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\Sudheer\AppData\Local\Temp\Temp1_100Years_RainfallDa
df
```

Out[2]:

	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
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2
...	...	...	...	...	...	...	...	...	...	...
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5

641 rows × 19 columns

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

Out[3]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	O
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	32
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	30
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	27
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	16
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	20

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

Out[4]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5

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 [6]:
```
df.fillna(method='ffill',inplace=True)
```

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]:
```
df.describe()
```

Out[8]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.033697
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.364643
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.600000
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.400000
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.700000
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.800000
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.900000

1

In [12]:
```
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 [10]:
```
df.columns
```

Out[10]:
```
Index(['STATE_UT_NAME', 'DISTRICT', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',
       'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb',
       'Mar-May', 'Jun-Sep', 'Oct-Dec'],
      dtype='object')
```

In [11]:
```
df.shape
```

Out[11]:
```
(641, 19)
```

In [13]:
```
df['ANNUAL'].value_counts()
```

Out[13]:
```
ANNUAL
747.1     9
2080.0    4
1336.5    3
1824.8    3
2814.4    3
         ..
1037.6    1
907.2     1
944.5     1
1003.3    1
3253.1    1
Name: count, Length: 591, dtype: int64
```

In [14]:
```
df['Jan-Feb'].value_counts()
```

Out[14]:
```
Jan-Feb
32.7     9
18.2     5
21.4     5
0.8      5
17.5     5
        ..
107.7    1
87.0     1
101.0    1
135.2    1
65.0     1
Name: count, Length: 399, dtype: int64
```

In [15]:
```
df['Mar-May'].value_counts()
```

Out[15]:
```
Mar-May
43.5     9
27.9     5
36.6     4
468.6    4
40.4     3
        ..
16.3     1
23.3     1
49.6     1
20.5     1
232.4    1
Name: count, Length: 511, dtype: int64
```

In [16]:
```
df['Jun-Sep'].value_counts()
```

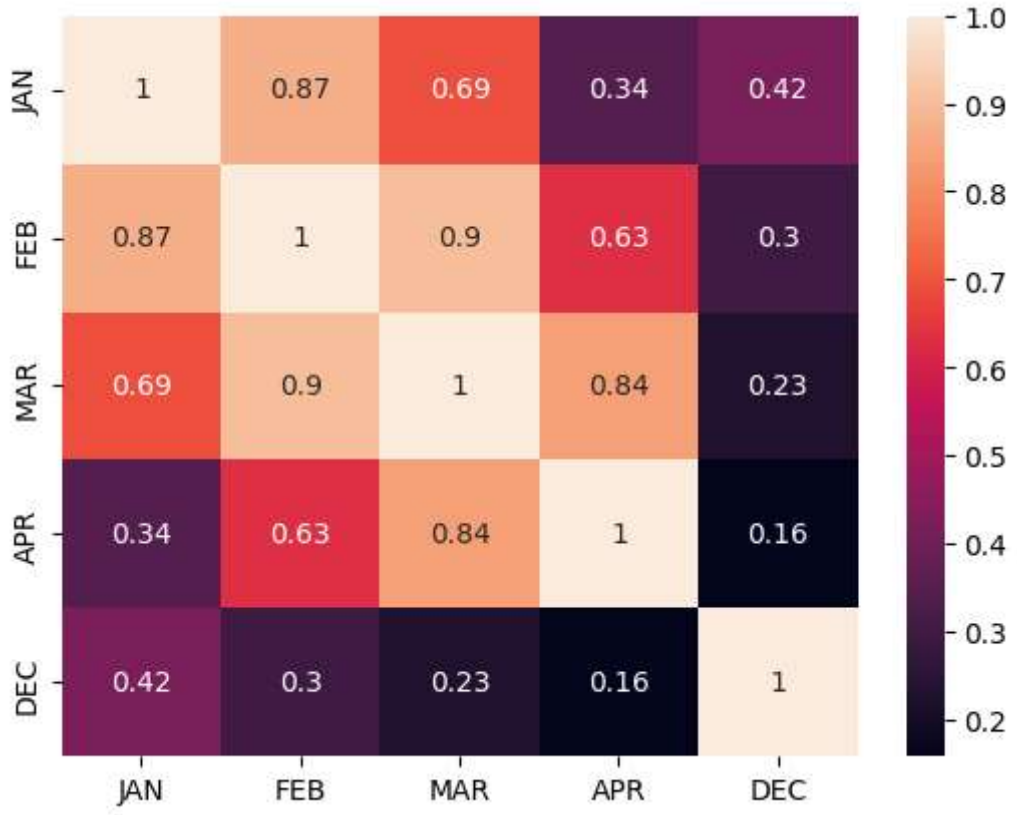
Out[16]:
```
Jun-Sep
636.2     9
1386.1    4
385.0     3
1122.3    3
1308.0    3
         ..
916.9     1
923.5     1
790.3     1
840.7     1
998.5     1
Name: count, Length: 592, dtype: int64
```

In [17]:
```
df['Oct-Dec'].value_counts()
```

Out[17]:
```
Oct-Dec
34.7     9
174.8    4
49.6     3
27.7     3
183.7    3
        ..
82.8     1
55.2     1
65.6     1
54.0     1
333.6    1
Name: count, Length: 524, dtype: int64
```

Exploratary Data Analysis

In [18]:
```
df=df[['JAN','FEB','MAR','APR','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [19]:
```
df.columns
```

Out[19]:
```
Index(['JAN', 'FEB', 'MAR', 'APR', 'DEC'], dtype='object')
```

In [20]:
```
x=df[["FEB"]]
y=df["JAN"]
```

In [21]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state
```

In [22]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,y_train)
print(reg.intercept_)
coeff_=pd.DataFrame(reg.coef_,x.columns,columns=['coefficient'])
coeff_
```

```
3.6728680241521268
```

Out[22]:

	coefficient
FEB	0.715365

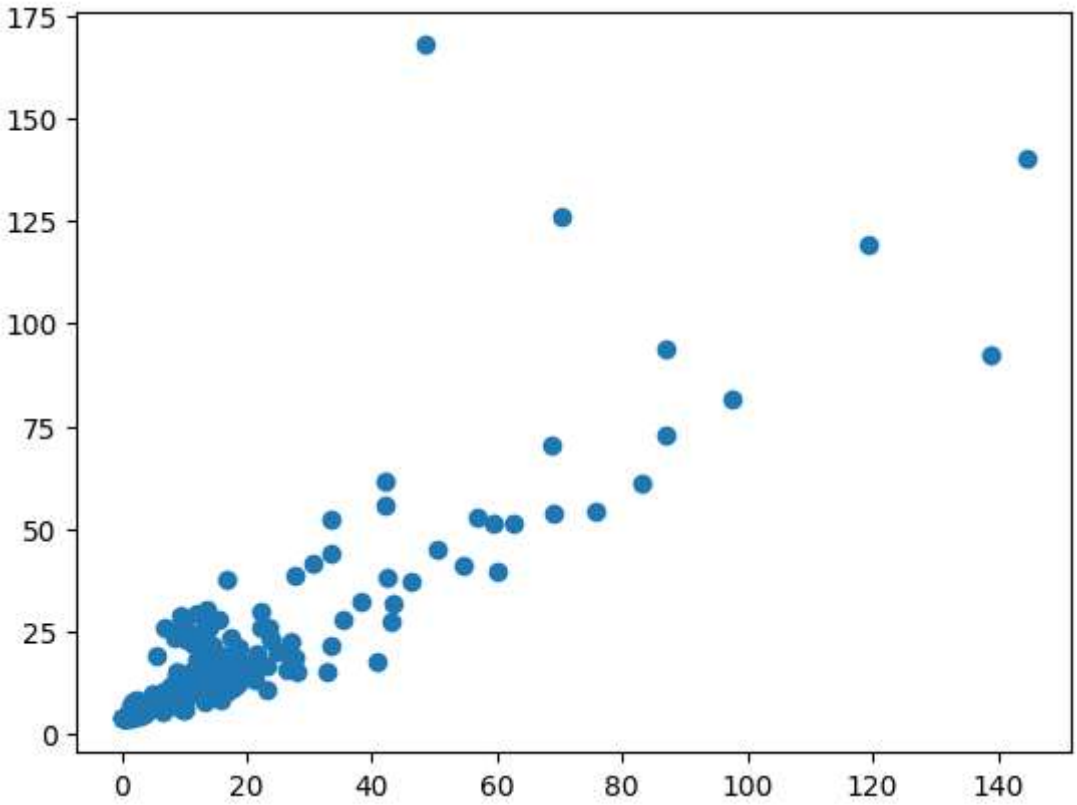
In [23]:
```
score=reg.score(X_test,y_test)
print(score)
```

```
0.6855837686354153
```

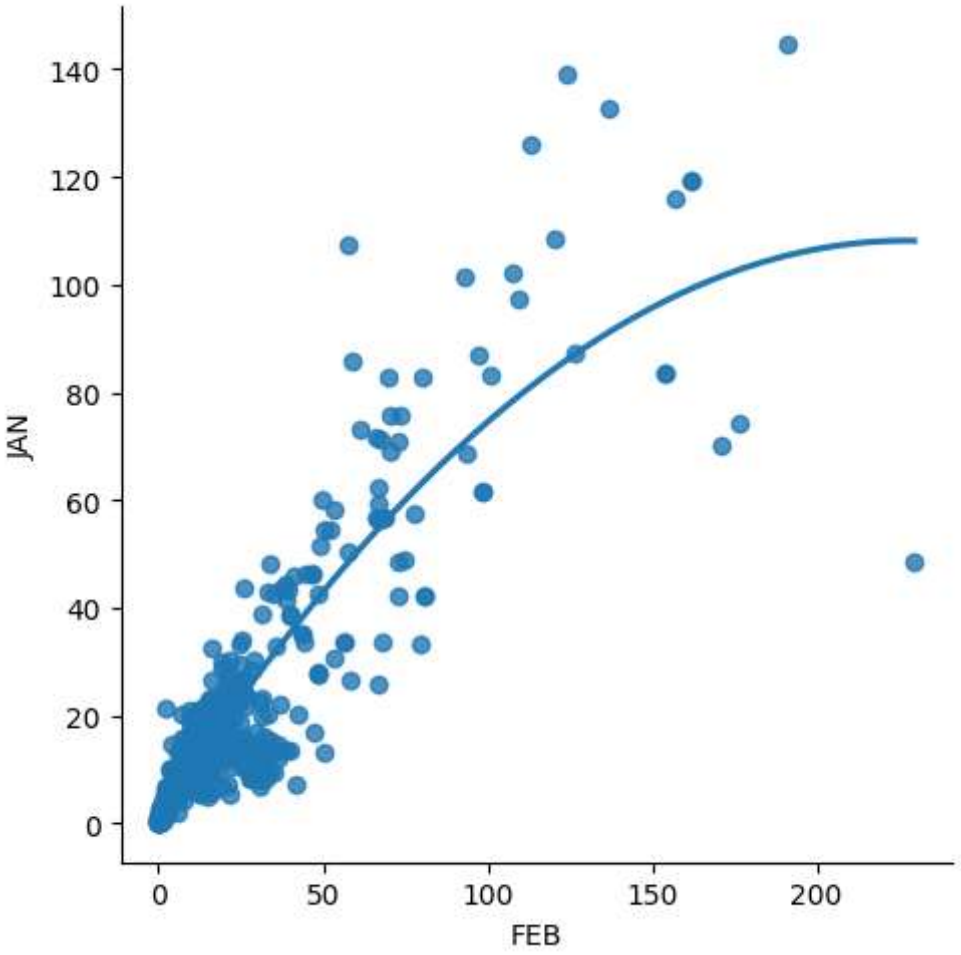
In [24]:
```
predictions=reg.predict(X_test)
```

In [25]:
```
plt.scatter(y_test,predictions)
```

Out[25]:
```
<matplotlib.collections.PathCollection at 0x26db6d09a50>
```



In [26]:
```
df500=df[:][:500]
sns.lmplot(x="FEB",y="JAN",order=2,ci=None,data=df500)
plt.show()
```



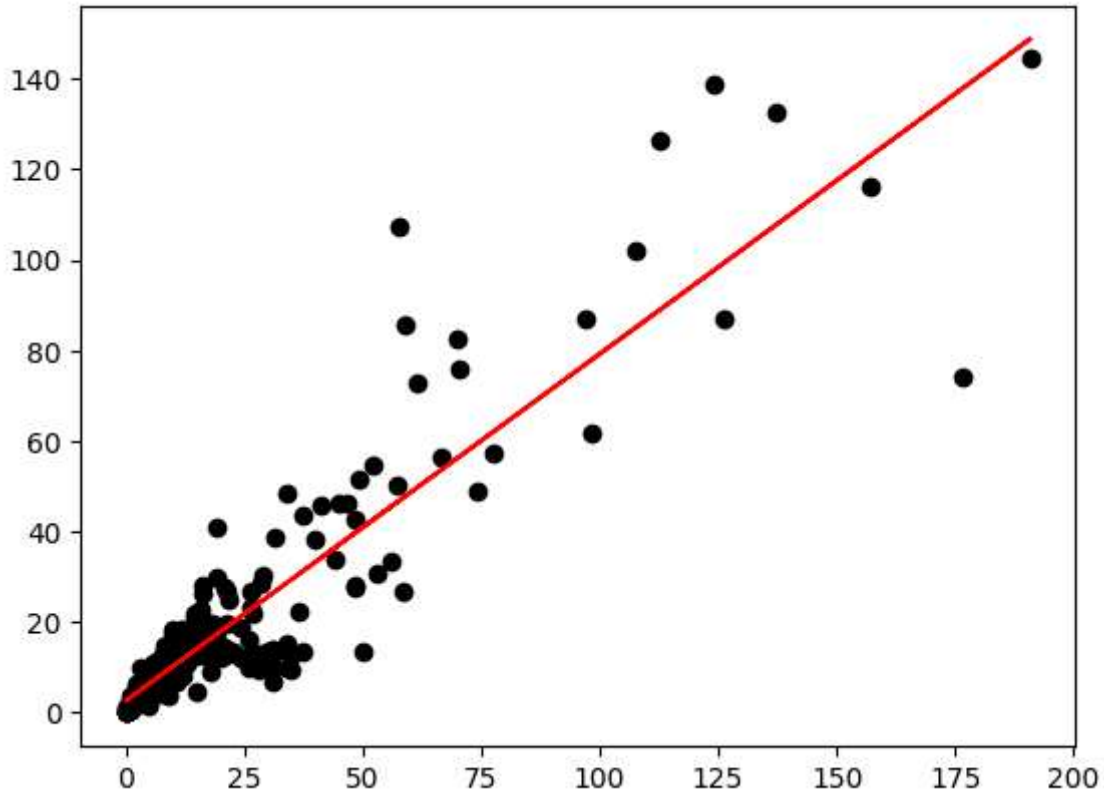
In [27]:
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
reg.fit(X_train,y_train)
reg.fit(X_test,y_test)
```

Out[27]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [28]:
```
y_pred=reg.predict(X_test)
plt.scatter(X_test,y_test,color='black')
plt.plot(X_test,y_pred,color='red')
plt.show()
```



In [29]:
```
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("R2 Score:",r2)
```

```
R2 Score: 0.7607093667102154
```

In [30]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [33]:
```
features= df.columns[0:5]
target= df.columns[-5]
```

In [34]:
```
x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[34], line 2
      1 x=np.array(df['JAN']).reshape(-1,1)
----> 2 y=np.array(df['FEB']).reshape(-1,2)

ValueError: cannot reshape array of size 641 into shape (2)
```

In [35]:
```
x= df[features].values
y= df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=
```

In [36]:
```
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)
```

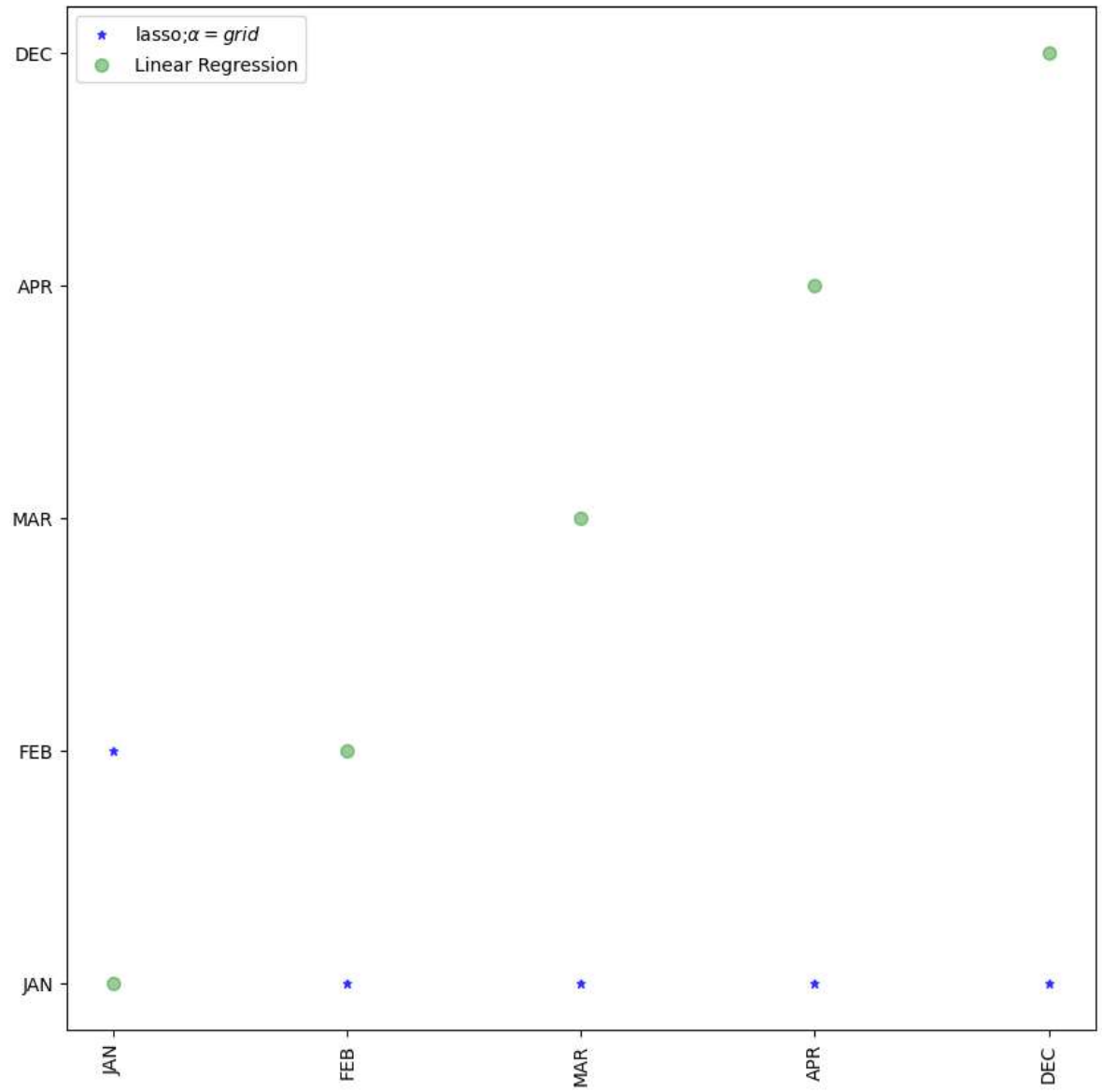
In [37]:
```
print("\n Ridge Model:\n")
print("the train score for ridge model is{}".format(train_score_ridge))
print("the test score for ridge model is{}".format(test_score_ridge))
```

```
 Ridge Model:

the train score for ridge model is0.9999999792491524
the test score for ridge model is0.9999999887465535
```

```
lr=LinearRegression()
```

