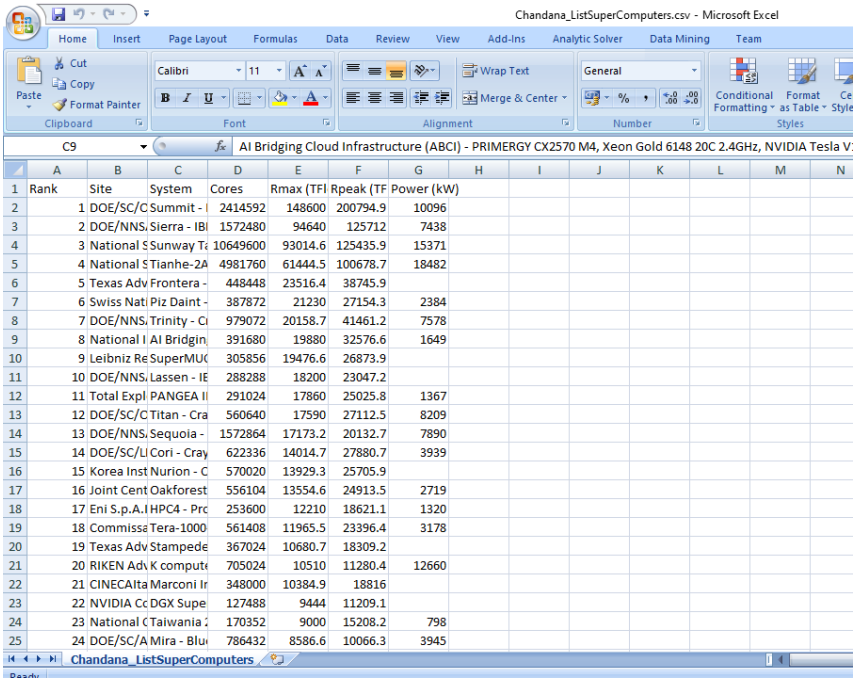
Using the BeautifulSoup we formed the **csv** file named **Chandana_ListSuperComputers.csv** with all the 500 Super Computer Records.

```
In [22]: list_comp=pd.concat([data_page1[0],data_page2[0],data_page3[0],data_page4[0],data_page5[0]])
         list_comp
         exporttocsv=list_comp.to_csv(r'C:\Users\chandana\Documents\Chandana_ListSuperComputers.csv')
```

Out[22]:

	Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
0	1	DOE/SC/Oak Ridge National LaboratoryUnited States	Summit - IBM Power System AC922, IBM POWER9 22...	2414592	148600.0	200794.9	10096.0
1	2	DOE/NNSA/LLNLUnited States	Sierra - IBM Power System S922LC, IBM POWER9 2...	1572480	94640.0	125712.0	7438.0
2	3	National Supercomputing Center in WuxiChina	Sunway TaihuLight - Sunway MPP, Sunway SW26010...	10649600	93014.6	125435.9	15371.0
3	4	National Super Computer Center in GuangzhouChina	Tianhe-2A - TH-IVB-FEP Cluster, Intel Xeon E5-...	4981760	61444.5	100678.7	18482.0
4	5	Texas Advanced Computing Center/Univ. of Texas...	Frontera - Dell C6420, Xeon Platinum 8280 28C ...	448448	23516.4	38745.9	NaN
...	...	...	...	...	...	...	...
95	496	Internet Service (B)China	Pdata-1 - Sugon I620, Xeon E5-2650v4 12C 2.2GH...	58800	1028.0	2069.8	735.0
96	497	Internet CompanyChina	PVideo-A - Sugon I620, Xeon E5-2620v3 6C 2.4GH...	54000	1026.0	2073.6	900.0
97	498	Network CompanyChina	Internet Company N D2 - Lenovo RD350, Xeon E5-...	60000	1022.0	2112.0	NaN
98	499	ROMEO HPC Center - Champagne-ArdenneFrance	Romeo - Bull Sequana X1125, Xeon Gold 6132 14C...	17640	1022.0	1484.0	127.0
99	500	Internet Service AChina	Inspur SA5212H5, Xeon E5-2682v4 16C 2.5GHz, NV...	27360	1021.0	2014.0	NaN

500 rows × 7 columns



	A	B	C	D	E	F	G
1	Rank	Site	System	Cores	Rmax (TFl	Rpeak (TF	Power (kW)
2	1	DOE/SC/C	Summit - I	2414592	148600	200794.9	10096
3	2	DOE/NNS	Sierra - IBI	1572480	94640	125712	7438
4	3	National S	Sunway Ta	10649600	93014.6	125435.9	15371
5	4	National S	Tianhe-2A	4981760	61444.5	100678.7	18482
6	5	Texas Adv	Frontera -	448448	23516.4	38745.9	
7	6	Swiss Nati	Piz Daint -	387872	21230	27154.3	2384
8	7	DOE/NNS	Trinity - Ci	979072	20158.7	41461.2	7578
9	8	National I	AI Bridgin	391680	19880	32576.6	1649
10	9	Leibniz Re	SuperMUC	305856	19476.6	26873.9	
11	10	DOE/NNS	Lassen - IE	288288	18200	23047.2	
12	11	Total Expl	PANGEA II	291024	17860	25025.8	1367
13	12	DOE/SC/C	Titan - Cra	560640	17590	27112.5	8209
14	13	DOE/NNS	Sequoia -	1572864	17173.2	20132.7	7890
15	14	DOE/SC/L	Cori - Cray	622336	14014.7	27880.7	3939
16	15	Korea Inst	Nurion - C	570020	13929.3	25705.9	
17	16	Joint Cent	Oakforest	556104	13554.6	24913.5	2719
18	17	Eni S.p.A.	HPC4 - Prc	253600	12210	18621.1	1320
19	18	Commissa	Tera-1000	561408	11965.5	23396.4	3178
20	19	Texas Adv	Stampede	367024	10680.7	18309.2	
21	20	RIKEN Adv	K compute	705024	10510	11280.4	12660
22	21	CINECA Ita	Marconi In	348000	10384.9	18816	
23	22	NVIDIA Cc	DGX Supe	127488	9444	11209.1	
24	23	National C	Taiwania 2	170352	9000	15208.2	798
25	24	DOE/SC/A	Mira - Blu	786432	8586.6	10066.3	3945

#Replacing the blank values with NA values using **errors='coerce'** and **to_numeric**() functions

```
In [158]: list_comp['Cores']=pd.to_numeric(list_comp['Cores'],errors='coerce')
          list_comp['Rpeak (TFlop/s)']=pd.to_numeric(list_comp['Rpeak (TFlop/s)'],errors='coerce')
          list_comp['Rmax (TFlop/s)']=pd.to_numeric(list_comp['Rmax (TFlop/s)'],errors='coerce')
          list_comp['Power (kW)']=pd.to_numeric(list_comp['Power (kW)'],errors='coerce')
```

```
In [161]: list_comp['Power (kW)']

Out[161]: 0     10096.0
          1      7438.0
          2     15371.0
          3     18482.0
          4         NaN
                 ...
          95      735.0
          96      900.0
          97        NaN
          98      127.0
          99        NaN
          Name: Power (kW), Length: 500, dtype: float64
```

Cleaning up of data is done by using filling the **NA** values using the **mean()** values with **fillna()** function.

```
In [163]: list_comp['Cores']=list_comp['Cores'].fillna(list_comp['Cores'].mean())
          list_comp['Rpeak (TFlop/s)']=list_comp['Rpeak (TFlop/s)'].fillna(list_comp['Rpeak (TFlop/s)'].mean())
          list_comp['Rmax (TFlop/s)']=list_comp['Rmax (TFlop/s)'].fillna(list_comp['Rmax (TFlop/s)'].mean())
          list_comp['Power (kW)']=list_comp['Power (kW)'].fillna(list_comp['Power (kW)'].mean())
```

```
In [164]: list_comp['Power (kW)']

Out[164]: 0     10096.000000
          1      7438.000000
          2     15371.000000
          3     18482.000000
          4      1756.617225
                    ...
          95      735.000000
          96      900.000000
          97     1756.617225
          98      127.000000
          99     1756.617225
          Name: Power (kW), Length: 500, dtype: float64
```

- **Summary of Cores**

```
In [88]: print(list_comp['Cores'].describe())

count    5.000000e+02
mean     1.182127e+05
std      5.472871e+05
min      1.259200e+04
25%      3.600000e+04
50%      5.760000e+04
75%      7.570000e+04
max      1.064960e+07
Name: Cores, dtype: float64
```

- **Summary of RPeak**

```
In [89]: print(list_comp['Rpeak (TFlop/s)'].describe())
```

```
count       500.000000
mean       4927.748800
std       13282.606456
min        1164.700000
25%        2119.700000
50%        2404.800000
75%        3779.200000
max      200794.900000
Name: Rpeak (TFlop/s), dtype: float64
```

- **Summary of RMax**

```
In [92]: print(list_comp['Rmax (TFlop/s)'].describe())
```

```
count       500.000000
mean       3119.151200
std        9556.759821
min        1021.000000
25%        1179.900000
50%        1646.050000
75%        1986.650000
max      148600.000000
Name: Rmax (TFlop/s), dtype: float64
```

- **Summary of Power**

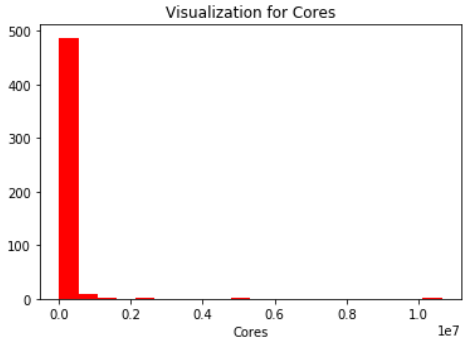
```
In [90]: print(list_comp['Power (kW)'].describe())
```

```
count      500.000000
mean      1756.617225
std       1668.810997
min         81.000000
25%       1187.500000
50%       1756.617225
75%       1756.617225
max      18482.000000
Name: Power (kW), dtype: float64
```

- **Visualization for Cores**

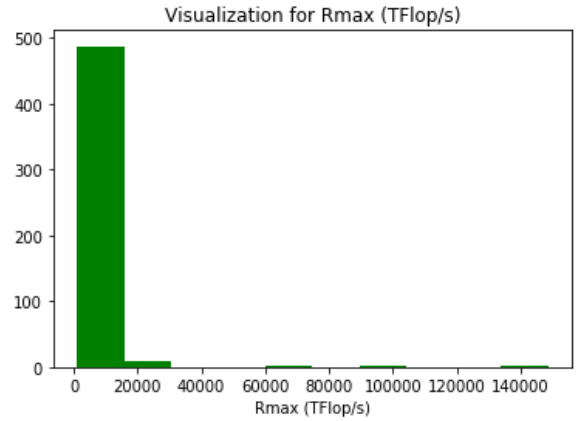
```
In [140]: plt.hist(list_comp['Cores'],color='red',bins=20)
          plt.xlabel('Cores')
          plt.title('Visualization for Cores')
```

```
Out[140]: Text(0.5, 1.0, 'Visualization for Cores')
```



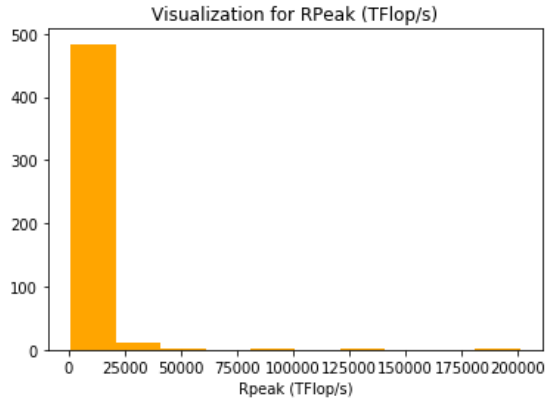
- **Visualization for RMax**

```
In [155]: plt.hist(list_comp['Rmax (TFlop/s)'],color='green')
          plt.xlabel('Rmax (TFlop/s)')
          plt.title('Visualization for Rmax (TFlop/s)')

Out[155]: Text(0.5, 1.0, 'Visualization for Rmax (TFlop/s)')
```



- **Visualization for RPeak**

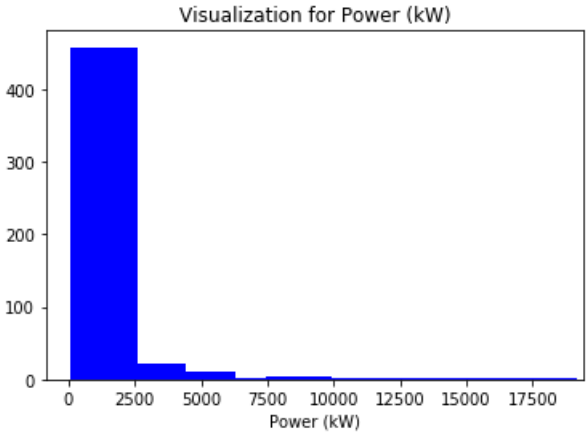
```
In [121]: plt.hist(list_comp['Rpeak (TFlop/s)'],color='orange')
          plt.xlabel('Rpeak (TFlop/s)')
          plt.title('Visualization for RPeak (TFlop/s)')

Out[121]: Text(0.5, 1.0, 'Visualization for RPeak (TFlop/s)')
```



- **Visualization for Power**

```
In [126]: plt.hist(list_comp['Power (kW)'],color='blue',width=2500)
          plt.xlabel('Power (kW)')
          plt.title('Visualization for Power (kW)')
```

```
Out[126]: Text(0.5, 1.0, 'Visualization for Power (kW)')
```



Relationship between Cores and RPeak

```
In [84]: plt.scatter(list_comp['Rpeak (TFlop/s)'],list_comp['Cores'],color='Green')
         plt.xlabel('Rpeak (TFlop/s)')
         plt.ylabel('Cores')
         plt.title('Relation between Cores and Rpeak(TFlop/s)')
```

```
Out[84]: Text(0.5, 1.0, 'Relation between Cores and Rpeak(TFlop/s)')
```