Mini project on Rainfall Data

Problem Statement: To predict which model best fits for the given dataset

Linear Regression

In [2]:

```
#importing the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
%matplotlib inline
```

Data Collection

In [3]:
```
df=pd.read_csv(r"C:\Users\rubin\Documents\district wise rainfall normal.csv")
df
```

Out[3]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2
...	...	...	...	...	...	...	...	...	...	...
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5

641 rows × 19 columns

Description about data

```
-The data is all about statewise and district wise  rainfall occured
during all the months.
-Number of columns are 19 and rows are 641
-The Annual Rainfall is recorded.
-And different months are taken rainfall is compared between those mo
nths.
-The rainfall is increasing monthly wise and decreasing vice versa.
```

Data Cleaning

In [4]:
```
df.head()
```

Out[4]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	O
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	32
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	455.6	30
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	454.8	27
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	16
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	20

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

Out[5]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5

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

Out[6]:
```
(641, 19)
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 641 entries, 0 to 640
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   STATE_UT_NAME  641 non-null    object 
 1   DISTRICT       641 non-null    object 
 2   JAN            641 non-null    float64
 3   FEB            641 non-null    float64
 4   MAR            641 non-null    float64
 5   APR            641 non-null    float64
 6   MAY            641 non-null    float64
 7   JUN            641 non-null    float64
 8   JUL            641 non-null    float64
 9   AUG            641 non-null    float64
 10  SEP            641 non-null    float64
 11  OCT            641 non-null    float64
 12  NOV            641 non-null    float64
 13  DEC            641 non-null    float64
 14  ANNUAL         641 non-null    float64
 15  Jan-Feb        641 non-null    float64
 16  Mar-May        641 non-null    float64
 17  Jun-Sep        641 non-null    float64
 18  Oct-Dec        641 non-null    float64
dtypes: float64(17), object(2)
memory usage: 95.3+ KB
```

Data Preprocessing

In [8]:

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

Out[8]:

```
STATE_UT_NAME    0
DISTRICT         0
JAN              0
FEB              0
MAR              0
APR              0
MAY              0
JUN              0
JUL              0
AUG              0
SEP              0
OCT              0
NOV              0
DEC              0
ANNUAL           0
Jan-Feb          0
Mar-May          0
Jun-Sep          0
Oct-Dec          0
dtype: int64
```

In [9]:

```
#From the above it is concluded that there are no null values in the given dat
```

In [10]:

```
#checking the number of columns in the data set
df.columns
```

Out[10]:

```
Index(['STATE_UT_NAME', 'DISTRICT', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',
       'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb',
       'Mar-May', 'Jun-Sep', 'Oct-Dec'],
      dtype='object')
```

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

Out[11]:
```
<bound method NDFrame.describe of                     STATE_UT_NAME        DIS
TRICT    JAN   FEB    MAR    APR
0    ANDAMAN And NICOBAR ISLANDS         NICOBAR  107.3  57.9   65.2  117.0
\
1    ANDAMAN And NICOBAR ISLANDS   SOUTH ANDAMAN   43.7  26.0   18.6   90.5
2    ANDAMAN And NICOBAR ISLANDS   N & M ANDAMAN   32.7  15.9    8.6   53.4
3              ARUNACHAL PRADESH           LOHIT   42.2  80.8  176.4  358.5
4              ARUNACHAL PRADESH      EAST SIANG   33.3  79.5  105.9  216.5
..                           ...             ...    ...   ...    ...    ...
636                       KERALA          IDUKKI   13.4  22.1   43.6  150.4
637                       KERALA        KASARGOD    2.3   1.0    8.4   46.9
638                       KERALA  PATHANAMTHITTA   19.8  45.2   73.9  184.9
639                       KERALA         WAYANAD    4.8   8.3   17.5   83.3
640                  LAKSHADWEEP     LAKSHADWEEP   20.8  14.7   11.8   48.9

       MAY    JUN     JUL    AUG    SEP    OCT    NOV    DEC  ANNUAL  Jan-Feb
0    358.5  295.5   285.0  271.9  354.8  326.0  315.2  250.9  2805.2    165.2
\
1    374.4  457.2   421.3  423.1  455.6  301.2  275.8  128.3  3015.7     69.7
2    343.6  503.3   465.4  460.9  454.8  276.1  198.6  100.0  2913.3     48.6
3    306.4  447.0   660.1  427.8  313.6  167.1   34.1   29.8  3043.8    123.0
4    323.0  738.3   990.9  711.2  568.0  206.9   29.5   31.7  4034.7    112.8
..     ...    ...     ...    ...    ...    ...    ...    ...     ...      ...
636  232.6  651.6   788.9  527.3  308.4  343.2  172.9   48.1  3302.5     35.5
637  217.6  999.6  1108.5  636.3  263.1  234.9   84.6   18.4  3621.6      3.3
638  294.7  556.9   539.9  352.7  266.2  359.4  213.5   51.3  2958.4     65.0
639  174.6  698.1  1110.4  592.9  230.7  213.1   93.6   25.8  3253.1     13.1
640  171.7  330.2   287.7  217.5  163.1  157.1  117.7   58.8  1600.0     35.5

     Mar-May  Jun-Sep  Oct-Dec
0      540.7   1207.2    892.1
1      483.5   1757.2    705.3
2      405.6   1884.4    574.7
3      841.3   1848.5    231.0
4      645.4   3008.4    268.1
..       ...      ...      ...
636    426.6   2276.2    564.2
637    272.9   3007.5    337.9
638    553.5   1715.7    624.2
639    275.4   2632.1    332.5
640    232.4    998.5    333.6

[641 rows x 19 columns]>
```

In [12]:

```
#To check data types
df.dtypes
```

Out[12]:

```
STATE_UT_NAME     object
DISTRICT          object
JAN              float64
FEB              float64
MAR              float64
APR              float64
MAY              float64
JUN              float64
JUL              float64
AUG              float64
SEP              float64
OCT              float64
NOV              float64
DEC              float64
ANNUAL           float64
Jan-Feb          float64
Mar-May          float64
Jun-Sep          float64
Oct-Dec          float64
dtype: object
```

In [13]:

```
#To check unique values
df.nunique()
```

Out[13]:

```
STATE_UT_NAME     35
DISTRICT         637
JAN              301
FEB              309
MAR              349
APR              372
MAY              457
JUN              547
JUL              570
AUG              569
SEP              543
OCT              507
NOV              349
DEC              263
ANNUAL           591
Jan-Feb          399
Mar-May          511
Jun-Sep          592
Oct-Dec          524
dtype: int64
```

In [14]:

```
df.duplicated().sum()
```

Out[14]:

```
0
```

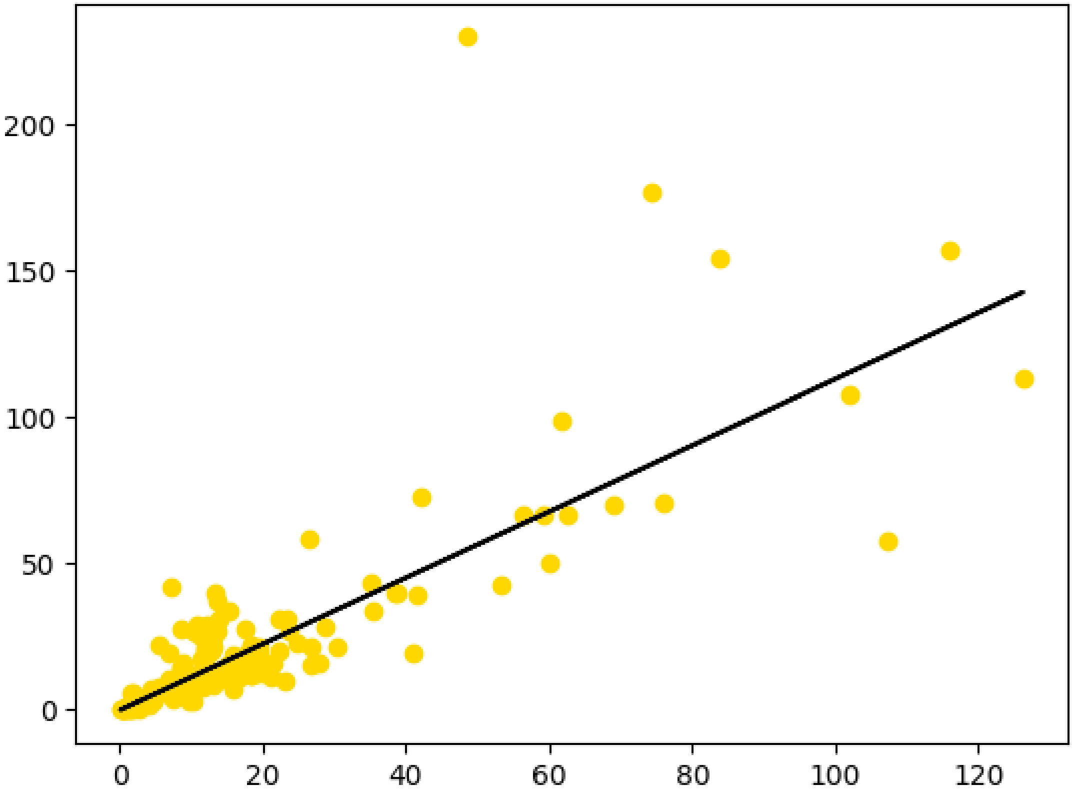
In [15]:
```
x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,1)
```

In [16]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.6375024350110565
```

For Linear Regression Model The accuracy is 63 percentage

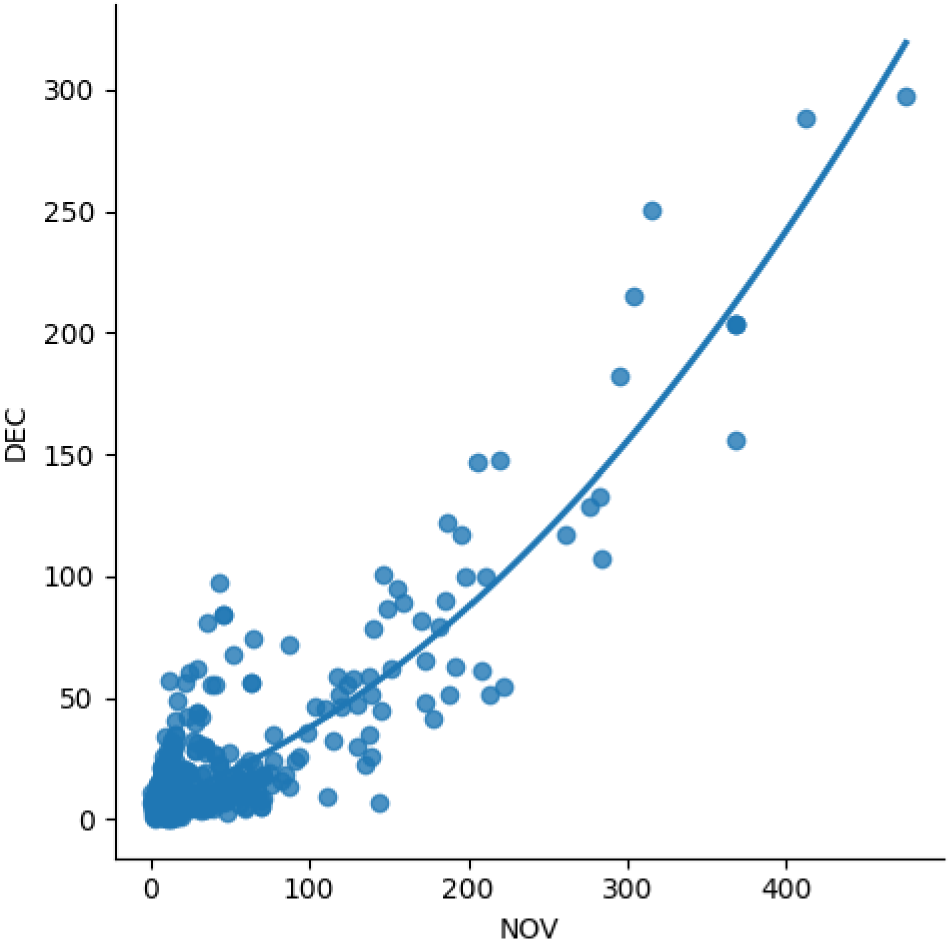
In [17]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='gold')
plt.plot(x_test,y_pred,color='black')
plt.show()
```



Data Visualization

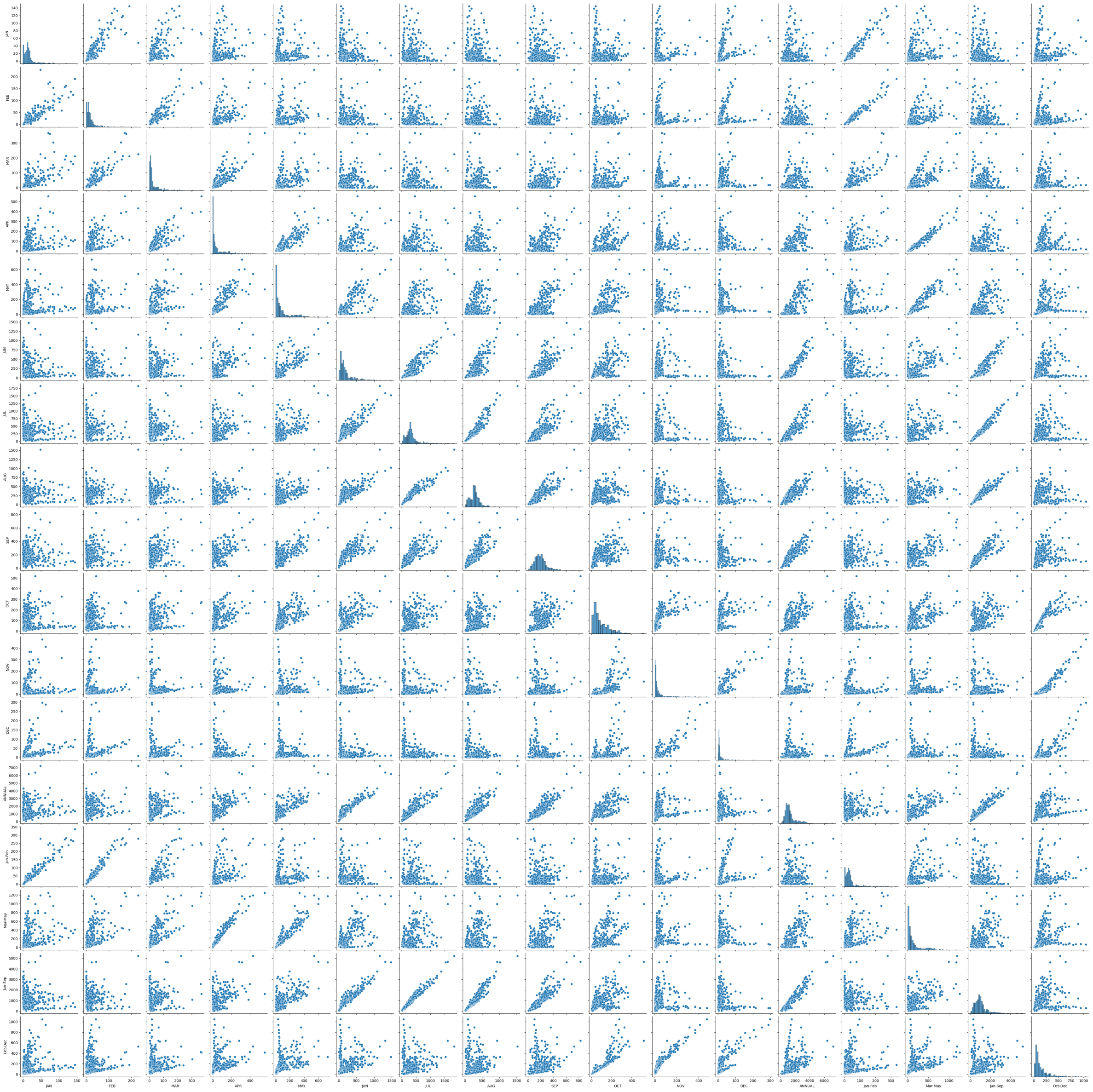
```
In [18]: sns.lmplot(x="NOV",y="DEC",data=df,order=2,ci=None)
```

```
Out[18]: <seaborn.axisgrid.FacetGrid at 0x1995bc45ff0>
```



In [19]:
```
sns.pairplot(df)
```

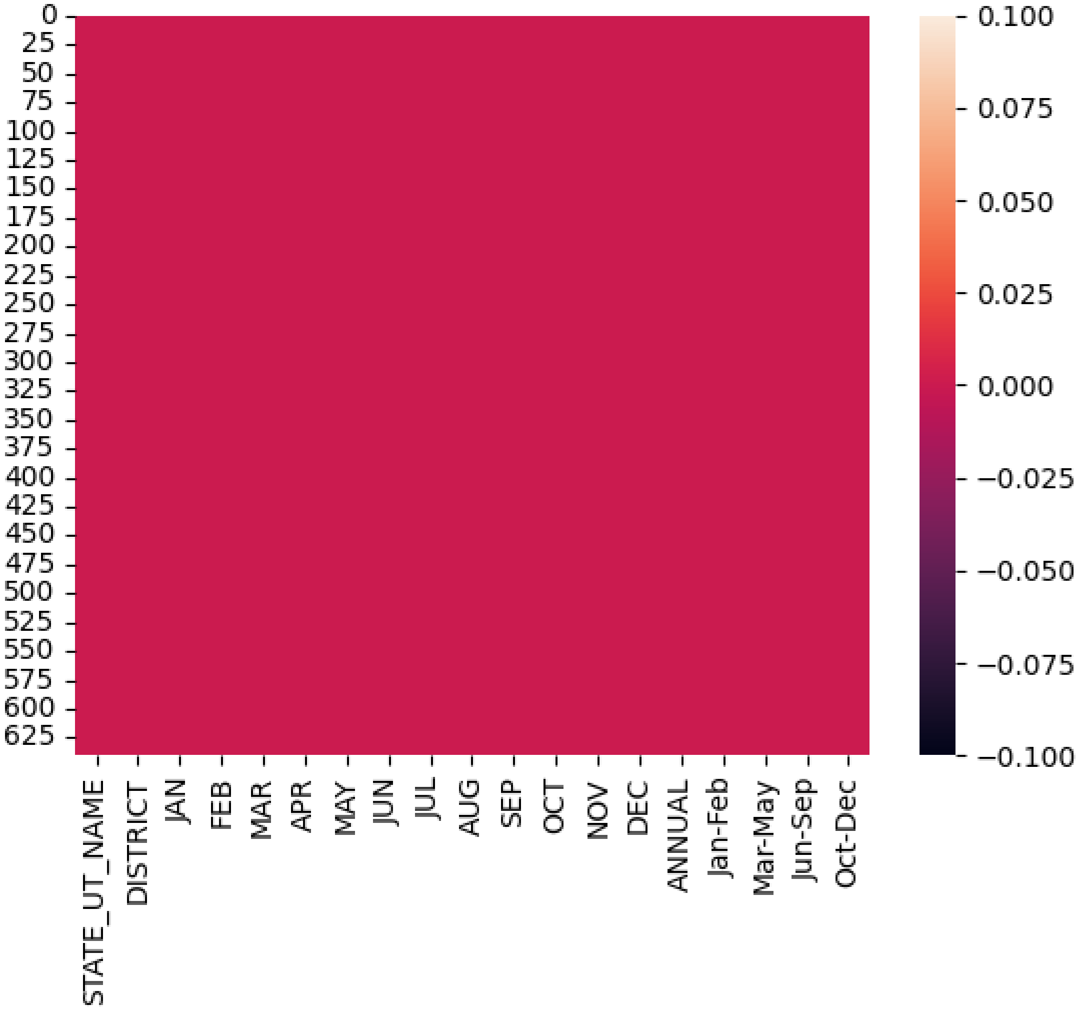
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x19921f7ae00>
```



-From the graphs we can say that the dataset is not normally distributed.

In [20]:
```
sns.heatmap(df.isnull())
```

Out[20]:
```
<Axes: >
```



In [21]:

```
convert={"STATE_UT_NAME":{"ANDAMAN And NICOBAR ISLANDS":1,"ARUNACHAL PRADESH":
df=df.replace(convert)
df
```

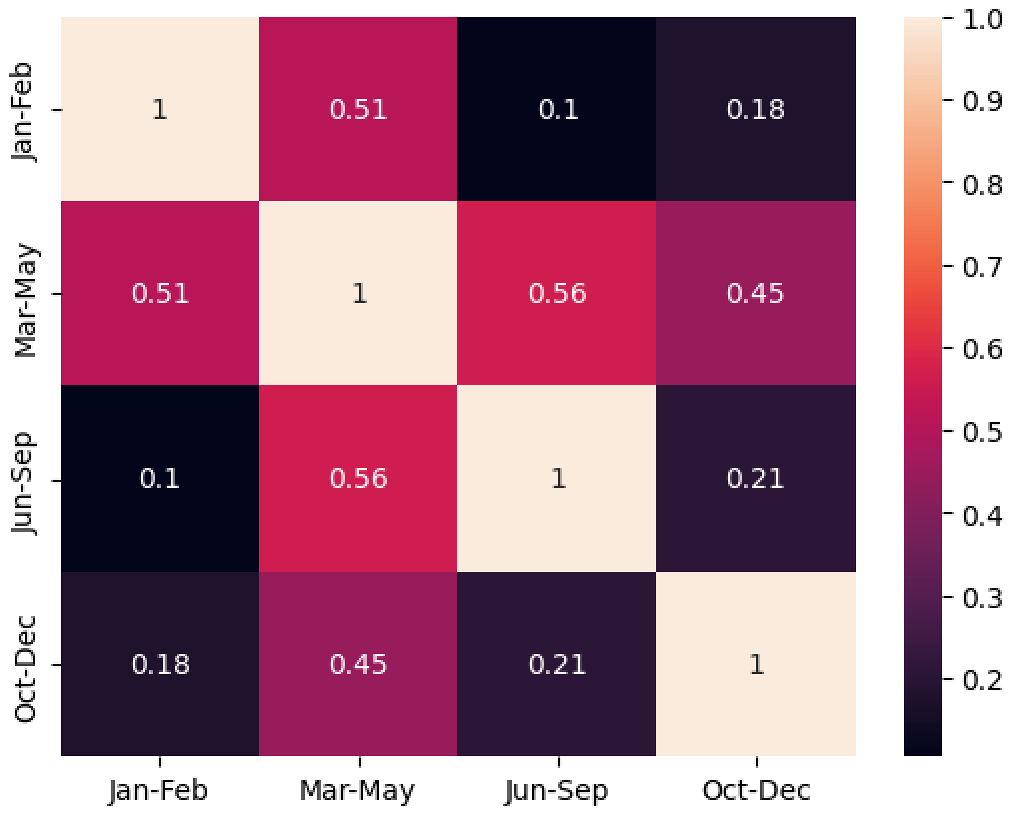
Out[21]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
0	1	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9
1	1	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1
2	1	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9
3	2	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8
4	2	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2
...	...	...	...	...	...	...	...	...	...	...
636	34	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3
637	34	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3
638	34	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7
639	34	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9
640	35	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5

641 rows × 19 columns

```
In [22]: df=df[['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec']]
         sns.heatmap(df.corr(),annot=True)
```

```
Out[22]: <Axes: >
```



```
In [23]: df['Jan-Feb'].value_counts()
```

```
Out[23]: Jan-Feb
         32.7     9
         18.2     5
         21.4     5
         0.8      5
         17.5     5
                 ..
         107.7    1
         87.0     1
         101.0    1
         135.2    1
         65.0     1
         Name: count, Length: 399, dtype: int64
```

ELASTIC NET

In [24]:
```
convert={"STATE_UT_NAME":{"ANDAMAN And NICOBAR ISLANDS":1,"ARUNACHAL PRADESH":
df=df.replace(convert)
df
```

Out[24]:

	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	165.2	540.7	1207.2	892.1
1	69.7	483.5	1757.2	705.3
2	48.6	405.6	1884.4	574.7
3	123.0	841.3	1848.5	231.0
4	112.8	645.4	3008.4	268.1
...	...	...	...	...
636	35.5	426.6	2276.2	564.2
637	3.3	272.9	3007.5	337.9
638	65.0	553.5	1715.7	624.2
639	13.1	275.4	2632.1	332.5
640	35.5	232.4	998.5	333.6

641 rows × 4 columns

In [25]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
#print(regr.coef_)
#print(regr.intercept_)
regr.score(x,y)
```

Out[25]:
```
0.7545114155274075
```

To the given data set for the Elastic Net Regression Model the accuracy is 75 percent.

Conclusion: For the given Rainfall data set we have performed different models and have got different accuracies.

Among all those the highest accuracy we got in Elastic Net Regression model.

So,Elastic net regression model best fits for the given dataframe

Implementing KMeans Clustering

In [45]:
```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

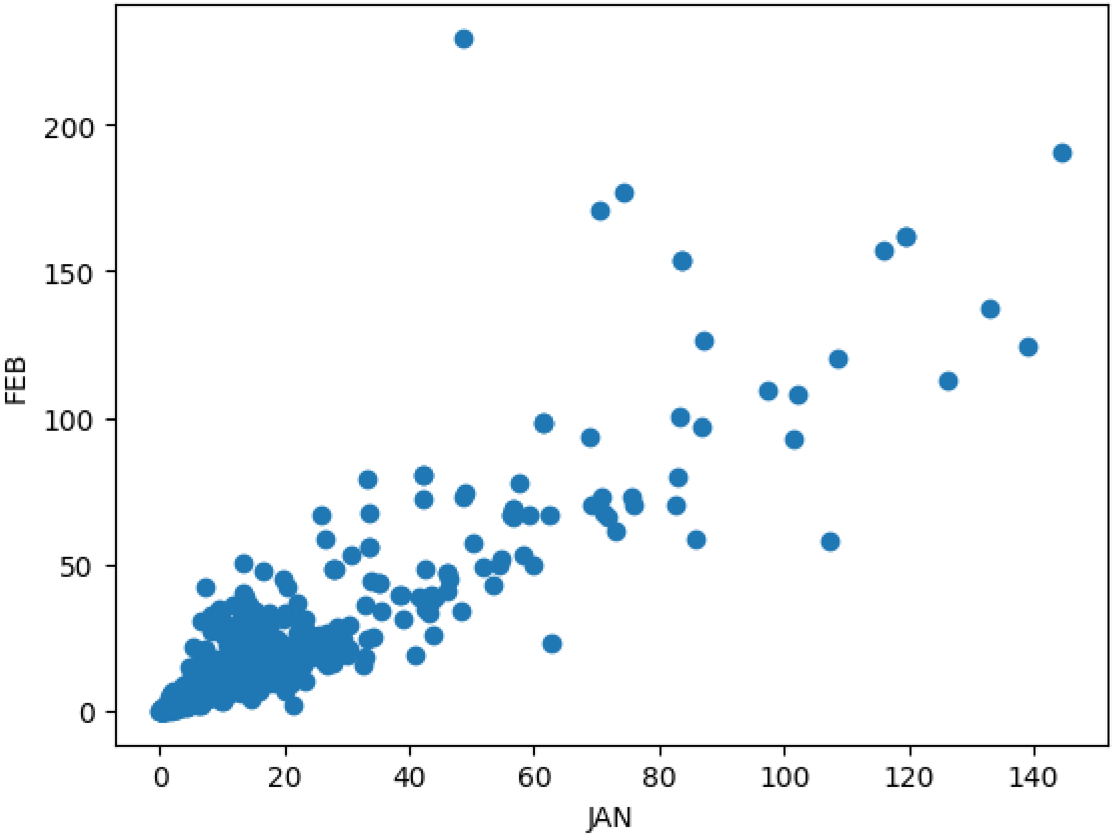
In [46]:
```
df=pd.read_csv(r"C:\Users\rubin\Documents\district wise rainfall normal.csv")
df.head()
```

Out[46]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	O
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	32
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	455.6	30
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	454.8	27
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	16
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	20

In [47]:
```
plt.scatter(df["JAN"],df["FEB"])
plt.xlabel("JAN")
plt.ylabel("FEB")
```

Out[47]:
```
Text(0, 0.5, 'FEB')
```



In [48]:
```
from sklearn.cluster import KMeans
```

In [49]:
```
km=KMeans()
km
```

Out[49]:
```
KMeans()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]:
```
y_predicted=km.fit_predict(df[["JAN","FEB"]])
y_predicted
```

```
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
```

Out[86]:
```
array([1, 4, 8, 1, 1, 4, 1, 1, 7, 7, 4, 6, 7, 4, 7, 4, 1, 4, 1, 0, 0, 8,
       0, 4, 0, 0, 0, 0, 2, 4, 0, 0, 0, 8, 0, 4, 0, 4, 0, 0, 0, 0, 0, 0,
       8, 8, 0, 4, 2, 2, 2, 0, 2, 0, 0, 7, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 8, 0, 2, 0, 0, 8, 0, 8, 8, 0, 0, 0, 0, 0, 2, 4,
       2, 2, 2, 2, 1, 4, 1, 4, 2, 2, 0, 0, 0, 8, 2, 2, 8, 0, 8, 0, 0, 0,
       2, 0, 0, 0, 0, 2, 0, 2, 0, 0, 2, 8, 0, 0, 0, 2, 2, 5, 2, 0, 0, 2,
       8, 2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 8, 2, 0, 2, 2, 2, 8, 8, 8, 8,
       2, 8, 2, 8, 8, 8, 8, 8, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 5, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 8, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 8, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 8, 2,
       2, 2, 2, 2, 2, 2, 8, 2, 2, 2, 2, 2, 2, 2, 8, 2, 2, 2, 2, 2, 2, 8,
       2, 8, 8, 8, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 8, 8, 8, 8, 8, 8, 8, 8,
       2, 2, 2, 2, 8, 8, 2, 2, 4, 1, 4, 4, 4, 4, 4, 1, 4, 4, 1, 8, 4, 4,
       2, 2, 8, 8, 2, 8, 2, 2, 8, 2, 8, 4, 8, 8, 2, 2, 2, 4, 2, 2, 4, 8,
       8, 8, 8, 8, 8, 8, 8, 8, 8, 2, 2, 4, 4, 8, 8, 8, 8, 4, 8, 2, 8, 4,
       8, 2, 2, 2, 8, 8, 1, 6, 1, 6, 6, 3, 1, 1, 1, 4, 1, 4, 6, 3, 3, 1,
       1, 2, 6, 1, 6, 4, 1, 8, 3, 1, 1, 1, 1, 1, 1, 3, 6, 3, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 2, 5, 2, 5, 2, 5, 5, 5, 5, 5, 5, 2, 5, 5, 2, 5, 5,
       5, 2, 5, 2, 5, 2, 5, 2, 2, 2, 2, 5, 5, 2, 2, 2, 5, 5, 5, 2, 5, 5,
       2, 5, 5, 2, 5, 2, 5, 2, 2, 5, 5, 5, 5, 5, 5, 8, 2, 8, 2, 8, 2, 8,
       2, 2, 8, 8, 8, 8, 2, 8, 8, 8, 8, 8, 2, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 2, 2, 5, 2, 2, 5, 2, 2, 2, 2, 2, 8, 5, 2, 5, 8, 2, 5, 2,
       2, 8, 8, 5, 5, 2, 2, 2, 5, 2, 2, 5, 5, 2, 2, 2, 2, 2, 5, 5, 2, 5,
       5, 5, 5, 5, 2, 5, 5, 2, 5, 5, 2, 2, 2, 8, 8, 2, 8, 8, 2, 8, 2, 4,
       2, 8, 2, 8, 2, 8, 2, 4, 2, 8, 2, 2, 8, 5, 2, 2, 8, 5, 8, 5, 8, 4,
       8, 8, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 0, 5, 2, 0, 5, 5, 5, 0, 5, 8, 0, 5,
       0, 5, 8])
```

In [51]:
```
df["Cluster"]=y_predicted
df.head()
```

Out[51]:

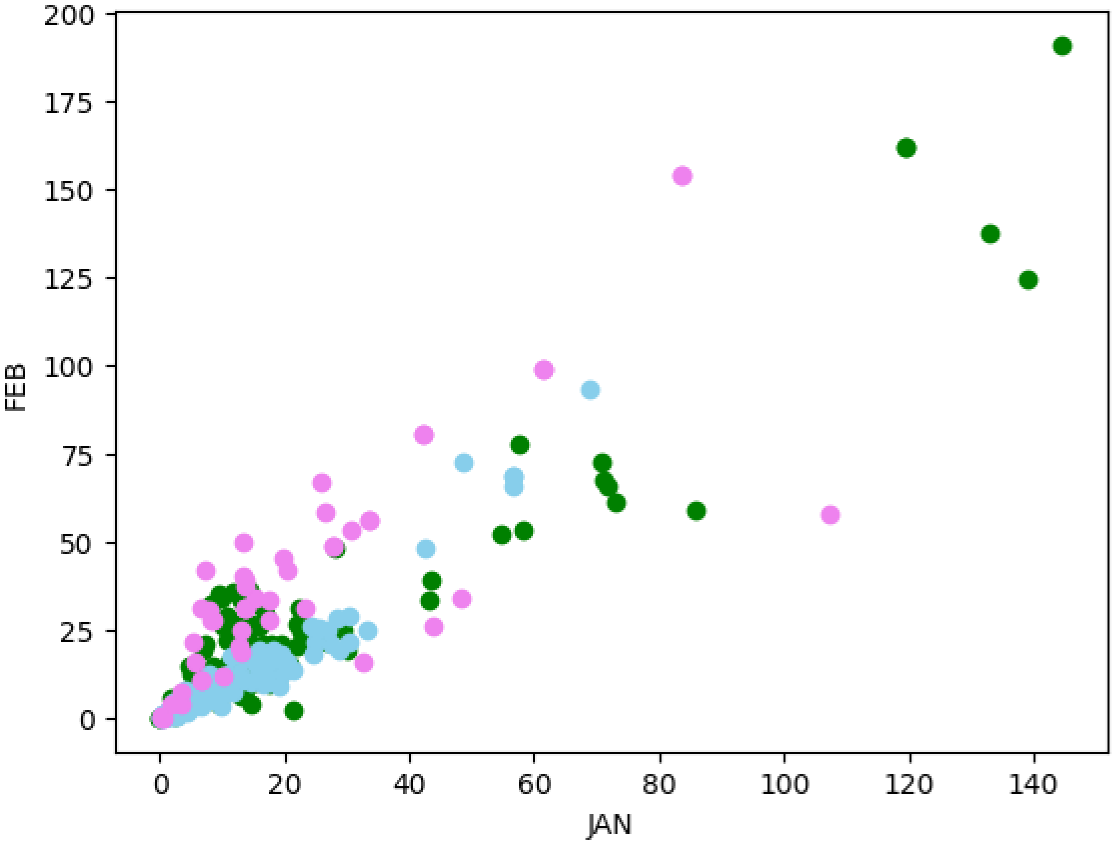
	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OC
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	326
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	455.6	301
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	454.8	276
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	167
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	206

In [87]:

```
df1=df[df.cluster==0]
df2=df[df.cluster==1]
df3=df[df.cluster==2]
plt.scatter(df1["JAN"],df1["FEB"],color="green")
plt.scatter(df2["JAN"],df2["FEB"],color="skyblue")
plt.scatter(df3["JAN"],df3["FEB"],color="violet")
plt.xlabel("JAN")
plt.ylabel("FEB")
```

Out[87]:

```
Text(0, 0.5, 'FEB')
```



In [70]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [71]:

```
Scaler=MinMaxScaler()
```

In [88]:

```
Scaler.fit(df[["JAN"]])
df["JAN"]=Scaler.transform(df[["JAN"]])
df.head()
```

Out[88]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	...
0	1	NICOBAR	0.742561	57.9	65.2	117.0	358.5	295.5	0.151108	271.9	...
1	1	SOUTH ANDAMAN	0.302422	26.0	18.6	90.5	374.4	457.2	0.226441	423.1	...
2	1	N & M ANDAMAN	0.226298	15.9	8.6	53.4	343.6	503.3	0.250815	460.9	...
3	2	LOHIT	0.292042	80.8	176.4	358.5	306.4	447.0	0.358426	427.8	...
4	2	EAST SIANG	0.230450	79.5	105.9	216.5	323.0	738.3	0.541259	711.2	...

5 rows × 21 columns

In [89]:

```
Scaler.fit(df[["FEB"]])
df["FEB"]=Scaler.transform(df[["FEB"]])
df.head()
```

Out[89]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
0	1	NICOBAR	0.742561	0.252178	65.2	117.0	358.5	295.5	0.151108	271.9
1	1	SOUTH ANDAMAN	0.302422	0.113240	18.6	90.5	374.4	457.2	0.226441	423.1
2	1	N & M ANDAMAN	0.226298	0.069251	8.6	53.4	343.6	503.3	0.250815	460.9
3	2	LOHIT	0.292042	0.351916	176.4	358.5	306.4	447.0	0.358426	427.8
4	2	EAST SIANG	0.230450	0.346254	105.9	216.5	323.0	738.3	0.541259	711.2

5 rows × 21 columns

In [74]:

```
km=KMeans()
km
```

Out[74]:

```
KMeans()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [90]:
```
y_predicted=km.fit_predict(df[["JAN","FEB"]])
y_predicted
```

```
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
```

Out[90]:
```
array([1, 3, 2, 3, 3, 3, 3, 3, 4, 4, 3, 1, 4, 3, 4, 3, 5, 3, 5, 6, 6, 6,
       6, 3, 6, 6, 6, 6, 8, 3, 2, 6, 6, 2, 6, 3, 6, 3, 6, 6, 6, 6, 6, 6,
       2, 8, 6, 3, 0, 8, 0, 6, 0, 6, 6, 4, 6, 6, 6, 3, 2, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 2, 2, 8, 6, 6, 2, 6, 2, 2, 6, 6, 6, 6, 6, 8, 3,
       8, 8, 8, 8, 5, 3, 5, 3, 8, 8, 6, 6, 6, 6, 8, 8, 8, 6, 8, 6, 6, 6,
       8, 6, 6, 6, 6, 0, 6, 0, 6, 6, 6, 2, 6, 6, 6, 6, 8, 0, 0, 6, 6, 8,
       6, 6, 6, 6, 6, 6, 0, 8, 8, 8, 8, 8, 2, 8, 6, 8, 8, 8, 2, 6, 2, 2,
       8, 8, 8, 2, 6, 8, 2, 2, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 8,
       8, 8, 8, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 2, 8, 8, 8, 0, 8,
       8, 8, 0, 8, 8, 2, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 2, 8,
       8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 2, 8, 8, 8, 8, 8, 8, 2,
       8, 8, 2, 2, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 2, 2, 2, 2, 2, 2, 8, 2,
       8, 8, 8, 8, 8, 2, 8, 8, 3, 5, 3, 3, 3, 3, 3, 5, 3, 3, 5, 2, 3, 3,
       8, 8, 8, 2, 8, 2, 8, 8, 2, 8, 2, 3, 8, 2, 8, 8, 8, 3, 8, 8, 3, 8,
       8, 8, 8, 8, 8, 8, 8, 8, 2, 8, 8, 5, 3, 2, 2, 2, 2, 3, 2, 8, 2, 3,
       2, 8, 8, 8, 2, 2, 5, 1, 5, 1, 1, 7, 5, 5, 5, 5, 5, 3, 1, 7, 7, 5,
       5, 0, 1, 5, 1, 3, 5, 2, 7, 5, 5, 5, 5, 5, 5, 7, 1, 7, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 8, 0, 0, 0, 8, 0, 0, 8, 8, 8, 0, 0, 8, 8, 8, 0, 0, 0, 8, 0, 0,
       8, 8, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 2, 8, 8, 8, 2, 8, 2,
       8, 8, 2, 2, 2, 2, 8, 2, 2, 2, 2, 2, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 8, 8, 0, 8, 8, 0, 8, 8, 0, 8, 0, 8, 0, 8, 0, 8, 8, 0, 8,
       8, 8, 2, 0, 0, 8, 8, 0, 0, 0, 0, 0, 0, 8, 0, 8, 8, 8, 0, 0, 8, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 2, 2, 8, 2, 2, 0, 2, 8, 3,
       0, 2, 8, 2, 8, 2, 8, 3, 8, 2, 8, 8, 8, 0, 8, 8, 2, 0, 2, 0, 2, 3,
       2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 8, 6, 0, 0, 0, 6, 0, 2, 6, 0,
       6, 0, 2])
```

In [91]:
```
df["New cluster"]=y_predicted
df.head()
```

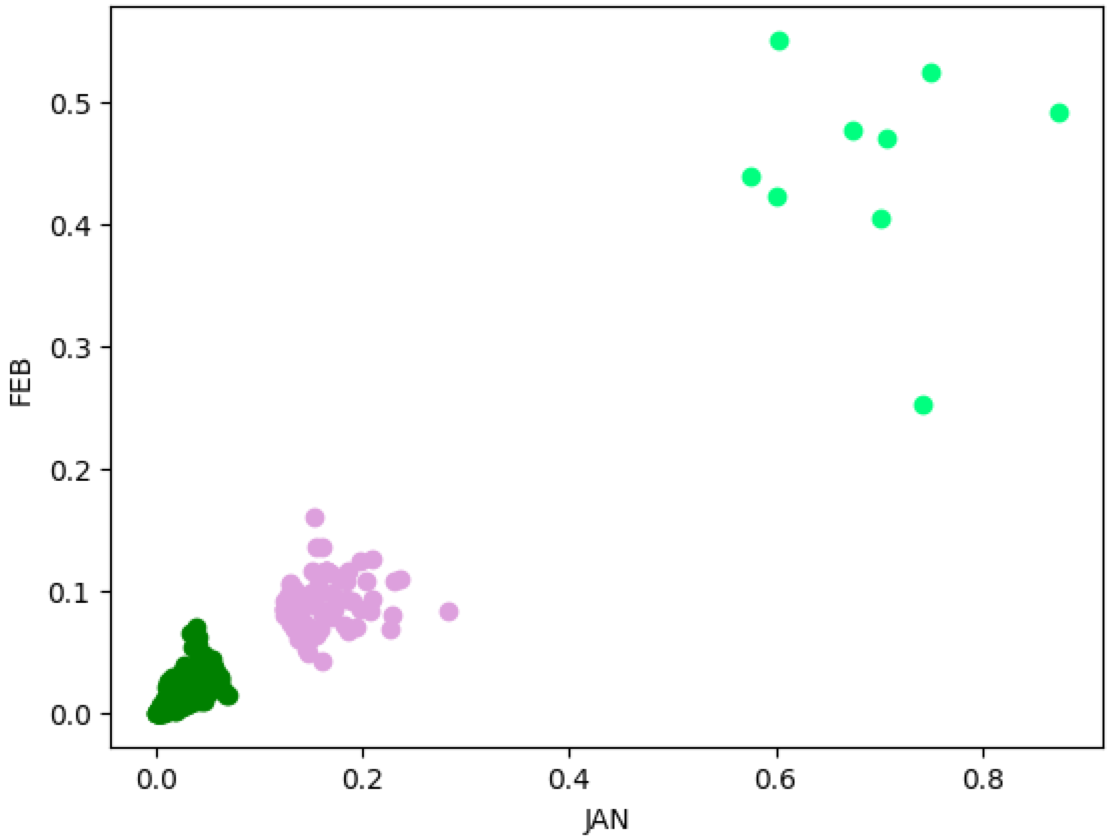
Out[91]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
0	1	NICOBAR	0.742561	0.252178	65.2	117.0	358.5	295.5	0.151108	271.9
1	1	SOUTH ANDAMAN	0.302422	0.113240	18.6	90.5	374.4	457.2	0.226441	423.1
2	1	N & M ANDAMAN	0.226298	0.069251	8.6	53.4	343.6	503.3	0.250815	460.9
3	2	LOHIT	0.292042	0.351916	176.4	358.5	306.4	447.0	0.358426	427.8
4	2	EAST SIANG	0.230450	0.346254	105.9	216.5	323.0	738.3	0.541259	711.2

5 rows × 21 columns

```
In [92]: df1=df[df["New cluster"]==0]
         df2=df[df["New cluster"]==1]
         df3=df[df["New cluster"]==2]
         plt.scatter(df1["JAN"],df1["FEB"],color="green")
         plt.scatter(df2["JAN"],df2["FEB"],color="SpringGreen")
         plt.scatter(df3["JAN"],df3["FEB"],color="plum")
         plt.xlabel("JAN")
         plt.ylabel("FEB")
```

```
Out[92]: Text(0, 0.5, 'FEB')
```



```
In [93]: km.cluster_centers_
```

```
Out[93]: array([[0.02562018, 0.01641456],
                [0.6922722 , 0.4476868 ],
                [0.16037197, 0.08769055],
                [0.28036332, 0.20850392],
                [0.49896194, 0.77090592],
                [0.45076817, 0.31008711],
                [0.0882526 , 0.1241017 ],
                [0.8893887 , 0.67704704],
                [0.09732043, 0.05072674]])
```

In [94]:
```
convert={"STATE_UT_NAME":{"ANDAMAN And NICOBAR ISLANDS":1,"ARUNACHAL PRADESH":
df=df.replace(convert)
df
```

Out[94]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JU
0	1	NICOBAR	0.742561	0.252178	65.2	117.0	358.5	295.5	0.1511
1	1	SOUTH ANDAMAN	0.302422	0.113240	18.6	90.5	374.4	457.2	0.2264
2	1	N & M ANDAMAN	0.226298	0.069251	8.6	53.4	343.6	503.3	0.2508
3	2	LOHIT	0.292042	0.351916	176.4	358.5	306.4	447.0	0.3584
4	2	EAST SIANG	0.230450	0.346254	105.9	216.5	323.0	738.3	0.5412
...	...	...	...	...	...	...	...	...	
636	34	IDUKKI	0.092734	0.096254	43.6	150.4	232.6	651.6	0.4296
637	34	KASARGOD	0.015917	0.004355	8.4	46.9	217.6	999.6	0.6062
638	34	PATHANAMTHITTA	0.137024	0.196864	73.9	184.9	294.7	556.9	0.2919
639	34	WAYANAD	0.033218	0.036150	17.5	83.3	174.6	698.1	0.6073
640	35	LAKSHADWEEP	0.143945	0.064024	11.8	48.9	171.7	330.2	0.1526

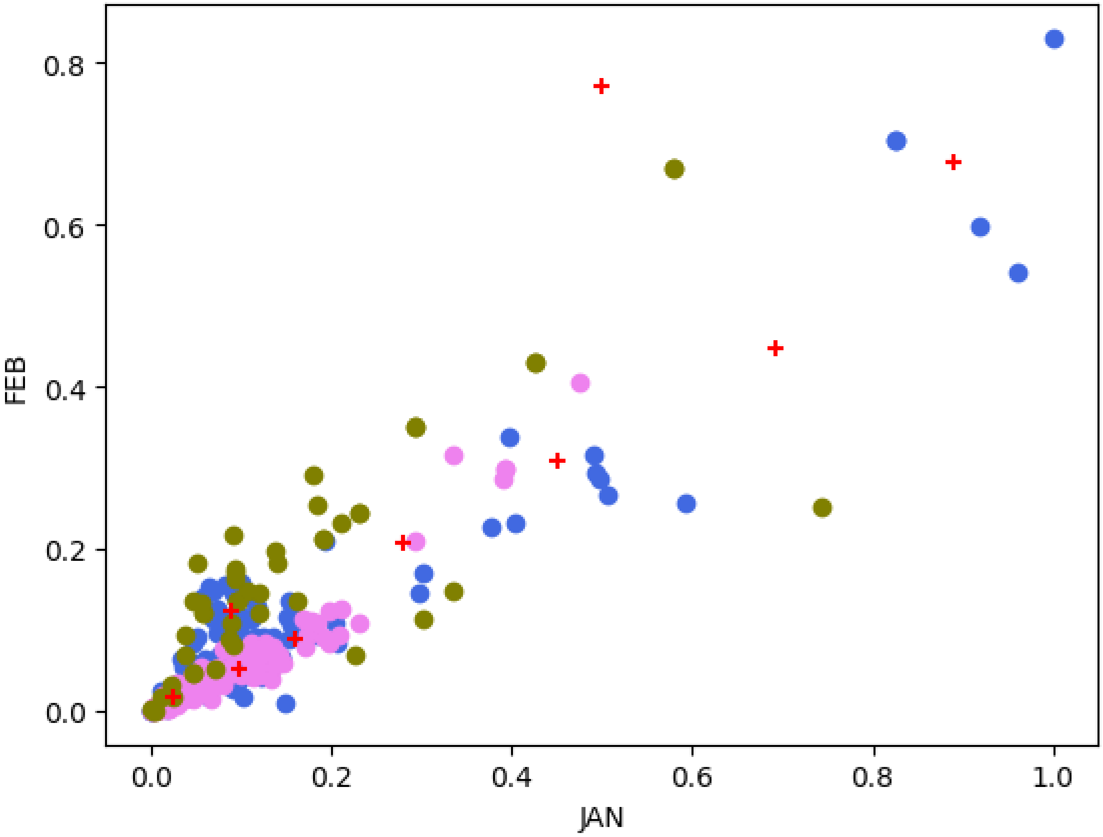
641 rows × 21 columns

In [95]:

```
df1=df[df.cluster==0]
df2=df[df.cluster==1]
df3=df[df.cluster==2]
plt.scatter(df1["JAN"],df1["FEB"],color="royalblue")
plt.scatter(df2["JAN"],df2["FEB"],color="violet")
plt.scatter(df3["JAN"],df3["FEB"],color="olive")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="red",mark
plt.xlabel("JAN")
plt.ylabel("FEB")
```

Out[95]:

```
Text(0, 0.5, 'FEB')
```

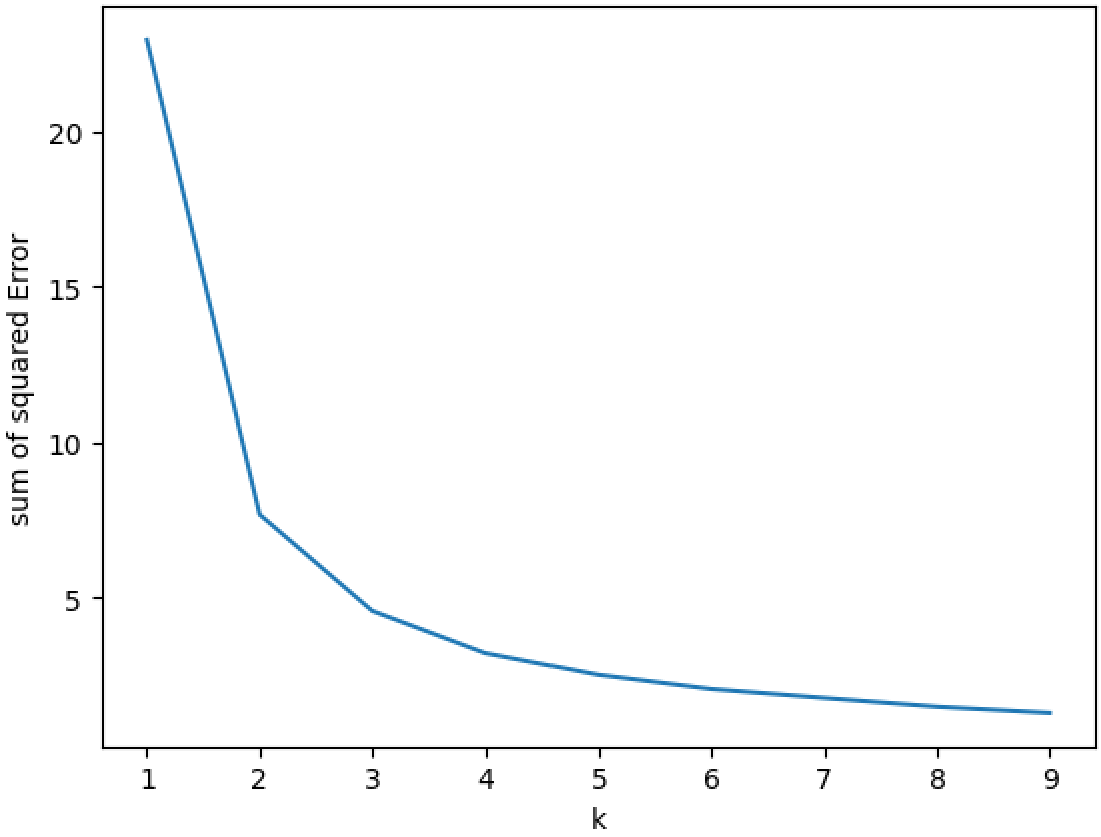


In [96]:
```
k_rng=range(1,10)
sse=[]
for k in k_rng:
    km=KMeans(n_clusters=k)
    km.fit(df[["JAN","FEB"]])
    sse.append(km.inertia_)
sse
```

```
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  warnings.warn(
Out[96]: [22.959082397157193,
          7.673772524692251,
          4.561202317275745,
          3.2008115349007453,
          2.4995513590096348,
          2.040669019576882,
          1.7526832226825582,
          1.4704410371248227,
          1.2717837227975586]
```

```
In [97]: plt.plot(k_rng,sse)
         plt.xlabel("k")
         plt.ylabel("sum of squared Error")
```

```
Out[97]: Text(0, 0.5, 'sum of squared Error')
```



For the given dataset we have performed KMeans cluster model.Based on the given data set we have grouped into different clusters.

```
In [ ]:
```