Missing Values

In [1]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np
```

In [3]:

```
# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from list
df = pd.DataFrame(dict)

print(df,"\n")
# using isnull() function
df.isnull()
```

```
   First Score  Second Score  Third Score
0        100.0          30.0          NaN
1         90.0          45.0         40.0
2          NaN          56.0         80.0
3         95.0           NaN         98.0
```

Out[3]:

	First Score	Second Score	Third Score
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

In [5]:

```
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\employees.csv"
df = pd.read_csv(path)
print(df)
```

```
    First Name  Gender  Start Date Last Login Time  Salary  Bonus %  \
0      Douglas    Male    8/6/1993        12:42 PM   97308    6.945   
1       Thomas    Male   3/31/1996         6:53 AM   61933    4.170   
2        Maria  Female   4/23/1993        11:17 AM  130590   11.858   
3        Jerry    Male    3/4/2005         1:00 PM  138705    9.340   
4        Larry    Male   1/24/1998         4:47 PM  101004    1.389   
..         ...     ...         ...             ...     ...      ...   
995      Henry     NaN  11/23/2014         6:09 AM  132483   16.655   
996    Phillip    Male   1/31/1984         6:30 AM   42392   19.675   
997    Russell    Male   5/20/2013        12:39 PM   96914    1.421   
998      Larry    Male   4/20/2013         4:45 PM   60500   11.985   
999     Albert    Male   5/15/2012         6:24 PM  129949   10.169   

    Senior Management                  Team  
0                True             Marketing  
1                True                   NaN  
2               False               Finance  
3                True               Finance  
4                True       Client Services  
```

```
..                ...                   ...
995             False          Distribution
996             False               Finance
997             False               Product
998             False  Business Development
999              True                 Sales

[1000 rows x 8 columns]
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   First Name         933 non-null    object 
 1   Gender             855 non-null    object 
 2   Start Date         1000 non-null   object 
 3   Last Login Time    1000 non-null   object 
 4   Salary             1000 non-null   int64  
 5   Bonus %            1000 non-null   float64
 6   Senior Management  933 non-null    object 
 7   Team               957 non-null    object 
dtypes: float64(1), int64(1), object(6)
memory usage: 62.6+ KB
```

In [7]:

```
df.describe()
```

Out[7]:

	Salary	Bonus %
count	1000.000000	1000.000000
mean	90662.181000	10.207555
std	32923.693342	5.528481
min	35013.000000	1.015000
25%	62613.000000	5.401750
50%	90428.000000	9.838500
75%	118740.250000	14.838000
max	149908.000000	19.944000

In [8]:

```
# creating bool series True for NaN values
bool_series = pd.isnull(df["Gender"])

# filtering data
# displaying data only with Gender = NaN
df[bool_series]
```

Out[8]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
20	Lois	NaN	4/22/1995	7:18 PM	64714	4.934	True	Legal
22	Joshua	NaN	3/8/2012	1:58 AM	90816	18.816	True	Client Services
27	Scott	NaN	7/11/1991	6:58 PM	122367	5.218	False	Legal
31	Joyce	NaN	2/20/2005	2:40 PM	88657	12.752	False	Product
41	Christine	NaN	6/28/2015	1:08 AM	66582	11.308	True	Business Development
...	...	...	...	...	...	...	...	...

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
...	...	...	...	...	...	...	...	...
961	Antonio	NaN	6/18/1989	9:37 PM	103050	3.050	False	Legal
972	Victor	NaN	7/28/2006	2:49 PM	76381	11.159	True	Sales
985	Stephen	NaN	7/10/1983	8:10 PM	85668	1.909	False	Legal
989	Justin	NaN	2/10/1991	4:58 PM	38344	3.794	False	Legal
995	Henry	NaN	11/23/2014	6:09 AM	132483	16.655	False	Distribution

145 rows × 8 columns

In [11]:

```
# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe using dictionary
df = pd.DataFrame(dict)

# using notnull() function
df.notnull()
```

Out[11]:

	First Score	Second Score	Third Score
0	True	True	False
1	True	True	True
2	False	True	True
3	True	False	True

In [12]:

```
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\employees.csv"
df = pd.read_csv(path)
# creating bool series True for NaN values
bool_series = pd.notnull(df["Gender"])

# filtering data
# displayind data only with Gender = Not NaN
df[bool_series]
```

Out[12]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
994	George	Male	6/21/2013	5:47 PM	98874	4.479	True	Marketing
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

855 rows × 8 columns

In [13]:

In [13]:

```
# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)
print(df)
# filling missing value using fillna()
df.fillna(0)
```

```
   First Score  Second Score  Third Score
0        100.0          30.0          NaN
1         90.0          45.0         40.0
2          NaN          56.0         80.0
3         95.0           NaN         98.0
```

Out[13]:

	First Score	Second Score	Third Score
0	100.0	30.0	0.0
1	90.0	45.0	40.0
2	0.0	56.0	80.0
3	95.0	0.0	98.0

In [1]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
  'Second Score': [30, 45, 56, np.nan],
  'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)
print(df)
# filling a missing value with
# previous ones
df.fillna(method ='pad')
```

```
   First Score  Second Score  Third Score
0        100.0          30.0          NaN
1         90.0          45.0         40.0
2          NaN          56.0         80.0
3         95.0           NaN         98.0
```

Out[1]:

	First Score	Second Score	Third Score
0	100.0	30.0	NaN
1	90.0	45.0	40.0
2	90.0	56.0	80.0
3	95.0	56.0	98.0

In [2]:

```
# importing pandas as pd
import pandas as pd
```

```
# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
  'Second Score': [30, 45, 56, np.nan],
  'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)
print(df)
# filling null value using fillna() function
df.fillna(method ='bfill')
```

```
   First Score  Second Score  Third Score
0        100.0          30.0          NaN
1         90.0          45.0         40.0
2          NaN          56.0         80.0
3         95.0           NaN         98.0
```

Out[2]:

	First Score	Second Score	Third Score
0	100.0	30.0	40.0
1	90.0	45.0	40.0
2	95.0	56.0	80.0
3	95.0	NaN	98.0

In [3]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\  \\DBDA\\Data Set\\employees.csv"
df = pd.read_csv(path)
# Printing the first 10 to 24 rows of
# the data frame for visualization
df[10:25]
```

Out[3]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
10	Louise	Female	8/12/1980	9:01 AM	63241	15.132	True	NaN
11	Julie	Female	10/26/1997	3:19 PM	102508	12.637	True	Legal
12	Brandon	Male	12/1/1980	1:08 AM	112807	17.492	True	Human Resources
13	Gary	Male	1/27/2008	11:40 PM	109831	5.831	False	Sales
14	Kimberly	Female	1/14/1999	7:13 AM	41426	14.543	True	Finance
15	Lillian	Female	6/5/2016	6:09 AM	59414	1.256	False	Product
16	Jeremy	Male	9/21/2010	5:56 AM	90370	7.369	False	Human Resources
17	Shawn	Male	12/7/1986	7:45 PM	111737	6.414	False	Product
18	Diana	Female	10/23/1981	10:27 AM	132940	19.082	False	Client Services
19	Donna	Female	7/22/2010	3:48 AM	81014	1.894	False	Product
20	Lois	NaN	4/22/1995	7:18 PM	64714	4.934	True	Legal
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing
22	Joshua	NaN	3/8/2012	1:58 AM	90816	18.816	True	Client Services
23	NaN	Male	6/14/2012	4:19 PM	125792	5.042	NaN	NaN
24	John	Male	7/1/1992	10:08 PM	97950	13.873	False	Client Services

In [5]:

```
df["Gender"].fillna("No Gender", inplace = True)
```

```
df
```

Out[5]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	11/23/2014	6:09 AM	132483	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

1000 rows × 8 columns

In [6]:

```
df[10:25]
```

Out[6]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
10	Louise	Female	8/12/1980	9:01 AM	63241	15.132	True	NaN
11	Julie	Female	10/26/1997	3:19 PM	102508	12.637	True	Legal
12	Brandon	Male	12/1/1980	1:08 AM	112807	17.492	True	Human Resources
13	Gary	Male	1/27/2008	11:40 PM	109831	5.831	False	Sales
14	Kimberly	Female	1/14/1999	7:13 AM	41426	14.543	True	Finance
15	Lillian	Female	6/5/2016	6:09 AM	59414	1.256	False	Product
16	Jeremy	Male	9/21/2010	5:56 AM	90370	7.369	False	Human Resources
17	Shawn	Male	12/7/1986	7:45 PM	111737	6.414	False	Product
18	Diana	Female	10/23/1981	10:27 AM	132940	19.082	False	Client Services
19	Donna	Female	7/22/2010	3:48 AM	81014	1.894	False	Product
20	Lois	No Gender	4/22/1995	7:18 PM	64714	4.934	True	Legal
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing
22	Joshua	No Gender	3/8/2012	1:58 AM	90816	18.816	True	Client Services
23	NaN	Male	6/14/2012	4:19 PM	125792	5.042	NaN	NaN
24	John	Male	7/1/1992	10:08 PM	97950	13.873	False	Client Services

In [7]:

```
df.replace(to_replace = np.nan, value = -99)
```

Out[7]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	-99
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	11/23/2014	6:09 AM	132483	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

1000 rows × 8 columns

In [8]:

```
import pandas as pd

# Creating the dataframe
df = pd.DataFrame({"A":[12, 4, 5, None, 1],
    "B":[None, 2, 54, 3, None],
    "C":[20, 16, None, 3, 8],
    "D":[14, 3, None, None, 6]})

# Print the dataframe
df
```

Out[8]:

	A	B	C	D
0	12.0	NaN	20.0	14.0
1	4.0	2.0	16.0	3.0
2	5.0	54.0	NaN	NaN
3	NaN	3.0	3.0	NaN
4	1.0	NaN	8.0	6.0

In [9]:

```
df.interpolate(method ='linear', limit_direction ='forward')
```

Out[9]:

	A	B	C	D
0	12.0	NaN	20.0	14.0
1	4.0	2.0	16.0	3.0
2	5.0	54.0	9.5	4.0
3	3.0	3.0	3.0	5.0
4	1.0	3.0	8.0	6.0

In [10]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np
```

```
# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
  'Second Score': [30, np.nan, 45, 56],
  'Third Score':[52, 40, 80, 98],
  'Fourth Score':[np.nan, np.nan, np.nan, 65]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

df
```

Out[10]:

	First Score	Second Score	Third Score	Fourth Score
0	100.0	30.0	52	NaN
1	90.0	NaN	40	NaN
2	NaN	45.0	80	NaN
3	95.0	56.0	98	65.0

In [11]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
  'Second Score': [30, np.nan, 45, 56],
  'Third Score':[52, 40, 80, 98],
  'Fourth Score':[np.nan, np.nan, np.nan, 65]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

# using dropna() function
df.dropna()
```

Out[11]:

	First Score	Second Score	Third Score	Fourth Score
3	95.0	56.0	98	65.0

In [12]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, np.nan, np.nan, 95],
  'Second Score': [30, np.nan, 45, 56],
  'Third Score':[52, np.nan, 80, 98],
  'Fourth Score':[np.nan, np.nan, np.nan, 65]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

df
```

Out[12]:

	First Score	Second Score	Third Score	Fourth Score

	First Score	Second Score	Third Score	Fourth Score
0	100.0	30.0	52.0	NaN
1	NaN	NaN	NaN	NaN
2	NaN	45.0	80.0	NaN
3	95.0	56.0	98.0	65.0

In [14]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, np.nan, np.nan, 95],
  'Second Score': [30, np.nan, 45, 56],
  'Third Score':[52, np.nan, 80, 98],
  'Fourth Score':[np.nan, np.nan, np.nan, 65]}

df = pd.DataFrame(dict)

# using dropna() function
df.dropna(how = 'all')
```

Out[14]:

	First Score	Second Score	Third Score	Fourth Score
0	100.0	30.0	52.0	NaN
2	NaN	45.0	80.0	NaN
3	95.0	56.0	98.0	65.0

In [15]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, np.nan, np.nan, 95],
  'Second Score': [30, np.nan, 45, 56],
  'Third Score':[52, np.nan, 80, 98],
  'Fourth Score':[60, 67, 68, 65]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

df
```

Out[15]:

	First Score	Second Score	Third Score	Fourth Score
0	100.0	30.0	52.0	60
1	NaN	NaN	NaN	67
2	NaN	45.0	80.0	68
3	95.0	56.0	98.0	65

In [16]:

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
```

```
import numpy as np

# dictionary of lists
dict = {'First Score':[100, np.nan, np.nan, 95],
  'Second Score': [30, np.nan, 45, 56],
  'Third Score':[52, np.nan, 80, 98],
  'Fourth Score':[60, 67, 68, 65]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

# using dropna() function
df.dropna(axis = 1)
```

Out[16]:

	Fourth Score
0	60
1	67
2	68
3	65

In [17]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\employees.csv"
data = pd.read_csv(path)

# making new data frame with dropped NA values
new_data = data.dropna(axis = 0, how ='any')

new_data
```

Out[17]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
5	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal
...	...	...	...	...	...	...	...	...
994	George	Male	6/21/2013	5:47 PM	98874	4.479	True	Marketing
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

764 rows × 8 columns

In [18]:

```
print("Old data frame length:", len(data))
print("New data frame length:", len(new_data))
print("Number of rows with at least 1 NA value: ", (len(data)-len(new_data)))
```

```
Old data frame length: 1000
New data frame length: 764
Number of rows with at least 1 NA value:  236
```

In [19]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\AcademicPerformance.csv"
df = pd.read_csv(path)
print(df)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0   NaN  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
96      97  152.0       1  56.0  Yes
97      98    1.8       1  28.0   No
98      99   35.0       0  44.0  NaN
99     100    4.0       0  24.0   No

[100 rows x 5 columns]
```

In [20]:

```
df.shape
```

Out[20]:

```
(100, 5)
```

In [21]:

```
print(df.isnull().sum())
```

```
Rollno     0
Marks      0
Gender     0
Age       16
PhD       13
dtype: int64
```

In [22]:

```
df.dropna(inplace=True)
print(df.isnull().sum())
```

```
Rollno    0
Marks     0
Gender    0
Age       0
PhD       0
dtype: int64
```

In [24]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\AcademicPerformance.csv"
df = pd.read_csv(path)
print(df)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0   NaN  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
96      97  152.0       1  56.0  Yes
97      98    1.8       1  28.0   No
98      99   35.0       0  44.0  NaN
99     100    4.0       0  24.0   No

[100 rows x 5 columns]
```

In [25]:

```
df["Age"] = df["Age"].replace(np.NaN,df["Age"].mean())
print(df["Age"][:10])
```

```
0    47.000000
1    65.000000
2    56.000000
3    23.000000
4    47.821429
5    27.000000
6    53.000000
7    47.821429
8    44.000000
9    63.000000
Name: Age, dtype: float64
```

In [26]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\  \\DBDA\\Data Set\\AcademicPerformance.csv"
df = pd.read_csv(path)
print(df)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0   NaN  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
96      97  152.0       1  56.0  Yes
97      98    1.8       1  28.0   No
98      99   35.0       0  44.0  NaN
99     100    4.0       0  24.0   No

[100 rows x 5 columns]
```

In [28]:

```
df["Age"] = df["Age"].replace(np.NaN,df["Age"].median())
print(df["Age"][:10])
```

```
0    47.0
1    65.0
2    56.0
3    23.0
4    50.0
5    27.0
6    53.0
7    50.0
8    44.0
9    63.0
Name: Age, dtype: float64
```

In [29]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\  \\DBDA\\Data Set\\AcademicPerformance.csv"
df = pd.read_csv(path)
print(df)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0   NaN  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
```

```
96      97  152.0       1  56.0  Yes
97      98    1.8       1  28.0   No
98      99   35.0       0  44.0  NaN
99     100    4.0       0  24.0   No

[100 rows x 5 columns]
```

In [30]:

```
import statistics
df["Age"] = df["Age"].replace(np.NaN, statistics.mode(df["Age"]))
print(df["Age"][:10])
```

```
0    47.0
1    65.0
2    56.0
3    23.0
4    65.0
5    27.0
6    53.0
7    65.0
8    44.0
9    63.0
Name: Age, dtype: float64
```

In [31]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\AcademicPerformance.csv"
df = pd.read_csv(path)
print(df)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0   NaN  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
96      97  152.0       1  56.0  Yes
97      98    1.8       1  28.0   No
98      99   35.0       0  44.0  NaN
99     100    4.0       0  24.0   No

[100 rows x 5 columns]
```

In [32]:

```
df.isnull().sum()
```

Out[32]:

```
Rollno     0
Marks      0
Gender     0
Age       16
PhD       13
dtype: int64
```

In [33]:

```
df["PhD"] = df["PhD"].fillna('U')
df.isnull().sum()
```

Out[33]:

```
Rollno     0
Marks      0
Gender     0
Age       16
PhD        0
dtype: int64
```

```
dtype: int64
```

In [34]:

```
print(df)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0   NaN  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
96      97  152.0       1  56.0  Yes
97      98    1.8       1  28.0   No
98      99   35.0       0  44.0    U
99     100    4.0       0  24.0   No

[100 rows x 5 columns]
```

In [35]:

```
import pandas as pd
path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\AcademicPerformance.csv"
dataset = pd.read_csv(path)

#LOCF - last observation carried forward

dataset["Age"] = dataset["Age"].fillna(method ='ffill')

dataset.isnull().sum()

print(dataset)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0  23.0  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
96      97  152.0       1  56.0  Yes
97      98    1.8       1  28.0   No
98      99   35.0       0  44.0  NaN
99     100    4.0       0  24.0   No

[100 rows x 5 columns]
```

In [36]:

```
import pandas as pd
import numpy as np

path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\AcademicPerformance.csv"
dataset = pd.read_csv(path)

#interpolation - linear

dataset["Age"] = dataset["Age"].interpolate(method='linear', limit_direction='forward',
axis=0)

print(dataset)

dataset.isnull().sum()
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0  25.0  Yes
```

```
4         5    80.0        0   23.0   Yes
..      ...     ...      ...    ...   ...
95       96    18.6        1   26.0    No
96       97   152.0        1   56.0   Yes
97       98     1.8        1   28.0    No
98       99    35.0        0   44.0   NaN
99      100     4.0        0   24.0    No

[100 rows x 5 columns]
```

Out[36]:

```
Rollno     0
Marks      0
Gender     0
Age        0
PhD       13
dtype: int64
```

In [37]:

```
#for knn imputation - we need to remove normalize the data and categorical data we need to convert
cat_variables = dataset[['PhD']]
cat_dummies = pd.get_dummies(cat_variables, drop_first=True)
cat_dummies.head()
dataset = dataset.drop(['PhD'], axis=1)
dataset = pd.concat([dataset, cat_dummies], axis=1)
dataset.head()

#removing unwanted features
dataset = dataset.drop(['Gender'], axis=1)
dataset.head()

#scaling mandatory before knn
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
dataset = pd.DataFrame(scaler.fit_transform(dataset), columns = dataset.columns)
dataset.head()

#knn imputer
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=3)
dataset = pd.DataFrame(imputer.fit_transform(dataset),columns = dataset.columns)

#checking for missing
dataset.isnull().sum()
```

Out[37]:

```
Rollno     0
Marks      0
Age        0
PhD_Yes    0
dtype: int64
```

In [2]:

```
import pandas as pd
import numpy as np

path="C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\AcademicPerformance.csv"
dataset = pd.read_csv(path)
print(dataset)
```

```
    Rollno  Marks  Gender   Age  PhD
0        1  140.0       1  47.0  Yes
1        2   30.0       0  65.0  Yes
2        3   35.1       0  56.0   No
3        4   30.0       1  23.0   No
4        5   80.0       0   NaN  Yes
..     ...    ...     ...   ...  ...
95      96   18.6       1  26.0   No
96      97  152.0       1  56.0  Yes
```

```
96        97   152.0         1   56.0  Yes
97        98     1.8         1   28.0   No
98        99    35.0         0   44.0  NaN
99       100     4.0         0   24.0   No

[100 rows x 5 columns]
```

In [39]:

```
dataset["PhD"].isnull()
```

Out[39]:

```
0      False
1      False
2      False
3      False
4      False
       ...
95     False
96     False
97     False
98      True
99     False
Name: PhD, Length: 100, dtype: bool
```

In [40]:

```
# Detecting numbers
cnt=0
for row in dataset['PhD']:
    try:
        int(row)
        dataset.loc[cnt, 'PhD']=np.nan
    except ValueError:
        pass
    cnt+=1
```

In [41]:

```
dataset["PhD"].isnull()
print(dataset)
```

```
    Rollno   Marks  Gender    Age  PhD
0        1   140.0       1   47.0  Yes
1        2    30.0       0   65.0  Yes
2        3    35.1       0   56.0   No
3        4    30.0       1   23.0   No
4        5    80.0       0    NaN  Yes
..     ...     ...     ...    ...  ...
95      96    18.6       1   26.0   No
96      97   152.0       1   56.0  Yes
97      98     1.8       1   28.0   No
98      99    35.0       0   44.0  NaN
99     100     4.0       0   24.0   No

[100 rows x 5 columns]
```

In [4]:

```
dataset.skew(axis=0)
```

Out[4]:

```
Rollno     0.000000
Marks      1.077026
Gender     0.000000
Age       -0.236916
dtype: float64
```

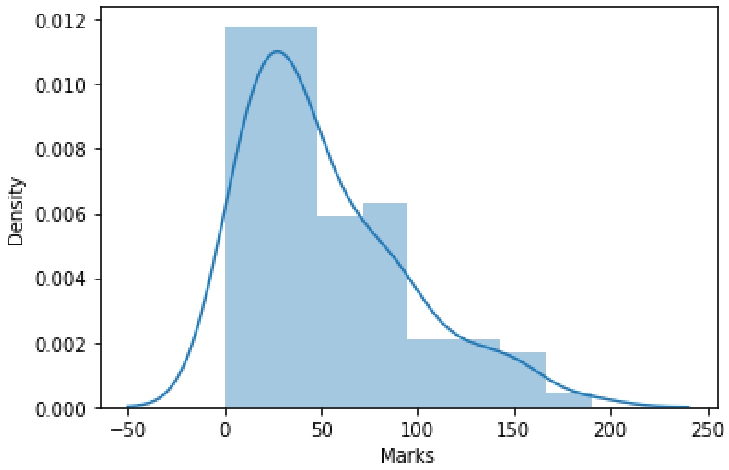
In [5]:

```
import seaborn as sn
import seaborn as sn
sn.distplot(dataset["Marks"])
```

```
D:\Program Files\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning
: `distplot` is a deprecated function and will be removed in a future version. Please ada
pt your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[5]:

```
<AxesSubplot:xlabel='Marks', ylabel='Density'>
```



In [6]:

```
np.log(1.077026)
```

Out[6]:

```
0.07420353901563533
```

In [7]:

```
log_Marks=np.log(dataset["Marks"])
```

In [8]:

```
log_Marks.head()
```

Out[8]:

```
0    4.941642
1    3.401197
2    3.558201
3    3.401197
4    4.382027
Name: Marks, dtype: float64
```

In [9]:

```
log_Marks.skew()
```

Out[9]:

```
-1.3980101345258154
```

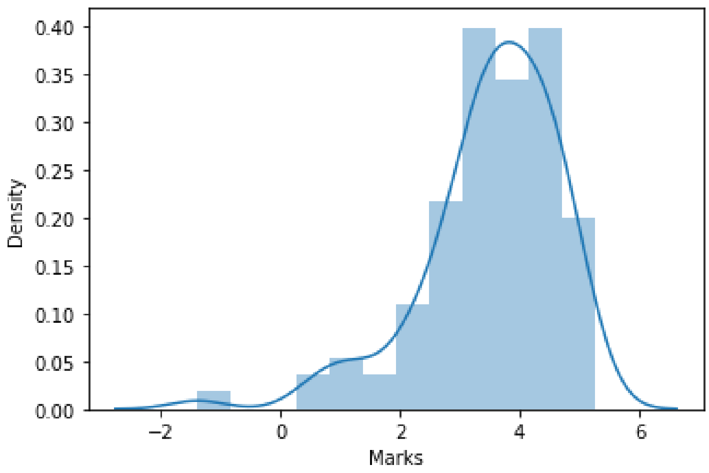
In [10]:

```
import seaborn as sn
sn.distplot(log_Marks)
```

```
D:\Program Files\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning
: `distplot` is a deprecated function and will be removed in a future version. Please ada
pt your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<AxesSubplot:xlabel='Marks', ylabel='Density'>
```



In [11]:

```
log_Marks_sq=np.sqrt(dataset["Marks"])
```

In [12]:

```
log_Marks_sq.head()
```

Out[12]:

```
0    11.832160
1     5.477226
2     5.924525
3     5.477226
4     8.944272
Name: Marks, dtype: float64
```

In [13]:

```
log_Marks_sq.skew()
```

Out[13]:

```
0.21202620353224017
```

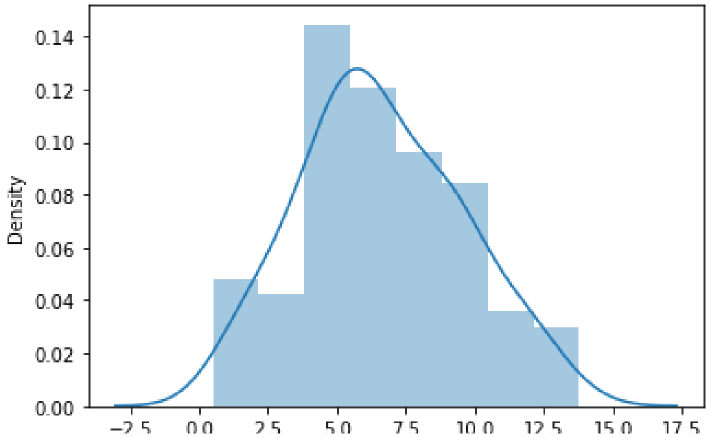
In [14]:

```
import seaborn as sn
sn.distplot(log_Marks_sq)
```

```
D:\Program Files\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning
: `distplot` is a deprecated function and will be removed in a future version. Please ada
pt your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[14]:

```
<AxesSubplot:xlabel='Marks', ylabel='Density'>
```



In [15]:

```
log_Marks_cb=np.cbrt(dataset["Marks"])
```

In [16]:

```
log_Marks_cb.head()
```

Out[16]:

```
0    5.192494
1    3.107233
2    3.274179
3    3.107233
4    4.308869
Name: Marks, dtype: float64
```

In [17]:

```
log_Marks_cb.head()
```

Out[17]:

```
0    5.192494
1    3.107233
2    3.274179
3    3.107233
4    4.308869
Name: Marks, dtype: float64
```

In [18]:

```
log_Marks_cb.skew()
```

Out[18]:

```
-0.18525230594632391
```

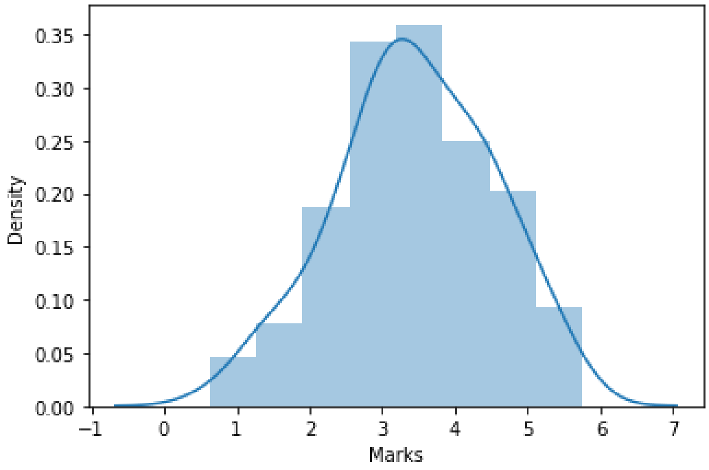
In [19]:

```
import seaborn as sn
sn.distplot(log_Marks_cb)
```

```
D:\Program Files\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning
: `distplot` is a deprecated function and will be removed in a future version. Please ada
pt your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[19]:

```
<AxesSubplot:xlabel='Marks', ylabel='Density'>
```



