Problem statement:-

```
        To predict the best model for the given Rainfall dataset based on accuracy.
```

Data Collection

In [1]:

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

In [3]:

```
df=pd.read_csv(r"C:\Users\sowmika\Downloads\rainfall.csv")
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.head()
```

Out[4]:

	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 [5]:

```
df.tail()
```

Out[5]:

	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 [6]:

```
df.describe
```

Out[6]:

```
<bound method NDFrame.describe of                       SUBDIVISION  YEAR   
JAN    FEB    MAR    APR    MAY    JUN  
0     ANDAMAN & NICOBAR ISLANDS  1901  49.2   87.1  29.2    2.3  528.8  51
7.5  \
1     ANDAMAN & NICOBAR ISLANDS  1902   0.0  159.8  12.2    0.0  446.1  53
7.1   
2     ANDAMAN & NICOBAR ISLANDS  1903  12.7  144.0   0.0    1.0  235.1  47
9.9   
3     ANDAMAN & NICOBAR ISLANDS  1904   9.4   14.7   0.0  202.4  304.5  49
5.1   
4     ANDAMAN & NICOBAR ISLANDS  1905   1.3    0.0   3.3   26.9  279.5  62
8.7   
...                         ...   ...   ...    ...   ...    ...    ...   
...   
4111               LAKSHADWEEP  2011   5.1    2.8   3.1   85.9  107.2  15
3.6   
4112               LAKSHADWEEP  2012  19.2    0.1   1.6   76.8   21.2  32
7.0   
4113               LAKSHADWEEP  2013  26.2   34.4  37.5    5.3   88.3  42
6.2   
4114               LAKSHADWEEP  2014  53.2   16.1   4.4   14.9   57.4  24
4.1   
4115               LAKSHADWEEP  2015   2.2    0.5   3.7   87.1  133.1  29
6.6   

        JUL    AUG    SEP    OCT    NOV    DEC  ANNUAL  Jan-Feb  Mar-May  
0     365.1  481.1  332.6  388.5  558.2   33.6  3373.2    136.3    560.3   
\
1     228.9  753.7  666.2  197.2  359.0  160.5  3520.7    159.8    458.3   
2     728.4  326.7  339.0  181.2  284.4  225.0  2957.4    156.7    236.1   
3     502.0  160.1  820.4  222.2  308.7   40.1  3079.6     24.1    506.9   
4     368.7  330.5  297.0  260.7   25.4  344.7  2566.7      1.3    309.7   
...     ...    ...    ...    ...    ...    ...     ...      ...      ...   
4111  350.2  254.0  255.2  117.4  184.3   14.9  1533.7      7.9    196.2   
4112  231.5  381.2  179.8  145.9   12.4    8.8  1405.5     19.3     99.6   
4113  296.4  154.4  180.0   72.8   78.1   26.7  1426.3     60.6    131.1   
4114  116.1  466.1  132.2  169.2   59.0   62.3  1395.0     69.3     76.7   
4115  257.5  146.4  160.4  165.4  231.0  159.0  1642.9      2.7    223.9   

      Jun-Sep  Oct-Dec  
0      1696.3    980.3  
1      2185.9    716.7  
2      1874.0    690.6  
3      1977.6    571.0  
4      1624.9    630.8  
...       ...      ...  
4111   1013.0    316.6  
4112   1119.5    167.1  
4113   1057.0    177.6  
4114    958.5    290.5  
4115    860.9    555.4  

[4116 rows x 19 columns]>
```

In [7]:

```
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 [8]:

```
df.shape
```

Out[8]:

```
(4116, 19)
```

In [9]:

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

Out[9]:

```
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 [10]:

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

In [11]:

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

Out[11]:

```
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 [12]:

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

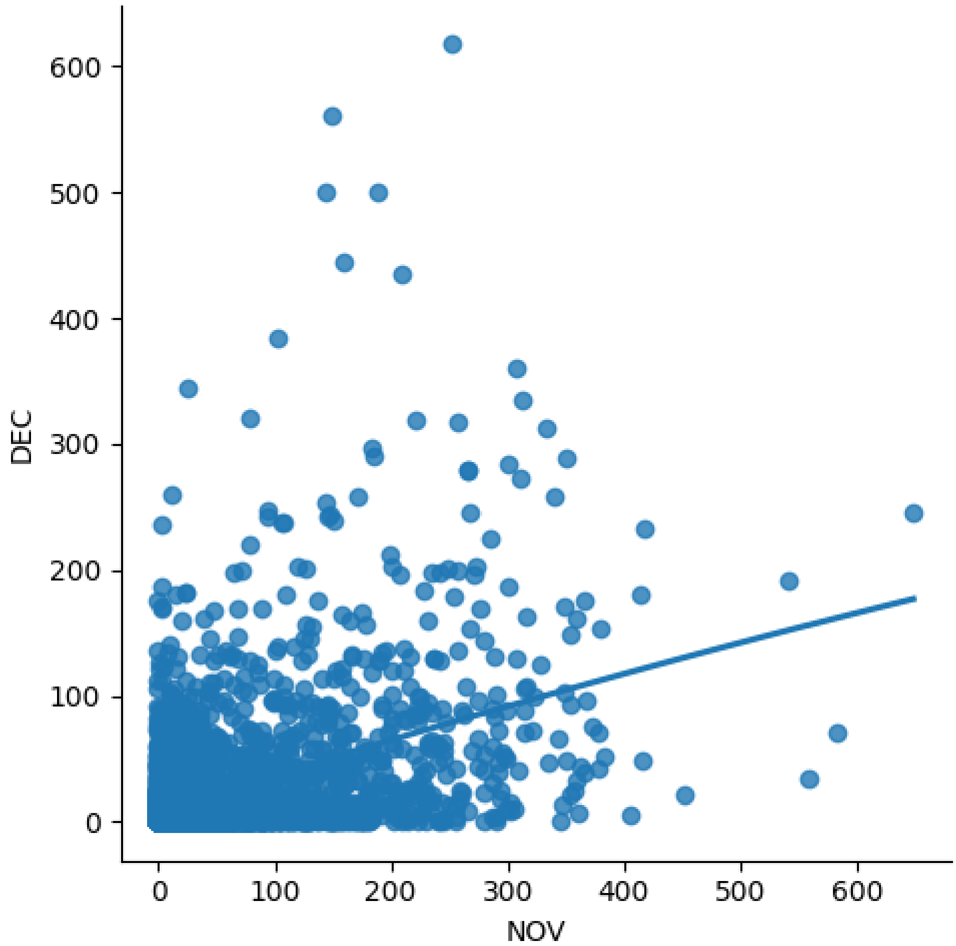
Out[12]:

```
YEAR
1963    36
2002    36
1976    36
1975    36
1974    36
        ..
1915    35
1918    35
1954    35
1955    35
1909    34
Name: count, Length: 115, dtype: int64
```

Exploratory data analysis

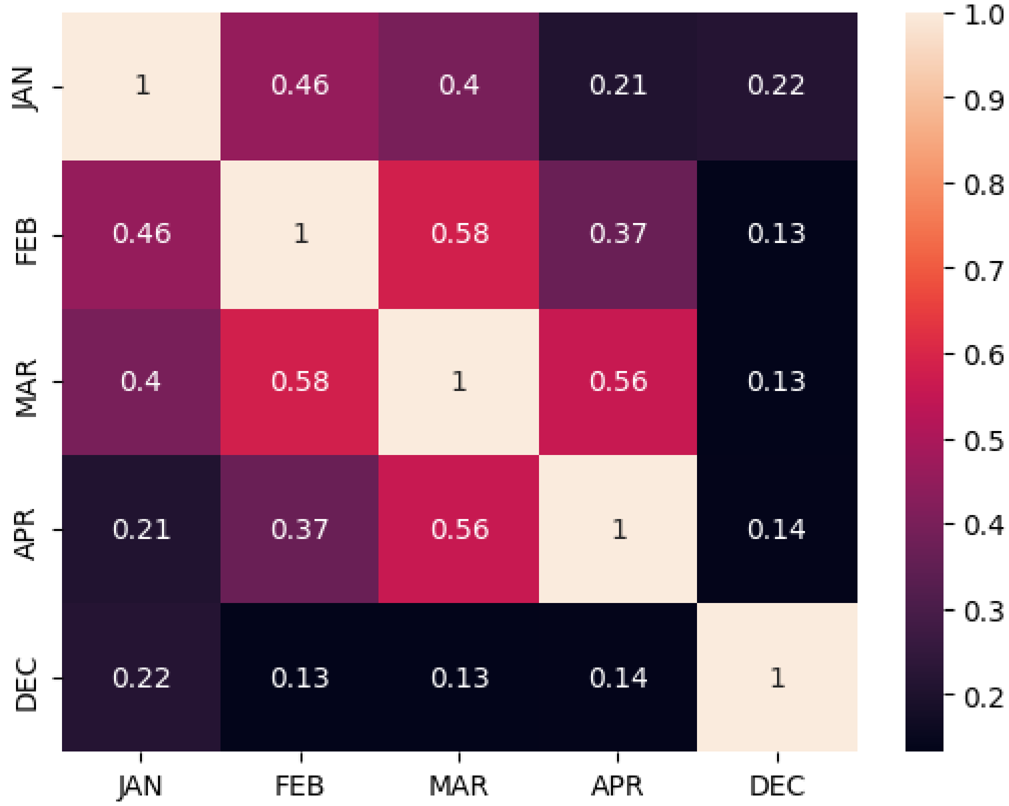
In [13]:

```
sns.lmplot(x='NOV',y='DEC',order=2,data=df,ci=None)
plt.show()
```



In [14]:

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

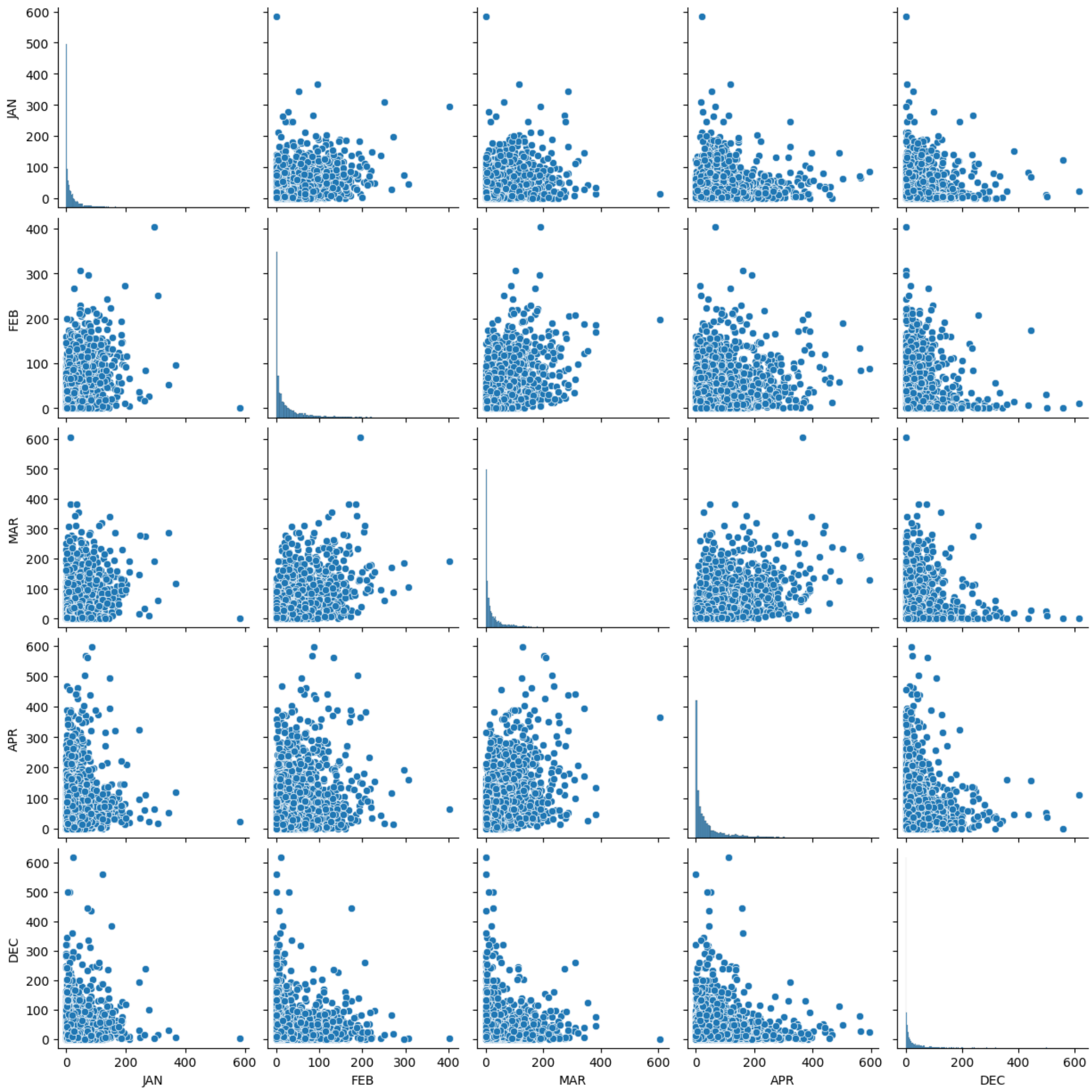


In [15]:

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

Out[15]:

```
<seaborn.axisgrid.PairGrid at 0x160dadc30d0>
```



Splitting the dataset into training data and test data

In [16]:

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

In [17]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

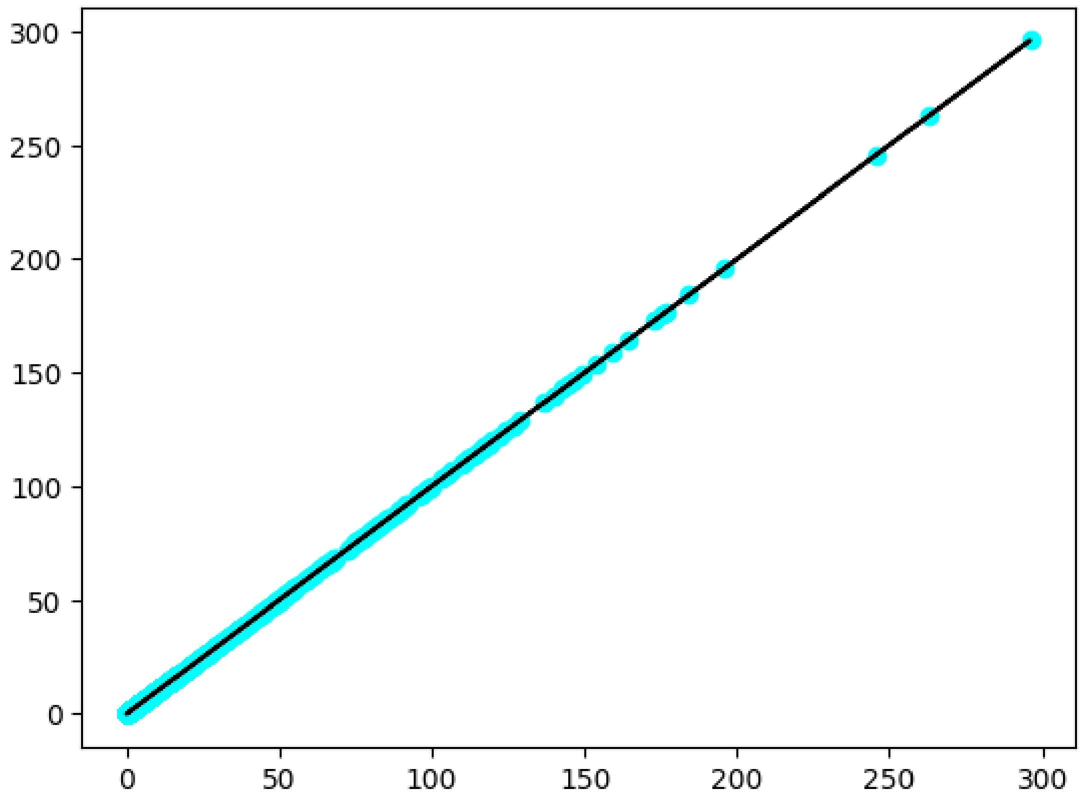
In [18]:

```
lrg=LinearRegression()
lrg.fit(x_train,y_train)
print(lrg.score(x_train,y_train))
```

1.0

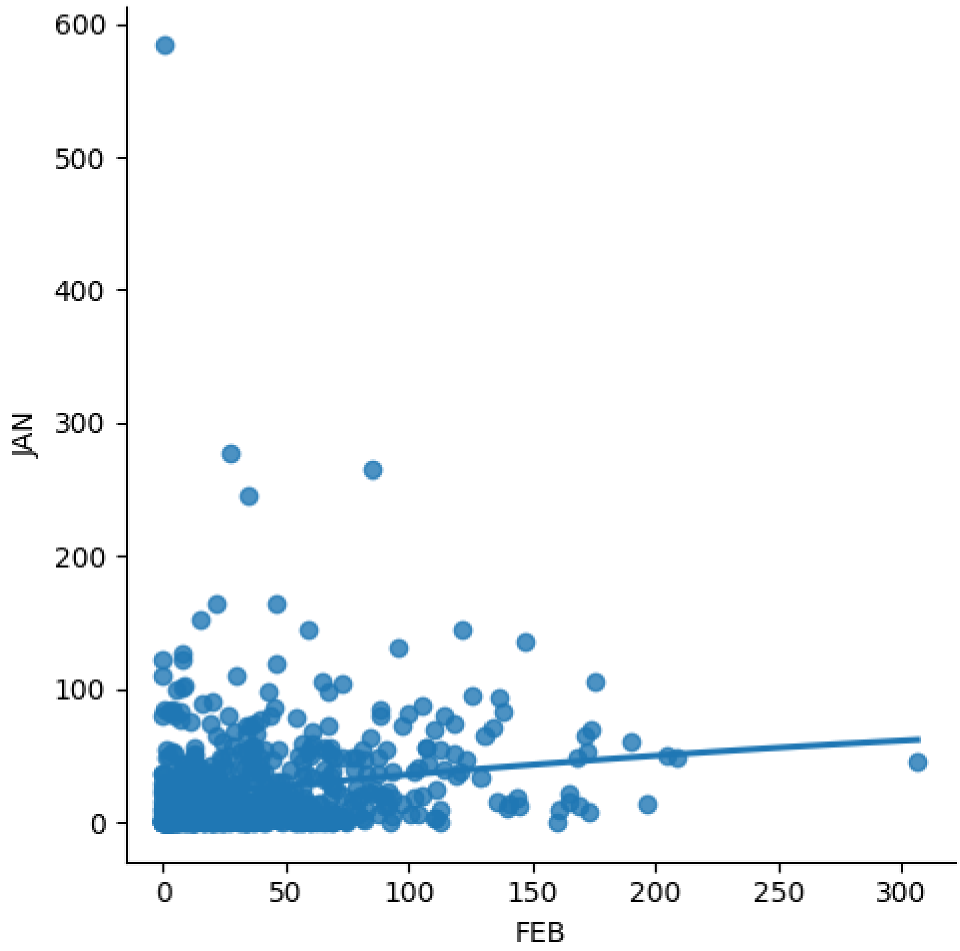
In [19]:

```
y_pred=lrg.predict(x_test)
plt.scatter(x_test,y_test,color='cyan')
plt.plot(x_test,y_pred,color='black')
plt.show()
```



In [20]:

```
df700=df[:][:700]
sns.lmplot(x='FEB',y='JAN',order=2,ci=None,data=df700)
plt.show()
```



In [21]:

```
df700.fillna(method='ffill',inplace=True)
```

In [22]:

```
x=np.array(df700['FEB']).reshape(-1,1)
y=x=np.array(df700['JAN']).reshape(-1,1)
```

In [23]:

