PROJECT 3

PROBLEM STATEMENT:

Importing packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

READ THE DATA

In [2]:
```
df=pd.read_csv(r"C:\Users\USER\Desktop\rainfall in india 1901-2015.csv")
df
```

Out[2]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	
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	558.2	33.6	3373.2	136.3	560.3	1
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	359.0	160.5	3520.7	159.8	458.3	2
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	284.4	225.0	2957.4	156.7	236.1	1
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	308.7	40.1	3079.6	24.1	506.9	1
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	25.4	344.7	2566.7	1.3	309.7	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	184.3	14.9	1533.7	7.9	196.2	1
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	12.4	8.8	1405.5	19.3	99.6	1
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	78.1	26.7	1426.3	60.6	131.1	1
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	59.0	62.3	1395.0	69.3	76.7	
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	231.0	159.0	1642.9	2.7	223.9	

4116 rows × 19 columns

DATA COLLECTION AND PREPROCESSING

In [3]:
```
df.head()
```

Out[3]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep
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	558.2	33.6	3373.2	136.3	560.3	1696.3
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	359.0	160.5	3520.7	159.8	458.3	2185.9
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	284.4	225.0	2957.4	156.7	236.1	1874.0
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	308.7	40.1	3079.6	24.1	506.9	1977.6
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	25.4	344.7	2566.7	1.3	309.7	1624.9

In [4]:
```
df.tail()
```

Out[4]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Ju S
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	184.3	14.9	1533.7	7.9	196.2	1013
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	12.4	8.8	1405.5	19.3	99.6	1119
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	78.1	26.7	1426.3	60.6	131.1	1057
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	59.0	62.3	1395.0	69.3	76.7	958
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	231.0	159.0	1642.9	2.7	223.9	860

In [5]:
```
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 [6]:
```
df.shape
```

Out[6]:
```
(4116, 19)
```

In [7]:
```
df.isnull().any()
```

Out[7]:
```
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 [8]:
```
df.isnull().sum()
```

Out[8]:
```
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 [9]:
```
df.fillna(method='ffill',inplace=True)
```

In [10]:
```
df.describe()
```

Out[10]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SE
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.0000
mean	1958.218659	18.957240	21.823251	27.415379	43.160641	85.788994	230.567979	347.177235	290.239796	197.5247
std	33.140898	33.576192	35.922602	47.045473	67.816588	123.220150	234.896056	269.321089	188.785639	135.5090
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.1000
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.475000	175.900000	155.850000	100.5750
50%	1958.000000	6.000000	6.700000	7.900000	15.700000	36.700000	138.900000	284.800000	259.400000	174.0000
75%	1987.000000	22.200000	26.800000	31.400000	50.125000	97.400000	306.150000	418.325000	377.800000	266.2250
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.0000

In [11]:
```
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          4116 non-null   float64
 3   FEB          4116 non-null   float64
 4   MAR          4116 non-null   float64
 5   APR          4116 non-null   float64
 6   MAY          4116 non-null   float64
 7   JUN          4116 non-null   float64
 8   JUL          4116 non-null   float64
 9   AUG          4116 non-null   float64
 10  SEP          4116 non-null   float64
 11  OCT          4116 non-null   float64
 12  NOV          4116 non-null   float64
 13  DEC          4116 non-null   float64
 14  ANNUAL       4116 non-null   float64
 15  Jan-Feb      4116 non-null   float64
 16  Mar-May      4116 non-null   float64
 17  Jun-Sep      4116 non-null   float64
 18  Oct-Dec      4116 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

In [12]:
```
df['Jan-Feb'].value_counts()
```

Out[12]:
```
Jan-Feb
0.0     238
0.1      80
0.2      52
0.3      38
0.4      32
        ...
23.3      1
95.2      1
76.9      1
66.5      1
69.3      1
Name: count, Length: 1220, dtype: int64
```

In [13]:
```
df['Mar-May'].value_counts()
```

Out[13]:
```
Mar-May
0.0      29
0.1      13
0.3      11
8.3      11
11.5     10
         ..
246.3     1
248.1     1
151.3     1
249.5     1
223.9     1
Name: count, Length: 2262, dtype: int64
```

