In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_excel('data_academic_performance.xlsx')
df
```

Out[2]:

	COD_S11	GENDER	EDU_FATHER	EDU_MOTHER	OCC_FATHER	OCC_MOTHER
0	SB11201210000129	F	Incomplete Professional Education	Complete technique or technology	Technical or professional level employee	Home
1	SB11201210000137	F	Complete Secundary	Complete professional education	Entrepreneur	Independent professional
2	SB11201210005154	M	Not sure	Not sure	Independent	Home
3	SB11201210007504	F	Not sure	Not sure	Other occupation	Independent
4	SB11201210007548	M	Complete professional education	Complete professional education	Executive	Home
...	...	...	...	...	...	...
12406	SB11201420568705	M	Ninguno	Complete Secundary	Other occupation	Auxiliary or Administrative
12407	SB11201420573045	M	Complete professional education	Complete Secundary	Executive	Other occupation
12408	SB11201420578809	M	Complete technique or technology	Complete technique or technology	Retired	Home
12409	SB11201420578812	F	Complete professional education	Complete professional education	Independent professional	Small entrepreneur
12410	SB11201420583232	M	Complete Secundary	Complete primary	Independent	Home

12411 rows × 45 columns

In [3]:
```
df.head() # It's showing top 5 result
```

Out[3]:

	COD_S11	GENDER	EDU_FATHER	EDU_MOTHER	OCC_FATHER	OCC_MOTHER	ST
0	SB11201210000129	F	Incomplete Professional Education	Complete technique or technology	Technical or professional level employee	Home	S
1	SB11201210000137	F	Complete Secundary	Complete professional education	Entrepreneur	Independent professional	S
2	SB11201210005154	M	Not sure	Not sure	Independent	Home	S
3	SB11201210007504	F	Not sure	Not sure	Other occupation	Independent	S
4	SB11201210007548	M	Complete professional education	Complete professional education	Executive	Home	S

5 rows × 45 columns

In [4]:
```
df.tail() # It's showing bottom 5 result
```

Out[4]:

	COD_S11	GENDER	EDU_FATHER	EDU_MOTHER	OCC_FATHER	OCC_MOTHER
12406	SB11201420568705	M	Ninguno	Complete Secundary	Other occupation	Auxiliary or Administrative
12407	SB11201420573045	M	Complete professional education	Complete Secundary	Executive	Other occupation
12408	SB11201420578809	M	Complete technique or technology	Complete technique or technology	Retired	Home
12409	SB11201420578812	F	Complete professional education	Complete professional education	Independent professional	Small entrepreneur
12410	SB11201420583232	M	Complete Secundary	Complete primary	Independent	Home

5 rows × 45 columns

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
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
```

In [7]:
```
df.dropna(inplace=True)
```

In [8]:
```
df.head()
```

Out[8]:

	COD_S11	GENDER	EDU_FATHER	EDU_MOTHER	OCC_FATHER	OCC_MOTHER	ST
0	SB11201210000129	F	Incomplete Professional Education	Complete technique or technology	Technical or professional level employee	Home	S
1	SB11201210000137	F	Complete Secundary	Complete professional education	Entrepreneur	Independent professional	S
2	SB11201210005154	M	Not sure	Not sure	Independent	Home	S
3	SB11201210007504	F	Not sure	Not sure	Other occupation	Independent	S
4	SB11201210007548	M	Complete professional education	Complete professional education	Executive	Home	S

5 rows × 44 columns

In [14]:
```
df.shape # Finding Dimensions of the data frame.
```

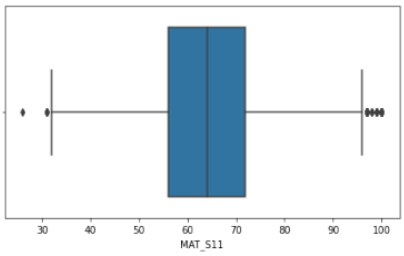
Out[14]:
```
(12411, 44)
```

Finding Outliers

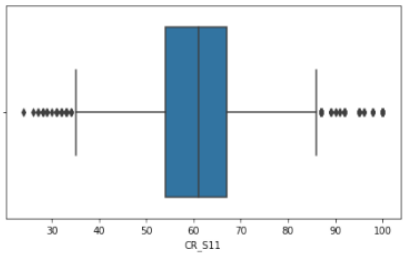
2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

In [15]:
```
def ploting(df,st):
    plt.figure(figsize=(16,4))
    plt.subplot(1,2,2)
    sns.boxplot(df[st])
    plt.show()
```

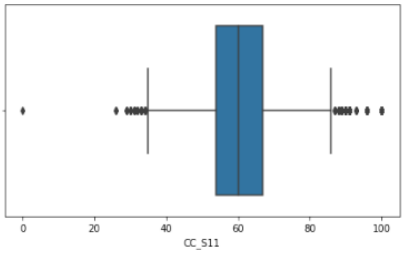
In [16]:
```
ploting(df,'MAT_S11')
```



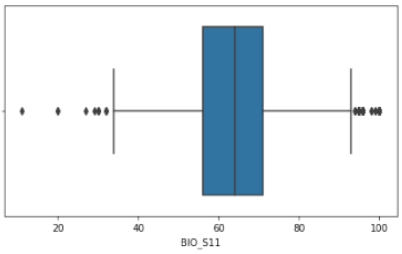
In [17]:
```
ploting(df,'CR_S11')
```



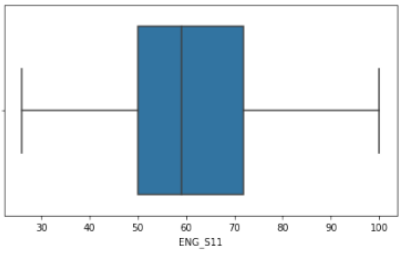
In [18]:
```
ploting(df,'CC_S11')
```



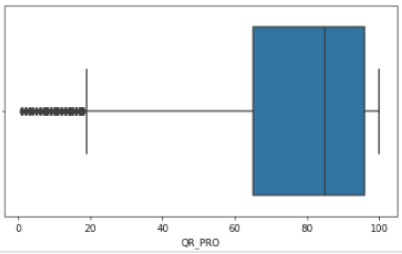
In [19]:
```
ploting(df,'BIO_S11')
```



In [20]:
```
ploting(df,'ENG_S11')
```



In [21]:
```
ploting(df,'QR_PRO')
```



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, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 26, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100]
```

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, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 26, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 28, 100, 100, 98, 98, 100, 26, 100, 98, 100, 98, 100, 100,
29, 100, 100, 100, 100, 100, 100, 29, 24, 100, 30, 29, 27, 100, 95, 100, 95,
100, 100, 95, 95, 100, 95, 95, 96, 95, 27, 95, 95, 95, 96, 100, 28, 28, 100,
28, 95, 96, 92, 92, 92, 92, 91, 98, 92, 92, 100, 92, 92, 92, 92, 92, 92, 91,
100]
```

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, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 26, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 28, 100, 100, 98, 98, 100, 26, 100, 98, 100, 98, 100, 100,
29, 100, 100, 100, 100, 100, 100, 29, 24, 100, 30, 29, 27, 100, 95, 100, 95,
100, 100, 95, 95, 100, 95, 95, 96, 95, 27, 95, 95, 95, 96, 100, 28, 28, 100,
28, 95, 96, 92, 92, 92, 92, 91, 98, 92, 92, 100, 92, 92, 92, 92, 92, 92, 91,
100, 29, 100, 96, 96, 96, 96, 30, 26, 100, 0, 96, 96, 96, 96, 96, 96, 96, 96,
96, 26, 29, 100, 100, 96, 96, 29, 96, 30, 96, 100, 100, 100, 100, 100, 30, 10
0, 100, 100, 100, 30, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 30, 100, 100, 100, 30, 93, 100, 93, 93, 3
0, 93]
```

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, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 26, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 28, 100, 100, 98, 98, 100, 26, 100, 98, 100, 98, 100, 100,
29, 100, 100, 100, 100, 100, 100, 29, 24, 100, 30, 29, 27, 100, 95, 100, 95,
100, 100, 95, 95, 100, 95, 95, 96, 95, 27, 95, 95, 95, 96, 100, 28, 28, 100,
28, 95, 96, 92, 92, 92, 92, 91, 98, 92, 92, 100, 92, 92, 92, 92, 92, 92, 91,
100, 29, 100, 96, 96, 96, 96, 30, 26, 100, 0, 96, 96, 96, 96, 96, 96, 96, 96,
96, 26, 29, 100, 100, 96, 96, 29, 96, 30, 96, 100, 100, 100, 100, 100, 30, 10
0, 100, 100, 100, 30, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 30, 100, 100, 100, 30, 93, 100, 93, 93, 3
0, 93, 100, 100, 100, 100, 100, 100, 20, 100, 100, 100, 100, 100, 100, 100, 1
00, 100, 100, 100, 11, 100, 100, 100, 100, 100, 100, 30, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 30, 100, 100, 100, 100, 20, 100,
100, 100, 100, 100, 30, 100, 100, 100, 30, 100, 100, 100, 100, 27, 100, 100,
100, 100, 100, 100, 100, 98, 100, 99, 100, 99, 100, 100, 98, 100, 100, 1
00, 29]
```

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, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 26, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 10
0, 100, 100, 100, 28, 100, 100, 98, 98, 100, 26, 100, 98, 100, 98, 100, 100,
29, 100, 100, 100, 100, 100, 100, 29, 24, 100, 30, 29, 27, 100, 95, 100, 95,
100, 100, 95, 95, 100, 95, 95, 96, 95, 27, 95, 95, 95, 96, 100, 28, 28, 100,
28, 95, 96, 92, 92, 92, 92, 91, 98, 92, 92, 100, 92, 92, 92, 92, 92, 92, 91,
100, 29, 100, 96, 96, 96, 96, 30, 26, 100, 0, 96, 96, 96, 96, 96, 96, 96, 96,
96, 26, 29, 100, 100, 96, 96, 29, 96, 30, 96, 100, 100, 100, 100, 100, 30, 10
0, 100, 100, 100, 30, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 30, 100, 100, 100, 30, 93, 100, 93, 93, 3
0, 93, 100, 100, 100, 100, 100, 20, 100, 100, 100, 100, 100, 100, 100, 1
00, 100, 100, 100, 11, 100, 100, 100, 100, 100, 100, 30, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 30, 100, 100, 100, 100, 20, 100,
100, 100, 100, 100, 30, 100, 100, 100, 30, 100, 100, 100, 100, 27, 100, 100,
100, 100, 100, 100, 100, 100, 98, 100, 99, 100, 99, 100, 100, 98, 100, 100, 1
00, 29]
```

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)
```

```
upper limit is 86.5
lower limit is 34.5
```

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]:
```
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)
```

```
upper limit is 93.5
lower limit is 33.5
```

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
```

Removing Outliers

In [68]:
```
#Removing Outliers
outliers_MAT_S11 = np.where(df['MAT_S11'] > upper_MAT_S11,True ,np.where(df['M
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_S
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_S
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['B
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_P
outliers_QR_PRO
```

Out[72]:
```
array([False, False,  True, ..., False, False, False])
```

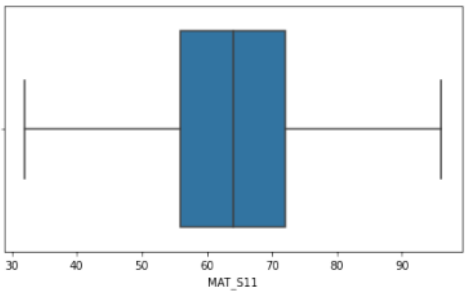
In [73]:
```
#Removing Outliers
outliers_FEP_PRO = np.where(df['FEP_PRO'] > upper_FEP_PRO,True ,np.where(df['F
outliers_FEP_PRO
```

Out[73]:
```
array([False, False, False, ..., False, False, False])
```

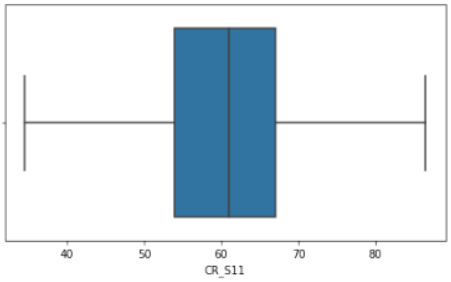
After Removing Outliers

In [77]:
```
def boxplt(df,st):
    plt.figure(figsize=(16,4))
    plt.subplot(1,2,2)
    sns.boxplot(df[st])
    plt.show()
```

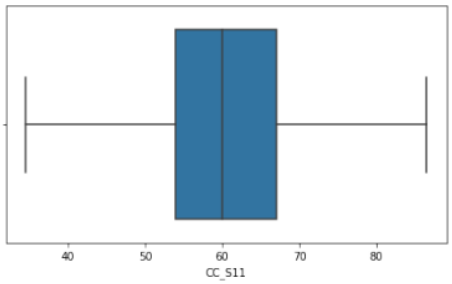
In [78]:
```
boxplt(df, 'MAT_S11')
```



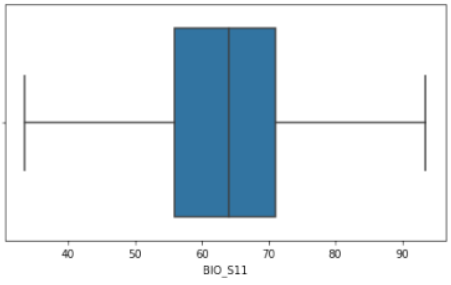
In [79]:
```
boxplt(df, 'CR_S11')
```



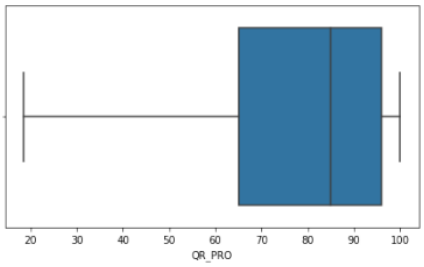
In [80]:
```
boxplt(df, 'CC_S11')
```



In [81]:
```
boxplt(df, 'BIO_S11')
```



In [82]:
```
boxplt(df, 'QR_PRO')
```



3. 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.

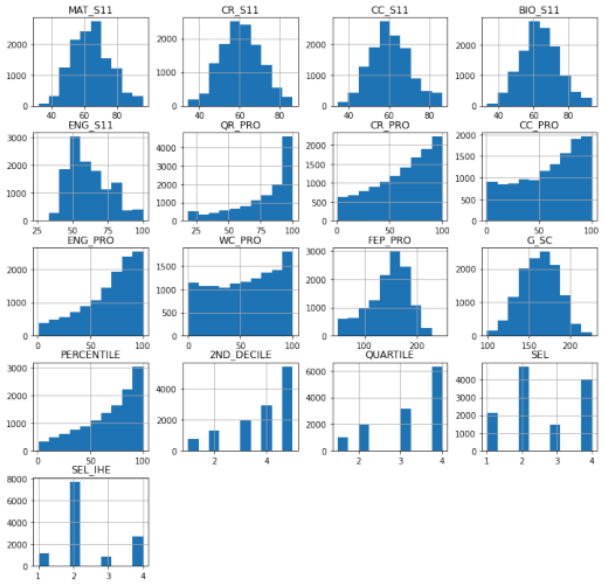
In [86]:
```
df.head()
```

Out[86]:

	COD_S11	GENDER	EDU_FATHER	EDU_MOTHER	OCC_FATHER	OCC_MOTHER	ST
0	SB11201210000129	F	Incomplete Professional Education	Complete technique or technology	Technical or professional level employee	Home	S
1	SB11201210000137	F	Complete Secundary	Complete professional education	Entrepreneur	Independent professional	S
2	SB11201210005154	M	Not sure	Not sure	Independent	Home	S
3	SB11201210007504	F	Not sure	Not sure	Other occupation	Independent	S
4	SB11201210007548	M	Complete professional education	Complete professional education	Executive	Home	S

5 rows × 44 columns

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	W
0	71.0	81.0	61.0	86.0	82	71.0	93	71	93	
1	83.0	75.0	66.0	93.5	88	97.0	38	86	98	
2	52.0	49.0	38.0	46.0	42	18.5	1	18	43	
3	56.0	55.0	51.0	64.0	73	65.0	35	76	80	
4	80.0	65.0	76.0	85.0	92	94.0	94	98	100	

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.        ]])
```