In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
traindf=pd.read_csv(r"C:\Users\shaha\OneDrive\Desktop\Excel\Rainfall train.csv")
traindf
```

Out[3]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AU
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.
...	...	...	...	...	...	...	...	...	...	.
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.

641 rows × 19 columns

In [4]:

```
testdf=pd.read_csv(r"C:\Users\shaha\OneDrive\Desktop\Excel\rainfall in india.csv")
testdf
```

Out[4]:

	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

In [5]:

```
traindf.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]:

```
testdf.head()
```

Out[6]:

	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 [7]:

```
traindf.tail()
```

Out[7]:

	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]:

```
testdf.tail()
```

Out[8]:

	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 [9]:

```
traindf.shape
```

Out[9]:

```
(641, 19)
```

In [10]:

```
testdf.shape
```

Out[10]:

```
(4116, 19)
```

In [11]:

```
traindf.describe()
```

Out[11]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.00000
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.03369
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.36464
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.60000
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.40000
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.70000
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.80000
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.90000

In [12]:

```
testdf.describe()
```

Out[12]:

	YEAR	JAN	FEB	MAR	APR	MAY	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.0
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.2
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.7
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.4
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.3
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.7
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.1
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.9

In [13]:

```
traindf.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 [14]:

```
testdf.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 [15]:

```
traindf.isnull().sum()
```

Out[15]:

```
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 [16]:

```
testdf.isnull().sum()
```

Out[16]:

```
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 [17]:

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

In [18]:

```
testdf.isnull().sum()
```

Out[18]:

```
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 [26]:

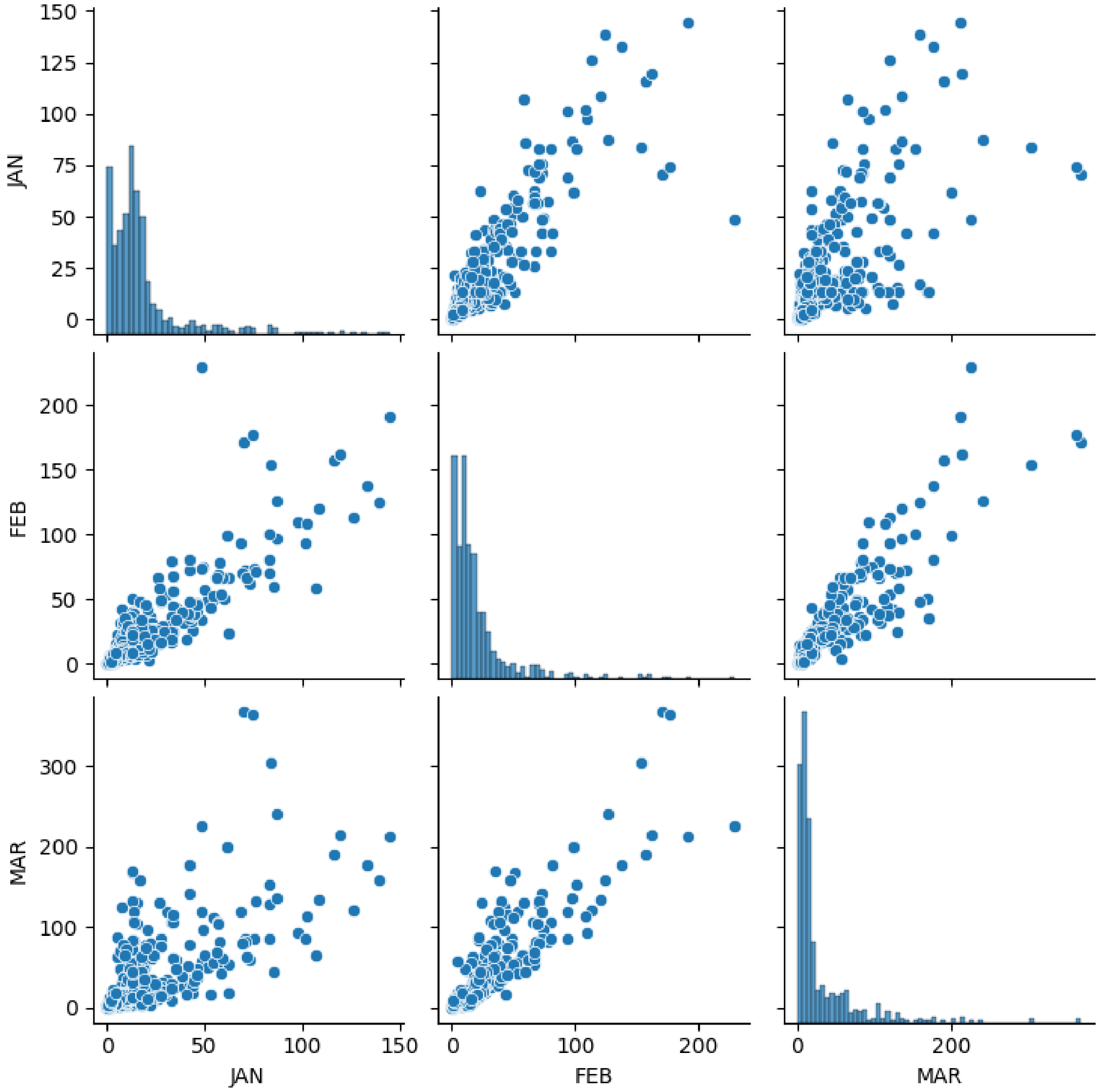
```
data=traindf[['JAN','FEB','MAR']]
```

In [28]:

```
sns.pairplot(data)
```

Out[28]:

```
<seaborn.axisgrid.PairGrid at 0x2cd56029db0>
```



In [29]:

```
traindf['STATE_UT_NAME'].value_counts()
```

Out[29]:

```
STATE_UT_NAME
UTTAR PRADESH                  71
MADHYA PRADESH                 50
BIHAR                          38
MAHARASHTRA                    35
RAJASTHAN                      33
TAMIL NADU                     32
KARNATAKA                      30
ORISSA                         30
ASSAM                          27
GUJARAT                        26
JHARKHAND                      24
ANDHRA PRADESH                 23
JAMMU AND KASHMIR              22
HARYANA                        21
PUNJAB                         20
WEST BENGAL                    19
CHATISGARH                     18
ARUNACHAL PRADESH              16
KERALA                         14
UTTARANCHAL                    13
HIMACHAL                       12
NAGALAND                       11
MIZORAM                         9
MANIPUR                         9
DELHI                           9
MEGHALAYA                       7
SIKKIM                          4
TRIPURA                         4
PONDICHERRY                     4
ANDAMAN And NICOBAR ISLANDS     3
GOA                             2
DAMAN AND DUI                   2
DADAR NAGAR HAVELI              1
CHANDIGARH                      1
LAKSHADWEEP                     1
Name: count, dtype: int64
```

In [30]:

```
traindf['DISTRICT'].value_counts()
```

Out[30]:

