Problem statement:To predict How Best the DataFits,To Predict the accuracy of the

Rainfall based on the given features 1)Data collection

In [2]:

```
#Importing libraries
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 [4]:

```
#Reading data
df=pd.read_csv(r"C:\Users\sruth\OneDrive\Desktop\rainfall.csv")
df
```

Out[4]:

	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	2
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	4
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	4
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	4
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	7
...	...	...	...	...	...	...	...	...	...	
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	5
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	6
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	3
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	5
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	2

641 rows × 19 columns

2)Data Cleaning and Preprocessing

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
df.tail()
```

Out[6]:

	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 [8]:

```
df.shape
```

Out[8]:

```
(641, 19)
```

In [9]:

```
df.describe
```

Out[9]:

```
<bound method NDFrame.describe of                  STATE_UT_NAME
DISTRICT    JAN   FEB    MAR    APR  \
0    ANDAMAN And NICOBAR ISLANDS         NICOBAR  107.3  57.9   65.2  11
7.0
1    ANDAMAN And NICOBAR ISLANDS   SOUTH ANDAMAN   43.7  26.0   18.6   9
0.5
2    ANDAMAN And NICOBAR ISLANDS   N & M ANDAMAN   32.7  15.9    8.6   5
3.4
3            ARUNACHAL PRADESH           LOHIT   42.2  80.8  176.4  35
8.5
4            ARUNACHAL PRADESH      EAST SIANG   33.3  79.5  105.9  21
6.5
..                         ...             ...    ...   ...    ...    ...
...
636                     KERALA          IDUKKI   13.4  22.1   43.6  15
0.4
637                     KERALA        KASARGOD    2.3   1.0    8.4   4
6.9
638                     KERALA  PATHANAMTHITTA   19.8  45.2   73.9  18
4.9
639                     KERALA         WAYANAD    4.8   8.3   17.5   8
3.3
640                LAKSHADWEEP     LAKSHADWEEP   20.8  14.7   11.8   4
8.9

       MAY    JUN     JUL    AUG    SEP    OCT    NOV    DEC  ANNUAL  Jan
-Feb  \
0    358.5  295.5   285.0  271.9  354.8  326.0  315.2  250.9  2805.2    1
65.2
1    374.4  457.2   421.3  423.1  455.6  301.2  275.8  128.3  3015.7
69.7
2    343.6  503.3   465.4  460.9  454.8  276.1  198.6  100.0  2913.3
48.6
3    306.4  447.0   660.1  427.8  313.6  167.1   34.1   29.8  3043.8    1
23.0
4    323.0  738.3   990.9  711.2  568.0  206.9   29.5   31.7  4034.7    1
12.8
..     ...    ...     ...    ...    ...    ...    ...    ...     ...
...
636  232.6  651.6   788.9  527.3  308.4  343.2  172.9   48.1  3302.5
35.5
637  217.6  999.6  1108.5  636.3  263.1  234.9   84.6   18.4  3621.6
3.3
638  294.7  556.9   539.9  352.7  266.2  359.4  213.5   51.3  2958.4
65.0
639  174.6  698.1  1110.4  592.9  230.7  213.1   93.6   25.8  3253.1
13.1
640  171.7  330.2   287.7  217.5  163.1  157.1  117.7   58.8  1600.0
35.5

     Mar-May  Jun-Sep  Oct-Dec
0      540.7   1207.2    892.1
1      483.5   1757.2    705.3
2      405.6   1884.4    574.7
3      841.3   1848.5    231.0
4      645.4   3008.4    268.1
..       ...      ...      ...
636    426.6   2276.2    564.2
637    272.9   3007.5    337.9
638    553.5   1715.7    624.2
639    275.4   2632.1    332.5
```

In [10]:

```
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
```

```
640     232.4     998.5     333.6
```

In [11]:

```
[641 rows x 19 columns]>
df.isnull().sum()
```

Out[11]:

```
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 [12]:

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

In [13]:

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

Out[13]:

```
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
```

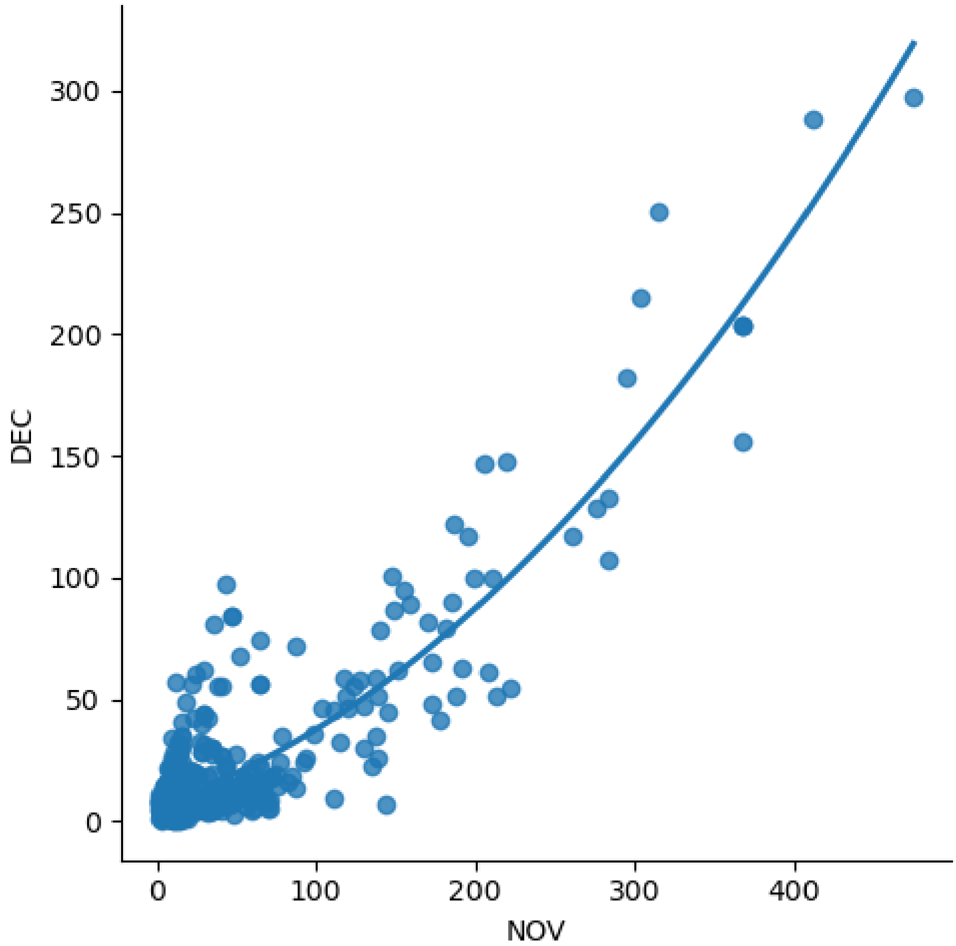
In [14]:

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

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call las
t)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3802, in I
ndex.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:138, in pandas.
_libs.index.IndexEngine.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:165, in pandas.
_libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashta
ble.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashta
ble.PyObjectHashTable.get_item()

KeyError: 'YEAR'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
t)
Cell In[14], line 1
----> 1 df['YEAR'].value_counts()

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3807, in DataFram
e.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3804, in I
ndex.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'YEAR'
```

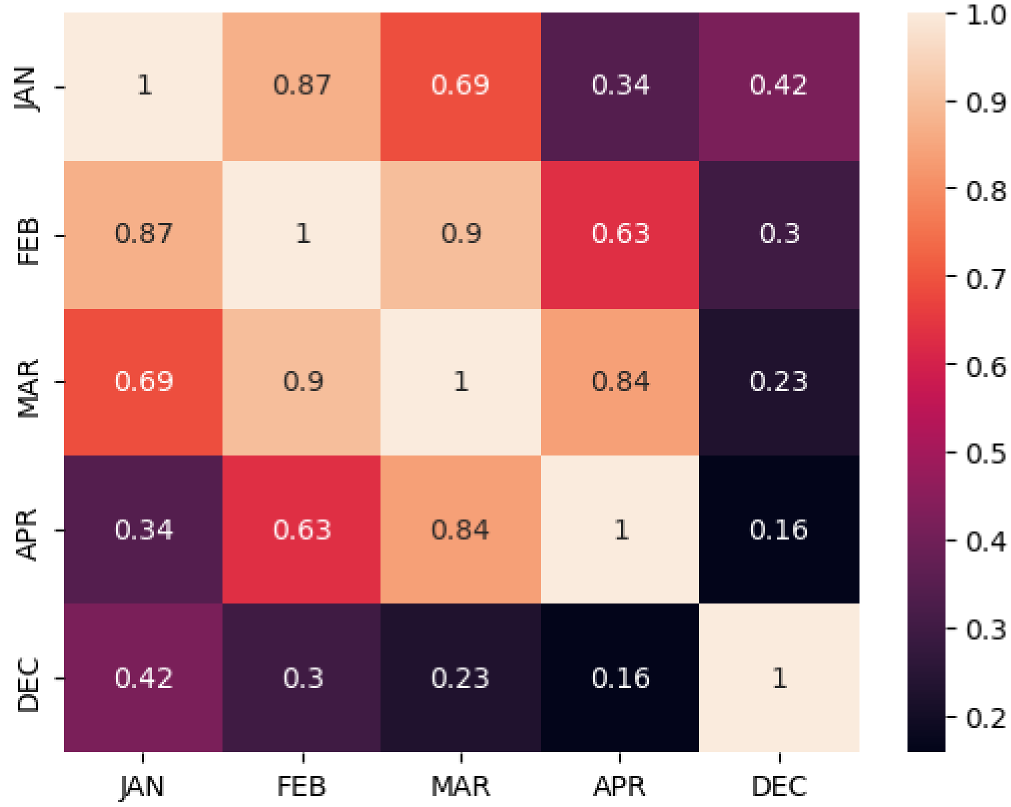
In [15]:

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



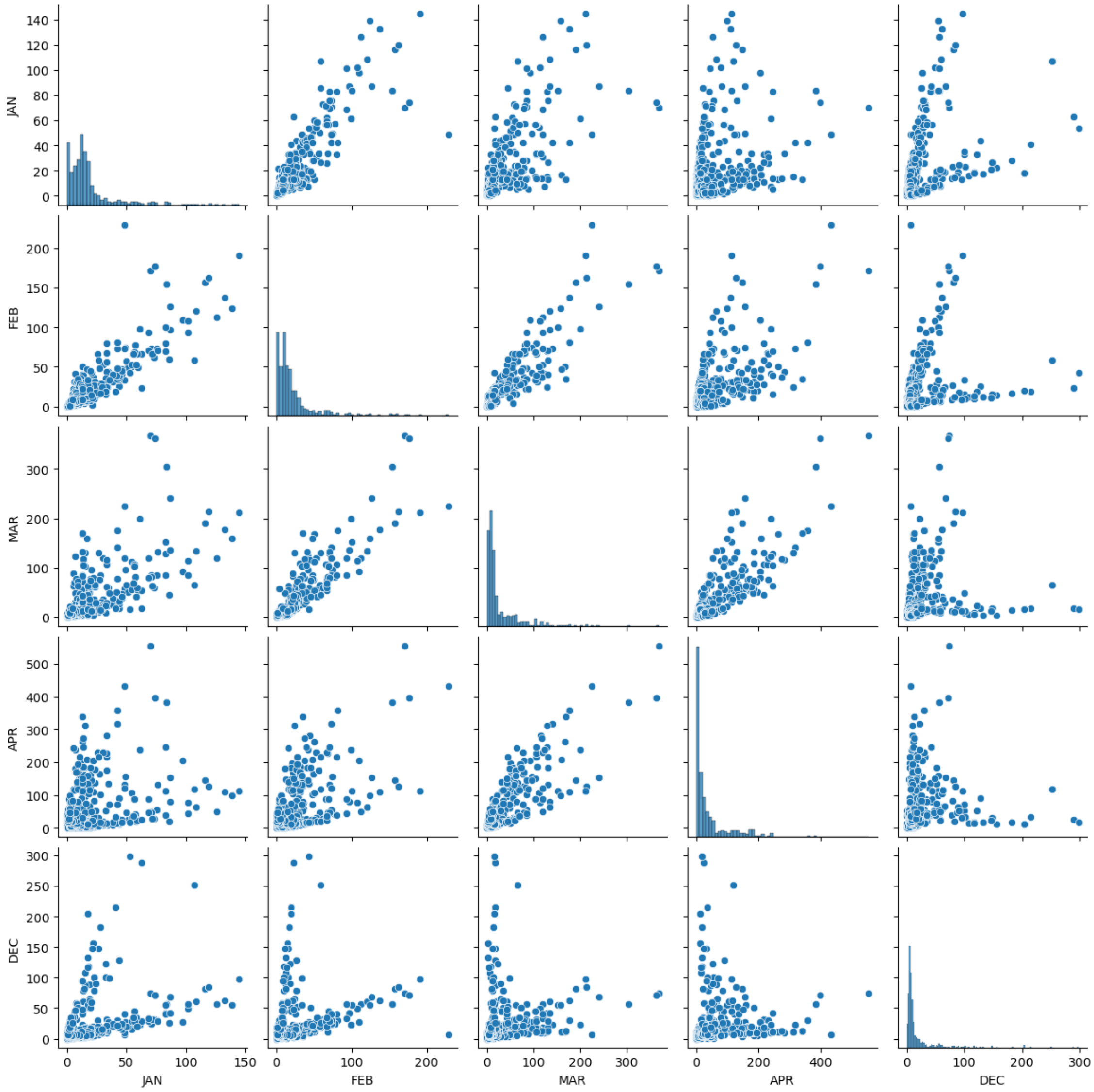
In [16]:

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



In [17]:

```
sns.pairplot(df)
plt.show()
```



In [18]:

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

In [19]:

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

In [20]:

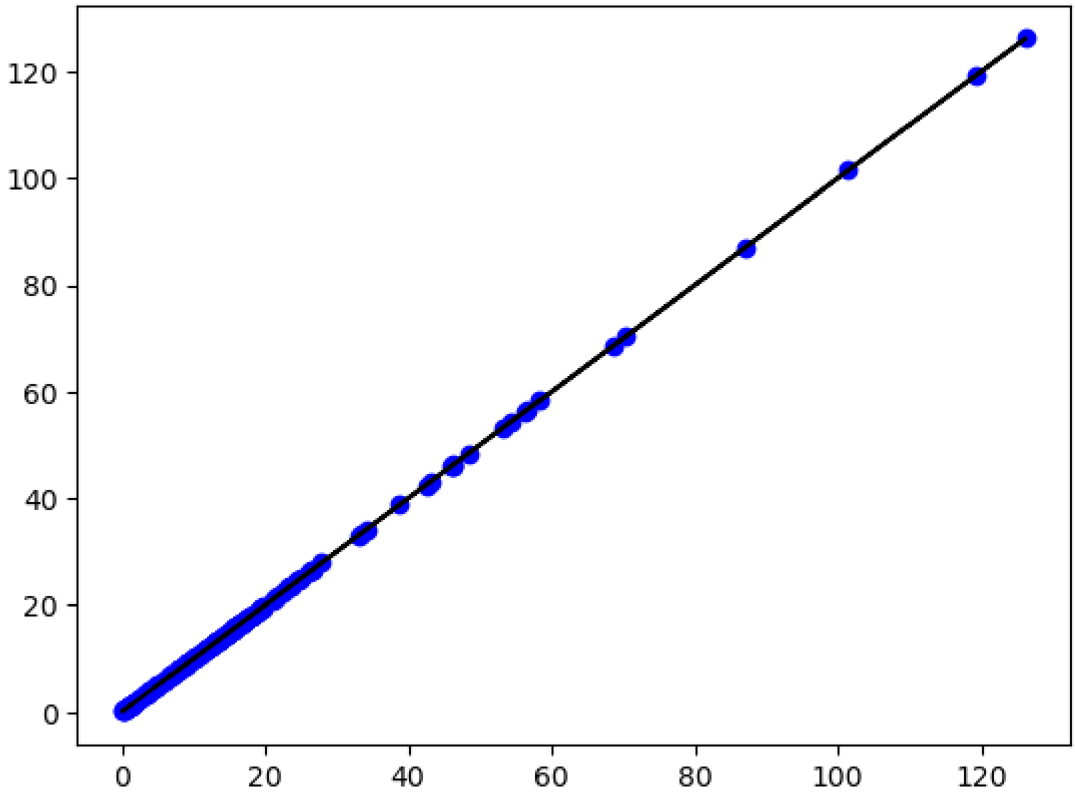
```
lin=LinearRegression()
lin.fit(x_train,y_train)
print(lin.score(x_train,y_train))
```

```
1.0
```

5)Exploring our Results

In [21]:

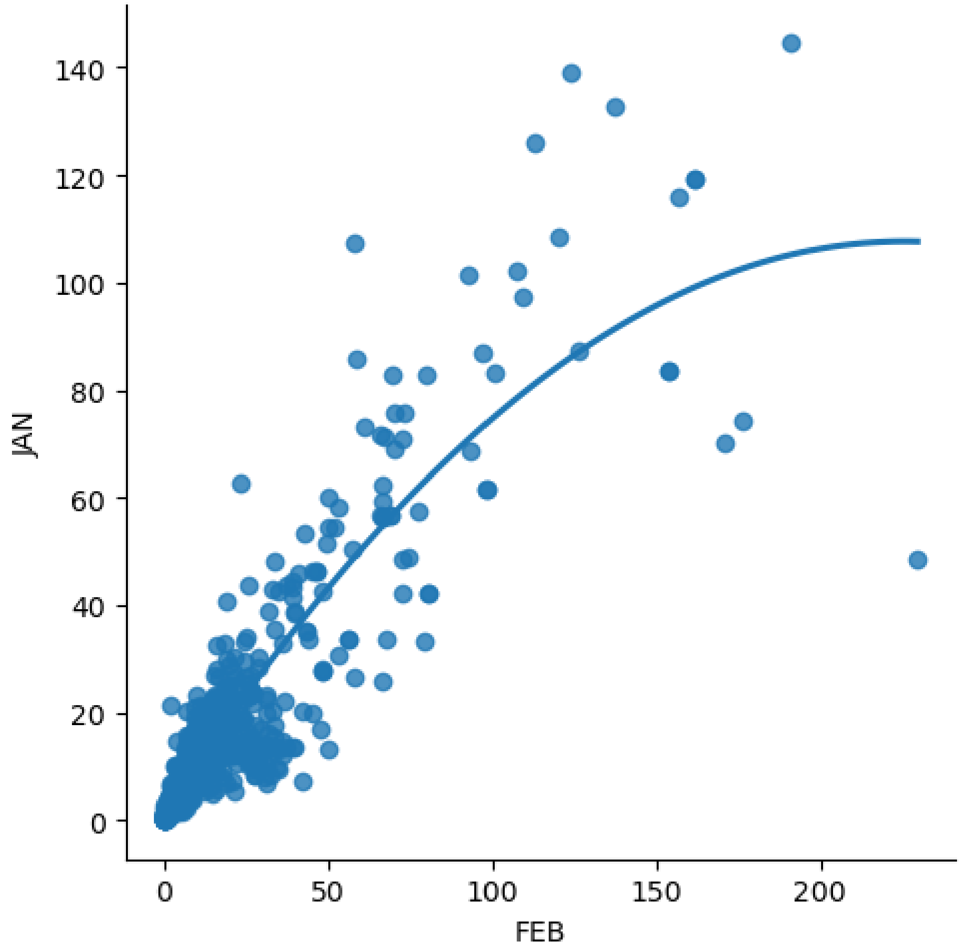
```
 y_pred=lin.predict(x_test)
plt.scatter(x_test,y_test,color='blue')
plt.plot(x_test,y_pred,color='black')
plt.show()
```



7)Working with subset of data

In [22]:

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



In [23]:

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

In [24]:

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

In [25]:

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

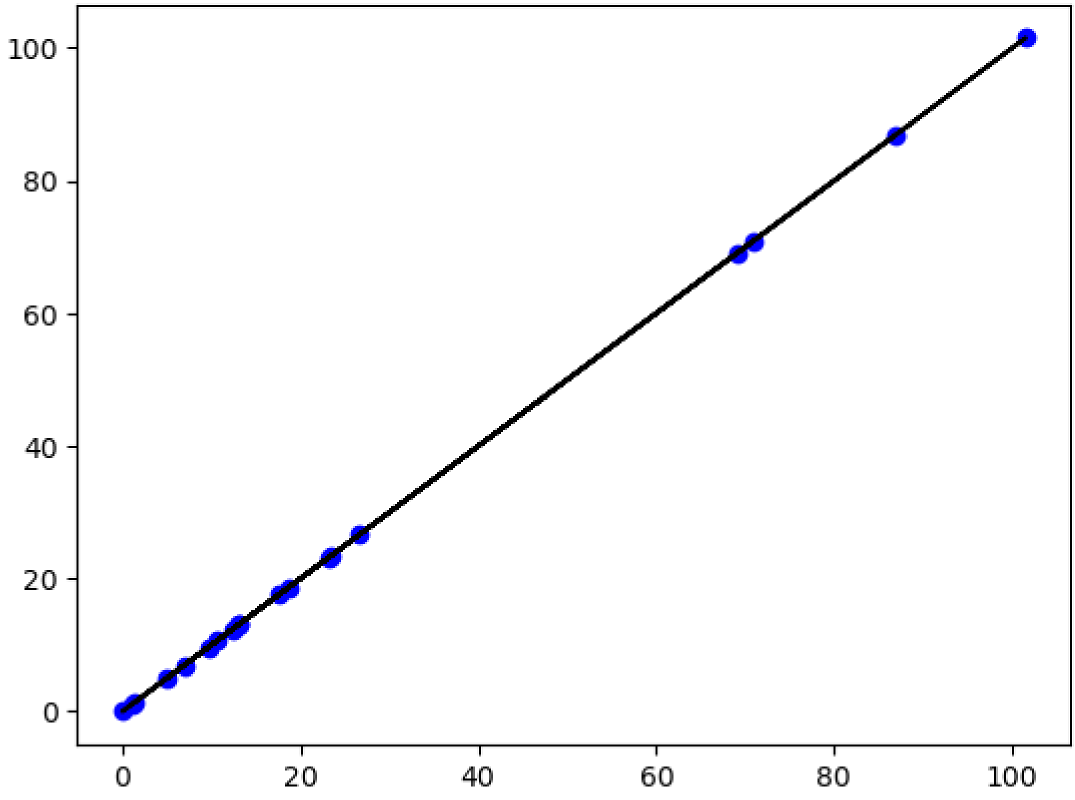
In [26]:

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

```
1.0
```

In [27]:

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



The accuracy of the Linear Regression is 1.0

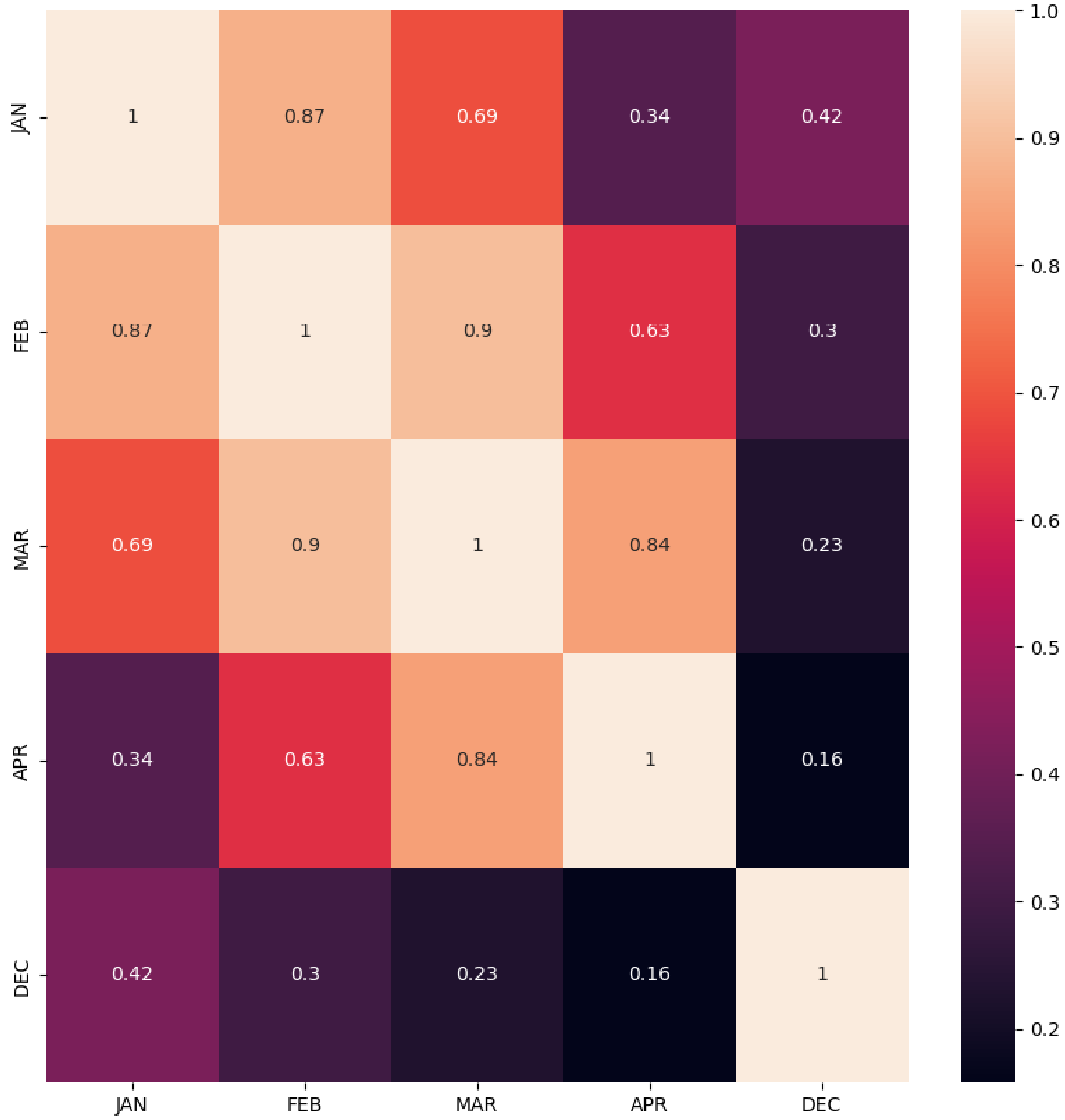
Ridge Regression

In [29]:

```
#Importing Libraries
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [31]:

```
plt.figure(figsize=(10,10))
sns.heatmap(df700.corr(),annot=True)
plt.show()
```



In [32]:

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

In [33]:

```
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 (448, 5)
The dimension of X_test is (193, 5)
```

In [34]:

```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
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))
```

```
Linear Regression Model:

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

In [35]:

```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor ridge regression
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 0.9999999796377905
The test score for ridge model is 0.9999999789383126
```

The accuracy of the Ridge Model is 0.99

Lasso Regression

In [36]:

```
#Importing libraries
lasso= Lasso(alpha=10)
lasso.fit(x_train,y_train)
#train and test scorefor ridge regression
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 0.9992614054347884
The test score for lasso model is 0.999097310714356
```

In [37]:

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

Out[37]:

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

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

In [38]:

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

In [39]:

```
#using the linear cv model
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.9999999999380964
0.9999999999416216
```

In [40]:

```
#using the linear cv model
from sklearn.linear_model import LassoCV
#cross validation
lasso_cv=LassoCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.9999999999999494
0.99999999999994
```

The accuracy of the Lasso Model is 0.20

Elastic Regression

In [41]:

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

In [42]:

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

```
[ 9.92162587e-01  4.43350275e-03  0.00000000e+00 -2.53789361e-04
  5.19490523e-04]
0.05076672333836285
```

Out[42]:

```
0.9999869961425129
```

In [43]:

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

In [44]:

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

```
0.004910953571739872
```

The accuracy of the ElasticNet Regression is 0.99999914

CONCLUSION: The given data is "Rain fall pridection".here we need to find the best fit model.As per the given data set I had applyed different types of models...in which different type of models got different type of accyuracies The accuracy of the Linear Regression is 1.0 The accuracy of the Ridge Model is 0.9999999999856 The accuracy of the Lasso Model is 0.20 The accuracy of the ElasticNet Regression is 0.99999914, comparing to all the models,Ridge Regression got the Highest Accuracy

Therefore Ridge Regression is the best fit for this Dataset

In [ ]:

```

```