In [46]:
```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markers
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="gr
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



Lasso Model

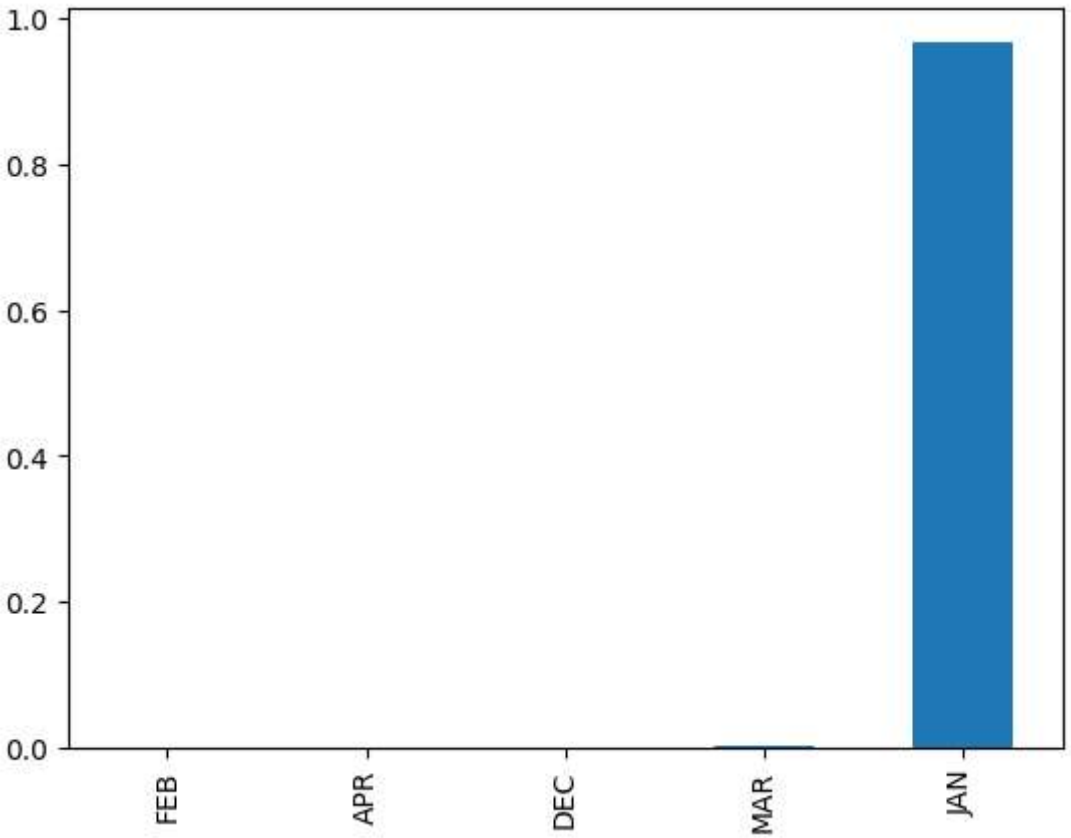
In [40]:
```
print("\n Lasso Model:\n")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_ls=lasso.score(x_train,y_train)
test_score_ls=lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is{}".format(test_score_ls))
```

```
 Lasso Model:

The train score for ls model is 0.99912857000705
The test score for ls model is0.9991969731663574
```

In [41]:
```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

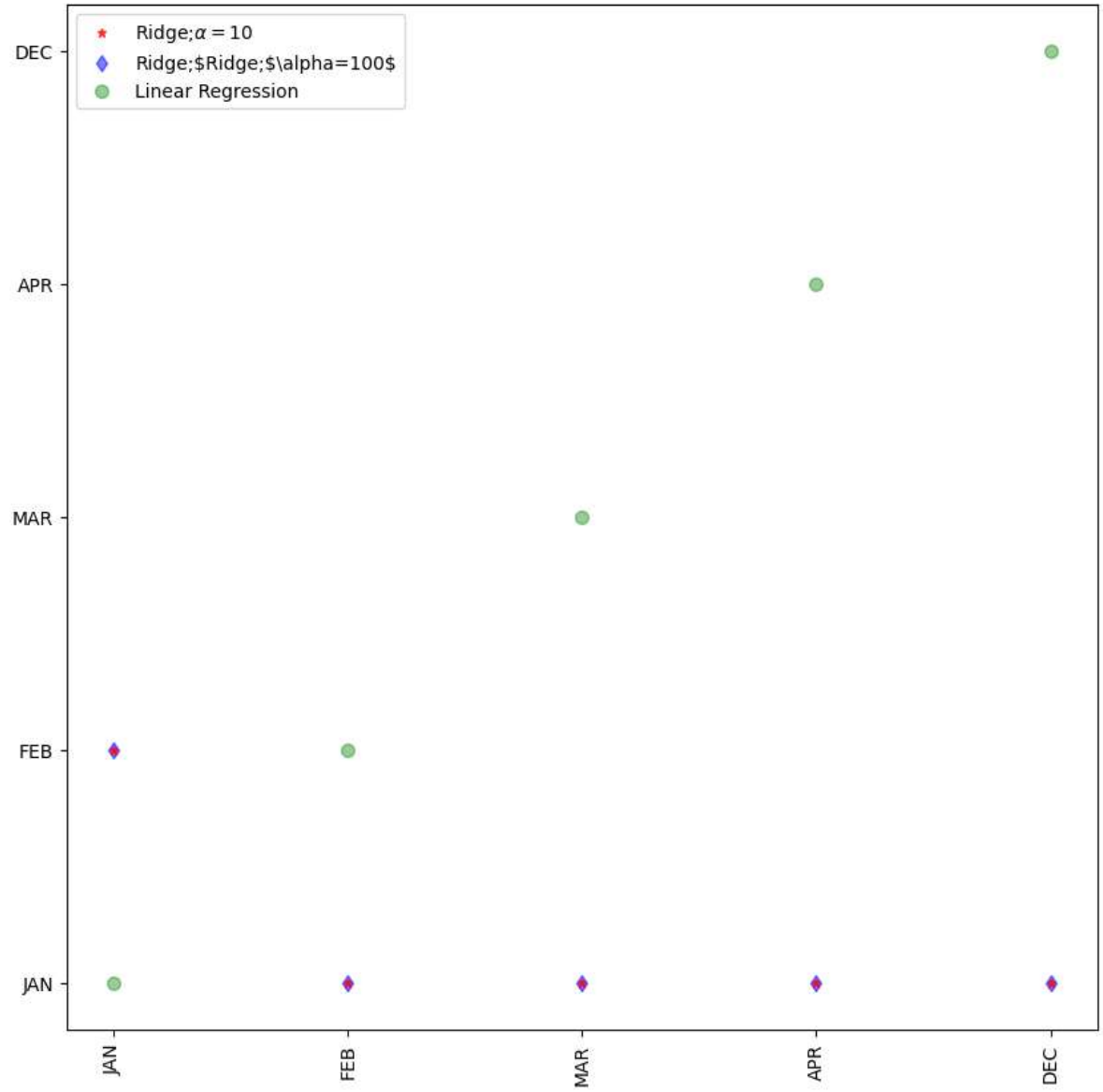
Out[41]: <Axes: >



In [42]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,1,10],random_state=0).fit(x_train,y
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.9999999999999198
0.9999999999999254
```

In [47]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,col
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gr
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



Elastic Net:-

In [44]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
print(regr.score(x,y))
```

```
[ 9.93078025e-01  4.17330119e-03  0.00000000e+00 -2.56844715e-04
  4.33064019e-04]
0.04331617537314614
0.9999905490942288
```

In [45]:
```
y_pred_elastic=regr.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

```
0.00427888850665954
```

Conclusion

THE SCORE OF LINEAR REGRESSION IS :-0.1793580786264921 THE SCORE OF RIDGE MODEL IS :- 0.99999999998833 THE SCORE OF LASSO MODEL IS :- 0.999999999999992 THE SCORE OF ELASTIC NET IS :- 0.9999992160905338 AMONG ALL MODELS LASSO YEILD HIGHEST ACCURACY.SO,WE PREFER LASSO MODEL FO R THIS DATA SET

In [ ]:
```
```