```
DISTRICT
BIJAPUR        2
BILASPUR       2
AURANGABAD     2
HAMIRPUR       2
NICOBAR        1
              ..
GONDA          1
GORAKHPUR      1
HARDOI         1
JAUNPUR        1
LAKSHADWEEP    1
Name: count, Length: 637, dtype: int64
```

Linear Regression

In [31]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [40]:

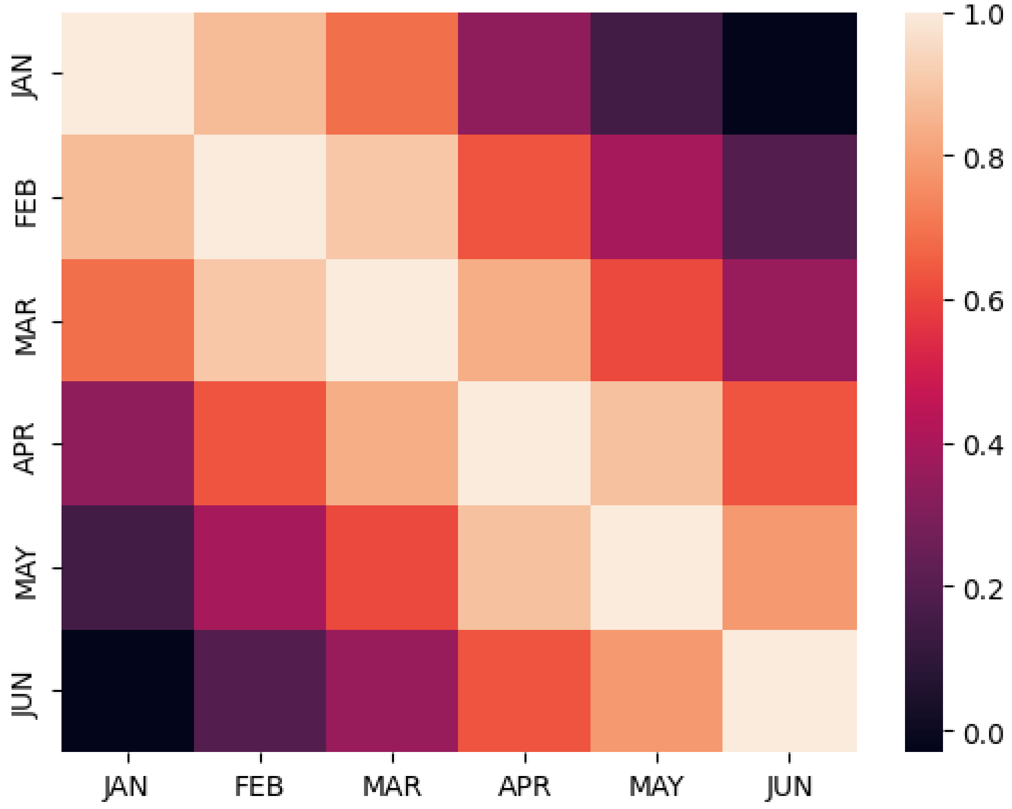
```
x=traindf[['JAN','FEB','MAR','APR','MAY','JUN']]
y=traindf['ANNUAL']
```

In [41]:

```
sns.heatmap(x.corr())
```

Out[41]:

```
<Axes: >
```



In [42]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)
```

In [43]:

```
regr = LinearRegression()
regr.fit(x_train,y_train)
print(regr.intercept_)
coeff_train_df=pd.DataFrame(regr.coef_,x.columns,columns=['coefficient'])
coeff_train_df
```

```
437.52725826332585
```

Out[43]:

	coefficient
JAN	3.017735
FEB	6.071487
MAR	-3.029124
APR	0.633801
MAY	0.825779
JUN	3.683982

In [44]:

```
score=regr.score(x_test,y_test)
print(score)
```

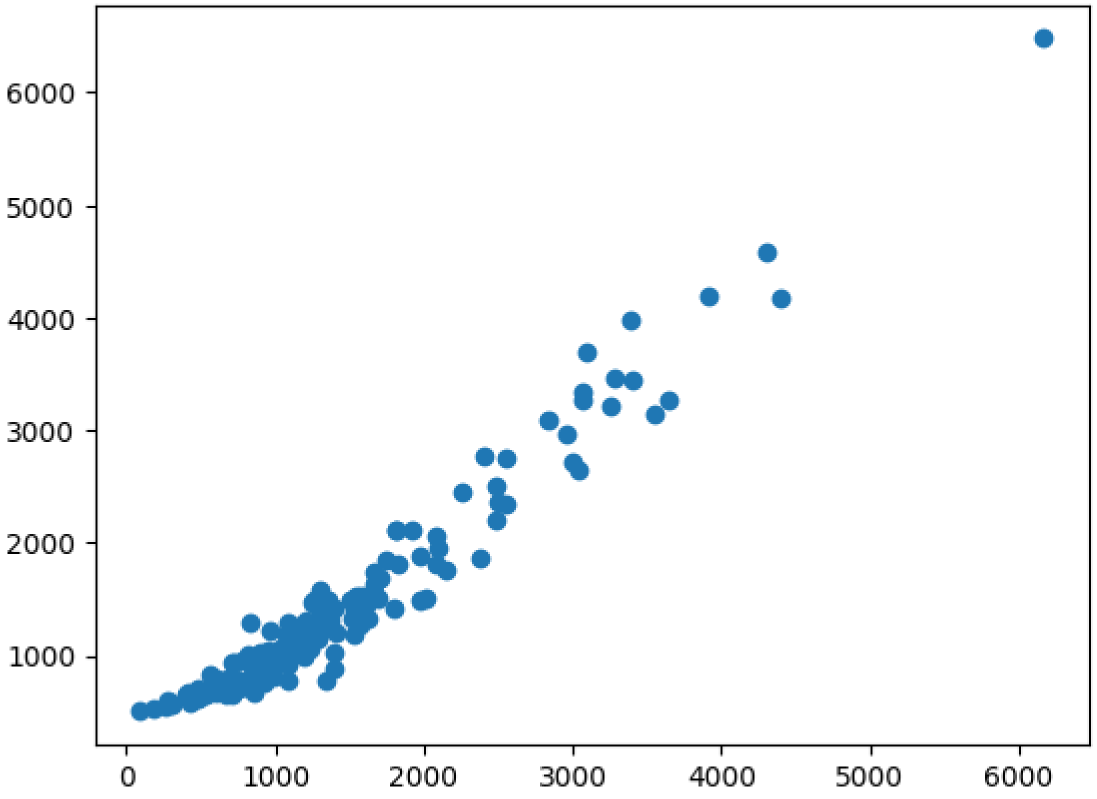
```
0.9488529666589274
```

In [45]:

```
predictions=regr.predict(x_test)
plt.scatter(y_test,predictions)
```

Out[45]:

```
<matplotlib.collections.PathCollection at 0x2cd6e870430>
```



In [49]:

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

In [50]:

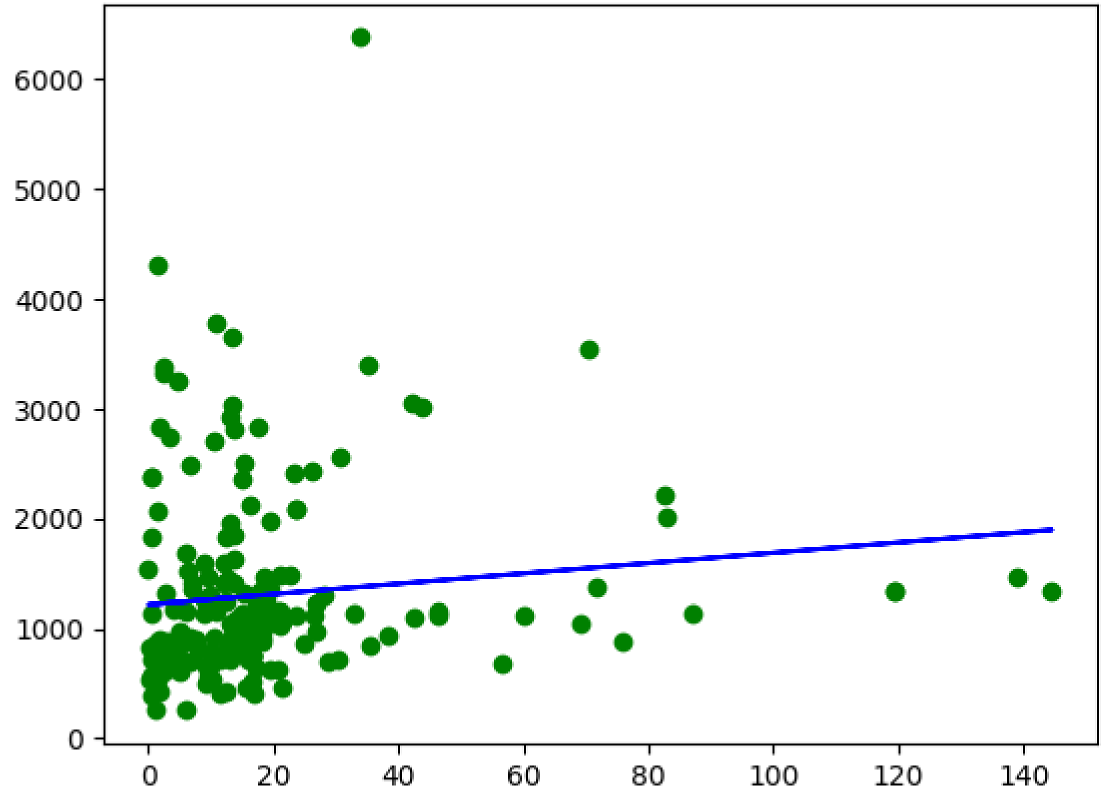
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(x_train,y_train)
regr.fit(x_test,y_test)
```

Out[50]:

```
▾ LinearRegression
LinearRegression()
```

In [51]:

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



Ridge and Lasso regression

In [59]:

```
#Ridge Regression MOdel
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [60]:

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

Out[60]:

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

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

In [115]:

```
features = traindf.columns[2:5]
target = traindf.columns[-1:]
#x and y values
x = traindf[features].values
y = traindf[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 (448, 3)
The dimension of x_test is (193, 3)
```

In [116]:

```
#Model
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))
```

```

Linear Regression Model:

The train score for lr model is 0.08412859673411976
The test score for lr model is 0.03895606778221439
```

In [117]:

```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test score for 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 0.08354804861559051
The test score for ridge model is 0.03961593206840308
```

In [118]:

```
#Lasso Regression model
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))
```

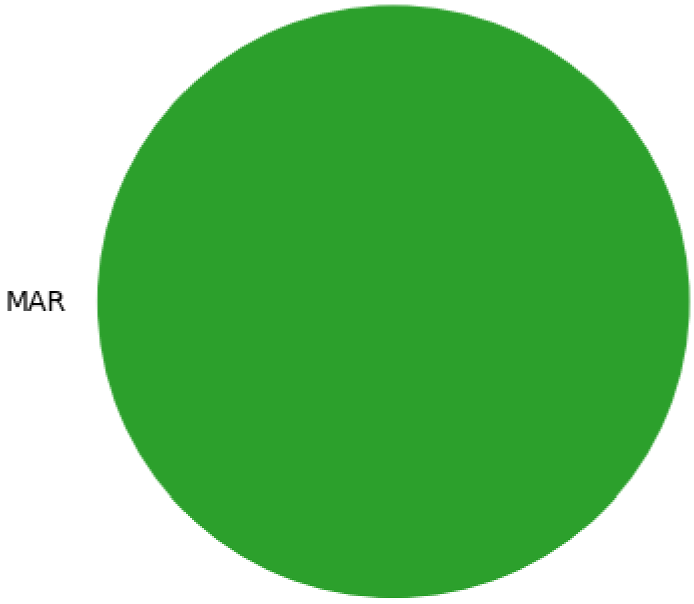
```
Lasso Model:

The train score for ls model is 0.07504785033558381
The test score for ls model is 0.052588033859716776
```

In [119]:

```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "pie")
```

Out[119]:

```
<Axes: >
```



In [120]:

```
#using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_t
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.07504785033558381
0.052588033859716776
```

In [122]:

```
#using the linear Cv model
from sklearn.linear_model import RidgeCV
#Ridge cross validation
ridge_cv = RidgeCV(alphas = [0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(x_train,y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(x_test,y_test)))
```

```
The train score for ridge model is 0.08354804861559062
The train score for ridge model is 0.039615932068407744
```

Elastic Net

In [123]:

```
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
regr.score(x,y)
```

```
[ 0.25536298 -1.31993694  1.50300354]
[120.58314834]
```

Out[123]:

```
0.07698323574326038
```

In [124]:

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

In [125]:

```
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 22500.582660635675
```

conclusion

from the given Rainfall dataset,we have performed LinearRegression model,Ridge and Lasso Regression,Elastic Net.After appling the models,Linear Regression gives the best score:0.9488529666589274.so,Linear Regression model is the Best fit.

In [ ]: