PROBLEM STATEMENT : To perform an Analytics Report on 100 Years of Rainfall Data

Importing Libraries

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the data

In [3]:

```
df=pd.read_csv(r"C:\Users\G S R  KARTHIK\Documents\100Years_RainfallDataset\rainfall in
df
```

Out[3]:

	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

Data Cleaning and Preprocessing

In [4]:

```
df.drop(['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec'],axis=1)
```

Out[4]:

	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 × 15 columns

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
df.tail()
```

Out[6]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OC
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.

In [7]:

```
df.describe()
```

Out[7]:

	YEAR	JAN	FEB	MAR	APR	MAY	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.0
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.2
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.7
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.4
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.3
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.7
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.1
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.9

In [8]:

```
df.shape
```

Out[8]:

```
(4116, 19)
```

In [9]:

```
df.columns
```

Out[9]:

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

In [10]:

```
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 [11]:

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

Out[11]:

```
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 [12]:

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

In [13]:

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

Out[13]:

```
SUBDIVISION    0
YEAR           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 [14]:

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

Out[14]:

```
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
```

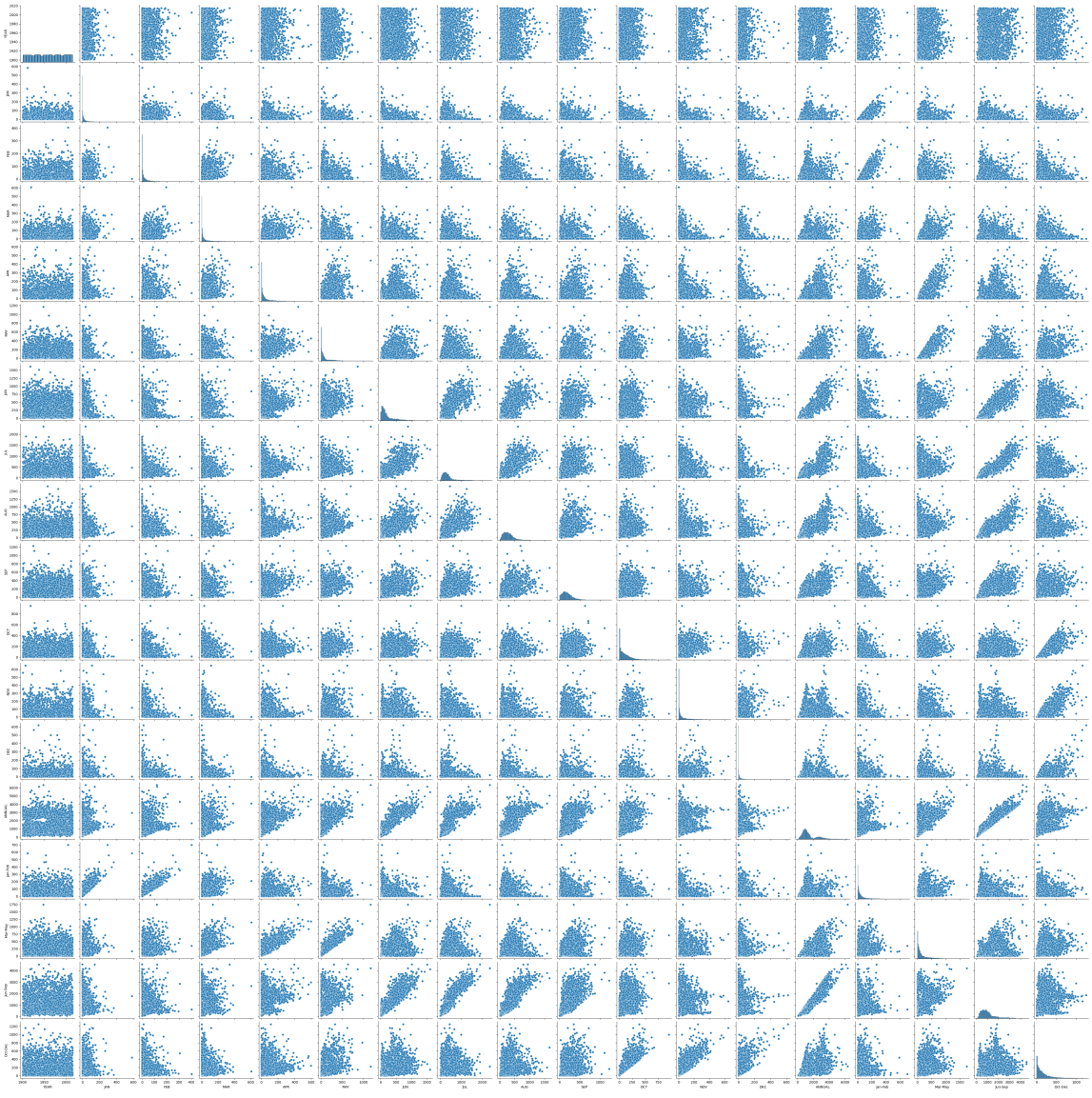
Data Visualization

In [15]:

```
#EDA
sns.pairplot(df)
```

Out[15]:

```
<seaborn.axisgrid.PairGrid at 0x1967daf6620>
```

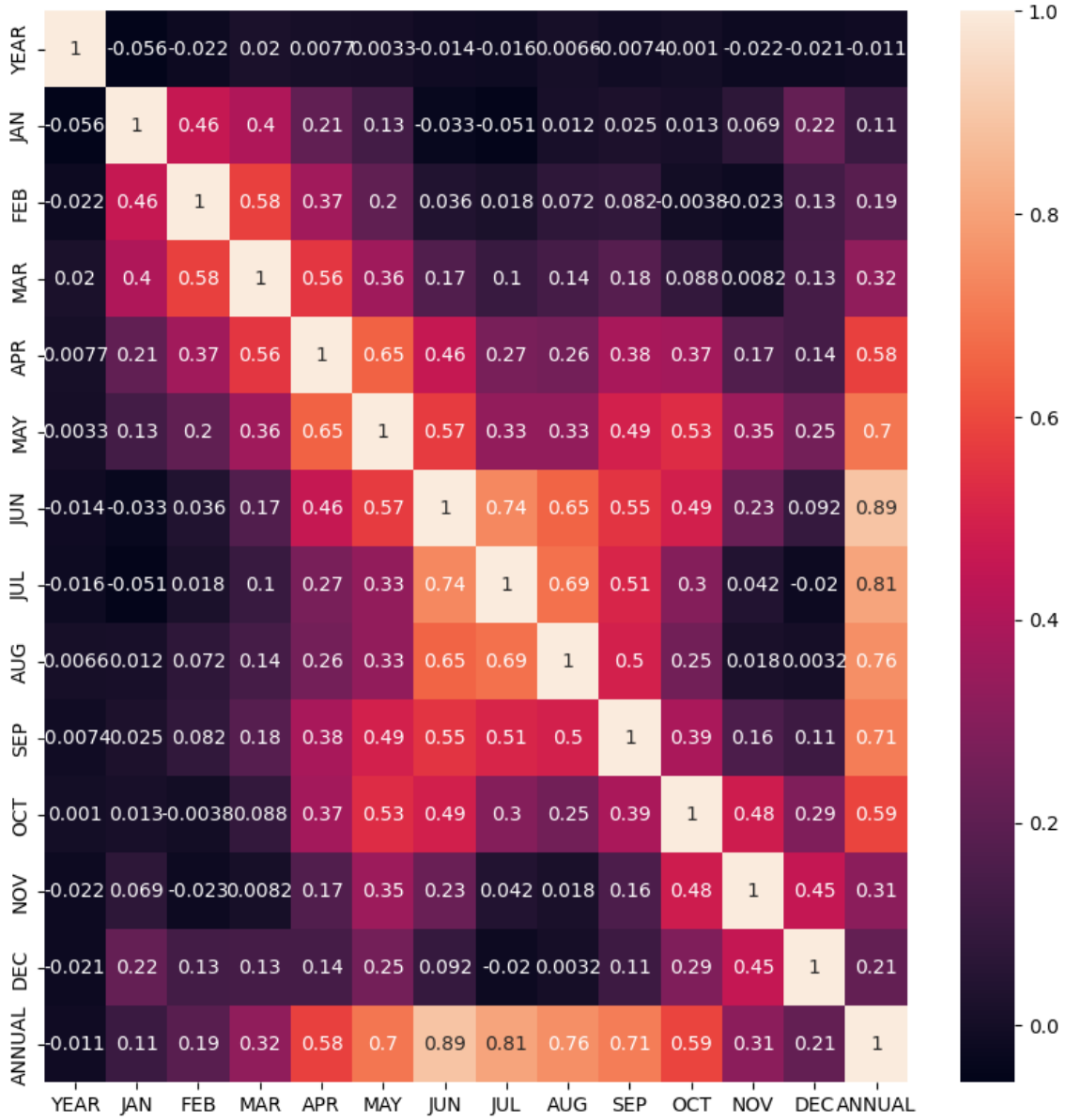


In [16]:

```
rdf=df[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL']]
plt.figure(figsize=(10,10))
sns.heatmap(rdf.corr(),annot=True)
```

Out[16]:

<Axes: >



In [17]:

```
states={"SUBDIVISION":{
"WEST MADHYA PRADESH":1,
"EAST RAJASTHAN":2,
"COASTAL KARNATAKA":3,
"TAMIL NADU":4,
"RAYALSEEMA":5,
"TELANGANA":6,
"COASTAL ANDHRA PRADESH":7,
"CHHATTISGARH":8,
"VIDARBHA":9,
"MATATHWADA":10,
"MADHYA MAHARASHTRA":11,
"KONKAN & GOA":12,
"SAURASHTRA & KUTCH":13,
"GUJARAT REGION":14,
"EAST MADHYA PRADESH":15,
"KERALA":16,
"WEST RAJASTHAN":17,
"SOUTH INTERIOR KARNATAKA":18,
"JAMMU & KASHMIR":19,
"HIMACHAL PRADESH":20,
"PUNJAB":21,
"HARYANA DELHI & CHANDIGARH":22,
"UTTARAKHAND":23,
"WEST UTTAR PRADESH":24,
"EAST UTTAR PRADESH":25,
"BIHAR":26,
"JHARKHAND":27,
"ORISSA":28,
"GANGETIC WEST BENGAL":29,
"SUB HIMALAYAN WEST BENGAL & SIKKIM":30,
"NAGA MANI MIZO TRIPURA":31,
"ASSAM & MEGHALAYA":32,
"NORTH INTERIOR KARNATAKA":33,
"LAKSHADWEEP":34,
"ANDAMAN & NICOBAR ISLANDS":35,
"ARUNACHAL PRADESH":36}}
df=df.replace(states)
df
```

Out[17]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT
0	35	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5
1	35	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2
2	35	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2
3	35	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2
4	35	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7
...	...	...	...	...	...	...	...	...	...	...	...	..
4111	34	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4
4112	34	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9
4113	34	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8
4114	34	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2
4115	34	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4

4116 rows × 19 columns

In [18]:

```
df.drop(['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec'],axis=1)
```

Out[18]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT
0	35	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5
1	35	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2
2	35	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2
3	35	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2
4	35	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7
...	...	...	...	...	...	...	...	...	...	...	...	..
4111	34	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4
4112	34	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9
4113	34	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8
4114	34	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2
4115	34	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4

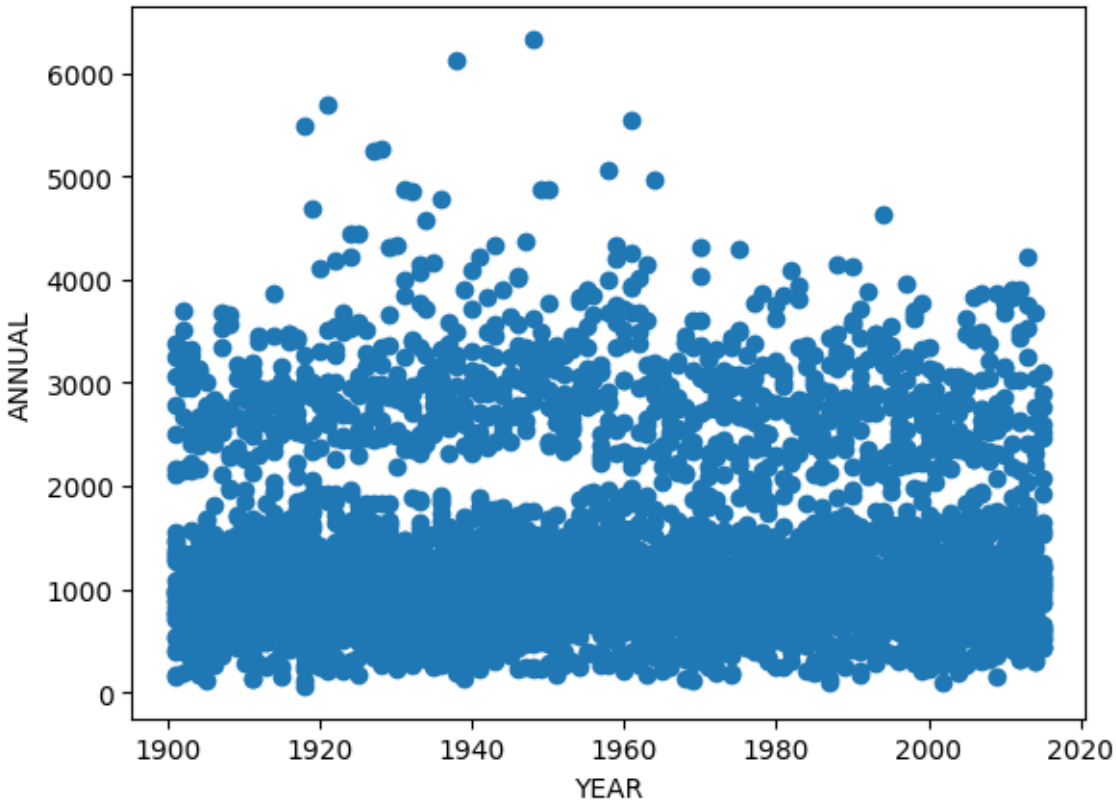
4116 rows × 15 columns

In [19]:

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

Out[19]:

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



Importing KMeans

In [22]:

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

In [23]:

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

Out[23]:

```
KMeans()
```

In [25]:

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

```
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-p
ackages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value o
f `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

Out[25]:

```
array([1, 1, 7, ..., 5, 2, 5])
```

In [26]:

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

Out[26]:

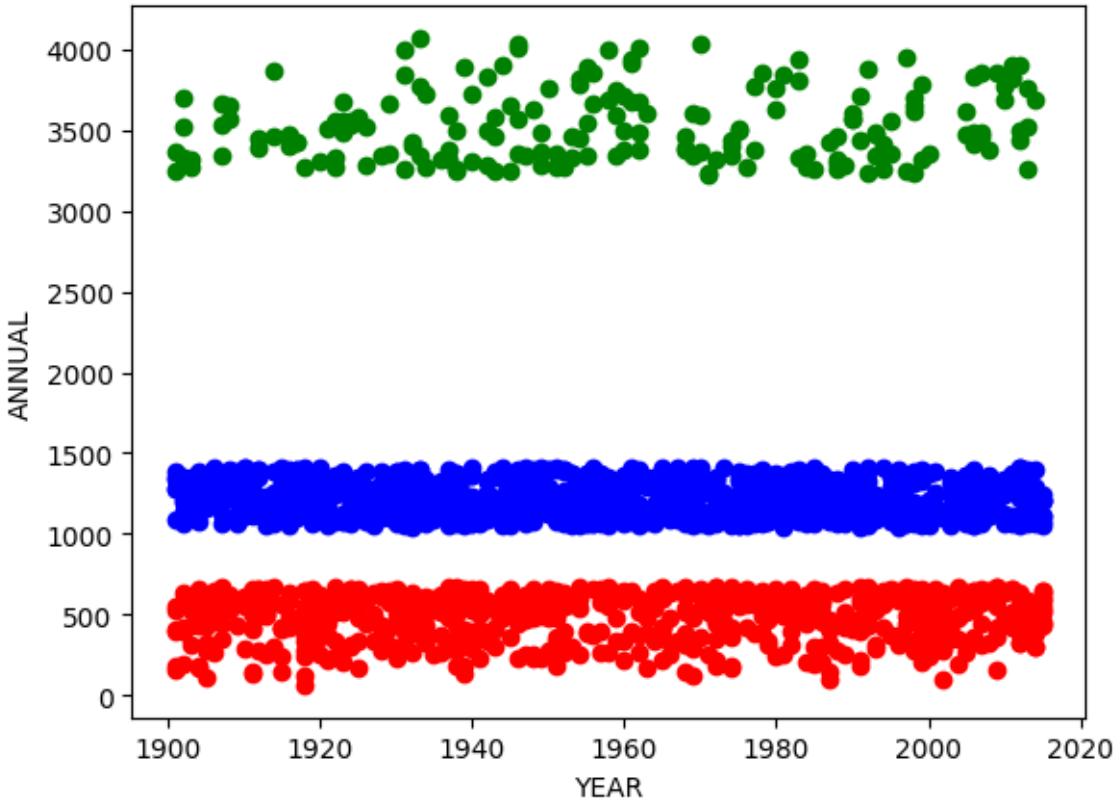
	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
0	35	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	5
1	35	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	3
2	35	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	2
3	35	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	3
4	35	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	

In [28]:

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

Out[28]:

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



In [29]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(df[["ANNUAL"]])
df["ANNUAL"]=scaler.transform(df[["ANNUAL"]])
df.head()
```

Out[29]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
0	35	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	5
1	35	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	3
2	35	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	2
3	35	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	3
4	35	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	

In [30]:

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

Out[30]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT
0	35	0.000000	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5
1	35	0.008772	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2
2	35	0.017544	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2
3	35	0.026316	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2
4	35	0.035088	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7

In [31]:

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

```
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-p
ackages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value o
f `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

Out[31]:

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

In [32]:

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

Out[32]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	...	OCT	N
0	35	0.000000	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	...	388.5	5
1	35	0.008772	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	...	197.2	3
2	35	0.017544	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	...	181.2	2
3	35	0.026316	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	...	222.2	3
4	35	0.035088	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	...	260.7	2

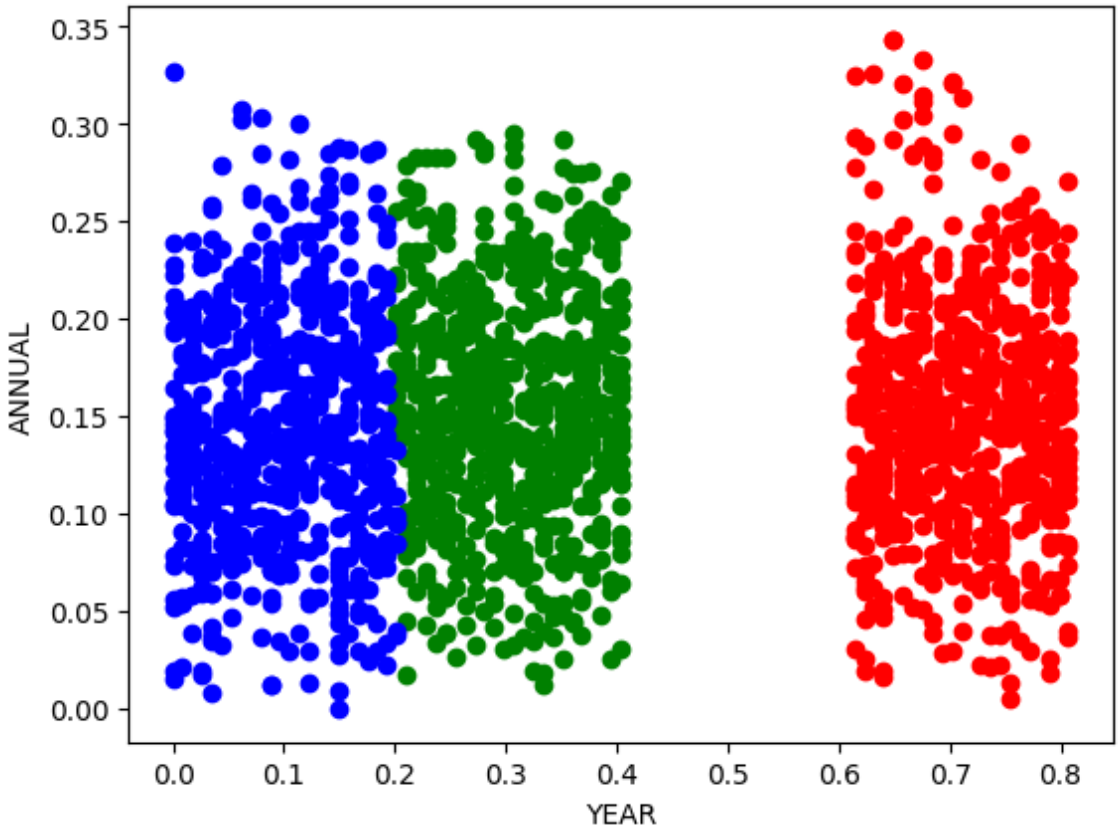
5 rows × 21 columns

In [33]:

```
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="red")
plt.scatter(df2["YEAR"],df2["ANNUAL"],color="green")
plt.scatter(df3["YEAR"],df3["ANNUAL"],color="blue")
plt.xlabel("YEAR")
plt.ylabel("ANNUAL")
```

Out[33]:

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



In [34]:

```
km.cluster_centers_
```

Out[34]:

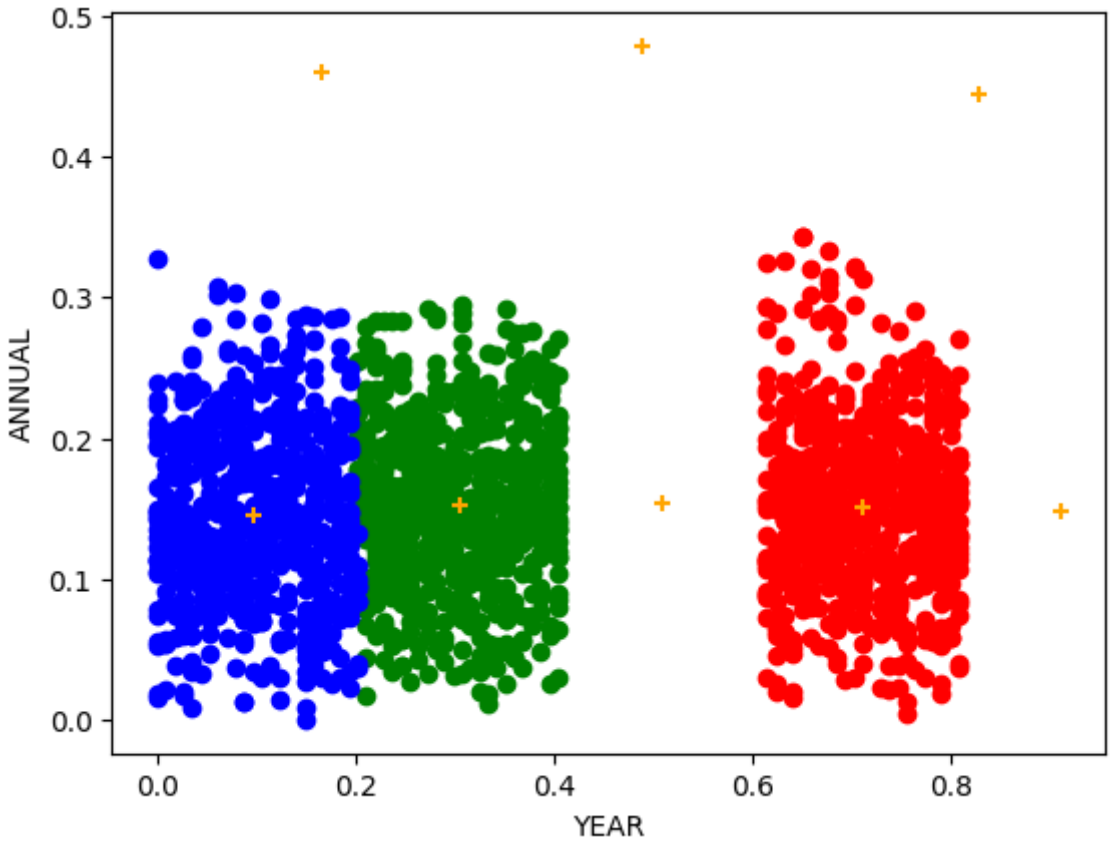
```
array([[0.70936297, 0.1515095 ],
       [0.30472806, 0.15299477],
       [0.09767881, 0.1450655 ],
       [0.90944067, 0.14824557],
       [0.82795322, 0.44391899],
       [0.48811746, 0.47865383],
       [0.50928396, 0.15377016],
       [0.1664858 , 0.46015296]])
```

In [35]:

```
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="red")
plt.scatter(df2["YEAR"],df2["ANNUAL"],color="green")
plt.scatter(df3["YEAR"],df3["ANNUAL"],color="blue")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="orange",marker="+")
plt.xlabel("YEAR")
plt.ylabel("ANNUAL")
```

Out[35]:

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



In [36]:

```
k_rng=range(1,10)
sse=[]
```

In [38]:

```
for k in k_rng:
    km=KMeans(n_clusters=k)
    km.fit(df[["YEAR","ANNUAL"]])
    sse.append(km.inertia_)
#km.inertia_ will give you the value of sum of square error
print(sse)
plt.plot(k_rng,sse)
plt.xlabel("K")
plt.ylabel("Sum of Squared Error")
```

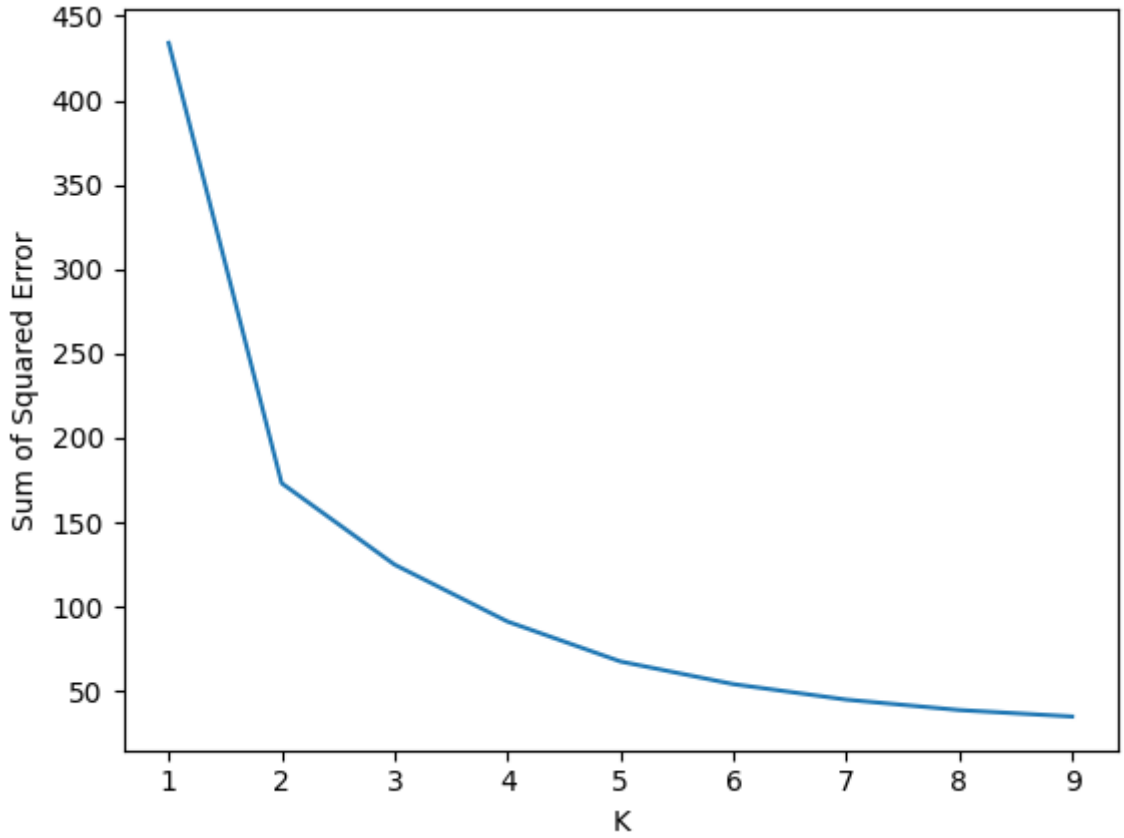
```
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\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 suppress the warning
  warnings.warn(
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\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 suppress the warning
  warnings.warn(
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\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 suppress the warning
  warnings.warn(
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\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 suppress the warning
  warnings.warn(
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\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 suppress the warning
  warnings.warn(
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\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 suppress the warning
  warnings.warn(
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\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 suppress the warning
  warnings.warn(

[434.01337392621843, 173.14952597578804, 124.9011426928695, 91.15705459917861, 67.48551964782972, 54.05475300669102, 44.87498165540285, 38.690187715642146, 34.86421887818578]
```

```
C:\Users\G S R  KARTHIK\AppData\Local\Programs\Python\Python310\lib\site-p
ackages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value o
f `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

Out[38]:

```
Text(0, 0.5, 'Sum of Squared Error')
```



CONCLUSION : The KMeans model is the best model for the given dataset

In [ ]: