100 YEARS RAINFALL

PROBLEM STATEMENT: TO PREDICT AND ANALYZE THE RAINFALL FOR AN YEAR IN DISTRICT.

In [2]:

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

DATA COLLECTION

In [3]:

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

Out[3]:

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

641 rows × 19 columns

DATA CLEANING

In [4]:

```
df.head()
```

Out[4]:

	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	35
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	45
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	45
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	31
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	56

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

Out[5]:

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

In [6]:
```
df.describe()
```

Out[6]:

	JAN	FEB	MAR	APR	MAY	JUN	
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.00
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.03
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.36
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.60
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.40
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.70
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.80
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.90

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
```

In [8]:
```
df.shape
```

Out[8]: (641, 19)

In [9]:
```
features=df[2:13]
target=df.columns[14]
```

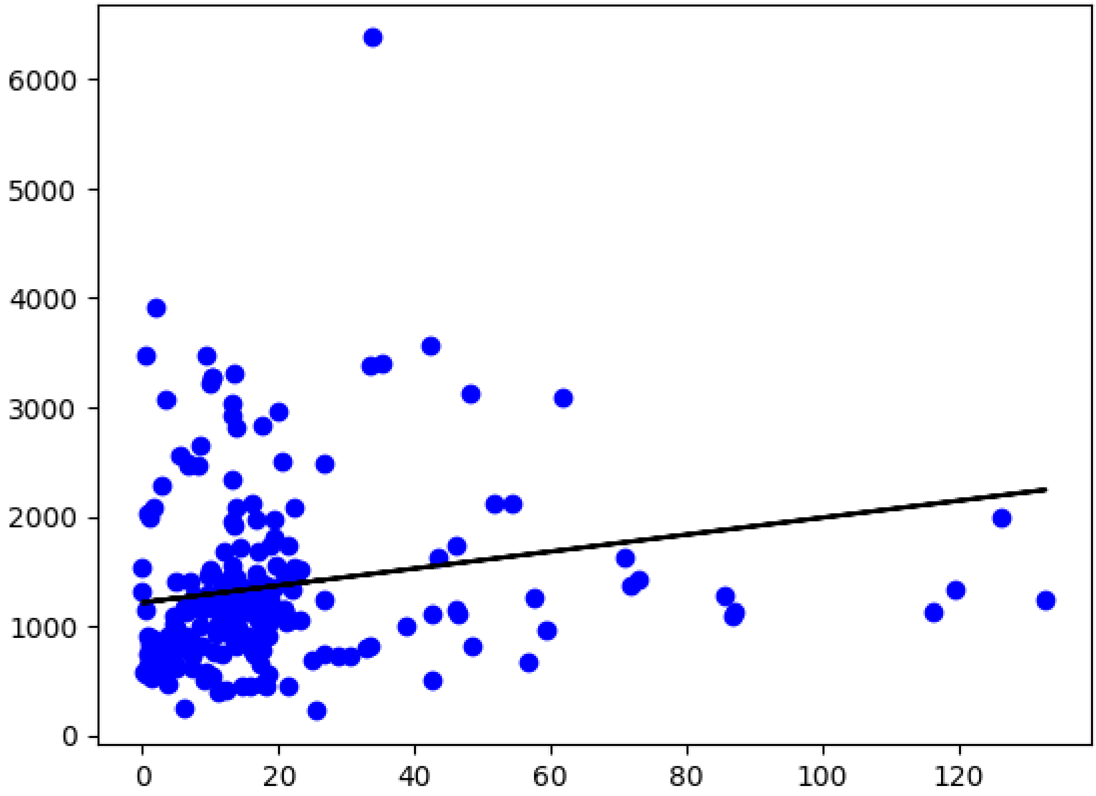
In [10]:
```
df.fillna(method='ffill',inplace=True)
```

In [14]:
```
X = np.array(df['JAN']).reshape(-1,1)
y = np.array(df['ANNUAL']).reshape(-1,1)
```

In [19]:
```
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))
```

```
0.0028369256345803784
```

In [20]:
```
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 [21]:
```
coeff_df=pd.DataFrame(regr.coef_)
coeff_df
```

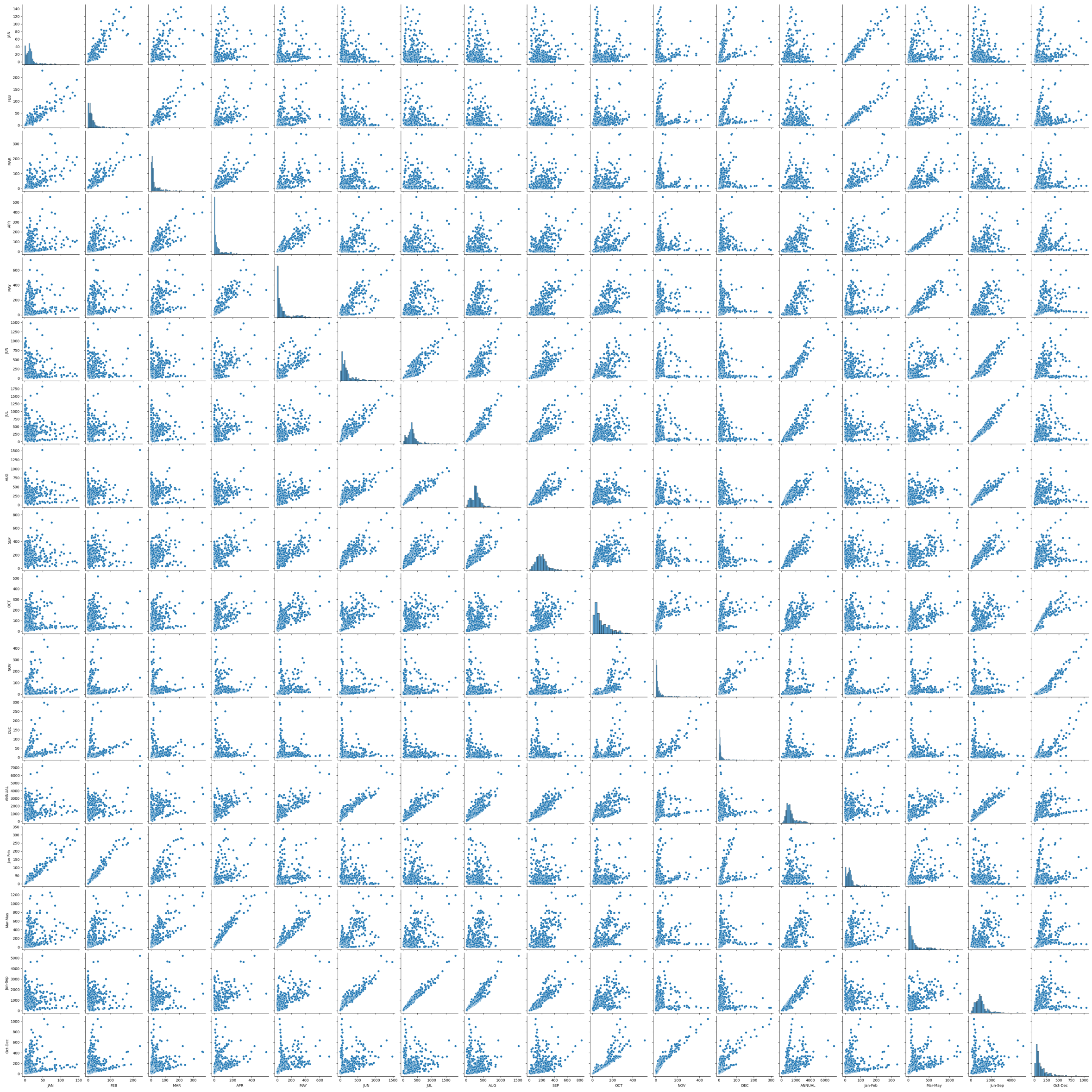
Out[21]:

	0
0	7.765861

EDA REPORT

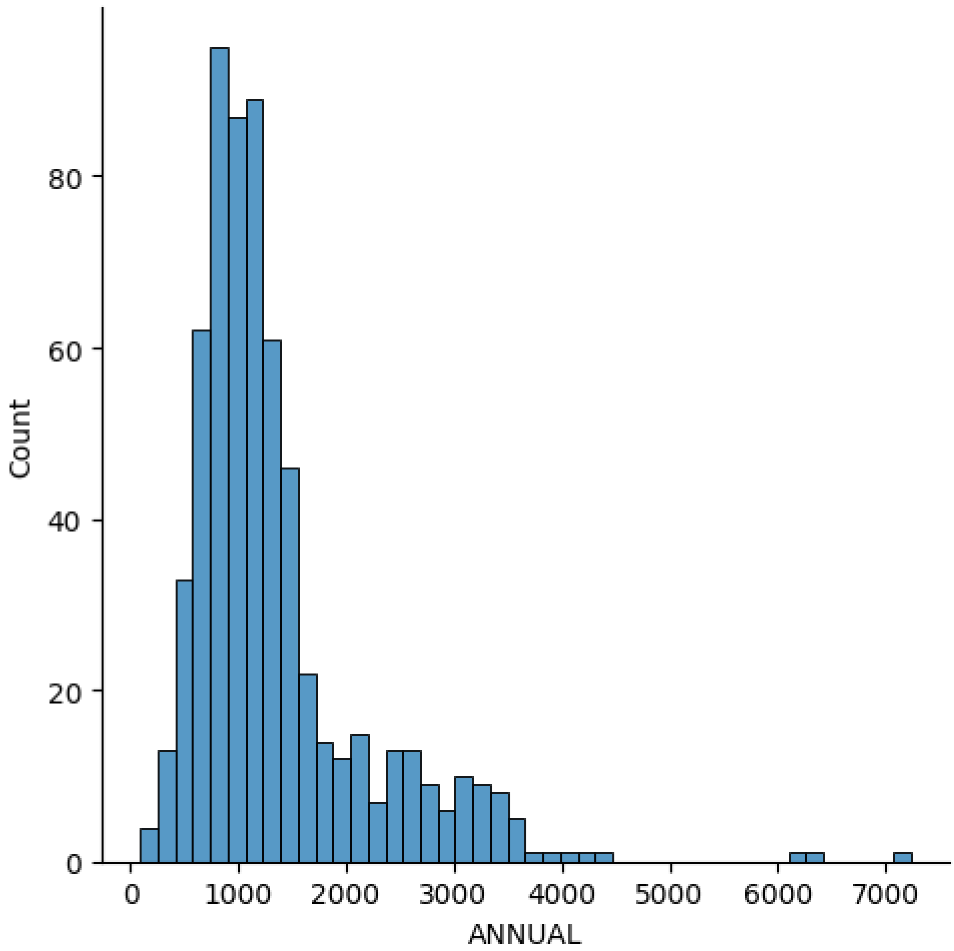
In [22]:
```
sns.pairplot(df)
```

Out[22]: <seaborn.axisgrid.PairGrid at 0x2308906ee00>



In [23]:
```
sns.displot(df['ANNUAL'])
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x2309d6df6a0>



RIDGE

In [24]:
```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
```

In [25]:
```
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 0.036524345824457O7
Test score for ridge model is 0.002839010973817113
```

LASSO

In [26]:
```
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 0.036524032343096646
Test score for lasso model is 0.002948290745804938
```

ELASTICNET

In [30]:
```python
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(X,y)
```

Out[30]:
```
▾ ElasticNet

ElasticNet()
```

In [31]:
```python
print(regr.coef_)
```

```
[6.48002837]
```

In [32]:
```python
print(regr.intercept_)
```

```
[1228.02820315]
```

In [33]:
```python
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 744132.7029669879
```

In [34]:
```python
regr.score(X_train,y_train)
```

Out[34]:
```
0.035458694514919675
```

CONCLUSION

THE SCORE OF THE REGRESSION IS COMPARED TO ALL (RIDGE,LASSO AND ELASTICNET ARE HIGHER THEN LINEAR REGRESSION).