|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | |
| **In** | **[1]:** | Practical No 02  Data Wrangling II  Create an “Academic performance” dataset of students and perform the following operations using Python.  Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.  Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.  Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.  Reason and document your approach properly.  **import numpy as np import pandas as pd**  **import matplotlib.pyplot as plt import seaborn as sns**  **import warnings warnings.filterwarnings("ignore")**  **df = pd.read\_excel('data\_academic\_performance.xlsx') df**  COD\_S11 GENDER EDU\_FATHER EDU\_MOTHER OCC\_FATHER OCC\_MOTHER STRATUM SISBEN PEOP | | | | | | | | | |
| **In** | **[2]:** |
| **Out** | **[2]:** |
|  |  | 0 | SB11201210000129 | F | Incomplete Professional Education | Complete technique or technology | Technical or professional level employee | Home | Stratum 4 | It is not classiﬁed by the SISBEN | Three |
| 1 | SB11201210000137 | F | Complete Secundary | Complete professional education | Entrepreneur | Independent professional | Stratum 5 | It is not classiﬁed by the SISBEN | Three |
| 2 | SB11201210005154 | M | Not sure | Not sure | Independent | Home | Stratum 2 | Level 2 | Five |
|  |  |  |  |  |  |  |  | It is not |  |
| 3 | SB11201210007504 | F | Not sure | Not sure | Other  occupation | Independent | Stratum 2 | classiﬁed by the | Three |
|  |  |  |  |  |  |  |  | SISBEN |  |
| 4 | SB11201210007548 | M | Complete professional education | Complete professional education | Executive | Home | Stratum 4 | It is not classiﬁed by the SISBEN | One |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
|  |  |  |  |  |  |  |  | It is not |  |
| 12406 | SB11201420568705 | M | Ninguno | Complete  Secundary | Other  occupation | Auxiliary or  Administrative | Stratum 2 | classiﬁed by the | Six |
|  |  |  |  |  |  |  |  | SISBEN |  |
| 12407 | SB11201420573045 | M | Complete professional education | Complete Secundary | Executive | Other occupation | Stratum 2 | Level 2 | Five |
|  |  |  | Complete | Complete |  |  |  |  |  |
| 12408 | SB11201420578809 | M | technique or | technique or | Retired | Home | Stratum 2 | Level 2 | Five |
|  |  |  | technology | technology |  |  |  |  |  |
| 12409 | SB11201420578812 | F | Complete professional education | Complete professional education | Independent professional | Small entrepreneur | Stratum 3 | It is not classiﬁed by the SISBEN | Seven |
| 12410 | SB11201420583232 | M | Complete Secundary | Complete primary | Independent | Home | Stratum 3 | Level 1 | Four |
| **In** | **[3]:** | 12411 rows × 45 columns  **df.head() # It's showing top 5 result** | | | | | | | | | |

L

|  |  |
| --- | --- |
| **Out [3]:** |  |
|  | COD\_S11 GENDER EDU\_FATHER EDU\_MOTHER OCC\_FATHER OCC\_MOTHER STRATUM SISBEN PEOPLE\_H |
| **In [4]:** | Incomplete Complete Technical or It is not   1. SB11201210000129 F Professional technique or professional Home Stratum 4 classiﬁed Three Education technology level by the   employee SISBEN  Complete It is not   1. SB11201210000137 F Complete professional Entrepreneur Independent Stratum 5 classiﬁed Three Secundary education professional by the   SISBEN   1. SB11201210005154 M Not sure Not sure Independent Home Stratum 2 Level 2 Five   It is not   1. SB11201210007504 F Not sure Not sure Other Independent Stratum 2 classiﬁed Three   occupation by the  SISBEN  Complete Complete It is not   1. SB11201210007548 M professional professional Executive Home Stratum 4 classiﬁed One education education by the   SISBEN   1. rows × 45 columns   **df.tail() # It's showing bottom 5 result**  COD\_S11 GENDER EDU\_FATHER EDU\_MOTHER OCC\_FATHER OCC\_MOTHER STRATUM SISBEN PEOP  It is not  12406 SB11201420568705 M Ninguno Complete Other Auxiliary or Stratum 2 classiﬁed Six  Secundary occupation Administrative by the  SISBEN  Complete Complete Other  12407 SB11201420573045 M professional Secundary Executive occupation Stratum 2 Level 2 Five  education  Complete Complete  12408 SB11201420578809 M technique or technique or Retired Home Stratum 2 Level 2 Five  technology technology  Complete Complete It is not  12409 SB11201420578812 F professional professional Independent Small Stratum 3 classiﬁed Seven  education education professional entrepreneur by the  SISBEN  12410 SB11201420583232 M Complete Complete Independent Home Stratum 3 Level 1 Four  Secundary primary |
| **Out [4]:** |
|  | 5 rows × 45 columns |
|  | 1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the |
|  | suitable techniques to deal with them. |
| **In [5]:** | **df.isnull().sum() # Caluclating the Null values** |
| **Out [5]:** | **COD\_S11 0**  **GENDER 0**  **EDU\_FATHER 0**  **EDU\_MOTHER 0**  **OCC\_FATHER 0**  **OCC\_MOTHER 0**  **STRATUM 0**  **SISBEN** **0**  **PEOPLE\_HOUSE 0**  **Unnamed: 9** **12411**  **INTERNET** **0**  **TV 0**  **COMPUTER 0**  **WASHING\_MCH** **0**  **MIC\_OVEN** **0**  **CAR 0**  **DVD 0**  **FRESH 0**  **PHONE 0**  **MOBILE** **0**  **REVENUE 0**  **JOB 0**  **SCHOOL\_NAME 0**  **SCHOOL\_NAT 0**  **SCHOOL\_TYPE 0**  **MAT\_S11 0**  **CR\_S11 0**  **CC\_S11 0**  **BIO\_S11** **0**  **ENG\_S11 0** |

