DSBDAL Assignment 2 - Data Wrangling II

Importing required Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Reading csv file

In [2]:
```
df=pd.read_csv('student_data.csv')
```

Displaying first 5 records

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

Out[3]:

	rollno	branch	marksPhy	marksChem	marksEM1	marksELEC	marksSME	Percentage	att
0	1	Comp	50.0	39.0	45.0	99.0	90.0	64.6	
1	2	IT	33.0	77.0	33.0	54.0	54.0	50.2	
2	3	IT	55.0	86.0	66.0	34.0	67.0	61.6	
3	4	IT	77.0	79.0	88.0	67.0	NaN	62.2	
4	5	Comp	1000.0	56.0	99.0	NaN	76.0	246.2	

Exploring Data

In [4]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[4]:
```
marksPhy       15
marksEM1       14
attendance     11
marksSME       10
marksELEC       4
marksChem       2
rollno          0
branch          0
Percentage      0
gender          0
dtype: int64
```

In [5]:
```
df.describe()
```

Out[5]:

	rollno	marksPhy	marksChem	marksEM1	marksELEC	marksSME	Percenta
count	100.000000	85.000000	98.000000	86.000000	96.000000	90.000000	100.0000
mean	50.500000	79.635294	71.316327	71.267442	83.760417	68.333333	68.1560
std	29.011492	118.834403	36.234800	38.161399	117.578171	53.115308	35.5031

	rollno	marksPhy	marksChem	marksEM1	marksELEC	marksSME	Percenta
min	1.000000	-55.000000	-44.000000	-45.000000	-99.000000	-88.000000	17.4000
25%	25.750000	54.000000	56.000000	56.000000	58.500000	54.250000	50.8000
50%	50.500000	67.000000	74.000000	67.000000	77.500000	67.000000	63.1000
75%	75.250000	88.000000	88.000000	88.000000	90.000000	87.750000	75.2000
max	100.000000	1000.000000	333.000000	333.000000	1111.000000	444.000000	274.0000

In [6]:
```
df.dtypes
```

Out[6]:
```
rollno          int64
branch         object
marksPhy      float64
marksChem     float64
marksEM1      float64
marksELEC     float64
marksSME      float64
Percentage    float64
attendance    float64
gender         object
dtype: object
```

In [7]:
```
print('Our data set contains {} rows and {} columns'.format(df.shape[0],df.sh
```

```
Our data set contains 100 rows and 10 columns
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   rollno      100 non-null    int64  
 1   branch      100 non-null    object 
 2   marksPhy    85 non-null     float64
 3   marksChem   98 non-null     float64
 4   marksEM1    86 non-null     float64
 5   marksELEC   96 non-null     float64
 6   marksSME    90 non-null     float64
 7   Percentage  100 non-null    float64
 8   attendance  89 non-null     float64
 9   gender      100 non-null    object 
dtypes: float64(7), int64(1), object(2)
memory usage: 7.9+ KB
```

In [9]:
```
df.branch.value_counts()
```

Out[9]:
```
ENTC    41
IT      33
Comp    26
Name: branch, dtype: int64
```

Preprocessing Data

Replacing out of bound values with Nan

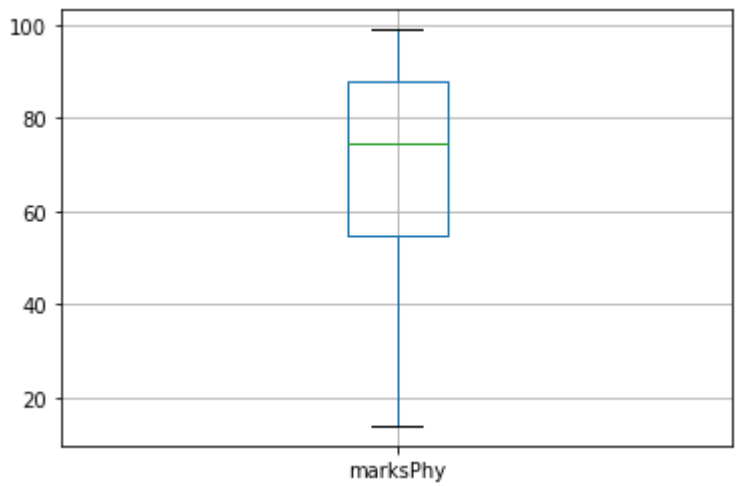
In [10]:
```
pd.options.mode.chained_assignment = None
```

```
for i in range(df.shape[0]):
    if(df.marksPhy[i]<0 or df.marksPhy[i]>100):
        df.marksPhy[i]=(np.nan)

for i in range(df.shape[0]):
    if(df.marksChem[i]<0 or df.marksChem[i]>100):
        df.marksChem[i]=(np.nan)

for i in range(df.shape[0]):
    if(df.marksEM1[i]<0 or df.marksEM1[i]>100):
        df.marksEM1[i]=(np.nan)

for i in range(df.shape[0]):
    if(df.marksELEC[i]<0 or df.marksELEC[i]>100):
        df.marksELEC[i]=(np.nan)

for i in range(df.shape[0]):
    if(df.marksSME[i]<0 or df.marksSME[i]>100):
        df.marksSME[i]=(np.nan)

for i in range(df.shape[0]):
    if(df.attendance[i]<0 or df.attendance[i]>100):
        df.attendance[i]=(np.nan)
```

Handling outliers

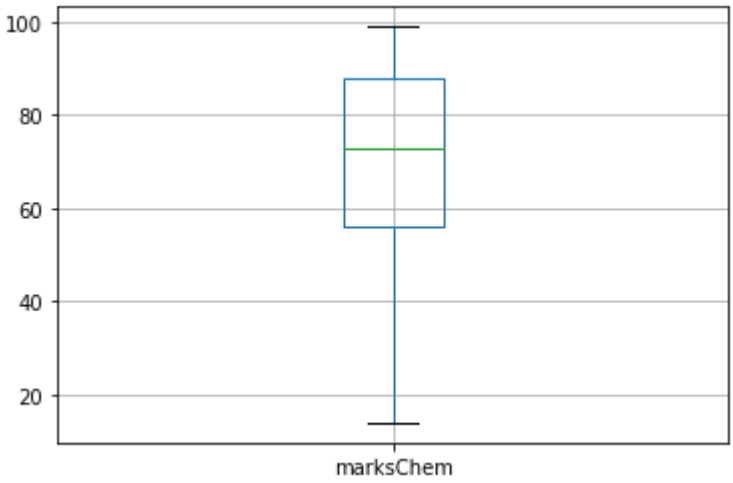
In [11]:
```
df.boxplot(column=['marksPhy'],return_type='axes')
```

Out[11]: <AxesSubplot:>



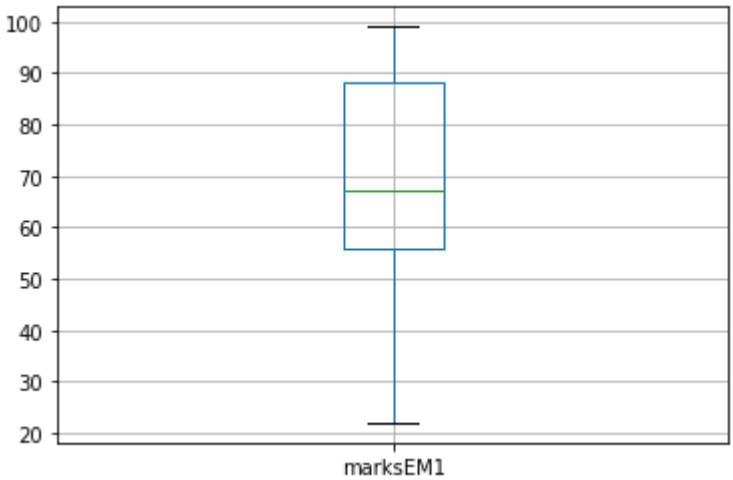
In [12]:
```
df.boxplot(column=['marksChem'],return_type='axes')
```

Out[12]: <AxesSubplot:>



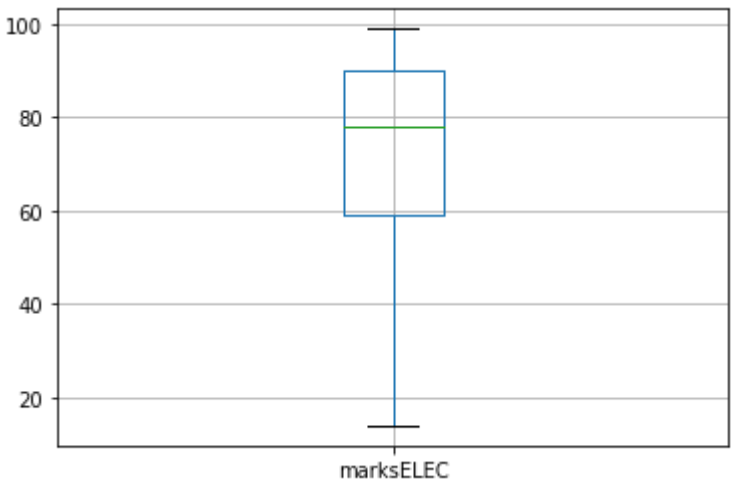
In [13]:
```
df.boxplot(column=['marksEM1'],return_type='axes')
```

Out[13]: <AxesSubplot:>



In [14]:
```
df.boxplot(column=['marksELEC'],return_type='axes')
```

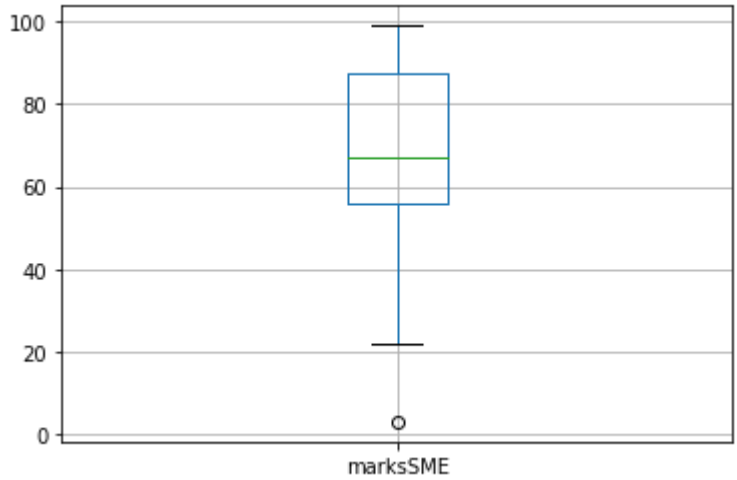
Out[14]: <AxesSubplot:>



marksSME column has outliers and is handeled by replcing records with median

In [15]:
```
df.boxplot(column=['marksSME'],return_type='axes')
```

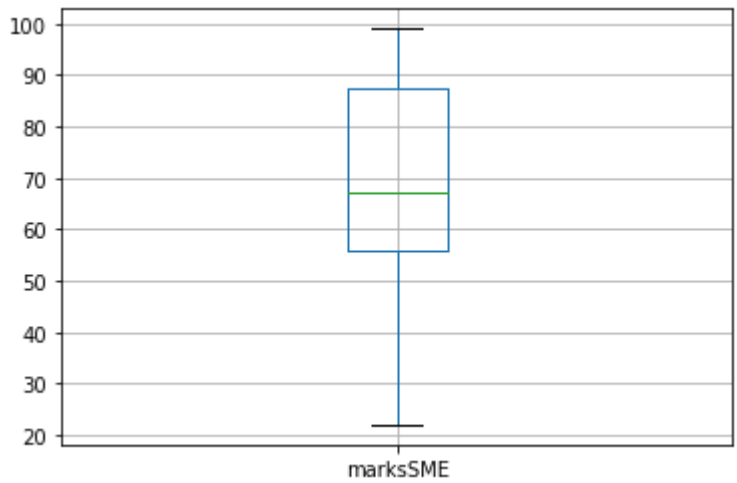
Out[15]: <AxesSubplot:>



In [16]:
```
Q1=df['marksSME'].quantile(0.25)
Q3=df['marksSME'].quantile(0.75)
IQR=Q3-Q1
Lower_Whisker=Q1-1.5*IQR
median = df.loc[df['marksSME']>Lower_Whisker, 'marksSME'].median()
df['marksSME']=np.where(df['marksSME']<Lower_Whisker,median,df['marksSME'])
# df.loc[df.marksSME > 75, 'Age'] = np.nan
# df.fillna(median,inplace=True)
#df=df[df['marksSME']>Lower_Whisker]
```

In [17]:
```
df.boxplot(column=['marksSME'],return_type='axes')
```

Out[17]: <AxesSubplot:>



Filling the null values with mean values

In [18]:
```
df['marksPhy'].fillna((df['marksPhy'].mean()),inplace=True)
df['marksChem'].fillna((df['marksChem'].mean()),inplace=True)
df['marksEM1'].fillna((df['marksEM1'].mean()),inplace=True)
df['marksELEC'].fillna((df['marksELEC'].mean()),inplace=True)
df['marksSME'].fillna((df['marksSME'].mean()),inplace=True)
```

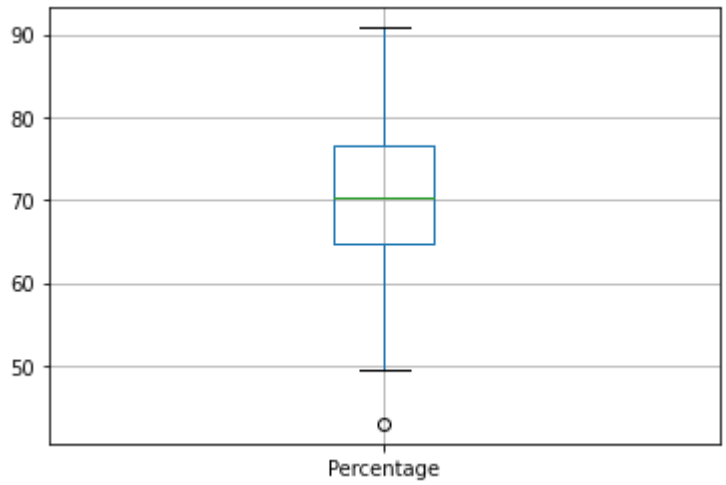
Calculating the percentage

In [19]:
```
df['Percentage']=(df['marksPhy']+df['marksChem']+df['marksEM1']+df['marksELEC
```

Percentage column has a outlier and is handeled by deleting that respective record

In [20]:
```
df.boxplot(column=['Percentage'],return_type='axes')
```

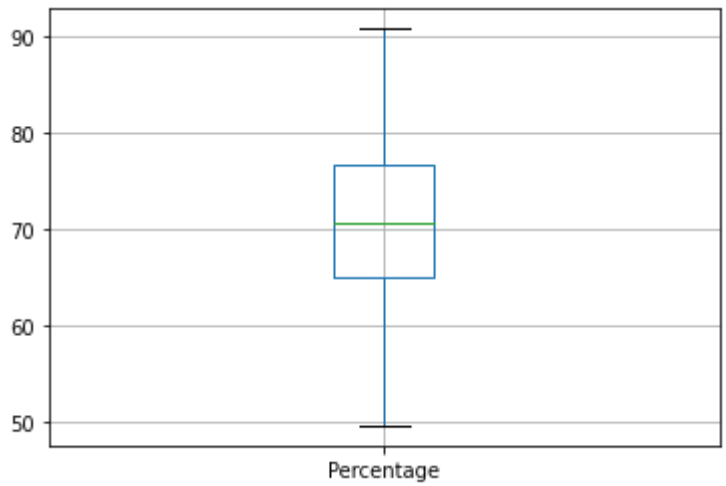
Out[20]: <AxesSubplot:>



In [21]:
```
Q1=df['Percentage'].quantile(0.25)
Q3=df['Percentage'].quantile(0.75)
IQR=Q3-Q1
Lower_Whisker=Q1-1.5*IQR
df=df[df['Percentage']>Lower_Whisker]
```

In [22]:
```
df.boxplot(column=['Percentage'],return_type='axes')
```

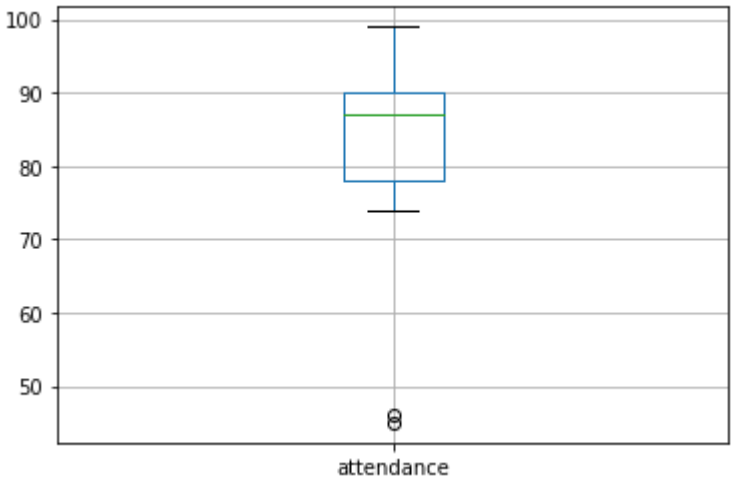
Out[22]: <AxesSubplot:>



Attendance column has outliers and is handeled by replcing records with median

In [23]:
```
df.boxplot(column=['attendance'],return_type='axes')
```

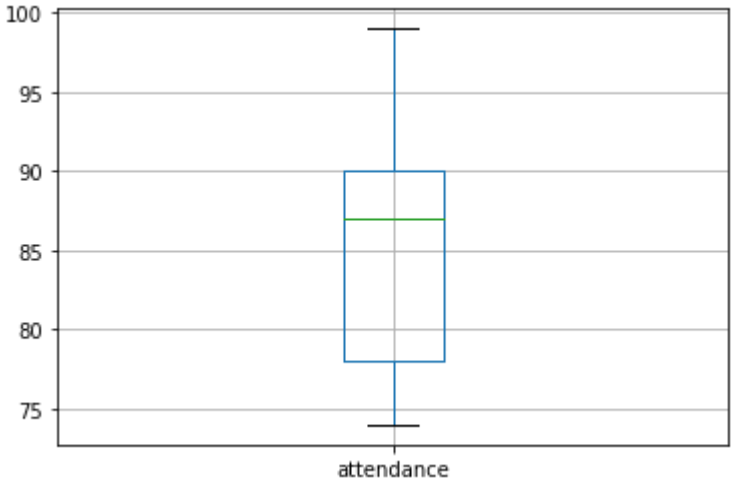
Out[23]: <AxesSubplot:>



In [24]:
```
Q1=df['attendance'].quantile(0.25)
Q3=df['attendance'].quantile(0.75)
IQR=Q3-Q1
Lower_Whisker=Q1-1.5*IQR
median = df.loc[df['attendance']>Lower_Whisker, 'attendance'].median()
df['attendance']=np.where(df['attendance']<Lower_Whisker,median,df['attendanc
```

In [25]:
```
df.boxplot(column=['attendance'],return_type='axes')
```

Out[25]: <AxesSubplot:>



In [26]:
```
df['attendance'].fillna((df['attendance'].mean()),inplace=True)
```

Using MinMaxScaler to scale the data

In [27]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [28]:
```
scaler = MinMaxScaler()
cols=['marksPhy','marksChem','marksEM1','marksELEC','marksSME']
scaler.fit(df[cols])
df[cols] = scaler.transform(df[cols])
df[cols].describe().loc[['min', 'max']]
```

Out[28]:

	marksPhy	marksChem	marksEM1	marksELEC	marksSME

	marksPhy	marksChem	marksEM1	marksELEC	marksSME
min	0.0	0.0	0.0	0.0	0.0
max	1.0	1.0	1.0	1.0	1.0

In [29]:
```
df.sample(10)
```

Out[29]:

	rollno	branch	marksPhy	marksChem	marksEM1	marksELEC	marksSME	Percentage	a
23	24	Comp	0.941176	0.858824	1.000000	0.611765	0.584416	82.600000	
70	71	ENTC	0.648718	1.000000	0.565789	0.482353	0.441558	69.028205	
3	4	IT	0.741176	0.764706	0.855263	0.623529	0.602814	75.883333	
84	85	ENTC	0.741176	0.952941	0.434211	0.894118	0.428571	74.600000	
80	81	Comp	0.470588	0.282353	0.447368	0.729412	0.441558	56.200000	
35	36	ENTC	0.494118	0.564706	1.000000	0.729412	0.602814	72.283333	
62	63	IT	0.648718	0.611765	0.628567	0.494118	0.602814	66.065755	
71	72	IT	0.588235	0.894118	0.710526	0.611765	0.844156	76.800000	
64	65	ENTC	0.648718	0.741176	0.000000	0.105882	0.441558	49.628205	
41	42	Comp	0.482353	0.611765	1.000000	0.752941	0.584416	73.000000	

In [30]:
```
df.to_csv("processed_student_data.csv")
```

In [ ]:
```

```