In [ ]:

```
#problem stmt:To find the best fitmodel for the given dataset
#import libraries
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

reading the dataframe

In [2]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\rainfall in india 1901-2015.csv")
df
```

Out[2]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OC
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388
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	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260
...	...	...	...	...	...	...	...	...	...	...	...	...
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165

4116 rows × 19 columns

In [3]:

```
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
```

performing the basic pre-processing steps

In [4]:

```
df.describe()
```

Out[4]:

	YEAR	JAN	FEB	MAR	APR	MAY	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.00
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.23
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.71
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.40
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.35
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.70
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.15
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.90

In [5]:

```
df.isna().any()
```

Out[5]:

```
SUBDIVISION    False
YEAR           False
JAN             True
FEB             True
MAR             True
APR             True
MAY             True
JUN             True
JUL             True
AUG             True
SEP             True
OCT             True
NOV             True
DEC             True
ANNUAL          True
Jan-Feb         True
Mar-May         True
Jun-Sep         True
Oct-Dec         True
dtype: bool
```

In [6]:

```
df["SUBDIVISION"].value_counts()
```

Out[6]:

```
SUBDIVISION
WEST MADHYA PRADESH                   115
EAST RAJASTHAN                        115
COASTAL KARNATAKA                     115
TAMIL NADU                            115
RAYALSEEMA                            115
TELANGANA                             115
COASTAL ANDHRA PRADESH                115
CHHATTISGARH                          115
VIDARBHA                              115
MATATHWADA                            115
MADHYA MAHARASHTRA                    115
KONKAN & GOA                          115
SAURASHTRA & KUTCH                    115
GUJARAT REGION                        115
EAST MADHYA PRADESH                   115
KERALA                                115
WEST RAJASTHAN                        115
SOUTH INTERIOR KARNATAKA              115
JAMMU & KASHMIR                       115
HIMACHAL PRADESH                      115
PUNJAB                                115
HARYANA DELHI & CHANDIGARH            115
UTTARAKHAND                           115
WEST UTTAR PRADESH                    115
EAST UTTAR PRADESH                    115
BIHAR                                 115
JHARKHAND                             115
ORISSA                                115
GANGETIC WEST BENGAL                  115
SUB HIMALAYAN WEST BENGAL & SIKKIM    115
NAGA MANI MIZO TRIPURA                115
ASSAM & MEGHALAYA                     115
NORTH INTERIOR KARNATAKA              115
LAKSHADWEEP                           114
ANDAMAN & NICOBAR ISLANDS             110
ARUNACHAL PRADESH                      97
Name: count, dtype: int64
```

dropping the columns

In [7]:

```
df.pop("SUBDIVISION")
```

Out[7]:

```
0       ANDAMAN & NICOBAR ISLANDS
1       ANDAMAN & NICOBAR ISLANDS
2       ANDAMAN & NICOBAR ISLANDS
3       ANDAMAN & NICOBAR ISLANDS
4       ANDAMAN & NICOBAR ISLANDS
                  ...
4111                  LAKSHADWEEP
4112                  LAKSHADWEEP
4113                  LAKSHADWEEP
4114                  LAKSHADWEEP
4115                  LAKSHADWEEP
Name: SUBDIVISION, Length: 4116, dtype: object
```

In [8]:

```
df.pop("Jan-Feb")
```

Out[8]:

```
0       136.3
1       159.8
2       156.7
3        24.1
4         1.3
        ...
4111      7.9
4112     19.3
4113     60.6
4114     69.3
4115      2.7
Name: Jan-Feb, Length: 4116, dtype: float64
```

In [9]:

```
df.describe()
```

Out[9]:

	YEAR	JAN	FEB	MAR	APR	MAY	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.00
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.23
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.71
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.40
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.35
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.70
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.15
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.90

In [10]:

```
df.pop("Mar-May")
```

Out[10]:

```
0       560.3
1       458.3
2       236.1
3       506.9
4       309.7
        ...  
4111    196.2
4112     99.6
4113    131.1
4114     76.7
4115    223.9
Name: Mar-May, Length: 4116, dtype: float64
```

In [11]:

```
df.pop("Jun-Sep")
```

Out[11]:

```
0       1696.3
1       2185.9
2       1874.0
3       1977.6
4       1624.9
         ...  
4111    1013.0
4112    1119.5
4113    1057.0
4114     958.5
4115     860.9
Name: Jun-Sep, Length: 4116, dtype: float64
```

In [12]:

```
df.pop("Oct-Dec")
```

Out[12]:

```
0       980.3
1       716.7
2       690.6
3       571.0
4       630.8
        ...  
4111    316.6
4112    167.1
4113    177.6
4114    290.5
4115    555.4
Name: Oct-Dec, Length: 4116, dtype: float64
```

In [13]:

```
df.describe()
```

Out[13]:

	YEAR	JAN	FEB	MAR	APR	MAY	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.00
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.23
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.71
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.40
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.35
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.70
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.15
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.90

filling the Nullvalues

In [14]:

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

In [15]:

```
features=df.columns[0:12]
target=df.columns[-1]
```

In [ ]:

```
x=df[features].values
y=df[target].values
```

In [ ]:

```
#Building a Model
```

In [17]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [18]:

```
a=LinearRegression()
a.fit(x_train,y_train)
print(a.score(x_test,y_test))
```

```
0.9924756266920182
```

In [19]:

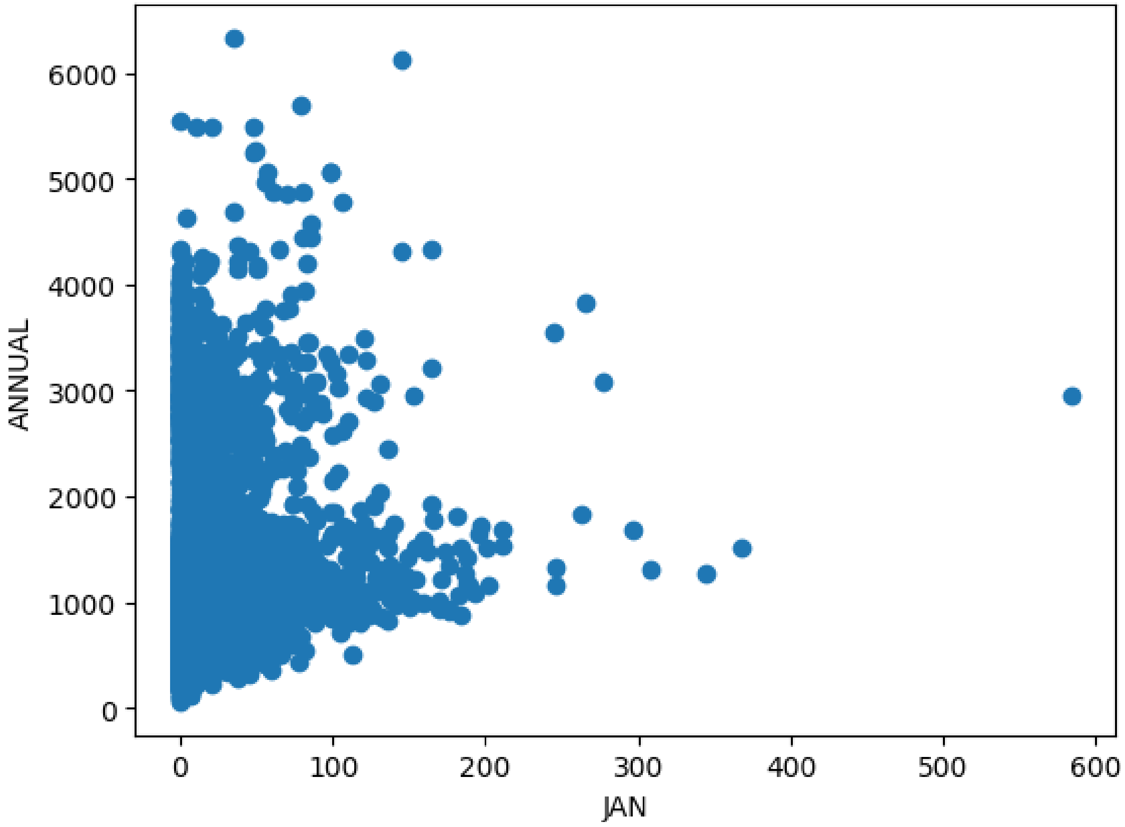
```
#k-Means
```

In [20]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [21]:

```
plt.scatter(df["JAN"],df["ANNUAL"])
plt.xlabel("JAN")
plt.ylabel("ANNUAL")
plt.show()
```



In [22]:

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

In [23]:

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

Out[23]:

```
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 [24]:

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

```
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
```

Out[24]:

```
array([1, 7, 1, ..., 3, 2, 3])
```

In [25]:

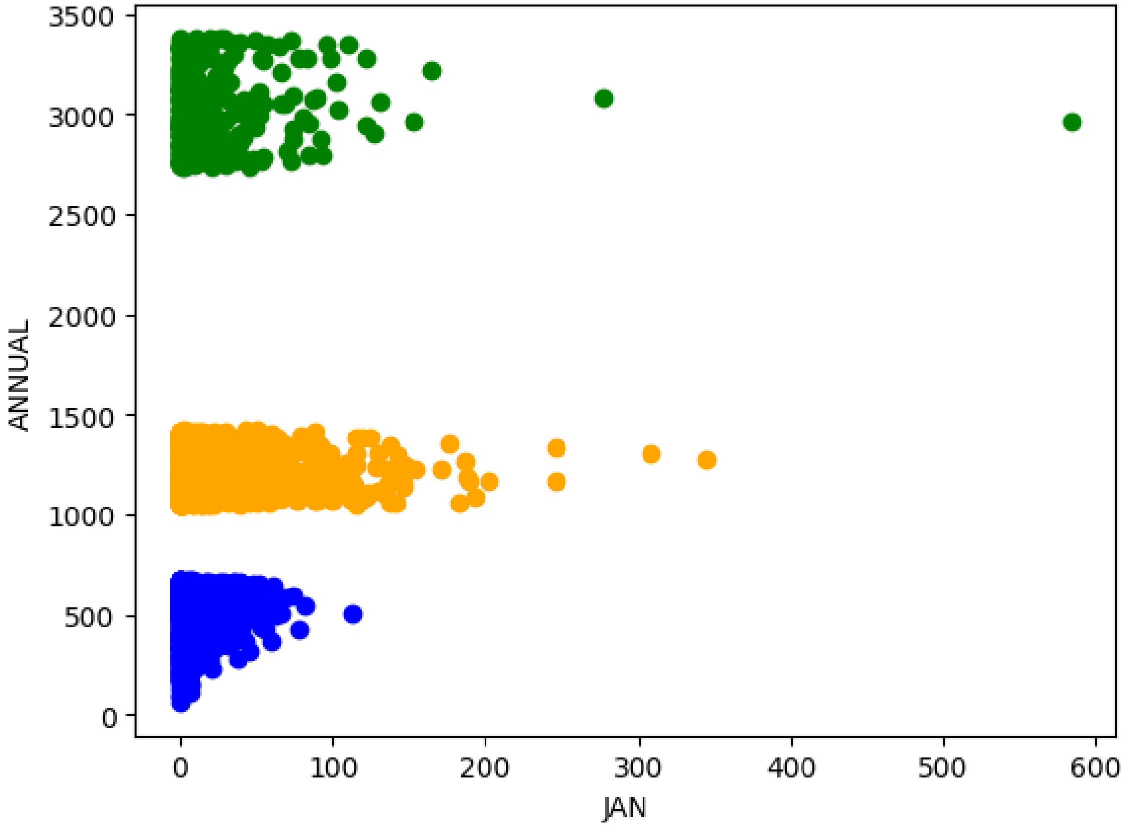
```
df["cluster"]=y_predicted
df.head()
```

Out[25]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	AN
0	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	
1	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	
2	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	
3	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	
4	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	

In [26]:

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



In [27]:

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

In [28]:

```
scaler=MinMaxScaler()
scaler.fit(df[["JAN"]])
s=scaler.transform(df[["JAN"]])
df.head()
```

Out[28]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	AN
0	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	
1	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	
2	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	
3	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	
4	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	

In [29]:

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

Out[29]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	AN
0	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	
1	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	
2	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	
3	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	
4	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	

In [30]:

```
km=KMeans()
km

```

Out[30]:

```
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 [31]:

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

```
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
```

Out[31]:

```
array([1, 3, 1, ..., 2, 4, 2])
```

In [32]:

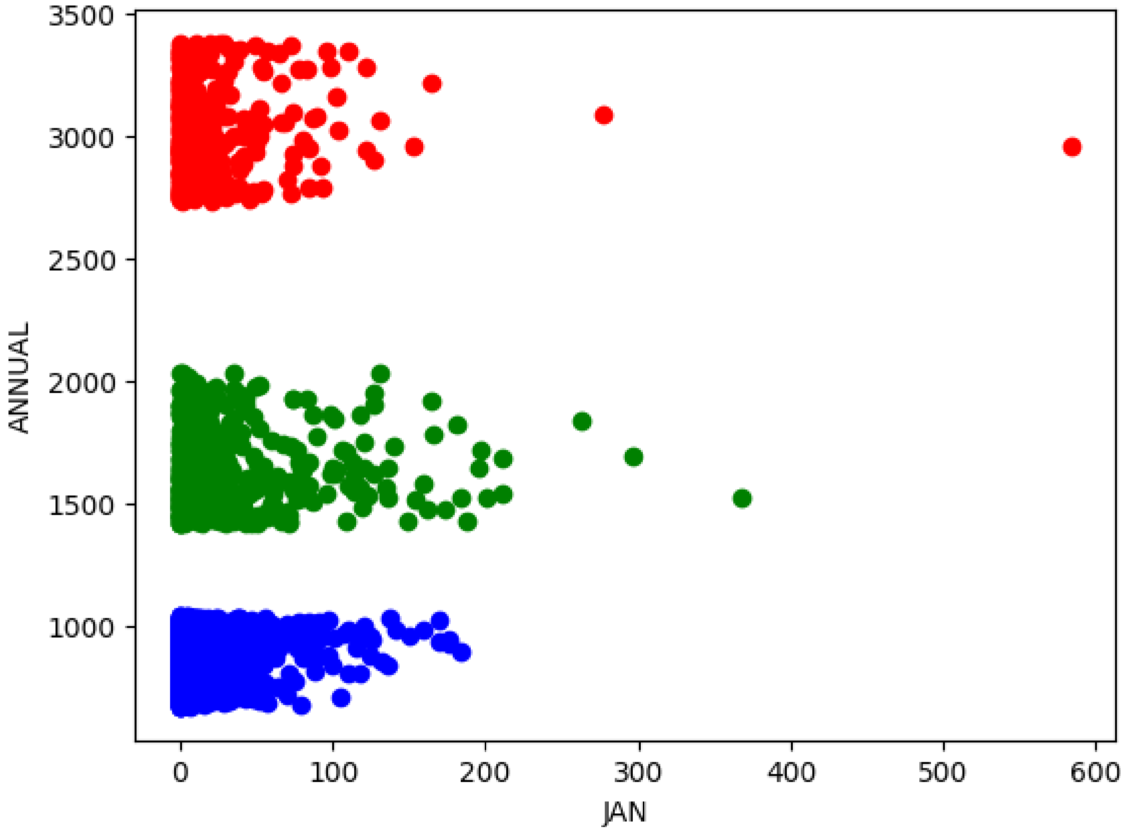
```
df["New cluster"]=y_predicted
y_predicted
```

Out[32]:

```
array([1, 3, 1, ..., 2, 4, 2])
```

In [33]:

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



In [34]:

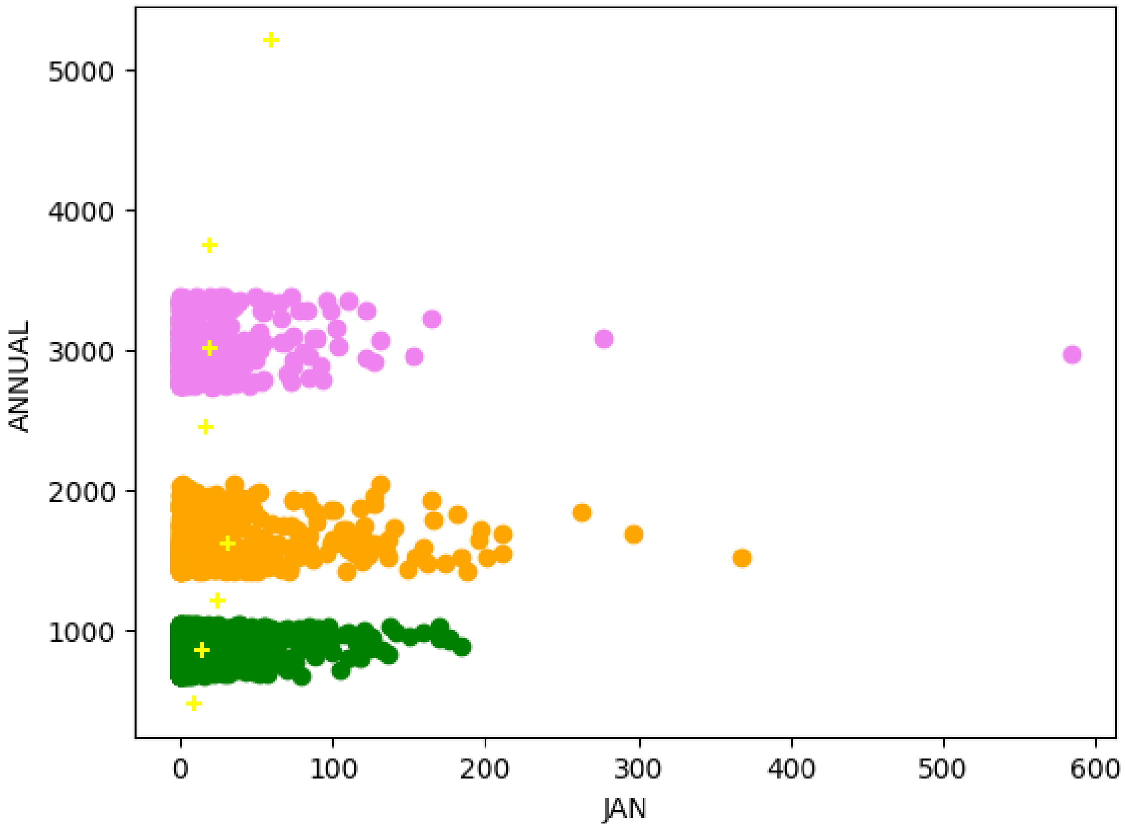
```
km.cluster_centers_
```

Out[34]:

```
array([[  14.08534031,  861.06928447],
       [  20.02631579, 3015.81218837],
       [  31.10688259, 1617.83137652],
       [  19.71904762, 3745.09251701],
       [  24.24035088, 1209.22827657],
       [  17.42192982, 2445.8994152 ],
       [   9.1967033 ,  476.29120879],
       [  59.27      , 5206.625     ]])
```

In [35]:

```
df1=df[df["New cluster"]==0]
df2=df[df["New cluster"]==1]
df3=df[df["New cluster"]==2]
plt.scatter(df1["JAN"],df1["ANNUAL"],color="green")
plt.scatter(df2["JAN"],df2["ANNUAL"],color="violet")
plt.scatter(df3["JAN"],df3["ANNUAL"],color="orange")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="yellow",marker=
plt.xlabel("JAN")
plt.ylabel("ANNUAL")
plt.show()
```



In [36]:

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

```

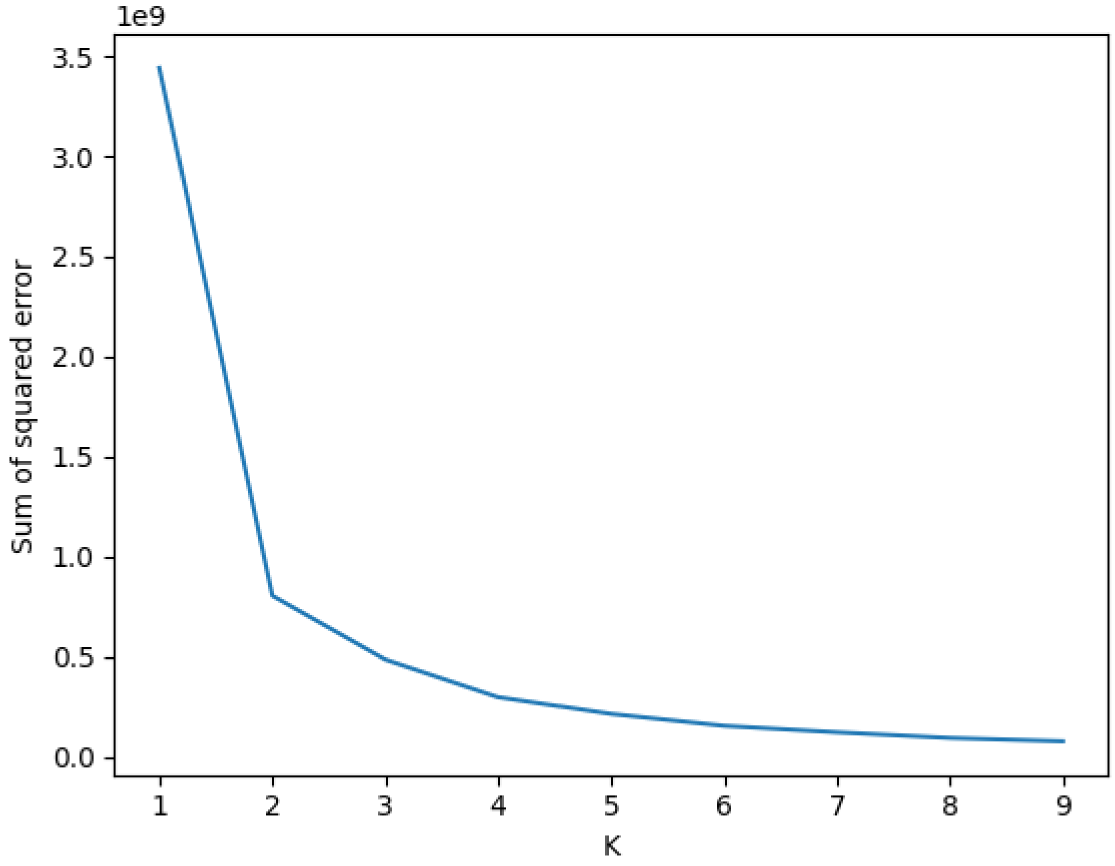
```
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_in
it` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explic
itly to suppress the warning
  warnings.warn(
```

In [37]:

```
plt.plot(p,sse)
plt.xlabel("K")
plt.ylabel("Sum of squared error")
```

Out[37]:

```
Text(0, 0.5, 'Sum of squared error')
```



In [38]:

```
#district wise#
```

In [39]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [40]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\district wise rainfall normal.csv")
df
```

Out[40]:

	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 [41]:

```
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 [42]:

```
df.pop("STATE_UT_NAME")
```

Out[42]:

```
0      ANDAMAN And NICOBAR ISLANDS
1      ANDAMAN And NICOBAR ISLANDS
2      ANDAMAN And NICOBAR ISLANDS
3                ARUNACHAL PRADESH
4                ARUNACHAL PRADESH
                  ...             
636                         KERALA
637                         KERALA
638                         KERALA
639                         KERALA
640                    LAKSHADWEEP
Name: STATE_UT_NAME, Length: 641, dtype: object
```

In [43]:

```
df.pop("DISTRICT")
```

Out[43]:

```
0             NICOBAR
1       SOUTH ANDAMAN
2       N & M ANDAMAN
3               LOHIT
4          EAST SIANG
            ...      
636            IDUKKI
637          KASARGOD
638    PATHANAMTHITTA
639           WAYANAD
640       LAKSHADWEEP
Name: DISTRICT, Length: 641, dtype: object
```

In [44]:

```
df.pop("Jan-Feb")
```

Out[44]:

```
0      165.2
1       69.7
2       48.6
3      123.0
4      112.8
       ...  
636     35.5
637      3.3
638     65.0
639     13.1
640     35.5
Name: Jan-Feb, Length: 641, dtype: float64
```

In [45]:

```
df.pop("Mar-May")
```

Out[45]:

```
0      540.7
1      483.5
2      405.6
3      841.3
4      645.4
       ...
636    426.6
637    272.9
638    553.5
639    275.4
640    232.4
Name: Mar-May, Length: 641, dtype: float64
```

In [46]:

```
df.pop("Jun-Sep")
```

Out[46]:

```
0      1207.2
1      1757.2
2      1884.4
3      1848.5
4      3008.4
        ...
636    2276.2
637    3007.5
638    1715.7
639    2632.1
640     998.5
Name: Jun-Sep, Length: 641, dtype: float64
```

In [47]:

```
df.pop("Oct-Dec")
```

Out[47]:

```
0      892.1
1      705.3
2      574.7
3      231.0
4      268.1
       ...
636    564.2
637    337.9
638    624.2
639    332.5
640    333.6
Name: Oct-Dec, Length: 641, dtype: float64
```

In [48]:

```
df.describe()
```

Out[48]:

	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 [53]:

```
features=df.columns[0:12]
target=df.columns[-1]
```

In [54]:

```
x=df[features].values
y=df[target].values
```

In [55]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [56]:

```
a=LinearRegression()
a.fit(x_train,y_train)
print(a.score(x_test,y_test))
```

```
1.0
```

In [57]:

```
    #lasso

```

In [59]:

```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
ridge=Ridge(alpha=10)
ridge.fit(x_train,y_train)
train_score_ridge=ridge.score(x_train,y_train)
test_score_ridge=ridge.score(x_test,y_test)
print(train_score_ridge)
print(test_score_ridge)
```

```
0.9999999999939446
0.9999999999874553
```

In [62]:

```
l=Lasso(alpha=10)
l.fit(x_train,y_train)
train_score_l=l.score(x_train,y_train)
test_score_l=l.score(x_test,y_test)
print(train_score_l)
print(test_score_l)
```

```
0.999999401432268
0.9999987435581194
```

In [ ]:

In [ ]: