Aim: Find the outlier using trimming and capping method

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df= pd.read_csv("placement.csv")
```

In [3]:
```
df.head()
```

Out[3]:

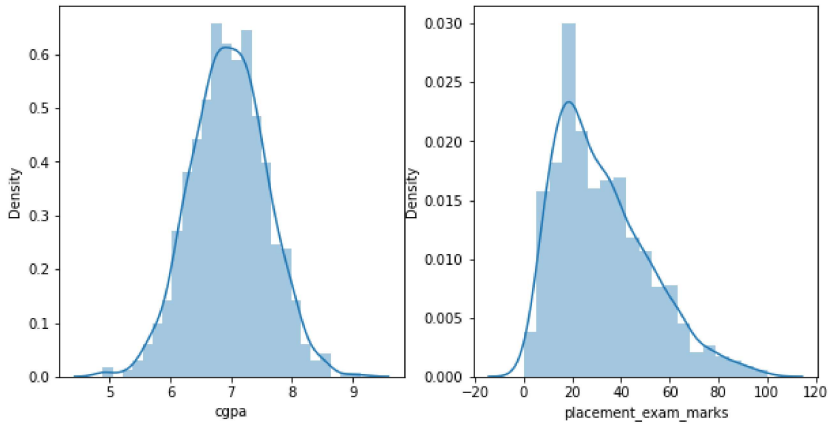
	cgpa	placement_exam_marks	placed
0	7.19	26	1
1	7.46	38	1
2	7.54	40	1
3	6.42	8	1
4	7.23	17	0

In [4]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [5]:
```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.distplot(df['cgpa'])

plt.subplot(1,2,2)
sns.distplot(df['placement_exam_marks'])
```

Out[5]: <AxesSubplot:xlabel='placement_exam_marks', ylabel='Density'>

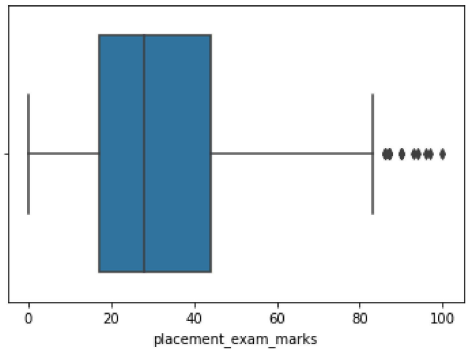


In [6]:
```
df['placement_exam_marks'].describe()
```

Out[6]:
```
count    1000.000000
mean       32.225000
std        19.130822
min         0.000000
25%        17.000000
50%        28.000000
75%        44.000000
max       100.000000
Name: placement_exam_marks, dtype: float64
```

In [7]:
```
sns.boxplot(df['placement_exam_marks'])
```

Out[7]:
```
<AxesSubplot:xlabel='placement_exam_marks'>
```



In [8]:
```
# finding boundaries value
print("Highest Boundary value of cgpa",df['cgpa'].mean()+3*df['cgpa'].std())
```

```
Highest Boundary value of cgpa 8.808933625397177
```

In [9]:
```
print("Lowest Boundary value of cgpa",df['cgpa'].mean()-3*df['cgpa'].std())
```

```
Lowest Boundary value of cgpa 5.113546374602842
```

In [10]:
```
# finding outliers
df[(df['cgpa']>8.80) | (df['cgpa']<5.11)]
```

Out[10]:

	cgpa	placement_exam_marks	placed
485	4.92	44	1
995	8.87	44	1
996	9.12	65	1
997	4.89	34	0
999	4.90	10	1

Trimming

In [11]:
```
df.shape
```

Out[11]:
```
(1000, 3)
```

In [12]:
```
new_df=df[(df['cgpa']<8.80)&(df['cgpa']>5.11)]
new_df
```

Out[12]:

	cgpa	placement_exam_marks	placed
0	7.19	26	1
1	7.46	38	1
2	7.54	40	1
3	6.42	8	1
4	7.23	17	0
...	...	...	...
991	7.04	57	0
992	6.26	12	0
993	6.73	21	1
994	6.48	63	0
998	8.62	46	1

995 rows × 3 columns

In [13]:
```
new_df.shape
```

Out[13]:
```
(995, 3)
```

Z Score

zi = xi - x_mean/S.D

In [14]:
```
df['cgpa_score']=(df['cgpa']-df['cgpa'].mean())/df['cgpa'].std()
```

In [15]:
```
df
```

Out[15]:

	cgpa	placement_exam_marks	placed	cgpa_score
0	7.19	26	1	0.371425
1	7.46	38	1	0.809810
2	7.54	40	1	0.939701
3	6.42	8	1	-0.878782
4	7.23	17	0	0.436371
...	...	...	...	...
995	8.87	44	1	3.099150
996	9.12	65	1	3.505062
997	4.89	34	0	-3.362960
998	8.62	46	1	2.693239
999	4.90	10	1	-3.346724

1000 rows × 4 columns

In [16]:
```
df['cgpa_score'].describe()
```

Out[16]:
```
count    1.000000e+03
mean    -1.600275e-14
std      1.000000e+00
min     -3.362960e+00
25%     -6.677081e-01
50%     -2.013321e-03
75%      6.636815e-01
max      3.505062e+00
Name: cgpa_score, dtype: float64
```

In [17]:
```
df[df['cgpa_score']>3]
```

Out[17]:

	cgpa	placement_exam_marks	placed	cgpa_score
995	8.87	44	1	3.099150
996	9.12	65	1	3.505062

In [18]:
```
df[df['cgpa_score']<-3]
```

Out[18]:

	cgpa	placement_exam_marks	placed	cgpa_score
485	4.92	44	1	-3.314251
997	4.89	34	0	-3.362960
999	4.90	10	1	-3.346724

In [19]:
```
new_dff= df[(df['cgpa_score']>3)|(df['cgpa_score']>-3)]
new_dff
```

Out[19]:

	cgpa	placement_exam_marks	placed	cgpa_score
0	7.19	26	1	0.371425
1	7.46	38	1	0.809810
2	7.54	40	1	0.939701
3	6.42	8	1	-0.878782
4	7.23	17	0	0.436371
...	...	...	...	...
993	6.73	21	1	-0.375452
994	6.48	63	0	-0.781363
995	8.87	44	1	3.099150
996	9.12	65	1	3.505062
998	8.62	46	1	2.693239

997 rows × 4 columns

In [20]:
```
new_dff.shape
```

Out[20]:
```
(997, 4)
```

Capping

In [21]:
```
upper_limit = df['cgpa'].mean()+3*df['cgpa'].std()
lower_limit = df['cgpa'].mean()-3*df['cgpa'].std()
lower_limit
```

Out[21]:
```
5.113546374602842
```

In [22]:
```
upper_limit
```

Out[22]:
```
8.808933625397177
```

In [23]:
```
df['cgpa_cap']=np.where(
    df['cgpa']>upper_limit,
    upper_limit,
    np.where(
    df['cgpa']<lower_limit,
    lower_limit,df['cgpa']
    )
)
```

In [24]:
```
df
```

Out[24]:

	cgpa	placement_exam_marks	placed	cgpa_score	cgpa_cap
0	7.19	26	1	0.371425	7.190000
1	7.46	38	1	0.809810	7.460000
2	7.54	40	1	0.939701	7.540000
3	6.42	8	1	-0.878782	6.420000
4	7.23	17	0	0.436371	7.230000
...	...	...	...	...	...
995	8.87	44	1	3.099150	8.808934
996	9.12	65	1	3.505062	8.808934
997	4.89	34	0	-3.362960	5.113546
998	8.62	46	1	2.693239	8.620000
999	4.90	10	1	-3.346724	5.113546

1000 rows × 5 columns

In [25]:
```
df.describe()
```

Out[25]:

	cgpa	placement_exam_marks	placed	cgpa_score	cgpa_cap
count	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000
mean	6.961240	32.225000	0.489000	-1.600275e-14	6.961499
std	0.615898	19.130822	0.500129	1.000000e+00	0.612688
min	4.890000	0.000000	0.000000	-3.362960e+00	5.113546
25%	6.550000	17.000000	0.000000	-6.677081e-01	6.550000
50%	6.960000	28.000000	0.000000	-2.013321e-03	6.960000
75%	7.370000	44.000000	1.000000	6.636815e-01	7.370000
max	9.120000	100.000000	1.000000	3.505062e+00	8.808934

In [26]:
```
df['placement_exam_marks'].skew()
```

Out[26]:
```
0.8356419499466834
```

In [27]:
```
percent25=df['placement_exam_marks'].quantile(0.25)
percent75=df['placement_exam_marks'].quantile(0.75)
```

In [28]:
```
percent25
```

Out[28]:
```
17.0
```

In [29]:
```
percent75
```

Out[29]:
```
44.0
```

In [30]:
```
iqr=percent75-percent25
iqr
```

Out[30]: 27.0

In [31]:
```
upper_limit1=percent75+1.5*iqr
upper_limit1
```

Out[31]: 84.5

In [32]:
```
lower_limit1=percent25-1.5*iqr
lower_limit1
```

Out[32]: -23.5

In [33]:
```
df[df['placement_exam_marks']>upper_limit1]
```

	cgpa	placement_exam_marks	placed	cgpa_score	cgpa_cap
61	7.51	86	0	0.890992	7.51
134	6.33	93	0	-1.024910	6.33
162	7.80	90	0	1.361849	7.80
283	7.09	87	0	0.209061	7.09
290	8.38	87	0	2.303564	8.38
311	6.97	87	1	0.014223	6.97
324	6.64	90	0	-0.521580	6.64
630	6.56	96	1	-0.651472	6.56
685	6.05	87	1	-1.479531	6.05
730	6.14	90	1	-1.333403	6.14
771	7.31	86	1	0.566263	7.31
846	6.99	97	0	0.046696	6.99
917	5.95	100	0	-1.641896	5.95

In [34]:
```
df[df['placement_exam_marks']<lower_limit1]
```

Out[34]:

cgpa	placement_exam_marks	placed	cgpa_score	cgpa_cap

In [35]:
```
new_dfff=df[df['placement_exam_marks']<upper_limit1]
new_dfff
```

Out[35]:

	cgpa	placement_exam_marks	placed	cgpa_score	cgpa_cap
0	7.19	26	1	0.371425	7.190000
1	7.46	38	1	0.809810	7.460000
2	7.54	40	1	0.939701	7.540000
3	6.42	8	1	-0.878782	6.420000
4	7.23	17	0	0.436371	7.230000
...	...	...	...	...	...
995	8.87	44	1	3.099150	8.808934
996	9.12	65	1	3.505062	8.808934
997	4.89	34	0	-3.362960	5.113546
998	8.62	46	1	2.693239	8.620000
999	4.90	10	1	-3.346724	5.113546

985 rows × 5 columns

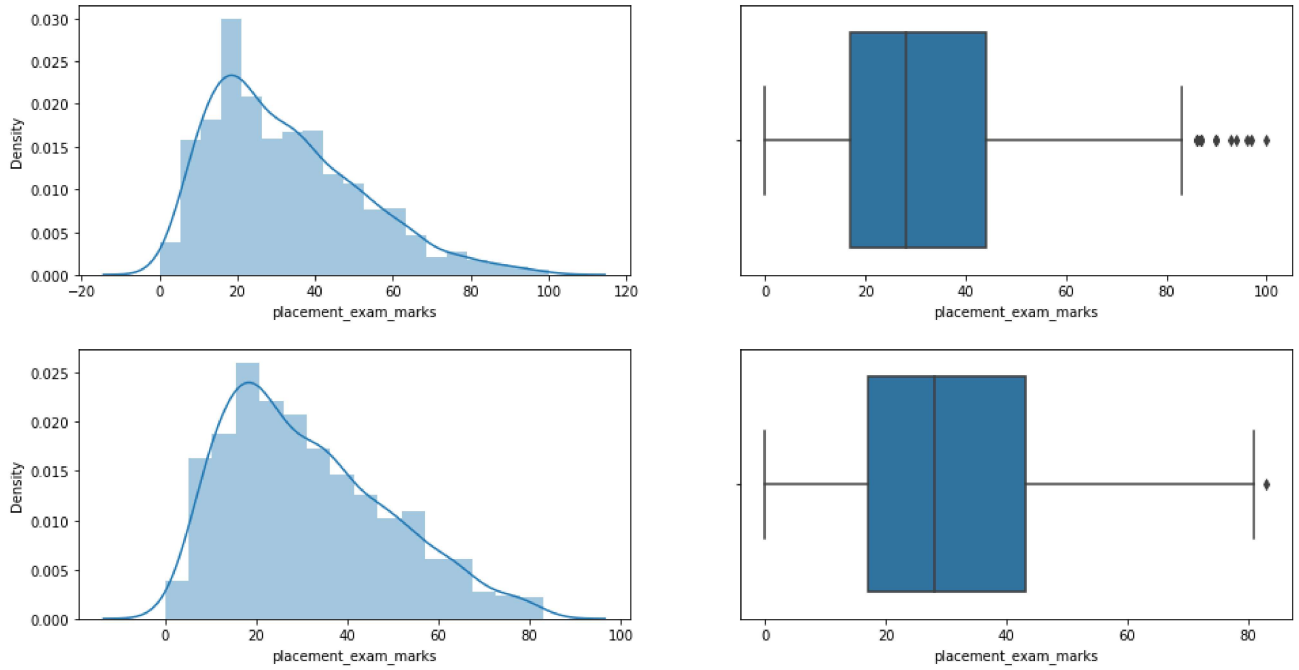
In [36]:
```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df['placement_exam_marks'])

plt.subplot(2,2,2)
sns.boxplot(df['placement_exam_marks'])

plt.figure(figsize=(16,8))
plt.subplot(2,2,3)
sns.distplot(new_dfff['placement_exam_marks'])

plt.subplot(2,2,4)
sns.boxplot(new_dfff['placement_exam_marks'])
```

Out[36]: <AxesSubplot:xlabel='placement_exam_marks'>



In [37]:
```
new_df_cap=df.copy()
new_df_cap['placement_exam_marks']=np.where(
    new_df_cap['placement_exam_marks']>upper_limit,
    upper_limit,
    np.where(
    new_df_cap['placement_exam_marks']<lower_limit,
    lower_limit,new_df_cap['placement_exam_marks']
    )
)
```

In [38]:
```
new_df_cap
```

Out[38]:

	cgpa	placement_exam_marks	placed	cgpa_score	cgpa_cap
0	7.19	8.808934	1	0.371425	7.190000
1	7.46	8.808934	1	0.809810	7.460000
2	7.54	8.808934	1	0.939701	7.540000
3	6.42	8.000000	1	-0.878782	6.420000
4	7.23	8.808934	0	0.436371	7.230000
...	...	...	...	...	...
995	8.87	8.808934	1	3.099150	8.808934
996	9.12	8.808934	1	3.505062	8.808934
997	4.89	8.808934	0	-3.362960	5.113546
998	8.62	8.808934	1	2.693239	8.620000
999	4.90	8.808934	1	-3.346724	5.113546

1000 rows × 5 columns

In [39]:
```
new_df_cap.shape
```

Out[39]:
```
(1000, 5)
```

In [ ]:
```

```