```
df700.dropna(inplace=True)
```

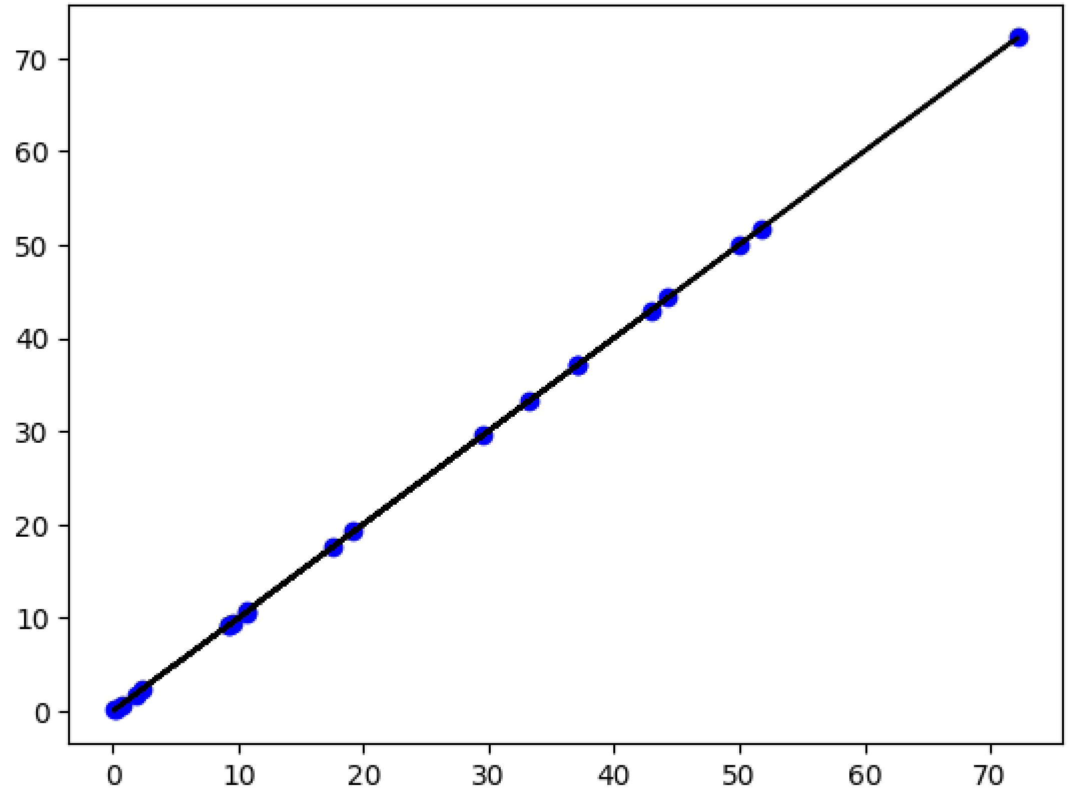
In [24]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.03)
lrg=LinearRegression()
lrg.fit(x_train,y_train)
print(lrg.score(x_test,y_test))
```

```
1.0
```

In [25]:

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



In [26]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [27]:

```
lrg=LinearRegression()
lrg.fit(x_train,y_train)
y_pred=lrg.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 1.0
```

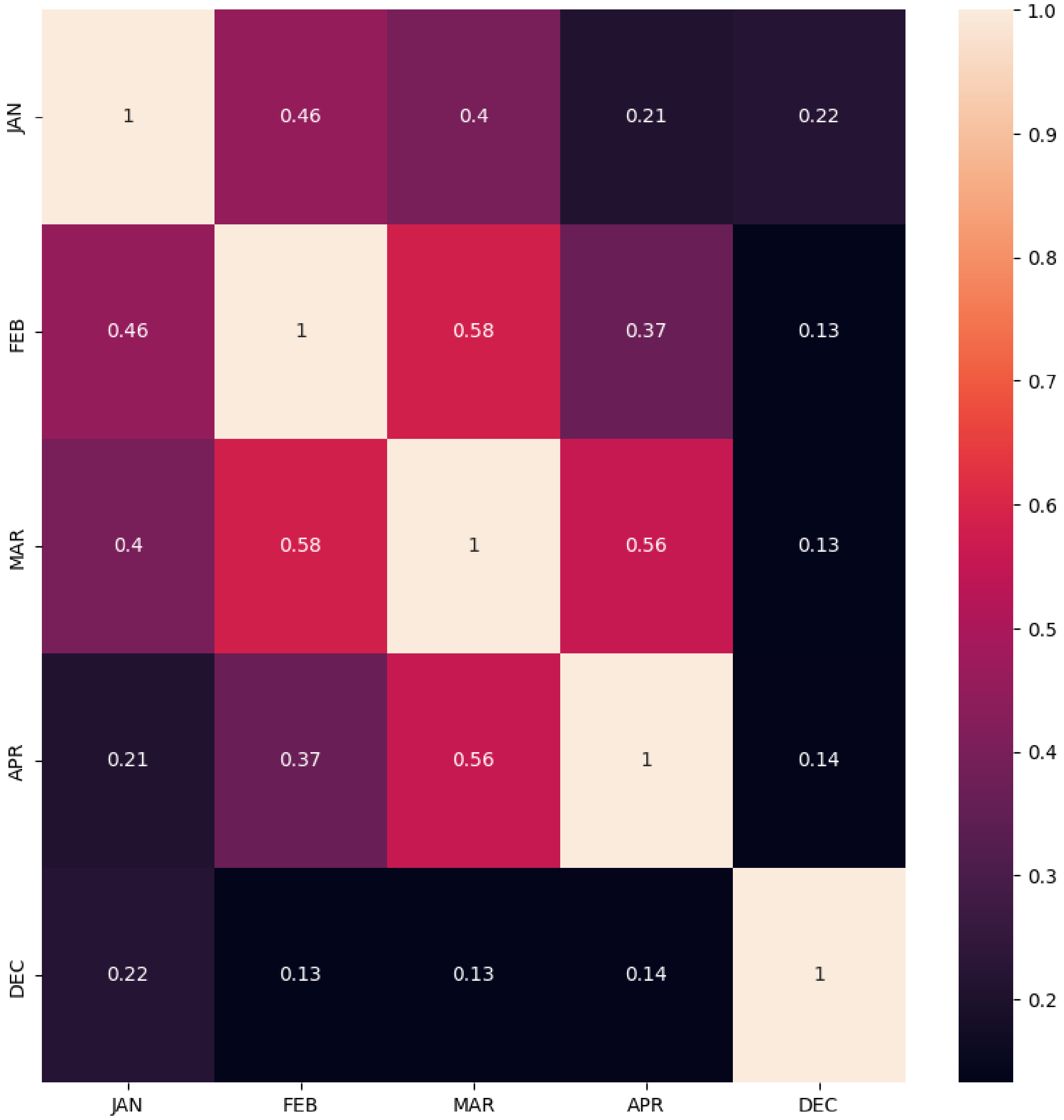
Ridge Regression

In [28]:

```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [29]:

```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [30]:

```
features=df.columns[0:5]
target=df.columns[-5]
```

In [31]:

```
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

```
The dimension of X_train is (2881, 5)
The dimension of X_test is (1235, 5)
```

In [32]:

```
lrg= LinearRegression()
#Fit model
lrg.fit(x_train, y_train)
actual = y_test
train_score_lrg = lrg.score(x_train, y_train)
test_score_lrg = lrg.score(x_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lrg))
print("The test score for lr model is {}".format(test_score_lrg))
```

```

Linear Regression Model:

The train score for lr model is 1.0
The test score for lr model is 1.0
```

In [33]:

```
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 0.9999999999856335
The test score for ridge model is 0.9999999999840021
```

Lasso Regression

In [35]:

```
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("\nLasso Model:\n")
print("The train score for lasso model is {}".format(train_score_ls))
print("The test score for lasso model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for lasso model is 0.9999147271297208
The test score for lasso model is 0.9999147248375002
```

In [36]:

```
plt.figure(figsize=(10,10))
```

Out[36]:

```
<Figure size 1000x1000 with 0 Axes>
```

```
<Figure size 1000x1000 with 0 Axes>
```

In [37]:

```
from sklearn.linear_model import LassoCV
```

In [38]:

```
from sklearn.linear_model import RidgeCV
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.9999999982836236
0.9999999986591067
```

Elastic Net

In [39]:

```
from sklearn.linear_model import ElasticNet
```

In [40]:

```
e=ElasticNet()
e.fit(x_train,y_train)
print(e.coef_)
print(e.intercept_)
print(e.score(x,y))
```

```
[9.99044548e-01 1.38835344e-05 4.58897515e-05 0.00000000e+00
 0.00000000e+00]
0.016565679683701262
0.9999991435191248
```

In [41]:

```
y_pred_elastic=e.predict(x_train)
```

In [51]:

```
mean_sqaured_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_sqaured_error)
```

```
0.0009226812593710402
```

Conclusion:

```
                   we conclude that "Ridge model" is the best model for Rainfall
Prediction dataset,because it got
    highest accuracy compared to other models
```