Mini project on Rainfall

Problem statement:Rain Dataset is to predict whether or not it will rain tomorrow. The Dataset contains about 114(1901-2015)years of daily weather observations of different locations and different districts in India.

Here we have to predict two things:

a) Design a predictive model with the use of machine learning algorithms in which district contains more rainfall.

b) Design a predictive model with the us of all models to predict how much rainfall could be there.

In [70]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing,svm
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
```

1.Data Collection: Read the data

In [71]:

```
df=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\Downloads\rain fall in India.csv")
df
```

Out[71]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	O
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	38
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	19
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	18
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	22
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	26
...	...	...	...	...	...	...	...	...	...	...	...	...
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	11
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	14
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	7
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	16
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	16

4116 rows × 19 columns

In [72]:

```
df=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\Downloads\district wise rainfall.csv")
df
```

Out[72]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	A
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	27
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	42
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	46
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	42
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	71
...	...	...	...	...	...	...	...	...	...	
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	52
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	63
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	35
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	59
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	21

641 rows × 19 columns

Discription:

1.Rain Dataset is to predict whether or not it will rain tomorrow. The Dataset contains about 114(1901-2015)years of daily weather observations of different locations and different districts in India.
2.Data- The data set contains monthly wise and yearly report of state and district rainfall.
3.Rainfall -The amount of rainfall recorded monthly wise and calculated annually.
4.Sunshine -The number of hours of bright sunshine in the data
5.Here I observe the data monthly wise the rainfall will change gradually
6.Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere at a given place and how the atmosphere will change.

2. Data cleaning and preprocessing:

In [73]:

```
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
```

In [74]:

```
df.head()
```

Out[74]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8
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
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
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0

In [75]:

```
df.tail()
```

Out[75]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AU
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217

In [76]:

```
df.shape
```

Out[76]:

```
(641, 19)
```

In [77]:

```
df.describe()
```

Out[77]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.033697
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.364643
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.600000
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.400000
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.700000
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.800000
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.900000

In [78]:

```
df.columns
```

Out[78]:

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

To find duplicated and null values

In [79]:

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

Out[79]:

```
0
```

In [80]:

```
df.isna().sum()
```

Out[80]:

```
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 [81]:

```
df.dtypes
```

Out[81]:

```
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 [82]:

```
df.nunique()
```

Out[82]:

```
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 [83]:

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

Out[83]:

```
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
```

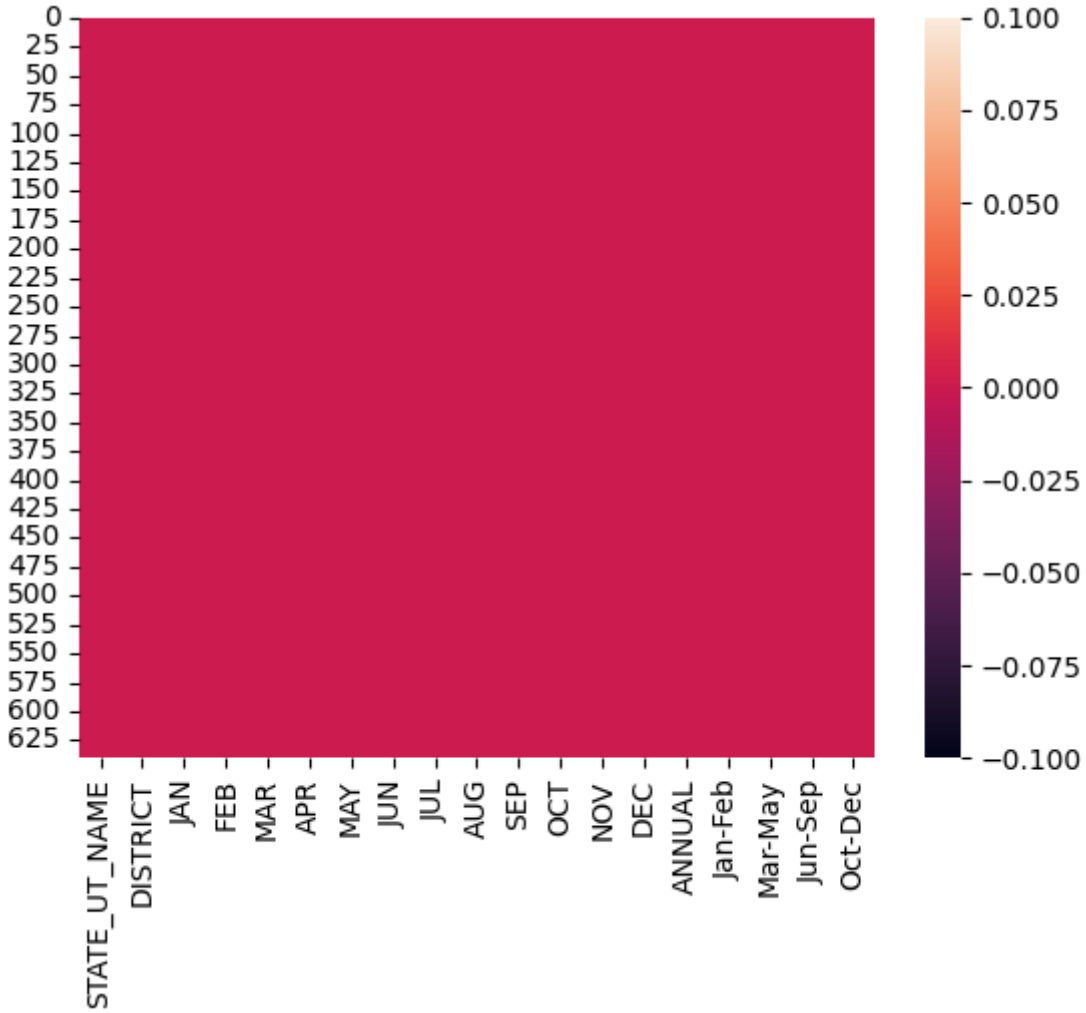
Type *Markdown* and LaTeX: α^2

In [84]:

```
sns.heatmap(df.isnull())
```

Out[84]:

<Axes: >



Type *Markdown* and LaTeX: α^2

We can clearly see that we have successfully treated all the null values and we dont have any null values in the dataset

split the data

In [85]:

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

In [86]:

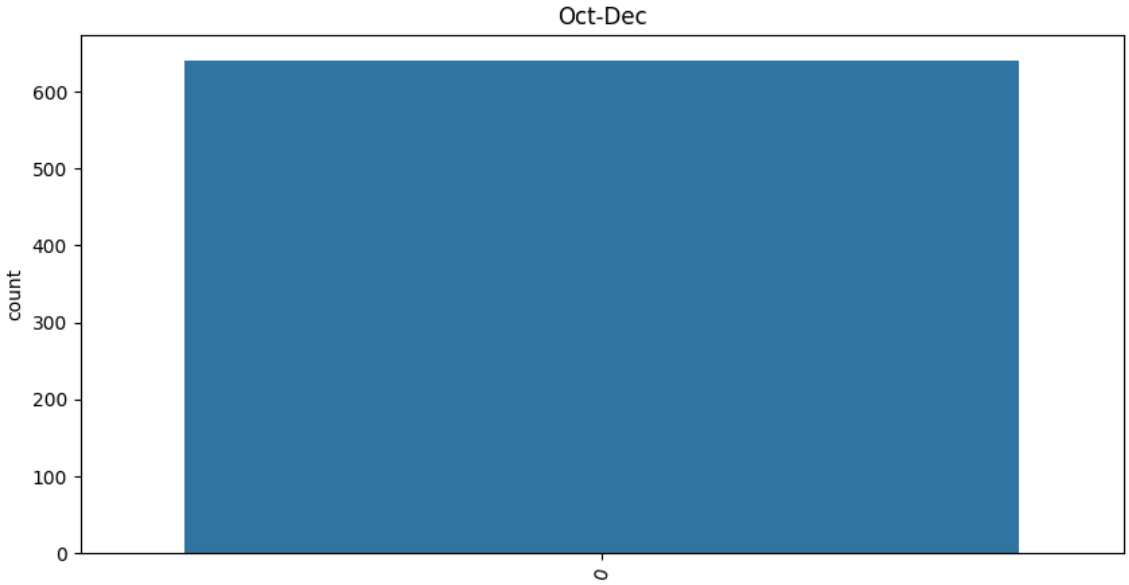
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.6156151436220155
```

3.Data Visualization:

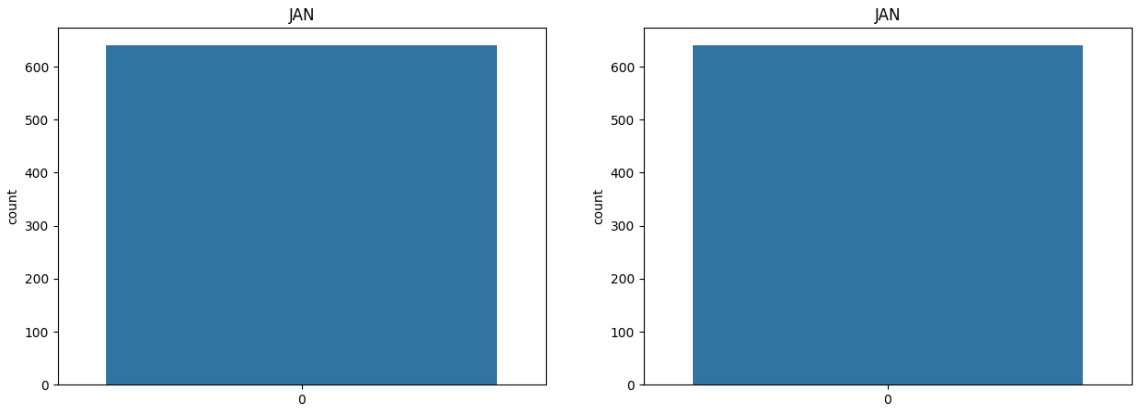
In [87]:

```
plt.figure(figsize=(10,5))
sns.countplot(df["Jan-Feb"])
plt.title("Oct-Dec")
plt.xticks(rotation=75)
plt.show()
```



In [88]:

```
plt.figure(figsize=(15,5))
plt.subplot(1, 2, 1)
plt.title('JAN')
sns.countplot(df['FEB'])
plt.subplot(1,2,2)
plt.title('JAN')
sns.countplot(df['FEB'])
plt.show()
```



In [ ]:

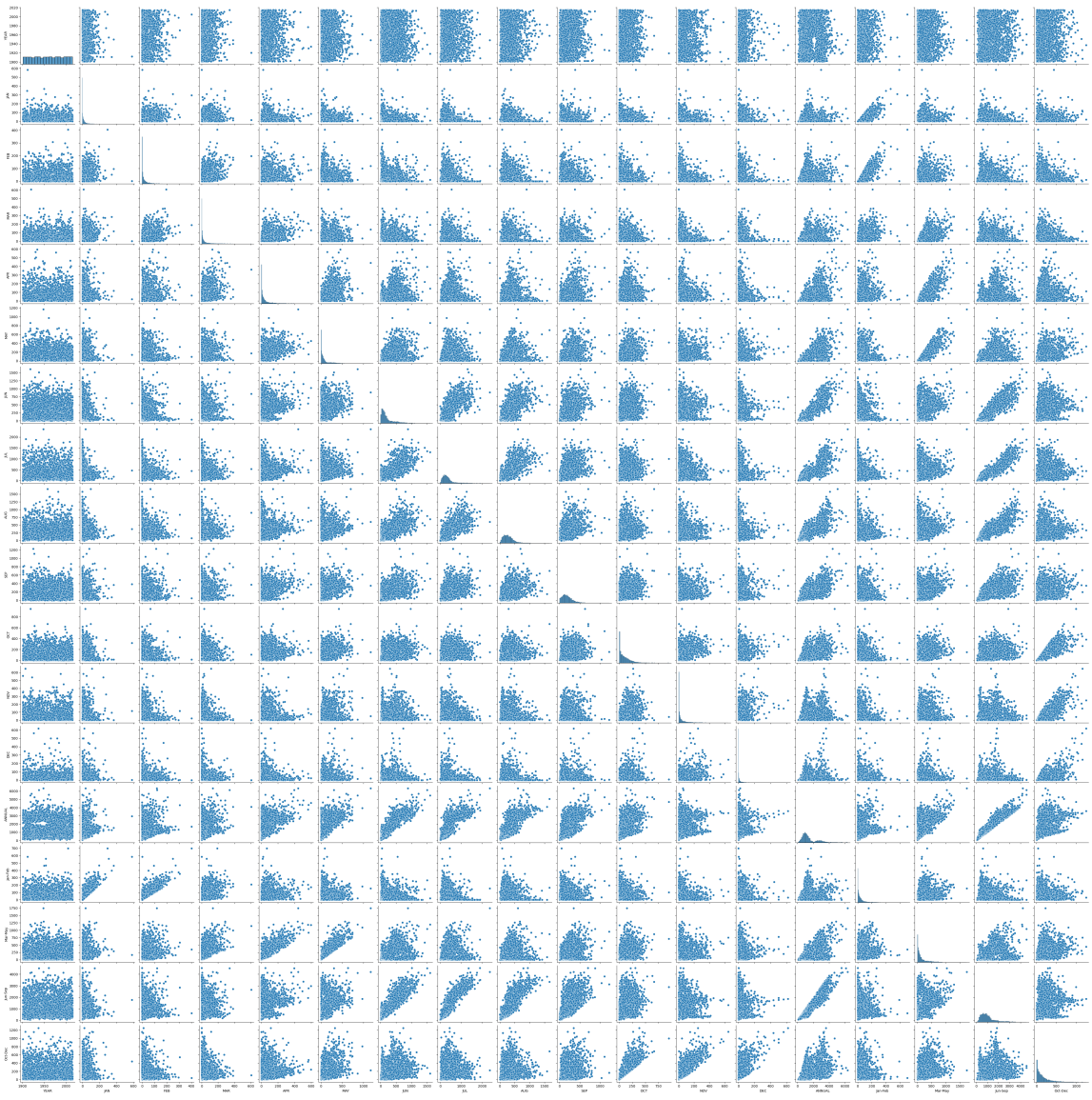
```
```

In [115]:

```
sns.pairplot(df)
```

Out[115]:

```
<seaborn.axisgrid.PairGrid at 0x209b8777880>
```



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

2.Jan month is normally distributed.

3.Feb is slightly skewed.

4.Anuall is more skewed.

5.Jan-Feb is not normally distributed is more skewed.

6.Mar-May is slightly skewed and Jun-Sep is normally distributed.

7.Mar,Aprl,May,Jun,Jul has no skewness however the data is not distributed uniformly.

4.Data modelling:Using different models

LinearRegression

In [90]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)
```

In [91]:

```
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
```

Out[91]:

```
LinearRegression()
```

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 [92]:

```
score=regr.score(x_test,y_test)
print(score)
```

```
0.7667452369097906
```

In [93]:

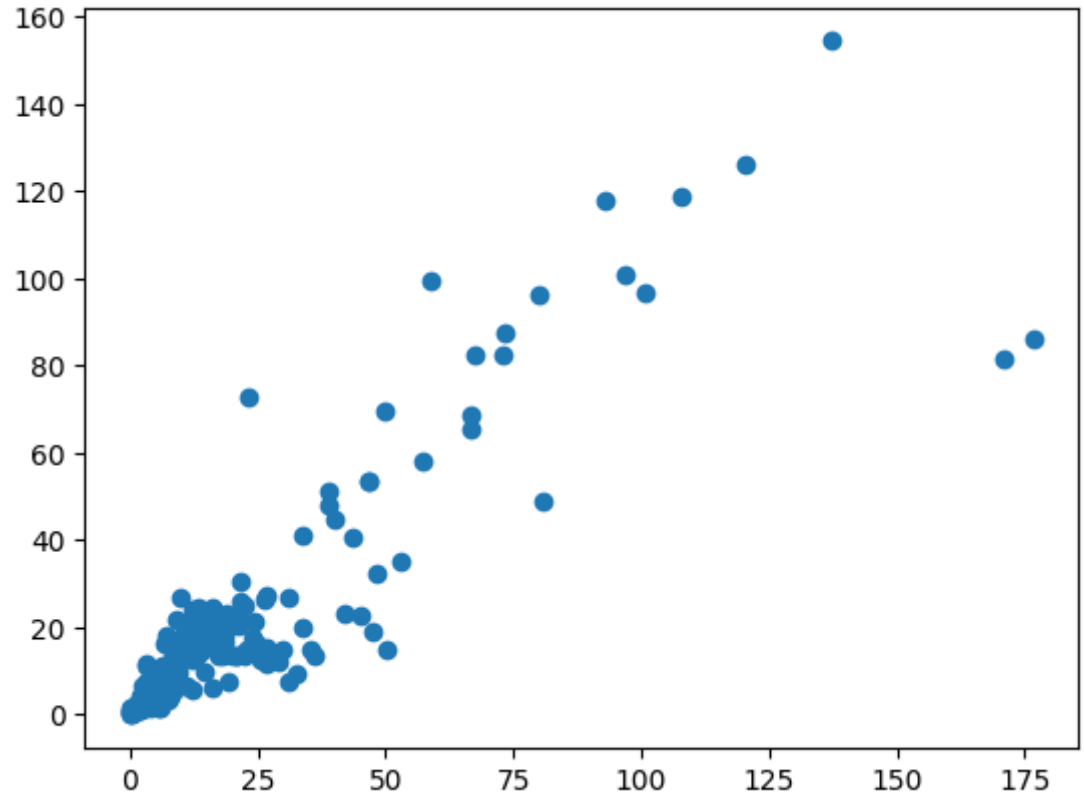
```
predictions=regr.predict(x_test)
```

In [94]:

```
plt.scatter(y_test,predictions)
```

Out[94]:

```
<matplotlib.collections.PathCollection at 0x209a35e2fb0>
```



In [95]:

```
x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,1)
df.dropna(inplace=True)
```

In [96]:

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

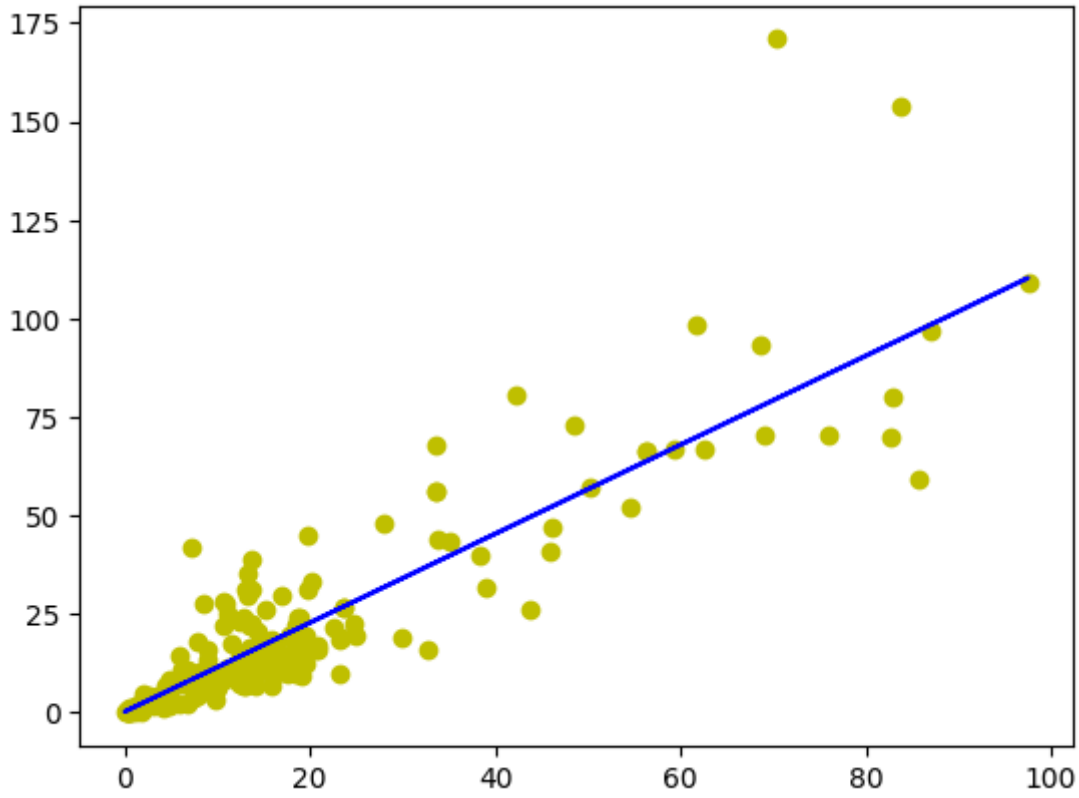
Out[96]:

```
LinearRegression()
```

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 [97]:

```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



Decision Tree

In [116]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [117]:

```
df=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\Downloads\rain fall in India.csv")
df
```

Out[117]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	O
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	38
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	19
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	18
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	22
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	26
...	...	...	...	...	...	...	...	...	...	...	...	...
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	11
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	14
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	7
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	16
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	16

4116 rows × 19 columns

In [118]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4116 entries, 0 to 4115
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   SUBDIVISION  4116 non-null   object 
 1   YEAR         4116 non-null   int64  
 2   JAN          4112 non-null   float64
 3   FEB          4113 non-null   float64
 4   MAR          4110 non-null   float64
 5   APR          4112 non-null   float64
 6   MAY          4113 non-null   float64
 7   JUN          4111 non-null   float64
 8   JUL          4109 non-null   float64
 9   AUG          4112 non-null   float64
 10  SEP          4110 non-null   float64
 11  OCT          4109 non-null   float64
 12  NOV          4105 non-null   float64
 13  DEC          4106 non-null   float64
 14  ANNUAL       4090 non-null   float64
 15  Jan-Feb      4110 non-null   float64
 16  Mar-May      4107 non-null   float64
 17  Jun-Sep      4106 non-null   float64
 18  Oct-Dec      4103 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

In [119]:

```
df['JAN'].value_counts()
```

Out[119]:

```
JAN
0.0      608
0.1      133
0.2      101
0.3       62
0.4       56
        ...
126.8      1
81.4       1
53.3       1
112.1      1
83.3       1
Name: count, Length: 802, dtype: int64
```

In [120]:

```
df['FEB'].value_counts()
```

Out[120]:

```
FEB
0.0      645
0.1      134
0.2       84
0.5       59
0.4       53
        ...
72.0       1
100.5      1
120.7      1
91.5       1
114.9      1
Name: count, Length: 898, dtype: int64
```

In [ ]:

```

```

In [121]:

```
x=["YEAR"]
y=["YES","NO"]
all_inputs=df[x]
all_classes=df["SUBDIVISION"]
```

In [122]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.5)
```

In [123]:

```
clf=DecisionTreeClassifier(random_state=0)
```

In [124]:

```
clf.fit(x_train,y_train)
```

Out[124]:

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [125]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
0.0
```

K-Means Cluster

In [126]:

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

In [127]:

```
df=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\Downloads\rain fall in India.csv")
df.head()
```

Out[127]:

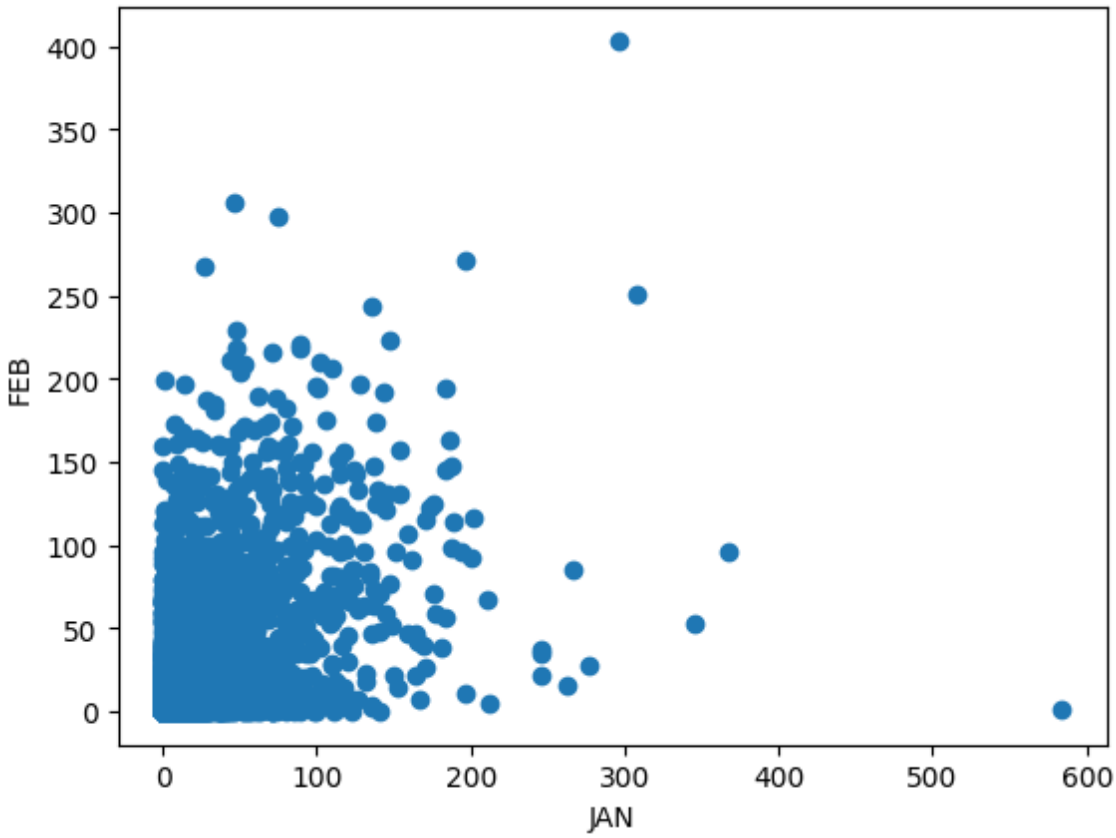
	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	5
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	3
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	2
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	3
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	

In [128]:

```
plt.xlabel("JAN")
plt.ylabel("FEB")
```

Out[128]:

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



In [129]:

```
from sklearn.cluster import KMeans
```

In [130]:

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

Out[130]:

```
KMeans()
```

In [139]:

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

Out[139]:

```
SUBDIVISION     0
YEAR            0
JAN             4
FEB             3
MAR             6
APR             4
MAY             3
JUN             5
JUL             7
AUG             4
SEP             6
OCT             7
NOV            11
DEC            10
ANNUAL         26
Jan-Feb         6
Mar-May         9
Jun-Sep        10
Oct-Dec        13
dtype: int64
```

In [140]:

```
df.fillna(method="ffill",inplace=True)
```

In [141]:

```
y_predicted=km.fit_predict(df[["YEAR","ANNUAL"]])
y_predicted
```

```
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
```

Out[141]:

```
array([6, 6, 1, ..., 5, 2, 5])
```

In [142]:

```
df["cluster"]=y_predicted
df.head()
```

Out[142]:

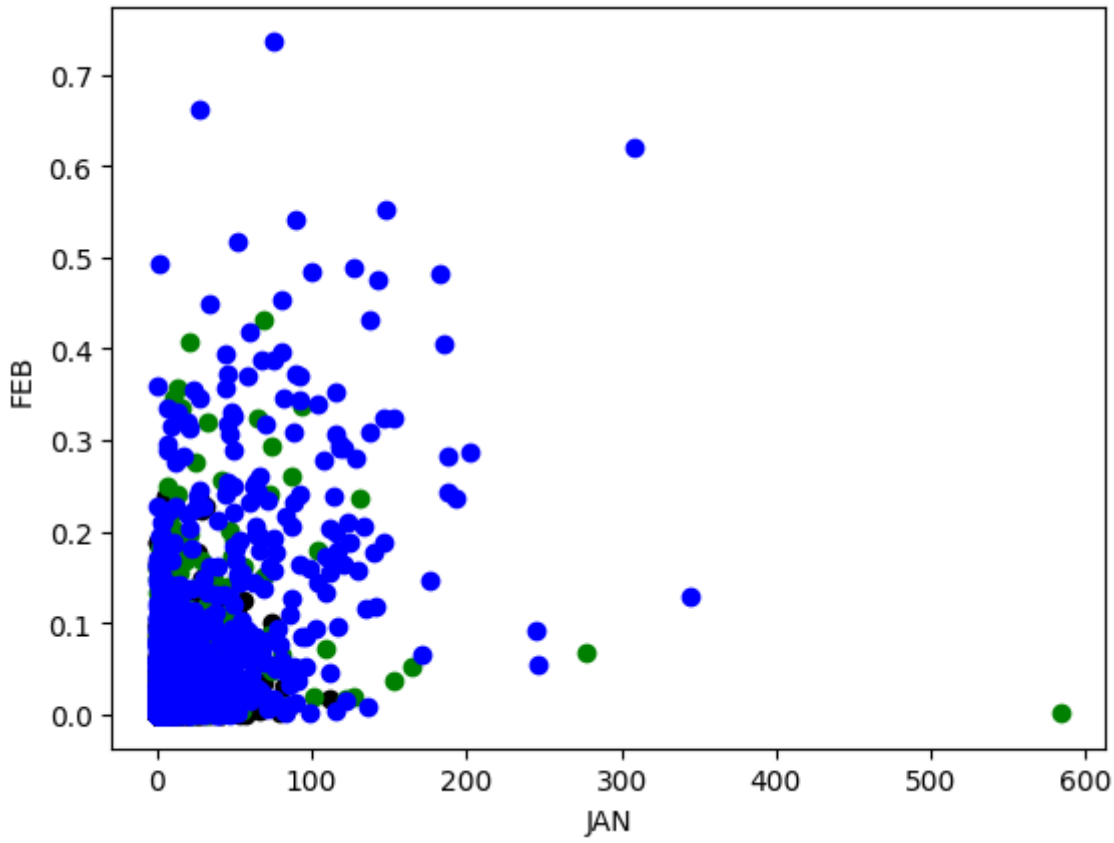
	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	5
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	3
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	2
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	3
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	

In [156]:

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

Out[156]:

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



In [144]:

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

In [145]:

```
Scaler=MinMaxScaler()
```

In [148]:

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

Out[148]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	0.215861	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	0.396035	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	0.356877	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	0.036431	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.000000	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7

In [150]:

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

Out[150]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	0.215861	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	0.396035	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	0.356877	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	0.036431	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.000000	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7

In [151]:

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

Out[151]:

KMeans

KMeans()

In [152]:

```
y_predicted=km.fit_predict(df[["YEAR","ANNUAL"]])
y_predicted
```

```
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
```

Out[152]:

```
array([3, 3, 3, ..., 2, 2, 2])
```

In [160]:

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

Out[160]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	...	NOV	
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	0.215861	29.2	2.3	528.8	517.5	365.1	481.1	...	558.2	
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	0.396035	12.2	0.0	446.1	537.1	228.9	753.7	...	359.0	1
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	0.356877	0.0	1.0	235.1	479.9	728.4	326.7	...	284.4	2
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	0.036431	0.0	202.4	304.5	495.1	502.0	160.1	...	308.7	
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.000000	3.3	26.9	279.5	628.7	368.7	330.5	...	25.4	3

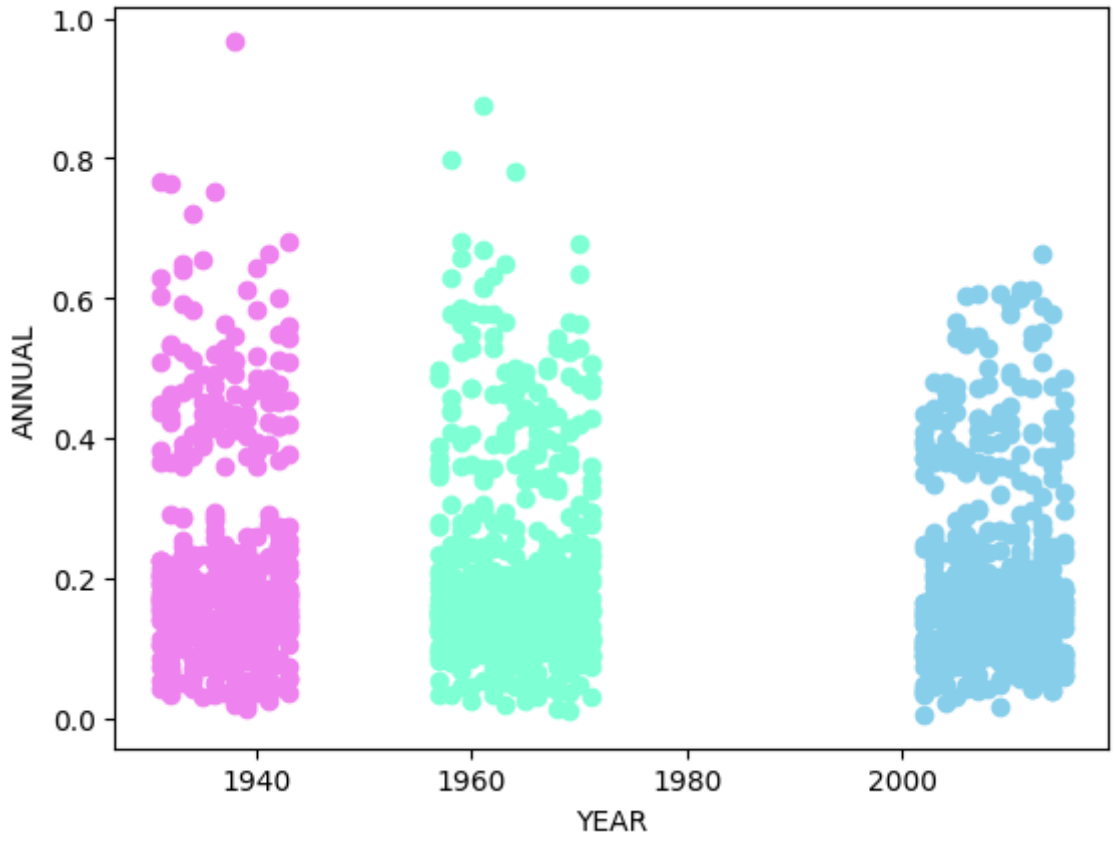
5 rows × 22 columns

In [162]:

```
df1=df[df["New cluster"]==0]
df2=df[df["New cluster"]==1]
df3=df[df["New cluster"]==2]
plt.scatter(df1["YEAR"],df1["ANNUAL"],color="aquamarine")
plt.scatter(df2["YEAR"],df2["ANNUAL"],color="violet")
plt.scatter(df3["YEAR"],df3["ANNUAL"],color="skyblue")
plt.xlabel("YEAR")
plt.ylabel("ANNUAL")
```

Out[162]:

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



In [163]:

```
km.cluster_centers_
```

Out[163]:

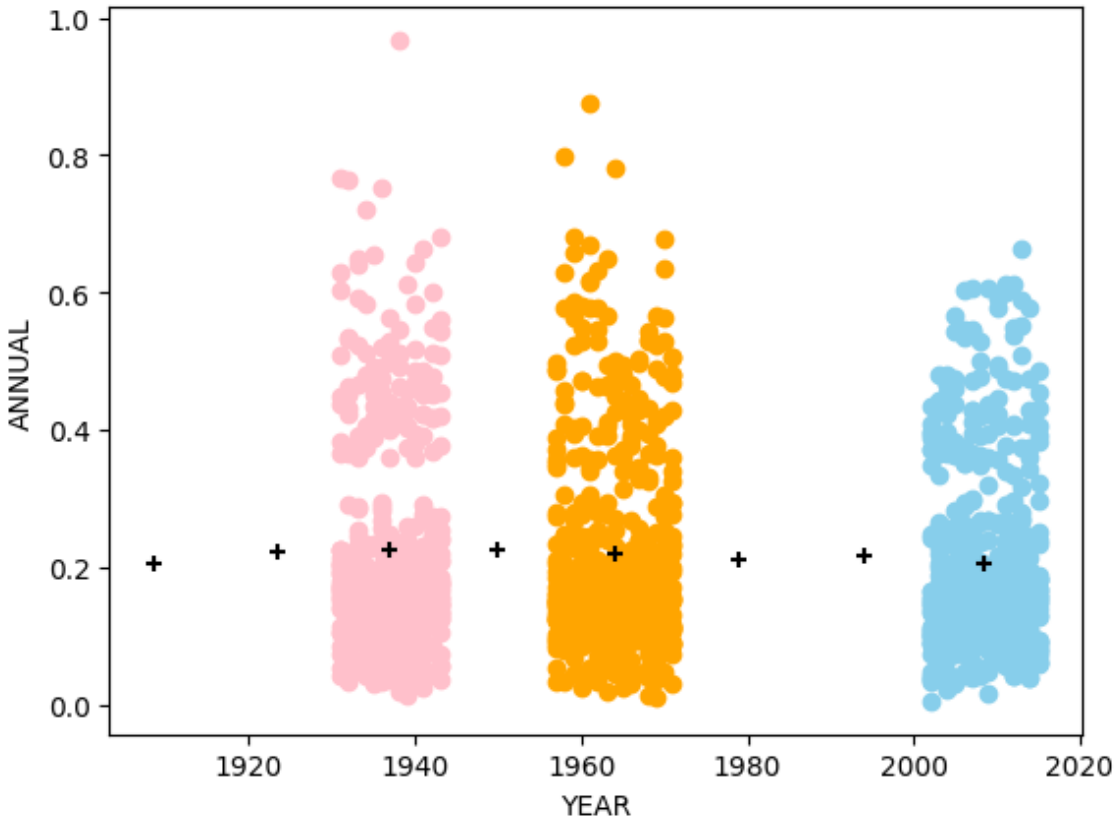
```
array([[1.96400000e+03, 2.20184943e-01],
       [1.93698715e+03, 2.25502485e-01],
       [2.00850000e+03, 2.05632822e-01],
       [1.90851250e+03, 2.04288472e-01],
       [1.97900000e+03, 2.10236704e-01],
       [1.99400000e+03, 2.17334495e-01],
       [1.92351093e+03, 2.22864669e-01],
       [1.94999567e+03, 2.25934864e-01]])
```

In [166]:

```
df1=df[df["New cluster"]==0]
df2=df[df["New cluster"]==1]
df3=df[df["New cluster"]==2]
plt.scatter(df1["YEAR"],df1["ANNUAL"],color="orange")
plt.scatter(df2["YEAR"],df2["ANNUAL"],color="pink")
plt.scatter(df3["YEAR"],df3["ANNUAL"],color="skyblue")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="black",marker="+")
plt.xlabel("YEAR")
plt.ylabel("ANNUAL")
```

Out[166]:

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



In [167]:

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

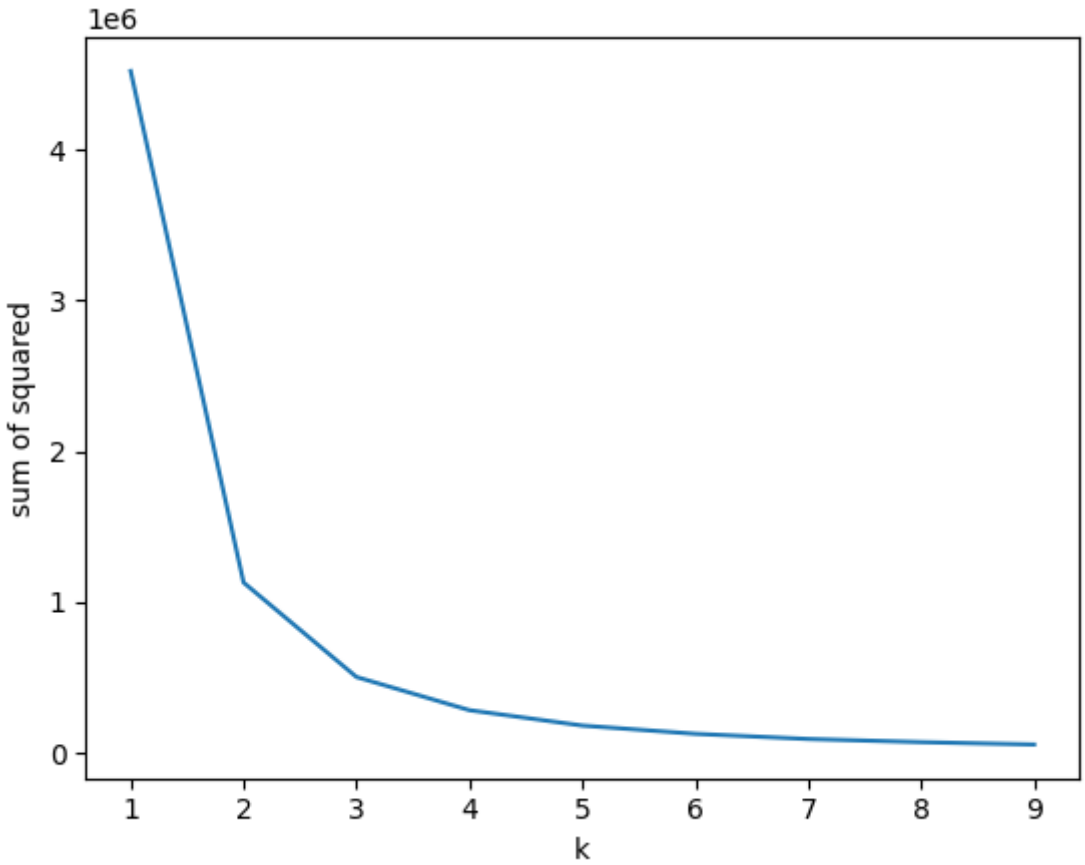
```
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default valu
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
```

In [169]:

```
plt.plot(k_rng,sse)
plt.xlabel("k")
plt.ylabel("sum of squared")
```

Out[169]:

```
Text(0, 0.5, 'sum of squared')
```



Conclusion:

In this data set used some models,I got the more accuracy for LinearRegression 76.This is the best fit model for this Rainfall Data.In K-Means Cluster from the above data we have devided groups into clustre.

In [ ]: