bar graph we will use the **bar()** function from the pyplot module.
    Syntax.
    plt.bar(x_value, y_values, width=width_value, colors=tuple_of_colors)



```
use_least_power_satellite_ = sorted_df.tail(5)
print(use_least_power_satellite_)

name = use_least_power_satellite_['Official Name of Satellite']
power = use_least_power_satellite_['Power (Watts)']

plt.xlabel("Official Name of Satellite")
plt.xticks(rotation='vertical')
plt.ylabel("Power (Watts)")


label = name
value = power
plt.bar(label, value,width=0.4, color=('red','yellow','green','pink','blue')) #bar-graph
```

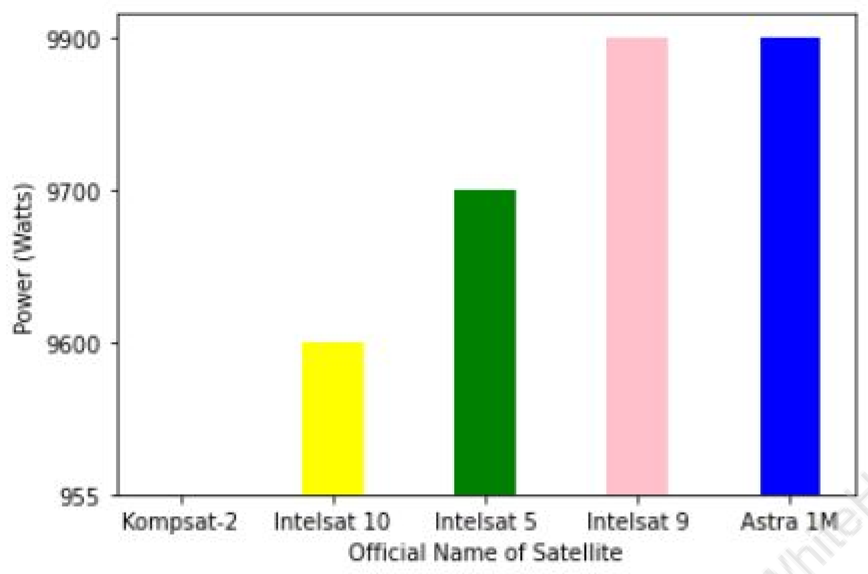
Output of this code is:

This is the output of **use_least_power_satellite_** as it holds the last 5 rows of the table



```
     Power (Watts) Official Name of Satellite
766            955                 Kompsat-2
621           9600               Intelsat 10
642           9700                Intelsat 5
647           9900                Intelsat 9
102           9900                  Astra 1M
```

This is the bar graph plot for the 5 countries satellites having least power.



What's NEXT?

We will continue learning about data visualization and learn how to group the data and plot graphs out of it.