O

L

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cod\_SPro 0**  **UNIVERSITY** **0**  **ACADEMIC\_PROGRAM** **0**  **QR\_PRO 0**  **CR\_PRO 0**  **CC\_PRO 0**  **ENG\_PRO 0**  **WC\_PRO** **0**  **FEP\_PRO 0**  **G\_SC 0**  **PERCENTILE** **0**  **2ND\_DECILE** **0**  **QUARTILE** **0**  **SEL 0**  **SEL\_IHE** **0**  **dtype: int64**  **In [6]: df.drop('Unnamed: 9',axis=1,inplace=True) # Droping Cabin Column becasue here lots of null values**  **In [7]: df.dropna(inplace=True)**  **In [8]: df.head()**  **Out [8]:** COD\_S11 GENDER EDU\_FATHER EDU\_MOTHER OCC\_FATHER OCC\_MOTHER STRATUM SISBEN PEOPLE\_H | | | | | | | | | | |
|  | 0 | SB11201210000129 | F | Incomplete Professional Education | Complete technique or technology | Technical or professional level employee | Home | Stratum 4 | It is not classiﬁed by the SISBEN | Three |
|  | 1 | SB11201210000137 | F | Complete Secundary | Complete professional education | Entrepreneur | Independent professional | Stratum 5 | It is not classiﬁed by the SISBEN | Three |
|  | 2 | SB11201210005154 | M | Not sure | Not sure | Independent | Home | Stratum 2 | Level 2 | Five |
| 3 | SB11201210007504 | F | Not sure | Not sure | Other occupation | Independent | Stratum 2 | It is not classiﬁed by the SISBEN | Three |
| 4 | SB11201210007548 | M | Complete professional education | Complete professional education | Executive | Home | Stratum 4 | It is not classiﬁed by the SISBEN | One |
| 5 rows × 44 columns  **In [9]: df.isnull().sum() # Caluclating the Null values**  **Out [9]: COD\_S11** **0**  **GENDER 0**  **EDU\_FATHER 0**  **EDU\_MOTHER 0**  **OCC\_FATHER 0**  **OCC\_MOTHER 0**  **STRATUM 0**  **SISBEN** **0**  **PEOPLE\_HOUSE 0**  **INTERNET** **0**  **TV 0**  **COMPUTER 0**  **WASHING\_MCH** **0**  **MIC\_OVEN** **0**  **CAR 0**  **DVD 0**  **FRESH 0**  **PHONE 0**  **MOBILE** **0**  **REVENUE 0**  **JOB 0**  **SCHOOL\_NAME 0**  **SCHOOL\_NAT 0**  **SCHOOL\_TYPE 0**  **MAT\_S11 0**  **CR\_S11 0**  **CC\_S11 0**  **BIO\_S11** **0**  **ENG\_S11 0**  **Cod\_SPro 0**  **UNIVERSITY** **0**  **ACADEMIC\_PROGRAM** **0**  **QR\_PRO 0**  **CR\_PRO 0**  **CC\_PRO 0**  **ENG\_PRO 0**  **WC\_PRO** **0**  **FEP\_PRO 0**  **G\_SC 0**  **PERCENTILE** **0**  **2ND\_DECILE** **0**  **QUARTILE** **0** | | | | | | | | | | |

O

**In [10]:**

**Out [10]:**

**SEL 0**

**SEL\_IHE** **0**

**dtype: int64**

df.describe() # Get some initial statistics.

MAT\_S11 CR\_S11 CC\_S11 BIO\_S11 ENG\_S11 QR\_PRO CR\_PRO CC\_PRO count 12411.000000 12411.000000 12411.000000 12411.000000 12411.000000 12411.000000 12411.000000 12411.00000

mean 64.320764 60.778422 60.705181 63.950528 61.801064 77.417291 62.199339 59.18677

std 11.873650

10.025876

10.120524

11.156869

14.297777

22.673444

27.666558

28.99184

min 26.000000

24.000000

0.000000

11.000000

26.000000

1.000000

1.000000

1.00000

25% 56.000000

54.000000

54.000000

56.000000

50.000000

65.000000

42.000000

36.00000

50% 64.000000

61.000000

60.000000

64.000000

59.000000

85.000000

67.000000

65.00000

75% 72.000000

67.000000

67.000000

71.000000

72.000000

96.000000

86.000000

85.00000

max 100.000000 100.000000 100.000000 100.000000 100.000000 100.000000 100.000000 100.00000

**In [11]:**

df.info() # Getting some informatation about dataset

**<class 'pandas.core.frame.DataFrame'> Int64Index: 12411 entries, 0 to 12410 Data columns (total 44 columns):**

**# Column Non-Null Count Dtype**

1. **COD\_S11 12411 non-null object**
2. **GENDER 12411 non-null object**
3. **EDU\_FATHER 12411 non-null object**
4. **EDU\_MOTHER 12411 non-null object**
5. **OCC\_FATHER 12411 non-null object**
6. **OCC\_MOTHER 12411 non-null object**
7. **STRATUM 12411 non-null object**
8. **SISBEN 12411 non-null object**
9. **PEOPLE\_HOUSE 12411 non-null object**
10. **INTERNET 12411 non-null object**
11. **TV 12411 non-null object**
12. **COMPUTER 12411 non-null object**
13. **WASHING\_MCH 12411 non-null object**
14. **MIC\_OVEN 12411 non-null object**
15. **CAR 12411 non-null object**
16. **DVD 12411 non-null object**
17. **FRESH 12411 non-null object**
18. **PHONE 12411 non-null object**
19. **MOBILE 12411 non-null object**
20. **REVENUE 12411 non-null object**
21. **JOB 12411 non-null object**
22. **SCHOOL\_NAME 12411 non-null object**
23. **SCHOOL\_NAT 12411 non-null object**
24. **SCHOOL\_TYPE 12411 non-null object**
25. **MAT\_S11 12411 non-null int64**
26. **CR\_S11 12411 non-null int64**
27. **CC\_S11 12411 non-null int64**
28. **BIO\_S11 12411 non-null int64**
29. **ENG\_S11 12411 non-null int64**
30. **Cod\_SPro 12411 non-null object**
31. **UNIVERSITY 12411 non-null object**
32. **ACADEMIC\_PROGRAM 12411 non-null object**
33. **QR\_PRO 12411 non-null int64**
34. **CR\_PRO 12411 non-null int64**
35. **CC\_PRO 12411 non-null int64**
36. **ENG\_PRO 12411 non-null int64**
37. **WC\_PRO 12411 non-null int64**
38. **FEP\_PRO 12411 non-null int64**
39. **G\_SC 12411 non-null int64**
40. **PERCENTILE 12411 non-null int64**
41. **2ND\_DECILE 12411 non-null int64**
42. **QUARTILE 12411 non-null int64**
43. **SEL 12411 non-null int64**
44. **SEL\_IHE 12411 non-null int64 dtypes: int64(17), object(27)**

**memory usage: 4.3+ MB**

**In [12]:**

df.dtypes # Finding Data Types

**Out [12]: COD\_S11 object**

**GENDER object**

**EDU\_FATHER object**

**EDU\_MOTHER object**

**OCC\_FATHER object**

**OCC\_MOTHER object**

**STRATUM object**

**SISBEN object**

**PEOPLE\_HOUSE object**

**INTERNET object**

**TV object**

**COMPUTER object**

**WASHING\_MCH object**

**MIC\_OVEN object**

**CAR object**

**DVD object**

**FRESH object**

**PHONE object**

**MOBILE object**

**In [14]:**

**REVENUE object**

**JOB object**

**SCHOOL\_NAME object**

**SCHOOL\_NAT object**

**SCHOOL\_TYPE object**

**MAT\_S11 int64**

**CR\_S11 int64**

**CC\_S11 int64**

**BIO\_S11 int64**

**ENG\_S11 int64**

**Cod\_SPro object**

**UNIVERSITY object ACADEMIC\_PROGRAM object QR\_PRO int64**

**CR\_PRO int64**

**CC\_PRO int64**

**ENG\_PRO int64**

**WC\_PRO int64**

**FEP\_PRO int64**

**G\_SC int64**

**PERCENTILE int64**

**2ND\_DECILE int64**

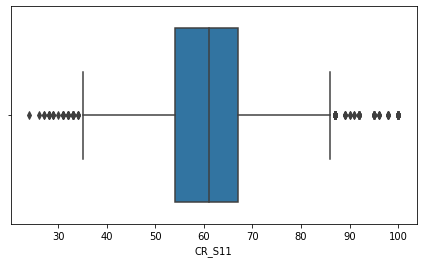
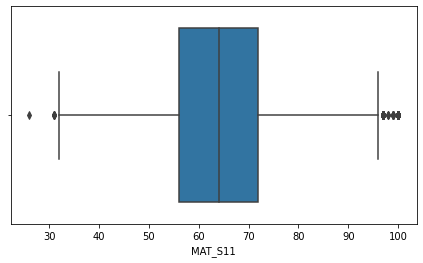
**QUARTILE int64**

**SEL int64**

**SEL\_IHE int64**

**dtype: object**

df.shape # Finding Dimensions of the data frame.



**Out [14]: (12411, 44)**

Finding Outliers

* 1. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

**In [15]:**

def ploting(df,st): plt.figure(figsize=(16,4)) plt.subplot(1,2,2) sns.boxplot(df[st]) plt.show()

**In [16]:**

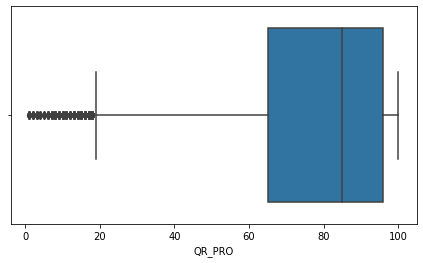
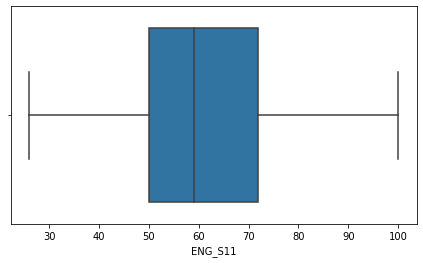
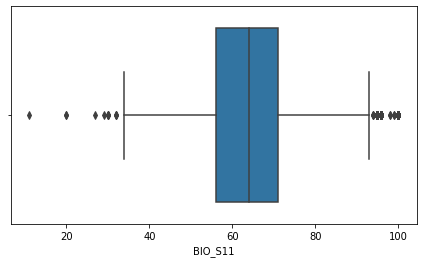
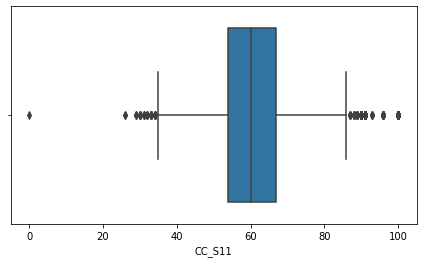
ploting(df,'MAT\_S11')

**In [17]:**

ploting(df,'CR\_S11')

**In [18]:**

ploting(df,'CC\_S11')



**In [19]:**

ploting(df,'BIO\_S11')

**In [20]:**

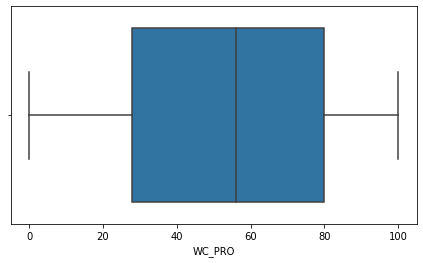
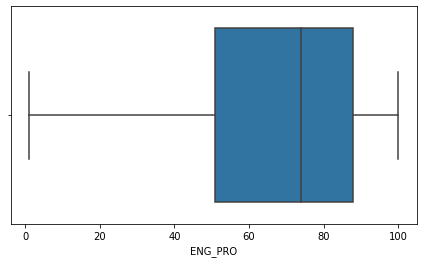
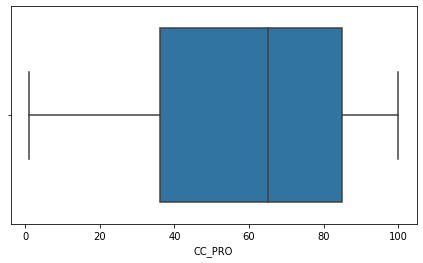
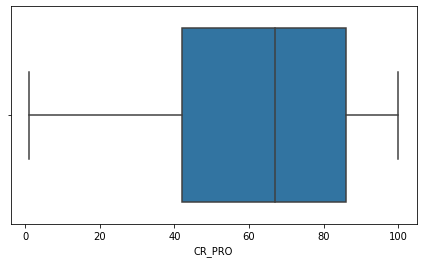
ploting(df,'ENG\_S11')

**In [21]:**

ploting(df,'QR\_PRO')

**In [22]:**

ploting(df,'CR\_PRO')



**In [23]:**

ploting(df,'CC\_PRO')

**In [24]:**

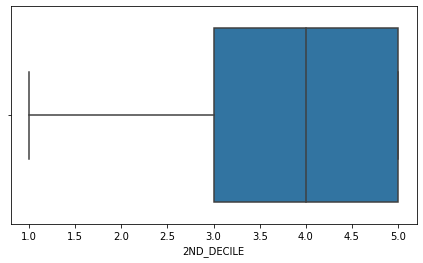
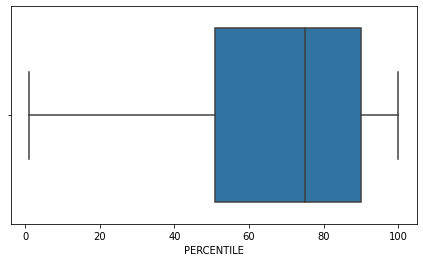
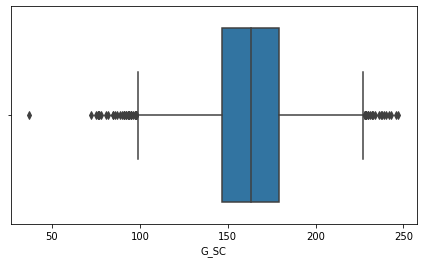
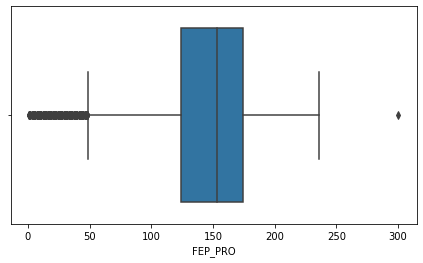
ploting(df,'ENG\_PRO')

**In [25]:**

ploting(df,'WC\_PRO')

**In [26]:**

ploting(df,'FEP\_PRO')



**In [27]:**

ploting(df,'G\_SC')

**In [28]:**

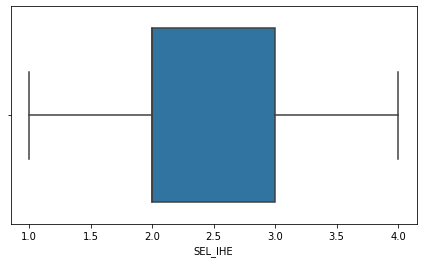
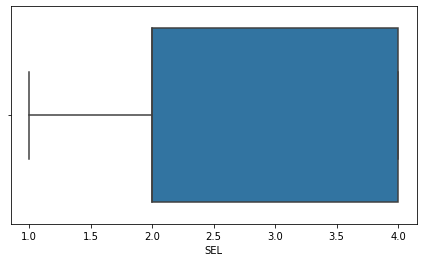
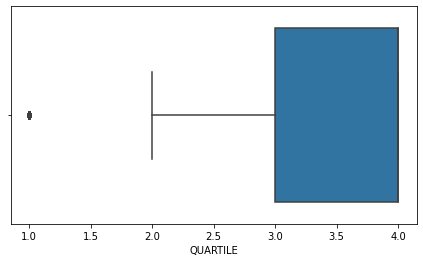
ploting(df,'PERCENTILE')

**In [29]:**

ploting(df,'2ND\_DECILE')

**In [30]:**

ploting(df,'QUARTILE')



**In [31]:**

ploting(df,'SEL')

**In [32]:**

ploting(df,'SEL\_IHE')

# Detecting Outliers

**In [33]:**

# Detecting Outliers import numpy as np outliers = []

def detect\_outliers\_zscore(df): thres = 3

mean = np.mean(df) std = np.std(df) # print(mean, std) for i in df:

z\_score = (i-mean)/std

if (np.abs(z\_score) > thres): outliers.append(i)

return outliers

**In [34]:**

mat = detect\_outliers\_zscore(df['MAT\_S11']) print("Outliers from Z-scores method: ", mat)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [35]:**

cr = detect\_outliers\_zscore(df['CR\_S11']) print("Outliers from Z-scores method: ", cr)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [36]:**

cc = detect\_outliers\_zscore(df['CC\_S11']) print("Outliers from Z-scores method: ", cc)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [37]:**

bio = detect\_outliers\_zscore(df['BIO\_S11']) print("Outliers from Z-scores method: ", bio)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [38]:**

eng = detect\_outliers\_zscore(df['ENG\_S11']) print("Outliers from Z-scores method: ", eng)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [39]:**

qr = detect\_outliers\_zscore(df['QR\_PRO']) print("Outliers from Z-scores method: ", qr)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [40]:**

crpro = detect\_outliers\_zscore(df['CR\_PRO']) print("Outliers from Z-scores method: ", crpro)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [41]:**

ccpro = detect\_outliers\_zscore(df['CC\_PRO']) print("Outliers from Z-scores method: ", ccpro)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [42]:**

engpro = detect\_outliers\_zscore(df['ENG\_PRO']) print("Outliers from Z-scores method: ", engpro)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [43]:**

wcpro = detect\_outliers\_zscore(df['WC\_PRO']) print("Outliers from Z-scores method: ", wcpro)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [44]:**

feppro = detect\_outliers\_zscore(df['FEP\_PRO']) print("Outliers from Z-scores method: ", feppro)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [45]:**

gsc = detect\_outliers\_zscore(df['G\_SC']) print("Outliers from Z-scores method: ", gsc)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [46]:**

percentile = detect\_outliers\_zscore(df['PERCENTILE']) print("Outliers from Z-scores method: ", percentile)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [47]:**

decile = detect\_outliers\_zscore(df['2ND\_DECILE']) print("Outliers from Z-scores method: ", decile)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [48]:**

quartile = detect\_outliers\_zscore(df['QUARTILE']) print("Outliers from Z-scores method: ", quartile)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [49]:**

sel = detect\_outliers\_zscore(df['SEL']) print("Outliers from Z-scores method: ", sel)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

**In [50]:**

selihe = detect\_outliers\_zscore(df['SEL\_IHE']) print("Outliers from Z-scores method: ", selihe)

**Outliers from Z-scores method: [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10**

# Finding IQR

**In [51]:**

def finding\_Iqr(df,st):

#lets find the IQR (inter quantile range) Q1 = df[st].quantile(0.25)

Q3 = df[st].quantile(0.75)

IQR = Q3-Q1

lower\_boundry = Q1 -1.5\*IQR upper\_boundry = Q3 +1.5\*IQR

return lower\_boundry , upper\_boundry

**In [52]:**

lower\_MAT\_S11, upper\_MAT\_S11 = finding\_Iqr(df,'MAT\_S11') print('upper limit is' , upper\_MAT\_S11)

print('lower limit is' , lower\_MAT\_S11)

**upper limit is 96.0 lower limit is 32.0**

**In [53]:**

lower\_CR\_S11, upper\_CR\_S11 = finding\_Iqr(df,'CR\_S11') print('upper limit is' , upper\_CR\_S11)

print('lower limit is' , lower\_CR\_S11)

**In [54]:**

lower\_CC\_S11, upper\_CC\_S11 = finding\_Iqr(df,'CC\_S11') print('upper limit is' , upper\_CC\_S11)

print('lower limit is' , lower\_CC\_S11)

**upper limit is 86.5 lower limit is 34.5**

**In [55]:**

**upper limit is 86.5 lower limit is 34.5**

**upper limit is 93.5 lower limit is 33.5**

lower\_BIO\_S11, upper\_BIO\_S11 = finding\_Iqr(df,'BIO\_S11') print('upper limit is' , upper\_BIO\_S11)

print('lower limit is' , lower\_BIO\_S11)

**In [56]:**

lower\_ENG\_S11, upper\_ENG\_S11 = finding\_Iqr(df,'ENG\_S11') print('upper limit is' , upper\_ENG\_S11)

print('lower limit is' , lower\_ENG\_S11)

**upper limit is 105.0 lower limit is 17.0**

**In [57]:**

lower\_QR\_PRO, upper\_QR\_PRO = finding\_Iqr(df,'QR\_PRO') print('upper limit is' , upper\_QR\_PRO)

print('lower limit is' , lower\_QR\_PRO)

**upper limit is 142.5 lower limit is 18.5**

**In [58]:**

lower\_CR\_PRO, upper\_CR\_PRO = finding\_Iqr(df,'CR\_PRO') print('upper limit is' , upper\_CR\_PRO)

print('lower limit is' , lower\_CR\_PRO)

**upper limit is 152.0 lower limit is -24.0**

**In [59]:**

lower\_ENG\_PRO, upper\_ENG\_PRO = finding\_Iqr(df,'ENG\_PRO') print('upper limit is' , upper\_CR\_PRO)

print('lower limit is' , lower\_CR\_PRO)

**In [60]:**

**upper limit is 152.0 lower limit is -24.0**

**upper limit is 158.0 lower limit is -50.0**

lower\_WC\_PRO, upper\_WC\_PRO = finding\_Iqr(df,'WC\_PRO') print('upper limit is' , upper\_WC\_PRO)

print('lower limit is' , lower\_WC\_PRO)

**In [61]:**

lower\_FEP\_PRO, upper\_FEP\_PRO = finding\_Iqr(df,'FEP\_PRO') print('upper limit is' , upper\_FEP\_PRO)

print('lower limit is' , lower\_FEP\_PRO)

**upper limit is 249.0 lower limit is 49.0**

**In [62]:**

lower\_G\_SC, upper\_G\_SC = finding\_Iqr(df,'G\_SC') print('upper limit is' , upper\_G\_SC) print('lower limit is' , lower\_G\_SC)

**In [63]:**

lower\_PERCENTILE, upper\_PERCENTILE = finding\_Iqr(df,'PERCENTILE') print('upper limit is' , upper\_PERCENTILE)

print('lower limit is' , lower\_PERCENTILE)

**upper limit is 227.0 lower limit is 99.0**

**In [64]:**

**upper limit is 148.5 lower limit is -7.5**

**upper limit is 8.0 lower limit is 0.0**

lower\_2ND\_DECILE, upper\_2ND\_DECILE = finding\_Iqr(df,'2ND\_DECILE') print('upper limit is' , upper\_2ND\_DECILE)

print('lower limit is' , lower\_2ND\_DECILE)

**In [65]:**

lower\_QUARTILE, upper\_QUARTILE = finding\_Iqr(df,'QUARTILE') print('upper limit is' , upper\_QUARTILE)

print('lower limit is' , lower\_QUARTILE)

**upper limit is 5.5 lower limit is 1.5**

**In [66]:**

lower\_SEL, upper\_SEL = finding\_Iqr(df,'SEL') print('upper limit is' , upper\_SEL) print('lower limit is' , lower\_SEL)

**upper limit is 7.0 lower limit is -1.0**

**In [67]:**

lower\_SEL\_IHE, upper\_SEL\_IHE = finding\_Iqr(df,'SEL\_IHE') print('upper limit is' , upper\_SEL\_IHE)

print('lower limit is' , lower\_SEL\_IHE)

**upper limit is 4.5 lower limit is 0.5**

# Removing Outliers

**In [68]:**

#Removing Outliers

outliers\_MAT\_S11 = np.where(df['MAT\_S11'] > upper\_MAT\_S11,True ,np.where(df['MAT\_S11']< lower\_MAT\_ outliers\_MAT\_S11

**Out [68]: array([False, False, False, ..., False, False, False]) In [69]:**

#Removing Outliers

outliers\_CR\_S11 = np.where(df['CR\_S11'] > upper\_CR\_S11,True ,np.where(df['CR\_S11']< lower\_CR\_S11, outliers\_CR\_S11

**Out [69]: array([False, False, False, ..., False, False, False])**

**In [70]:**

#Removing Outliers

outliers\_CC\_S11 = np.where(df['CC\_S11'] > upper\_CC\_S11,True ,np.where(df['CC\_S11']< lower\_CC\_S11, outliers\_CC\_S11

**Out [70]: array([False, False, False, ..., False, False, False]) In [71]:**

#Removing Outliers

outliers\_BIO\_S11 = np.where(df['BIO\_S11'] > upper\_BIO\_S11,True ,np.where(df['BIO\_S11']< lower\_BIO\_ outliers\_BIO\_S11

**Out [71]: array([False, True, False, ..., False, False, False]) In [72]:**

#Removing Outliers

outliers\_QR\_PRO = np.where(df['QR\_PRO'] > upper\_QR\_PRO,True ,np.where(df['QR\_PRO']< lower\_QR\_PRO, outliers\_QR\_PRO

**Out [72]: array([False, False, True, ..., False, False, False]) In [73]:**

#Removing Outliers

outliers\_FEP\_PRO = np.where(df['FEP\_PRO'] > upper\_FEP\_PRO,True ,np.where(df['FEP\_PRO']< lower\_FEP\_ outliers\_FEP\_PRO

**Out [73]: array([False, False, False, ..., False, False, False]) In [74]:**

#Removing Outliers

outliers\_G\_SC = np.where(df['G\_SC'] > upper\_G\_SC,True ,np.where(df['G\_SC']< lower\_G\_SC, True , Fal outliers\_G\_SC

**Out [74]: array([False, False, False, ..., False, False, False]) In [75]:**

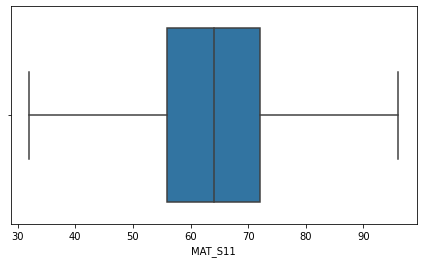
#Removing Outliers

outliers\_QUARTILE = np.where(df['QUARTILE'] > upper\_QUARTILE,True ,np.where(df['QUARTILE']< lower\_ outliers\_QUARTILE

**Out [75]: array([False, False, True, ..., False, False, False])**

**In [76]:**

df['MAT\_S11']= np.where(df['MAT\_S11']> upper\_MAT\_S11 , upper\_MAT\_S11,np.where(df['MAT\_S11'] < lowe df['CR\_S11']= np.where(df['CR\_S11']> upper\_CR\_S11 , upper\_CR\_S11,np.where(df['CR\_S11'] < lower\_CR\_ df['CC\_S11']= np.where(df['CC\_S11']> upper\_CC\_S11 , upper\_CC\_S11,np.where(df['CC\_S11'] < lower\_CC\_ df['BIO\_S11']= np.where(df['BIO\_S11']> upper\_BIO\_S11 , upper\_BIO\_S11,np.where(df['BIO\_S11'] < lowe df['QR\_PRO']= np.where(df['QR\_PRO']> upper\_QR\_PRO , upper\_QR\_PRO,np.where(df['QR\_PRO'] < lower\_QR\_ df['FEP\_PRO']= np.where(df['FEP\_PRO']> upper\_FEP\_PRO , upper\_FEP\_PRO,np.where(df['FEP\_PRO'] < lowe df['G\_SC']= np.where(df['G\_SC']> upper\_G\_SC , upper\_G\_SC,np.where(df['G\_SC'] < lower\_G\_SC , lower\_ df['QUARTILE']= np.where(df['QUARTILE']> upper\_QUARTILE , upper\_QUARTILE,np.where(df['QUARTILE'] <



# After Removing Outliers

**In [77]:**

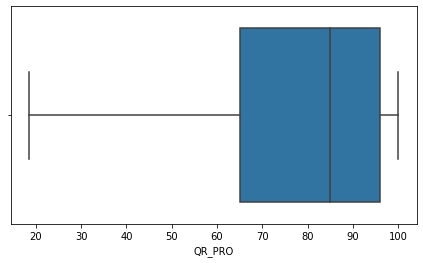
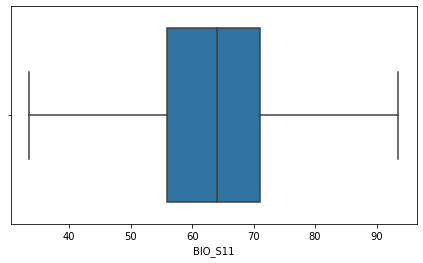
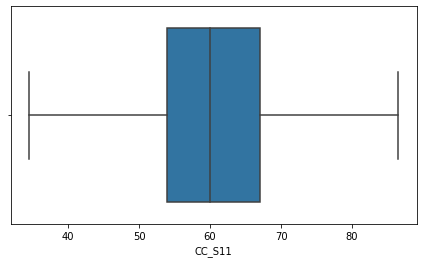
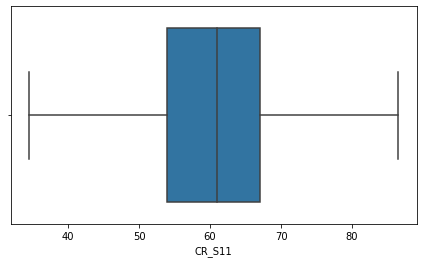
def boxplt(df,st): plt.figure(figsize=(16,4)) plt.subplot(1,2,2) sns.boxplot(df[st]) plt.show()

**In [78]:**

boxplt(df, 'MAT\_S11')

**In [79]:**

boxplt(df, 'CR\_S11')



**In [80]:**

boxplt(df, 'CC\_S11')

**In [81]:**

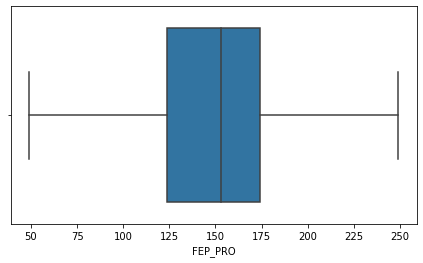
boxplt(df, 'BIO\_S11')

**In [82]:**

boxplt(df, 'QR\_PRO')

**In [83]:**

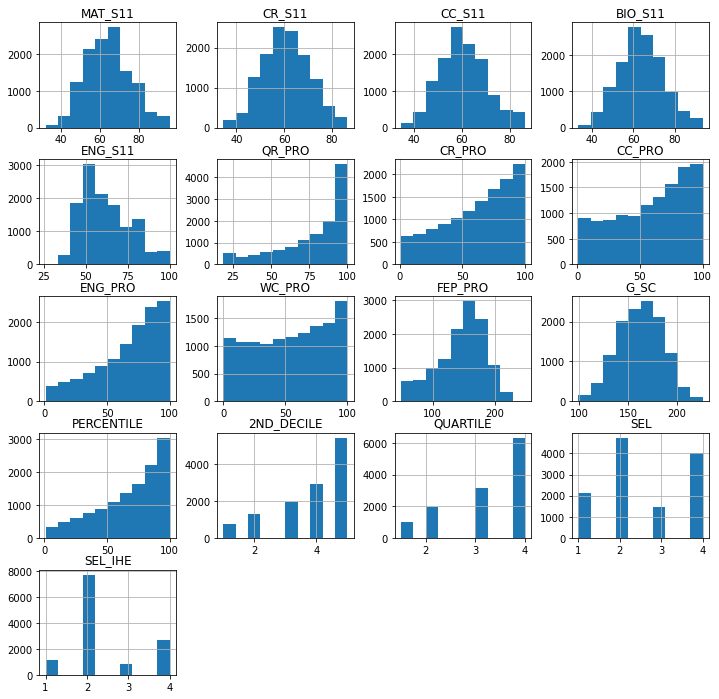
boxplt(df, 'FEP\_PRO')



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **In [84]: boxplt(df, 'G\_SC')**    **In [85]: boxplt(df, 'QUARTILE')**    1. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.  **In [86]: df.head()**  **Out [86]:** COD\_S11 GENDER EDU\_FATHER EDU\_MOTHER OCC\_FATHER OCC\_MOTHER STRATUM SISBEN PEOPLE\_H | | | | | | | | | | |
|  | 0 | SB11201210000129 | F | Incomplete Professional Education | Complete technique or technology | Technical or professional level employee | Home | Stratum 4 | It is not classiﬁed by the SISBEN | Three |
|  | 1 | SB11201210000137 | F | Complete Secundary | Complete professional education | Entrepreneur | Independent professional | Stratum 5 | It is not classiﬁed by the SISBEN | Three |
|  | 2 | SB11201210005154 | M | Not sure | Not sure | Independent | Home | Stratum 2 | Level 2 | Five |
| 3 | SB11201210007504 | F | Not sure | Not sure | Other occupation | Independent | Stratum 2 | It is not classiﬁed by the SISBEN | Three |
| 4 | SB11201210007548 | M | Complete professional education | Complete professional education | Executive | Home | Stratum 4 | It is not classiﬁed by the SISBEN | One |
| 5 rows × 44 columns | | | | | | | | | | |

O

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **In** | **[87]:** | **df.hist(figsize=(12,12)) plt.show()** | | | | | |  |
|  |  |  |  |  |  |  |  | |
| **In** | **[88]:** | **X = df.iloc[:,[24,25,26,27,28,32,33,34,35,36,37,38,39,40]]** |  |  |  |  |  | |
| **In** | **[89]:** | **X.head(5)** |  |  |  |  |  | |
| **Out** | **[89]:** | MAT\_S11 CR\_S11 CC\_S11 BIO\_S11 ENG\_S11 QR\_PRO CR\_PRO CC\_PRO | ENG\_PRO | WC\_PRO | FEP\_PRO | G\_SC | PERC | |
|  |  | 0 71.0 81.0 61.0 86.0 82 71.0 93 71 | 93 | 79 | 181.0 | 180.0 | 91 | |
|  |  | 1 83.0 75.0 66.0 93.5 88 97.0 38 86 | 98 | 78 | 201.0 | 182.0 | 92 | |
|  |  | 2 52.0 49.0 38.0 46.0 42 18.5 1 18 | 43 | 22 | 113.0 | 113.0 | 7 | |
|  |  | 3 56.0 55.0 51.0 64.0 73 65.0 35 76 | 80 | 48 | 137.0 | 157.0 | 67 | |
|  |  | 4 80.0 65.0 76.0 85.0 92 94.0 94 98 | 100 | 71 | 189.0 | 198.0 | 98 | |
| **In** | **[90]:** | **from sklearn.preprocessing import MinMaxScaler** | | | | | | |
| **In** | **[91]:** | **scaler=MinMaxScaler(feature\_range=(0, 1))** | | | | | | |
|  |  | **scaler.fit(X)** | | | | | | |
| **Out** | **[91]:** | **MinMaxScaler()** | | | | | | |
| **In** | **[92]:** | **scaled\_data=scaler.transform(X)** | | | | | | |
| **In** | **[93]:** | **scaled\_data** | | | | | | |
| **Out** | **[93]:** | **array([[0.609375 , 0.89423077, 0.50961538, ..., 0.6328125 , 0.90909091,** | | | | | | |
|  |  | **1. ],** | | | | | | |
|  |  | **[0.796875 , 0.77884615, 0.60576923, ..., 0.6484375 , 0.91919192,** | | | | | | |
|  |  | **1. ],** | | | | | | |
|  |  | **[0.3125 , 0.27884615, 0.06730769, ..., 0.109375 , 0.06060606,** | | | | | | |
|  |  | **0. ],** | | | | | | |
|  |  | **...,** | | | | | | |
|  |  | **[0.53125 , 0.66346154, 0.77884615, ..., 0.6953125 , 0.94949495,** | | | | | | |
|  |  | **1. ],** | | | | | | |
|  |  | **[0.328125 , 0.66346154, 0.56730769, ..., 0.3671875 , 0.49494949,** | | | | | | |
|  |  | **0.5 ],** | | | | | | |



**[0.734375 , 0.58653846, 0.52884615, ..., 0.6171875 , 0.88888889,**

**1. ]])**

---------------END---------------