In [21]:

```
Marks_reci=np.reciprocal(dataset["Marks"])
```

In [22]:

```
Marks_reci.head()
```

Out[22]:

```
0    0.007143
1    0.033333
2    0.028490
3    0.033333
4    0.012500
Name: Marks, dtype: float64
```

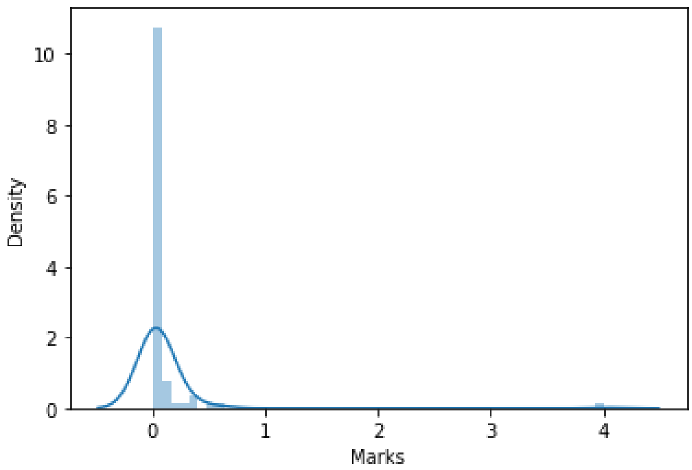
In [23]:

```
import seaborn as sn
sn.distplot(Marks_reci)
```

```
D:\Program Files\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning
: `distplot` is a deprecated function and will be removed in a future version. Please ada
pt your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[23]:

```
<AxesSubplot:xlabel='Marks', ylabel='Density'>
```



In [24]:

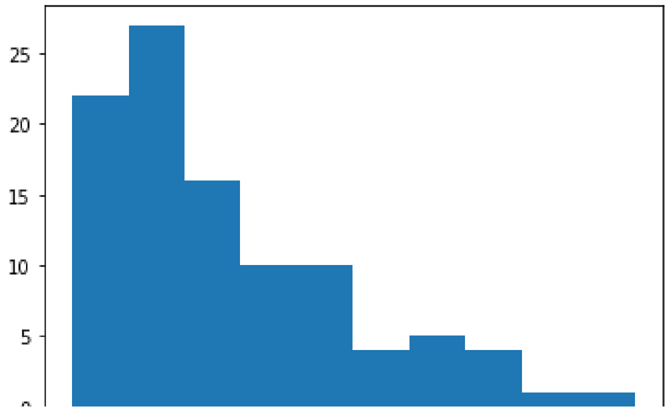
```
Marks_reci.skew()
```

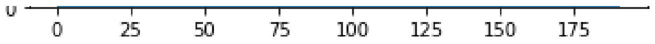
Out[24]:

```
9.14246062263327
```

In [25]:

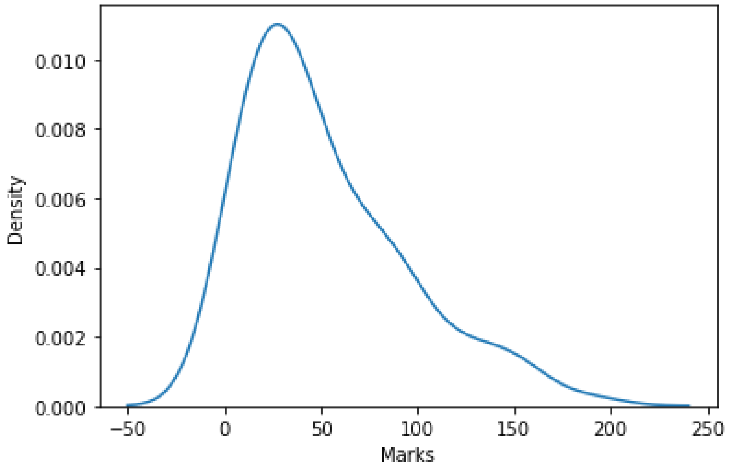
```
import matplotlib.pyplot as plt
his_Marks_cplt=plt.hist(dataset["Marks"])
```





In [26]:

```
plot_marks=sn.kdeplot(dataset["Marks"])
```



In [27]:

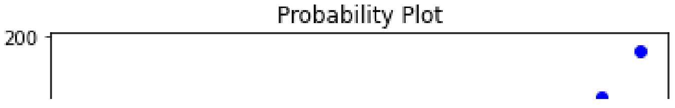
```
import scipy.stats as stats
import pylab
```

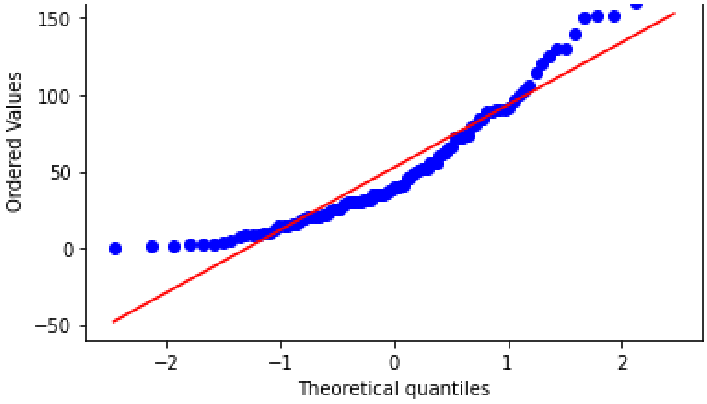
In [28]:

```
stats.probplot(dataset["Marks"],plot=pylab)
```

Out[28]:

```
((array([-2.46203784, -2.12570747, -1.93122778, -1.79044653, -1.67819304,
         -1.58381122, -1.50174123, -1.42869743, -1.36256869, -1.30191411,
         -1.24570419, -1.19317644, -1.14374949, -1.09696931, -1.05247413,
         -1.00997067, -0.96921765, -0.93001393, -0.89218993, -0.85560121,
         -0.82012357, -0.78564937, -0.75208458, -0.71934648, -0.68736185,
         -0.65606548, -0.62539893, -0.59530962, -0.56574992, -0.53667655,
         -0.50804994, -0.47983378, -0.45199463, -0.42450149, -0.39732558,
         -0.37044003, -0.34381966, -0.31744076, -0.29128096, -0.26531902,
         -0.23953472, -0.21390872, -0.18842244, -0.16305799, -0.13779803,
         -0.1126257 , -0.08752455, -0.06247843, -0.03747145, -0.01248789,
          0.01248789,  0.03747145,  0.06247843,  0.08752455,  0.1126257 ,
          0.13779803,  0.16305799,  0.18842244,  0.21390872,  0.23953472,
          0.26531902,  0.29128096,  0.31744076,  0.34381966,  0.37044003,
          0.39732558,  0.42450149,  0.45199463,  0.47983378,  0.50804994,
          0.53667655,  0.56574992,  0.59530962,  0.62539893,  0.65606548,
          0.68736185,  0.71934648,  0.75208458,  0.78564937,  0.82012357,
          0.85560121,  0.89218993,  0.93001393,  0.96921765,  1.00997067,
          1.05247413,  1.09696931,  1.14374949,  1.19317644,  1.24570419,
          1.30191411,  1.36256869,  1.42869743,  1.50174123,  1.58381122,
          1.67819304,  1.79044653,  1.93122778,  2.12570747,  2.46203784]),
  array([  0.25,   1.7 ,   1.8 ,   2.5 ,   3.  ,   3.  ,   4.  ,   4.6 ,
           7.  ,   9.  ,   9.  ,   9.  ,   9.5 ,  10.  ,  12.  ,  14.7 ,
          15.  ,  15.  ,  15.2 ,  16.  ,  18.6 ,  19.  ,  20.  ,  20.  ,
          20.  ,  20.  ,  22.  ,  22.3 ,  24.  ,  25.  ,  25.  ,  25.8 ,
          28.  ,  28.6 ,  30.  ,  30.  ,  30.  ,  30.  ,  30.  ,  31.1 ,
          32.  ,  32.  ,  34.8 ,  35.  ,  35.  ,  35.  ,  35.1 ,  36.  ,
          38.  ,  38.8 ,  39.8 ,  40.  ,  40.7 ,  41.  ,  45.6 ,  46.  ,
          48.  ,  50.  ,  51.  ,  52.  ,  52.  ,  52.  ,  55.  ,  55.  ,
          55.  ,  60.  ,  62.  ,  63.  ,  65.  ,  66.  ,  72.  ,  72.  ,
          72.  ,  73.  ,  74.  ,  80.  ,  81.  ,  84.  ,  84.  ,  89.  ,
          89.  ,  90.  ,  90.  ,  90.  ,  92.  ,  96.  , 100.  , 102.  ,
         106.  , 115.  , 120.  , 125.  , 130.  , 130.  , 140.  , 150.  ,
         152.  , 152.  , 160.  , 190.  ])),
 (40.79054296233955, 52.52449999999999, 0.9515395328716016))
```



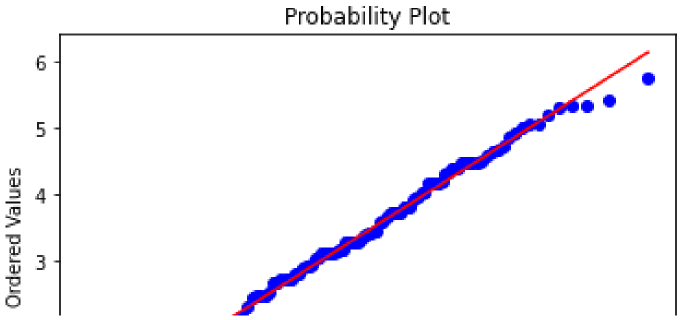


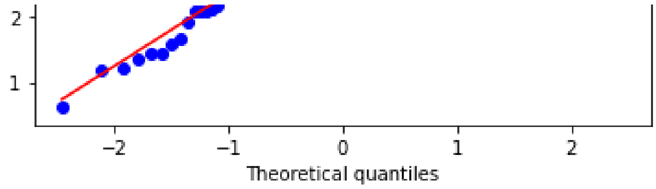
In [29]:

```
stats.probplot(log_Marks_cb,plot=pylab)
```

Out[29]:

```
((array([-2.46203784, -2.12570747, -1.93122778, -1.79044653, -1.67819304,
         -1.58381122, -1.50174123, -1.42869743, -1.36256869, -1.30191411,
         -1.24570419, -1.19317644, -1.14374949, -1.09696931, -1.05247413,
         -1.00997067, -0.96921765, -0.93001393, -0.89218993, -0.85560121,
         -0.82012357, -0.78564937, -0.75208458, -0.71934648, -0.68736185,
         -0.65606548, -0.62539893, -0.59530962, -0.56574992, -0.53667655,
         -0.50804994, -0.47983378, -0.45199463, -0.42450149, -0.39732558,
         -0.37044003, -0.34381966, -0.31744076, -0.29128096, -0.26531902,
         -0.23953472, -0.21390872, -0.18842244, -0.16305799, -0.13779803,
         -0.1126257 , -0.08752455, -0.06247843, -0.03747145, -0.01248789,
          0.01248789,  0.03747145,  0.06247843,  0.08752455,  0.1126257 ,
          0.13779803,  0.16305799,  0.18842244,  0.21390872,  0.23953472,
          0.26531902,  0.29128096,  0.31744076,  0.34381966,  0.37044003,
          0.39732558,  0.42450149,  0.45199463,  0.47983378,  0.50804994,
          0.53667655,  0.56574992,  0.59530962,  0.62539893,  0.65606548,
          0.68736185,  0.71934648,  0.75208458,  0.78564937,  0.82012357,
          0.85560121,  0.89218993,  0.93001393,  0.96921765,  1.00997067,
          1.05247413,  1.09696931,  1.14374949,  1.19317644,  1.24570419,
          1.30191411,  1.36256869,  1.42869743,  1.50174123,  1.58381122,
          1.67819304,  1.79044653,  1.93122778,  2.12570747,  2.46203784]),
  array([0.62996052, 1.19348319, 1.2164404 , 1.35720881, 1.44224957,
         1.44224957, 1.58740105, 1.6631035 , 1.91293118, 2.08008382,
         2.08008382, 2.08008382, 2.11791179, 2.15443469, 2.28942849,
         2.44965982, 2.46621207, 2.46621207, 2.47712466, 2.5198421 ,
         2.64954306, 2.66840165, 2.71441762, 2.71441762, 2.71441762,
         2.71441762, 2.80203933, 2.81471841, 2.88449914, 2.92401774,
         2.92401774, 2.95488036, 3.03658897, 3.05812578, 3.10723251,
         3.10723251, 3.10723251, 3.10723251, 3.10723251, 3.14475486,
         3.1748021 , 3.1748021 , 3.2648238 , 3.27106631, 3.27106631,
         3.27106631, 3.27417865, 3.30192725, 3.36197541, 3.38540456,
         3.41424245, 3.41995189, 3.43978636, 3.44821724, 3.57263198,
         3.58304787, 3.63424119, 3.6840315 , 3.70842977, 3.73251116,
         3.73251116, 3.73251116, 3.80295246, 3.80295246, 3.80295246,
         3.91486764, 3.95789161, 3.97905721, 4.02072576, 4.04124002,
         4.16016765, 4.16016765, 4.16016765, 4.1793392 , 4.19833645,
         4.30886938, 4.32674871, 4.37951914, 4.37951914, 4.4647451 ,
         4.4647451 , 4.48140475, 4.48140475, 4.48140475, 4.51435744,
         4.57885697, 4.64158883, 4.67232873, 4.73262349, 4.86294413,
         4.93242415, 5.        , 5.06579702, 5.06579702, 5.1924941 ,
         5.31329285, 5.3368033 , 5.3368033 , 5.42883523, 5.74889708])),
 (1.0964930316814503, 3.441077741563151, 0.9963217176950497))
```



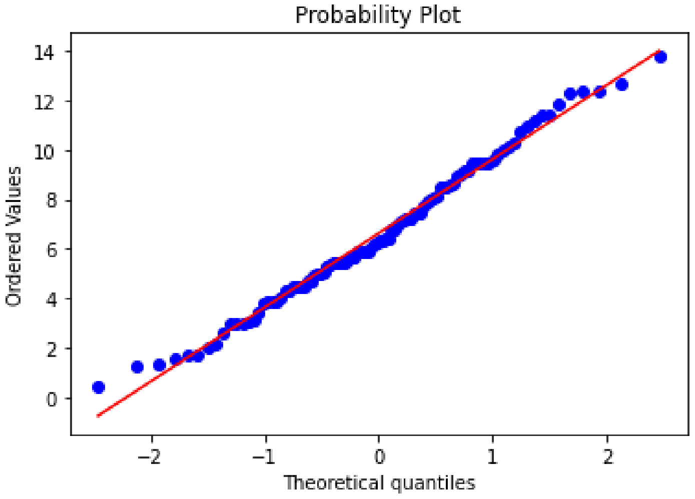


In [30]:

```
stats.probplot(log_Marks_sq,plot=pylab)
```

Out[30]:

```
((array([-2.46203784, -2.12570747, -1.93122778, -1.79044653, -1.67819304,
         -1.58381122, -1.50174123, -1.42869743, -1.36256869, -1.30191411,
         -1.24570419, -1.19317644, -1.14374949, -1.09696931, -1.05247413,
         -1.00997067, -0.96921765, -0.93001393, -0.89218993, -0.85560121,
         -0.82012357, -0.78564937, -0.75208458, -0.71934648, -0.68736185,
         -0.65606548, -0.62539893, -0.59530962, -0.56574992, -0.53667655,
         -0.50804994, -0.47983378, -0.45199463, -0.42450149, -0.39732558,
         -0.37044003, -0.34381966, -0.31744076, -0.29128096, -0.26531902,
         -0.23953472, -0.21390872, -0.18842244, -0.16305799, -0.13779803,
         -0.1126257 , -0.08752455, -0.06247843, -0.03747145, -0.01248789,
          0.01248789,  0.03747145,  0.06247843,  0.08752455,  0.1126257 ,
          0.13779803,  0.16305799,  0.18842244,  0.21390872,  0.23953472,
          0.26531902,  0.29128096,  0.31744076,  0.34381966,  0.37044003,
          0.39732558,  0.42450149,  0.45199463,  0.47983378,  0.50804994,
          0.53667655,  0.56574992,  0.59530962,  0.62539893,  0.65606548,
          0.68736185,  0.71934648,  0.75208458,  0.78564937,  0.82012357,
          0.85560121,  0.89218993,  0.93001393,  0.96921765,  1.00997067,
          1.05247413,  1.09696931,  1.14374949,  1.19317644,  1.24570419,
          1.30191411,  1.36256869,  1.42869743,  1.50174123,  1.58381122,
          1.67819304,  1.79044653,  1.93122778,  2.12570747,  2.46203784]),
  array([ 0.5       ,  1.30384048,  1.34164079,  1.58113883,  1.73205081,
          1.73205081,  2.        ,  2.14476106,  2.64575131,  3.        ,
          3.        ,  3.        ,  3.082207  ,  3.16227766,  3.46410162,
          3.8340579 ,  3.87298335,  3.87298335,  3.89871774,  4.        ,
          4.31277173,  4.35889894,  4.47213595,  4.47213595,  4.47213595,
          4.47213595,  4.69041576,  4.72228758,  4.89897949,  5.        ,
          5.        ,  5.07937004,  5.29150262,  5.34789678,  5.47722558,
          5.47722558,  5.47722558,  5.47722558,  5.47722558,  5.5767374 ,
          5.65685425,  5.65685425,  5.89915248,  5.91607978,  5.91607978,
          5.91607978,  5.9245253 ,  6.        ,  6.164414  ,  6.2289646 ,
          6.30872412,  6.32455532,  6.37965516,  6.40312424,  6.75277721,
          6.78232998,  6.92820323,  7.07106781,  7.14142843,  7.21110255,
          7.21110255,  7.21110255,  7.41619849,  7.41619849,  7.41619849,
          7.74596669,  7.87400787,  7.93725393,  8.06225775,  8.1240384 ,
          8.48528137,  8.48528137,  8.48528137,  8.54400375,  8.60232527,
          8.94427191,  9.        ,  9.16515139,  9.16515139,  9.43398113,
          9.43398113,  9.48683298,  9.48683298,  9.48683298,  9.59166305,
          9.79795897, 10.        , 10.09950494, 10.29563014, 10.72380529,
         10.95445115, 11.18033989, 11.40175425, 11.40175425, 11.83215957,
         12.24744871, 12.32882801, 12.32882801, 12.64911064, 13.78404875])),
 (2.983044720739973, 6.6254088687442305, 0.9951899042212309))
```

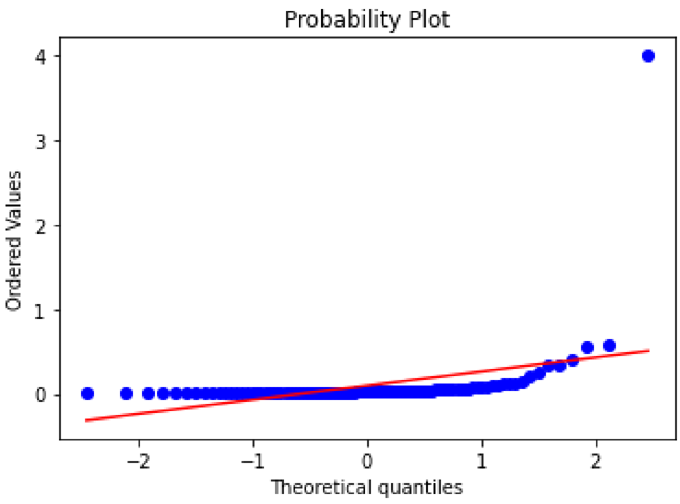


In [31]:

```
stats.probplot(Marks_reci,plot=pylab)
```

Out[31]:

```
((array([-2.46203784, -2.12570747, -1.93122778, -1.79044653, -1.67819304,
         -1.58381122, -1.50174123, -1.42869743, -1.36256869, -1.30191411,
         -1.24570419, -1.19317644, -1.14374949, -1.09696931, -1.05247413,
         -1.00997067, -0.96921765, -0.93001393, -0.89218993, -0.85560121,
         -0.82012357, -0.78564937, -0.75208458, -0.71934648, -0.68736185,
         -0.65606548, -0.62539893, -0.59530962, -0.56574992, -0.53667655,
         -0.50804994, -0.47983378, -0.45199463, -0.42450149, -0.39732558,
         -0.37044003, -0.34381966, -0.31744076, -0.29128096, -0.26531902,
         -0.23953472, -0.21390872, -0.18842244, -0.16305799, -0.13779803,
         -0.1126257 , -0.08752455, -0.06247843, -0.03747145, -0.01248789,
          0.01248789,  0.03747145,  0.06247843,  0.08752455,  0.1126257 ,
          0.13779803,  0.16305799,  0.18842244,  0.21390872,  0.23953472,
          0.26531902,  0.29128096,  0.31744076,  0.34381966,  0.37044003,
          0.39732558,  0.42450149,  0.45199463,  0.47983378,  0.50804994,
          0.53667655,  0.56574992,  0.59530962,  0.62539893,  0.65606548,
          0.68736185,  0.71934648,  0.75208458,  0.78564937,  0.82012357,
          0.85560121,  0.89218993,  0.93001393,  0.96921765,  1.00997067,
          1.05247413,  1.09696931,  1.14374949,  1.19317644,  1.24570419,
          1.30191411,  1.36256869,  1.42869743,  1.50174123,  1.58381122,
          1.67819304,  1.79044653,  1.93122778,  2.12570747,  2.46203784]),
  array([0.00526316, 0.00625   , 0.00657895, 0.00657895, 0.00666667,
         0.00714286, 0.00769231, 0.00769231, 0.008     , 0.00833333,
         0.00869565, 0.00943396, 0.00980392, 0.01      , 0.01041667,
         0.01086957, 0.01111111, 0.01111111, 0.01111111, 0.01123596,
         0.01123596, 0.01190476, 0.01190476, 0.01234568, 0.0125    ,
         0.01351351, 0.01369863, 0.01388889, 0.01388889, 0.01388889,
         0.01515152, 0.01538462, 0.01587302, 0.01612903, 0.01666667,
         0.01818182, 0.01818182, 0.01818182, 0.01923077, 0.01923077,
         0.01923077, 0.01960784, 0.02      , 0.02083333, 0.02173913,
         0.02192982, 0.02439024, 0.02457002, 0.025     , 0.02512563,
         0.0257732 , 0.02631579, 0.02777778, 0.02849003, 0.02857143,
         0.02857143, 0.02857143, 0.02873563, 0.03125   , 0.03125   ,
         0.03215434, 0.03333333, 0.03333333, 0.03333333, 0.03333333,
         0.03333333, 0.03496503, 0.03571429, 0.03875969, 0.04      ,
         0.04      , 0.04166667, 0.04484305, 0.04545455, 0.05      ,
         0.05      , 0.05      , 0.05      , 0.05263158, 0.05376344,
         0.0625    , 0.06578947, 0.06666667, 0.06666667, 0.06802721,
         0.08333333, 0.1       , 0.10526316, 0.11111111, 0.11111111,
         0.11111111, 0.14285714, 0.2173913 , 0.25      , 0.33333333,
         0.33333333, 0.4       , 0.55555556, 0.58823529, 4.        ])),
 (0.16654238388625658, 0.0958160800017085, 0.4031270817229625))
```

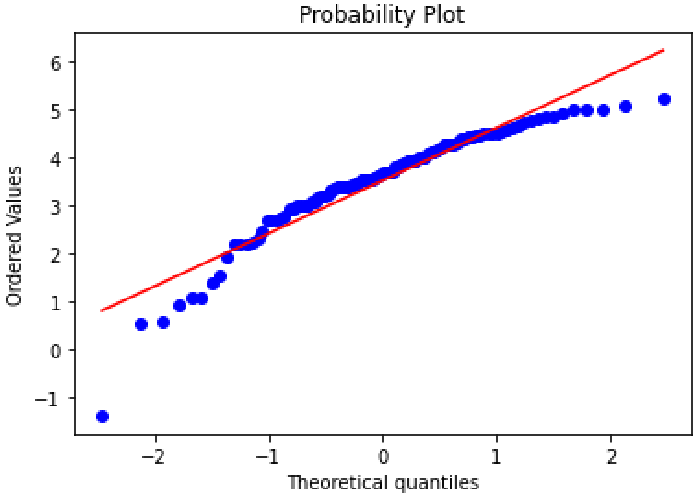


In [32]:

```
stats.probplot(log_Marks,plot=pylab)
```

Out[32]:

```
((array([-2.46203784, -2.12570747, -1.93122778, -1.79044653, -1.67819304,
        -1.58381122, -1.50174123, -1.42869743, -1.36256869, -1.30191411,
        -1.24570419, -1.19317644, -1.14374949, -1.09696931, -1.05247413,
        -1.00997067, -0.96921765, -0.93001393, -0.89218993, -0.85560121,
        -0.82012357, -0.78564937, -0.75208458, -0.71934648, -0.68736185,
        -0.65606548, -0.62539893, -0.59530962, -0.56574992, -0.53667655,
        -0.50804994, -0.47983378, -0.45199463, -0.42450149, -0.39732558,
        -0.37044003, -0.34381966, -0.31744076, -0.29128096, -0.26531902,
        -0.23953472, -0.21390872, -0.18842244, -0.16305799, -0.13779803,
        -0.1126257 , -0.08752455, -0.06247843, -0.03747145, -0.01248789,
         0.01248789,  0.03747145,  0.06247843,  0.08752455,  0.1126257 ,
         0.13779803,  0.16305799,  0.18842244,  0.21390872,  0.23953472,
         0.26531902,  0.29128096,  0.31744076,  0.34381966,  0.37044003,
         0.39732558,  0.42450149,  0.45199463,  0.47983378,  0.50804994,
         0.53667655,  0.56574992,  0.59530962,  0.62539893,  0.65606548,
         0.68736185,  0.71934648,  0.75208458,  0.78564937,  0.82012357,
         0.85560121,  0.89218993,  0.93001393,  0.96921765,  1.00997067,
         1.05247413,  1.09696931,  1.14374949,  1.19317644,  1.24570419,
         1.30191411,  1.36256869,  1.42869743,  1.50174123,  1.58381122,
         1.67819304,  1.79044653,  1.93122778,  2.12570747,  2.46203784]),
 array([-1.38629436,  0.53062825,  0.58778666,  0.91629073,  1.09861229,
         1.09861229,  1.38629436,  1.5260563 ,  1.94591015,  2.19722458,
         2.19722458,  2.19722458,  2.2512918 ,  2.30258509,  2.48490665,
         2.68784749,  2.7080502 ,  2.7080502 ,  2.72129543,  2.77258872,
         2.92316158,  2.94443898,  2.99573227,  2.99573227,  2.99573227,
         2.99573227,  3.09104245,  3.10458668,  3.17805383,  3.21887582,
         3.21887582,  3.25037449,  3.33220451,  3.35340672,  3.40119738,
         3.40119738,  3.40119738,  3.40119738,  3.40119738,  3.43720782,
         3.4657359 ,  3.4657359 ,  3.54961739,  3.55534806,  3.55534806,
         3.55534806,  3.55820113,  3.58351894,  3.63758616,  3.65842025,
         3.68386691,  3.68887945,  3.70622809,  3.71357207,  3.81990772,
         3.8286414 ,  3.87120101,  3.91202301,  3.93182563,  3.95124372,
         3.95124372,  3.95124372,  4.00733319,  4.00733319,  4.00733319,
         4.09434456,  4.12713439,  4.14313473,  4.17438727,  4.18965474,
         4.27666612,  4.27666612,  4.27666612,  4.29045944,  4.30406509,
         4.38202663,  4.39444915,  4.4308168 ,  4.4308168 ,  4.48863637,
         4.48863637,  4.49980967,  4.49980967,  4.49980967,  4.52178858,
         4.56434819,  4.60517019,  4.62497281,  4.66343909,  4.74493213,
         4.78749174,  4.82831374,  4.86753445,  4.86753445,  4.94164242,
         5.01063529,  5.02388052,  5.02388052,  5.07517382,  5.24702407])),
(1.1033972747040552, 3.522558182933606, 0.9494491339528739))
```



]