In [85]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [86]:

```
import pandas as pd
df=pd.read_csv('abalone.csv')
df.head()
```

Out[86]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [87]:

```
df.shape
```

Out[87]:

```
(4177, 9)
```

In [88]:

```
df
```

Out[88]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
5	I	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.1200	8
6	F	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.3300	20
7	F	0.545	0.425	0.125	0.7680	0.2940	0.1495	0.2600	16
8	M	0.475	0.370	0.125	0.5095	0.2165	0.1125	0.1650	9
9	F	0.550	0.440	0.150	0.8945	0.3145	0.1510	0.3200	19

In [89]:

```
# This code will display all the rows in output(needed because we have 61 coulumns)
pd.set_option('display.max_rows', None)

```

In [90]:

```
#it shows the statistical summary,
#describpe function is used for continues values not for categorical values hence for c
df.describe()
```

Out[90]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell we
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005

In [91]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4177 non-null   object 
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64  
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

In [92]:

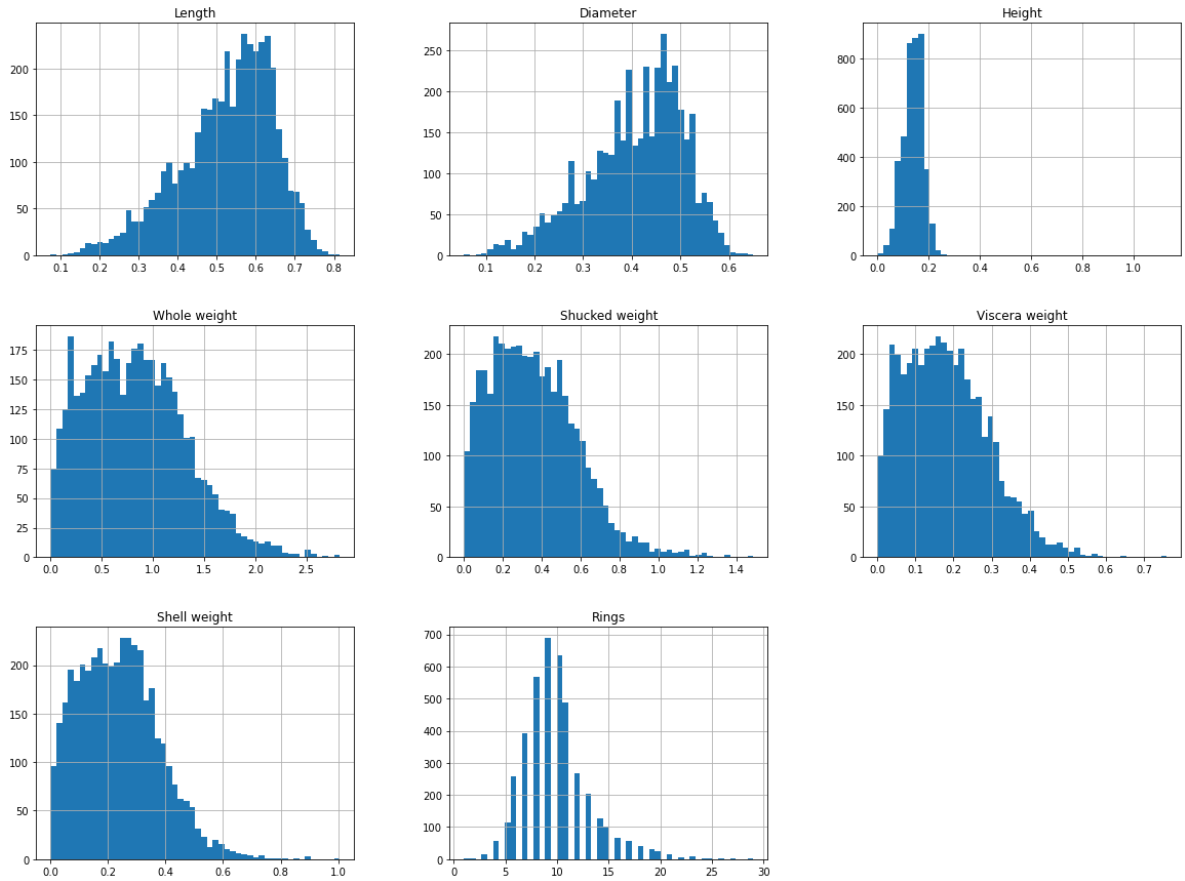
```
1 df.Sex.value_counts()
```

Out[92]:

```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

In [93]:

```
1 df.hist(bins=50, figsize=(20,15))
2 plt.show()
```



In [94]:

```
1 df.duplicated().sum()
```

Out[94]:

```
0
```

In [95]:

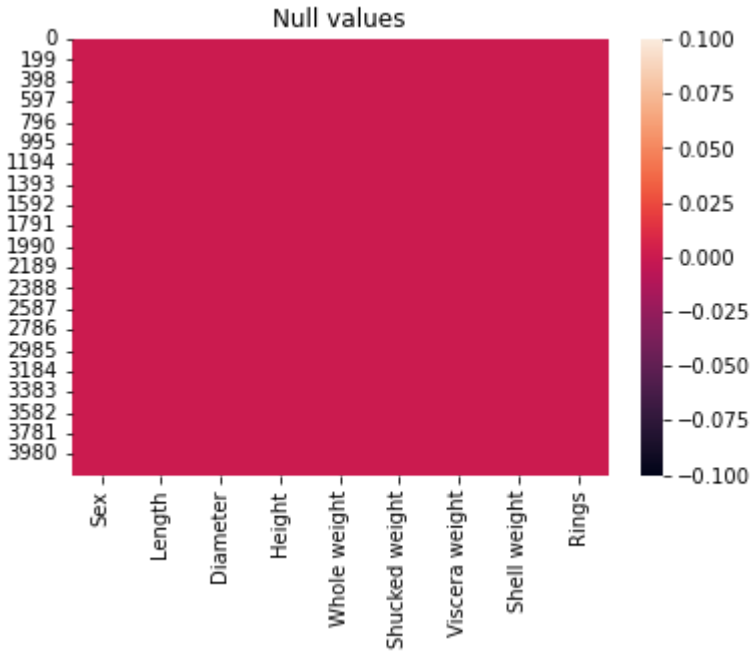
```
#similarly
df.isnull().sum()
```

Out[95]:

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

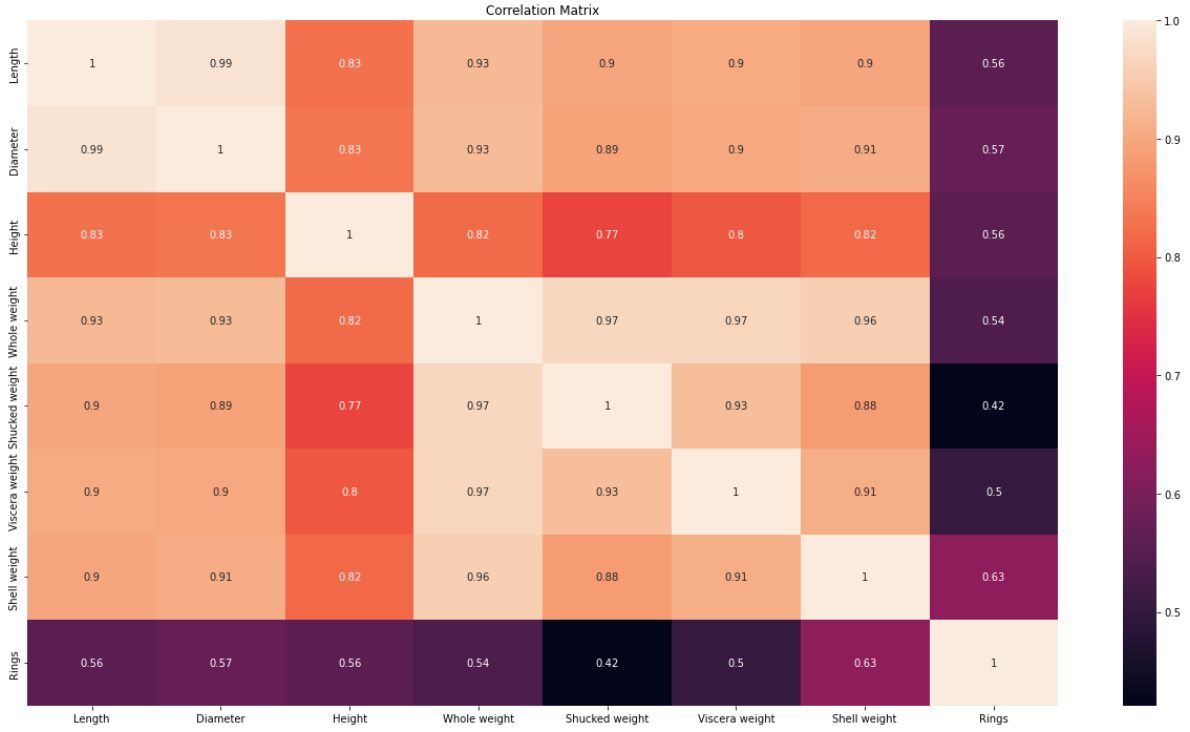
In [96]:

```
sns.heatmap(df.isnull())
plt.title("Null values")
plt.show()
```



In [97]:

```
corr_mat=df.corr()
plt.figure(figsize=[22,12])
sns.heatmap(corr_mat, annot=True)
plt.title("Correlation Matrix")
plt.show()
```



In [98]:

```
corr_matrix=df.corr()
corr_matrix
```

Out[98]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

In [99]:

```
#we want only the correlation of class and +/-1 is higly corrrelated hence will display

corr_matrix=df.corr()
corr_matrix['Rings'].sort_values(ascending=False)
```

Out[99]:

```
Rings             1.000000
Shell weight      0.627574
Diameter          0.574660
Height            0.557467
Length            0.556720
Whole weight      0.540390
Viscera weight    0.503819
Shucked weight    0.420884
Name: Rings, dtype: float64
```

splitting the independent and target variables in x and y before removing the skewness

In [100]:

```
## Splittng the dataset to train,test and split
from sklearn.model_selection import train_test_split

X = df.iloc[:,0:-1]
Y = df.iloc[:,-1]
```

In [101]:

```
#similarl to above

x=df.drop("Rings", axis=1)
y=df["Rings"]
```

In [102]:

```
corr_matrix=df.corr()
corr_matrix['Rings'].sort_values(ascending=False)
```

Out[102]:

```
Rings             1.000000
Shell weight      0.627574
Diameter          0.574660
Height            0.557467
Length            0.556720
Whole weight      0.540390
Viscera weight    0.503819
Shucked weight    0.420884
Name: Rings, dtype: float64
```

In [103]:

```
df
```

Out[103]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
5	I	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.1200	8
6	F	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.3300	20
7	F	0.545	0.425	0.125	0.7680	0.2940	0.1495	0.2600	16
8	M	0.475	0.370	0.125	0.5095	0.2165	0.1125	0.1650	9
9	F	0.550	0.440	0.150	0.8945	0.3145	0.1510	0.3200	19

As Sex is a string data we have to convert it to numeric data

In [104]:

```
#we have only one column string data class

from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
df['Sex']=LE.fit_transform(df['Sex'])
df['Sex'].value_counts()

```

Out[104]:

```
2    1528
1    1342
0    1307
Name: Sex, dtype: int64
```

In [105]:

```
df
```

Out[105]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
5	1	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.1200	8
6	0	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.3300	20
7	0	0.545	0.425	0.125	0.7680	0.2940	0.1495	0.2600	16
8	2	0.475	0.370	0.125	0.5095	0.2165	0.1125	0.1650	9
9	0	0.550	0.440	0.150	0.8945	0.3145	0.1510	0.3200	19

In [106]:

```
y.value_counts()
```

Out[106]:

```
9     689
10    634
8     568
11    487
7     391
12    267
6     259
13    203
14    126
5     115
15    103
16     67
17     58
4      57
18     42
19     32
20     26
3      15
21     14
23      9
22      6
27      2
24      2
1       1
26      1
29      1
2       1
25      1
Name: Rings, dtype: int64
```

As the output is imbalance we have to scale the data

Scaling

In [107]:

```
x.shape
```

Out[107]:

```
(4177, 8)
```

In [108]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
y = scaler.fit_transform(np.array(y).reshape(-1,1))

```

In [109]:

```
1 y
```

Out[109]:

```
array([[ 1.57154357],
       [-0.91001299],
       [-0.28962385],
       ...,
       [-0.28962385],
       [ 0.02057072],
       [ 0.64095986]])
```

In [112]:

```
1 from sklearn.preprocessing import StandardScaler
2 scaler = StandardScaler()
3 x = scaler.fit_transform(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\Users\ACERAS~1\AppData\Local\Temp/ipykernel_1168/2036666876.py in <module
>
      1 from sklearn.preprocessing import StandardScaler
      2 scaler = StandardScaler()
----> 3 x = scaler.fit_transform(x)

~\AppData\Roaming\Python\Python39\site-packages\sklearn\base.py in fit_trans
form(self, X, y, **fit_params)
    850         if y is None:
    851             # fit method of arity 1 (unsupervised transformation)
--> 852             return self.fit(X, **fit_params).transform(X)
    853         else:
    854             # fit method of arity 2 (supervised transformation)

~\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing\_data.
py in fit(self, X, y, sample_weight)
    804         # Reset internal state before fitting
    805         self._reset()
--> 806         return self.partial_fit(X, y, sample_weight)
    807
    808     def partial_fit(self, X, y=None, sample_weight=None):

~\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing\_data.
py in partial_fit(self, X, y, sample_weight)
    839         """
    840         first_call = not hasattr(self, "n_samples_seen_")
--> 841         X = self._validate_data(
    842             X,
    843             accept_sparse=("csr", "csc"),

~\AppData\Roaming\Python\Python39\site-packages\sklearn\base.py in _validate
_data(self, X, y, reset, validate_separately, **check_params)
    564             raise ValueError("Validation should be done on X, y or b
oth.")
    565         elif not no_val_X and no_val_y:
--> 566             X = check_array(X, **check_params)
    567             out = X
    568         elif no_val_X and not no_val_y:

~\AppData\Roaming\Python\Python39\site-packages\sklearn\utils\validation.py
 in check_array(array, accept_sparse, accept_large_sparse, dtype, order, cop
y, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_fea
tures, estimator)
    744                     array = array.astype(dtype, casting="unsafe", co
py=False)
    745                 else:
--> 746                     array = np.asarray(array, order=order, dtype=dty
pe)
    747             except ComplexWarning as complex_warning:
    748                 raise ValueError(

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray
(a, dtype, order, like)
```

```
    100         return _asarray_with_like(a, dtype=dtype, order=order, like=
like)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in __array
__(self, dtype)
   1991
   1992     def __array__(self, dtype: NpDtype | None = None) -> np.ndarray:
-> 1993         return np.asarray(self._values, dtype=dtype)
   1994
   1995     def __array_wrap__(

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray
(a, dtype, order, like)
    100         return _asarray_with_like(a, dtype=dtype, order=order, like=
like)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104

ValueError: could not convert string to float: 'M'
```

In [115]:

```
1 df
```

1220	1	0.330	0.250	0.095	0.2085	0.1020	0.0395	0.0520	7
1221	1	0.330	0.205	0.095	0.1595	0.0770	0.0320	0.0435	5
1222	1	0.335	0.245	0.090	0.2015	0.0960	0.0405	0.0480	7
1223	1	0.340	0.250	0.090	0.1790	0.0775	0.0330	0.0550	6
1224	1	0.345	0.255	0.095	0.1945	0.0925	0.0370	0.0550	6
1225	1	0.345	0.255	0.085	0.2005	0.1050	0.0370	0.0500	5
1226	1	0.350	0.270	0.075	0.2150	0.1000	0.0360	0.0650	6
1227	1	0.350	0.255	0.090	0.1785	0.0855	0.0305	0.0525	8
1228	1	0.360	0.270	0.085	0.1960	0.0875	0.0350	0.0640	4
1229	1	0.365	0.270	0.085	0.1875	0.0810	0.0420	0.0580	6
1230	1	0.365	0.270	0.085	0.1960	0.0825	0.0375	0.0600	7
1231	1	0.365	0.265	0.085	0.2130	0.0945	0.0490	0.0600	7
1232	1	0.370	0.290	0.090	0.2445	0.0890	0.0655	0.0750	7

In [ ]:

```
1
```