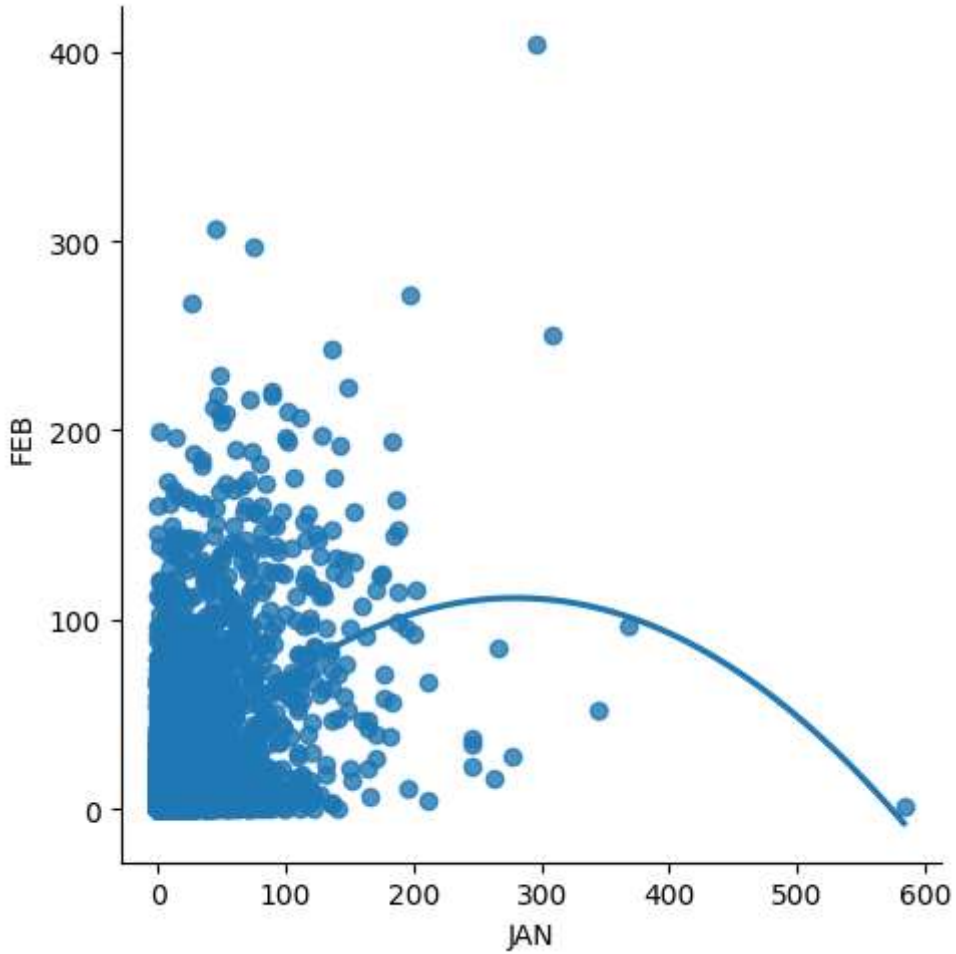
In [14]:
```
df['Jun-Sep'].value_counts()
```

Out[14]:
```
Jun-Sep
434.3     4
334.8     4
573.8     4
613.3     4
1082.3    3
         ..
301.6     1
380.9     1
409.3     1
229.4     1
958.5     1
Name: count, Length: 3683, dtype: int64
```

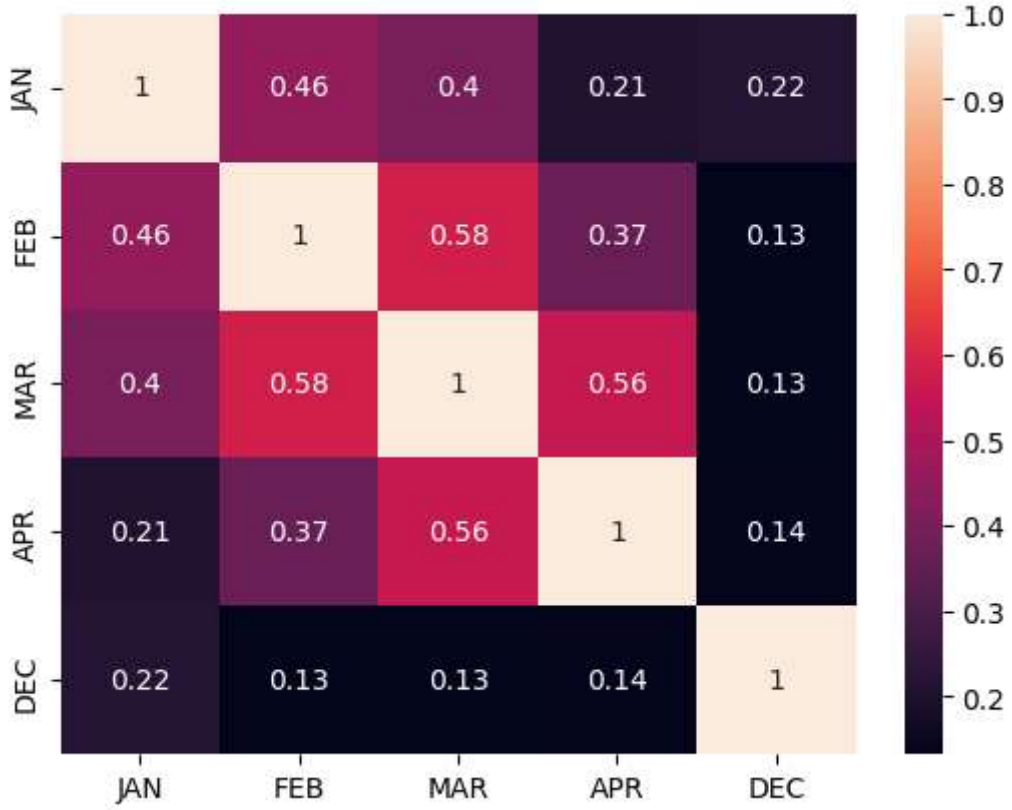
In [15]:
```
df['Oct-Dec'].value_counts()
```

Out[15]:
```
Oct-Dec
0.0      16
0.1      15
0.5      13
0.6      12
0.7      11
         ..
191.5     1
124.5     1
139.1     1
41.5      1
555.4     1
Name: count, Length: 2389, dtype: int64
```

In [22]:
```
sns.lmplot(x='JAN',y='FEB',order=2,data=df,ci=None)
plt.show()
```



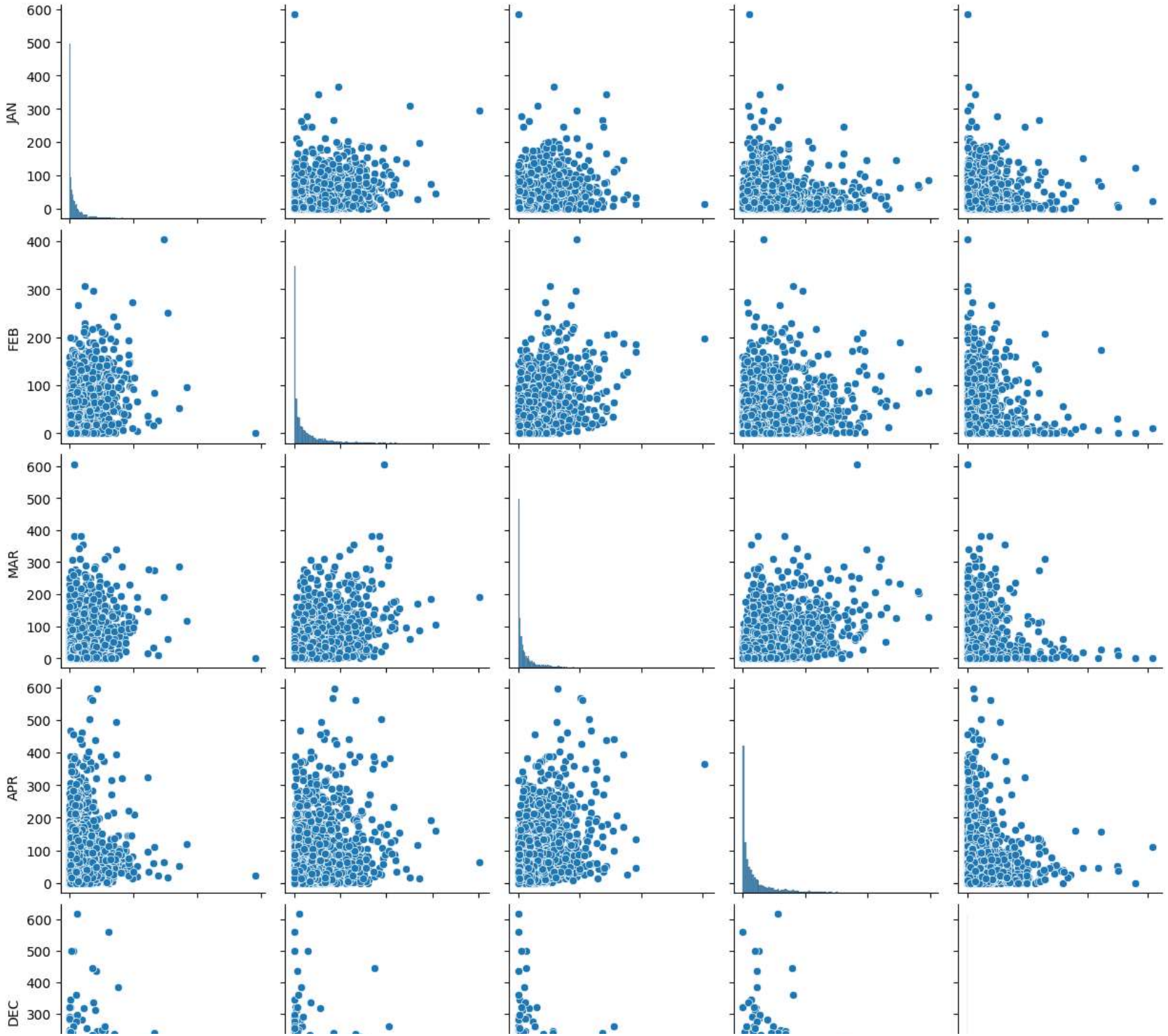
In [16]:
```
df=df[['JAN','FEB','MAR','APR','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```

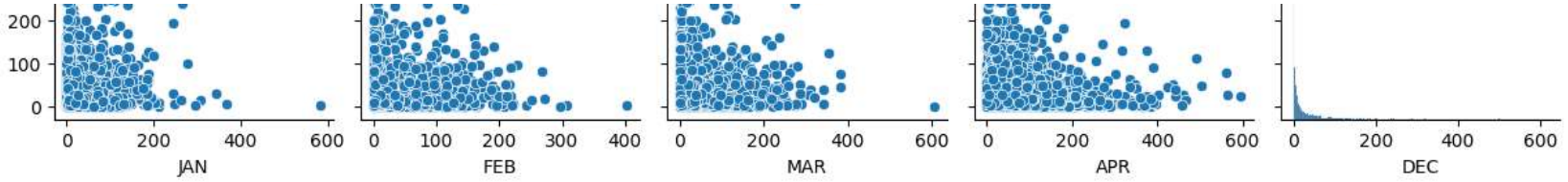


In [17]:
```
df.columns
```

Out[17]:
```
Index(['JAN', 'FEB', 'MAR', 'APR', 'DEC'], dtype='object')
```

In [18]:
```
sns.pairplot(df)
plt.show()
```


JAN
FEB
MAR
APR
DEC
0
100
200
300
400
500
600



LINEAR REGRESSION

In [23]:
```
x=np.array(df['FEB']).reshape(-1,1)
y=np.array(df['JAN']).reshape(-1,1)
```

In [24]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

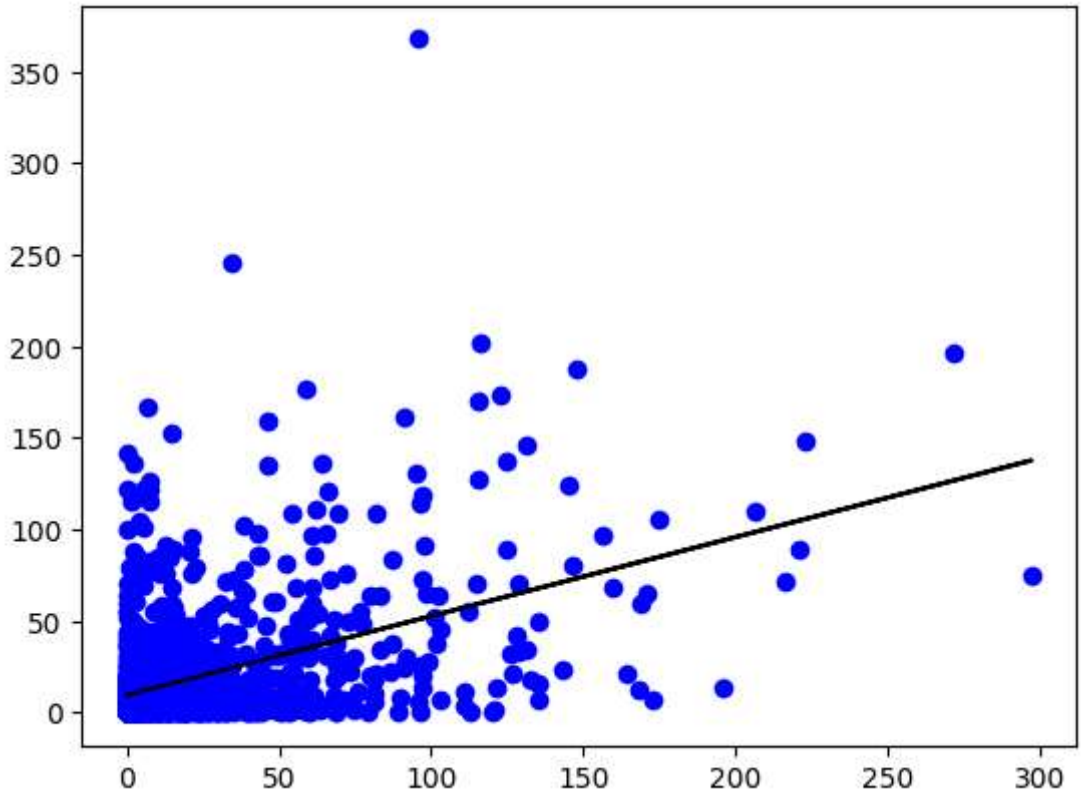
In [25]:
```
from sklearn.linear_model import LinearRegression
lin=LinearRegression()
lin.fit(x_train,y_train)
print(lin.score(x_test,y_test))
```

```
0.20614384038054023
```

In [27]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
lin.fit(x_train,y_train)
lin.fit(x_train,y_train)
```

Out[27]:
```
LinearRegression
LinearRegression()
```

In [29]:
```
y_pred=lin.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [31]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print('r2score:',r2)
```

```
r2score: 0.20274393027607263
```

RIDGE REGRESSION

In [39]:
```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [40]:
```
features = df.columns[0:2]
target = df.columns[-1]
#X and y values
x = df[features].values
y = df[target].values
#splot
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=17)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
#Scale features
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```
The dimension of X_train is (2881, 2)
The dimension of X_test is (1235, 2)
```

In [41]:
```
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("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
Ridge model:

The train score for ridge model is 0.046304274733072526
The test score for ridge model is 0.053755295092347666
```

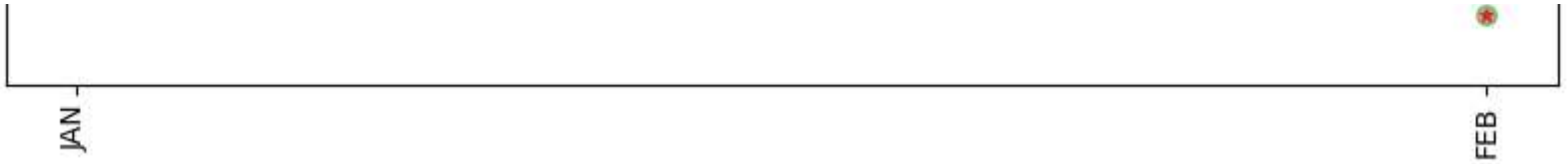
In [43]:
```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
actual = y_test
train_score_lr = lr.score(x_train, y_train)
test_score_lr = lr.score(x_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
#print(lr.score(y_test,prediction))
```

```
Linear Regression Model:

The train score for lr model is 0.04630466058267135
The test score for lr model is 0.0538161021443585
```

In [44]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Ridge; $\alpha =
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regr
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```


Ridge; $\alpha = 10$
Linear Regression
7
6
5
4
3



Lasso Regression

In [45]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

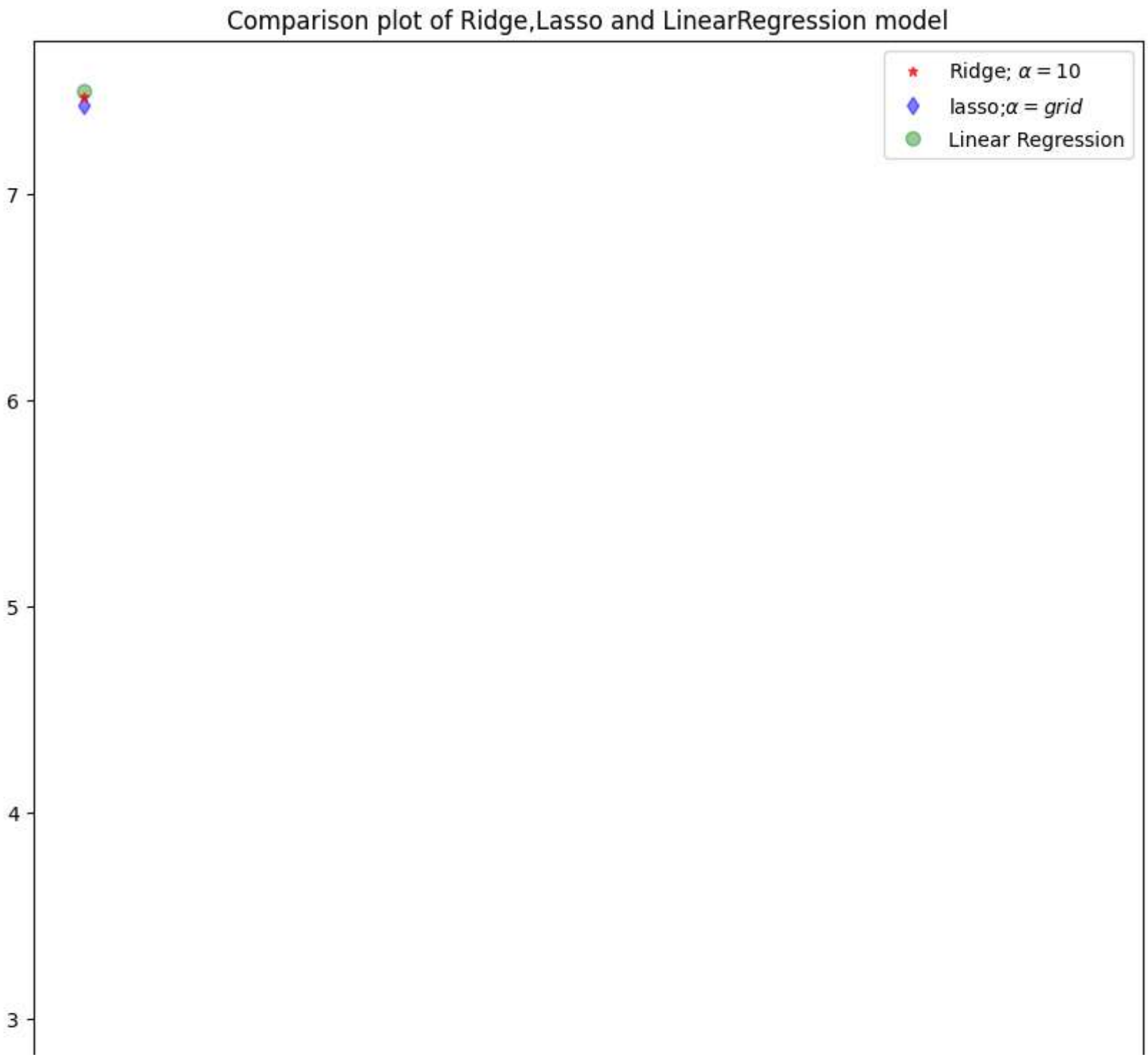
```
0.04629672768152171
0.05378649687813064
```

In [46]:
```
print("\nLasso model:\n")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_ls=lasso.score(x_train,y_train)
test_score_ls=lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso model:

The train score for ls model is 0.0
The test score for ls model is -0.0005263316941488405
```

In [47]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*', markersize=5, color='red',label=r'Ridg
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso;$\alpha=
plt.plot(features, lr.coef_, alpha=0.4, linestyle='none', marker='o', markersize=7, color='green',label='Linea
plt.xticks(rotation=90)
plt.legend()
plt.title('Comparison plot of Ridge,Lasso and LinearRegression model')
plt.show()
```


Comparison plot of Ridge,Lasso and LinearRegression model
Ridge; $\alpha = 10$
lasso;$\alpha = grid$
Linear Regression
7
6
5
4
3

Elastic Net

In [51]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x_train,y_train)
print(regr.coef_)
print(regr.intercept_)
regr.score(x,y)
```

```
[4.79965199 2.25704115]
18.73707046164526
```

Out[51]:
```
-33.03296434682958
```

In [49]:
```
y_pred_elastic=regr.predict(x_train)
```

In [50]:
```
mean_squared_error=np.mean((y_pred_elastic- y_train)**2)
print("mean _squared_error",mean_squared_error)
```

```
mean _squared_error 1683.4027958654087
```

conclusion:

The given data is "Rain fall prediction".here we need to find the best fit model.as per the given data set i had applied different types of models....in which different type of models got different types of accuracies

In [ ]: