PROBLEM STATEMENT:- To predict the rainfall based on various feat of the dataset

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df=pd.read_csv(r"C:\Users\Teju\Downloads\RainFall.csv")
df
```

Out[2]:

	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

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.tail()
```

Out[4]:

	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 [5]:

```
df.isnull().any()
```

Out[5]:

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

In [7]:

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

Out[7]:

```
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 [8]:

```python
df.describe()
```

Out[8]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.00000
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.03369
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.36464
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.60000
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.40000
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.70000
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.80000
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.90000

In [9]:

```python
df.shape
```

Out[9]:

```
(641, 19)
```

In [10]:

```python
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 [11]:

```
features=df[2:13]
target=df.columns[14]
```

In [12]:

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

In [13]:

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

In [14]:

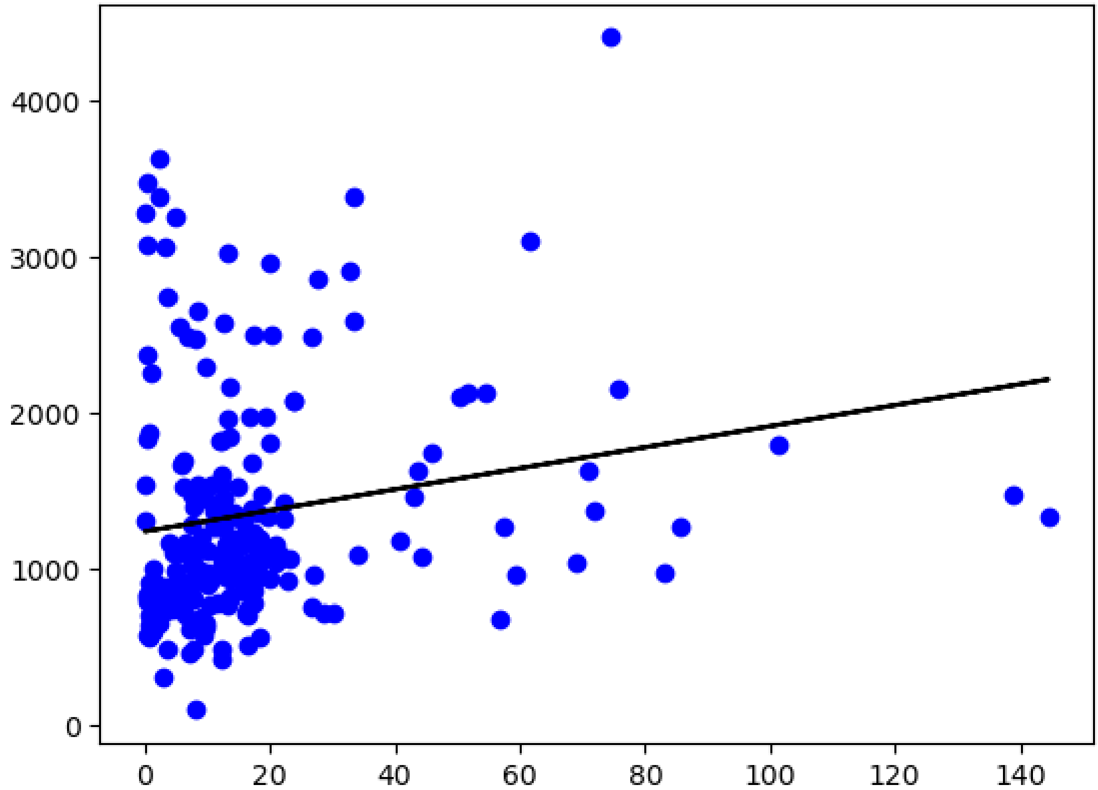
```
X_train,x_test,y_train,y_test = train_test_split(X,y,train_size=0.65)
regr = LinearRegression()
regr.fit(X_train,y_train)
print(regr.score(x_test, y_test))
```

```
0.024794843823667034
```

In [15]:

```

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



In [16]:

```
coeff_df=pd.DataFrame(regr.coef_)
coeff_df
```

Out[16]:

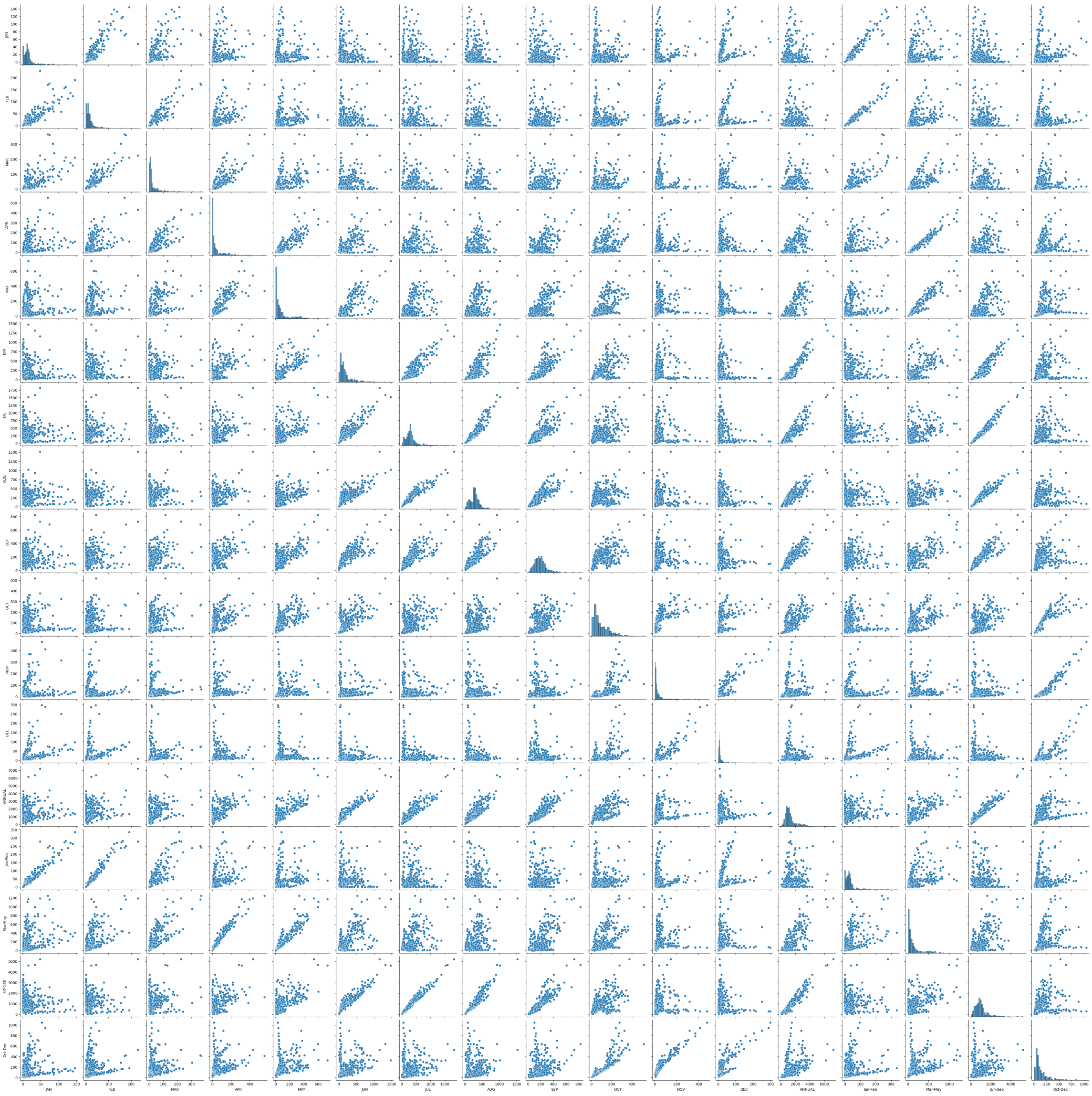
	0
0	6.748372

In [17]:

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

Out[17]:

<seaborn.axisgrid.PairGrid at 0x210f8602080>

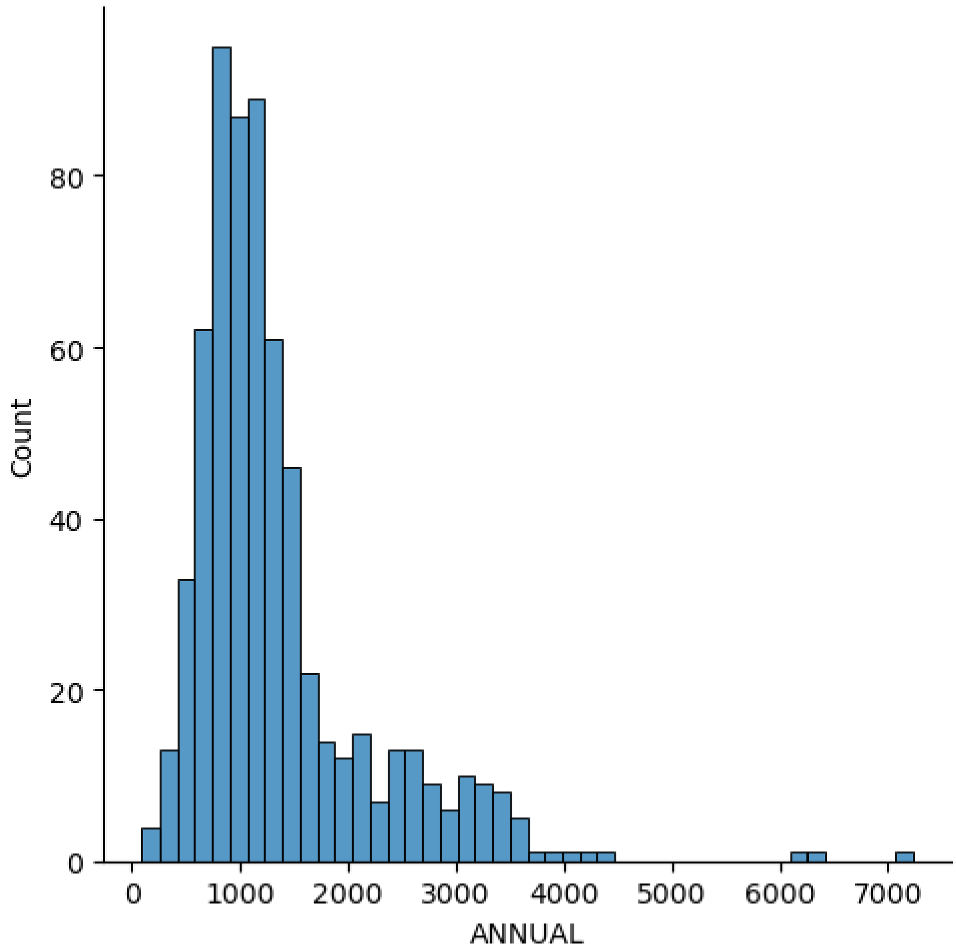


In [18]:

```
sns.displot(df['ANNUAL'])
```

Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x2108fb31210>
```



In [19]:

```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
```

In [20]:

```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
train_score_ridge = ridgeReg.score(X_train,y_train)
test_score_ridge = ridgeReg.score(x_test,y_test)
print('\nRidge model\n')
print('Train score for ridge model is {}'.format(train_score_ridge))
print('Test score for ridge model is {}'.format(test_score_ridge))
```

```
Ridge model

Train score for ridge model is 0.025710107063416476
Test score for ridge model is 0.024795270334125208
```

In [21]:

```
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(x_test,y_test)
print('\nLasso Model\n')
print('Train score for lasso model is {}'.format(train_score_lasso))
print('Test score for lasso model is {}'.format(test_score_lasso))
```

```
Lasso Model

Train score for lasso model is 0.025709834964721345
Test score for lasso model is 0.024820742347072322
```

In [22]:

```
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(X,y)
```

Out[22]:

```
▾ ElasticNet
ElasticNet()
```

In [23]:

```
print(regr.coef_)
```

```
[6.48002837]
```

In [24]:

```
print(regr.intercept_)
```

```
[1228.02820315]
```

In [25]:

```

y_pred_elastic = regr.predict(X_train)
mean_squared_error = np.mean((y_pred_elastic-y_train)**2)
print('Mean squared error on test set',mean_squared_error)
```

```
Mean squared error on test set 826039.9941899101
```

In [26]:

```
regr.score(X_train,y_train)
```

Out[26]:

```
0.025435858893807617
```

CONCLUSION:

```
        Based on accuracy of all models we can conclude that Linear Regression is
the best model for given dataset
```

In [ ]:

```

```