AIM:

To Read Dataset using Pandas , Show the Descriptive statistics Apply preprocessing methods, Cleaning Feature engineering and Outlier analysis, Find Std Dev Mean and Error .Report using Line plot, Scatter plot, Histogram, Boxplot etc..

DESCRIPTION:

Descriptive Statistics:

- Provides summary statistics like mean, standard deviation
- Useful for understanding the central tendency and variability of the data.

Preprocessing and Cleaning:

- Removed duplicates and handled missing values.
- Essential to ensure data quality before analysis.

Feature Engineering:

- Created a new feature, 'New_Feature,' as a combination of existing features.
- Removed unwanted features and produce clean dataset

Outlier Analysis:

- Helps identify extreme values that might impact analysis

Visualization:

- Used line plot to show the trend of 'Feature_A.'
- Employed a scatter plot for visualizing the relationship between two features
- Created a histogram to illustrate the distribution of 'Feature_A.'

CODE:

READ THE DATASET

```
#READ THE DATASET USING PANDAS LIBRARY
import pandas as pd
df=pd.read_csv("./district wise rainfall normal.csv")
df
df.head()
```

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	326.0	315.2	250.9	2805.2	1
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	301.2	275.8	128.3	3015.7	
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	276.1	198.6	100.0	2913.3	
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	167.1	34.1	29.8	3043.8	1
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	206.9	29.5	31.7	4034.7	1

DESCRIPTIVE STATISTICS

MEASURES OF CENTRAL TENDENCY MEAN, MEDIAN , MODE

```
df["ANNUAL"].mean()
```

```
1346.9695787831513
```

```
df["Jan-Feb"].mode()
```

```
0    32.7
Name: Jan-Feb, dtype: float64
```

+ Code + Markdown

```
df["Mar-May"].median()
```

```
67.2
```

MEASURES OF VARIABILITY(STANDARD DEVIATION, VARIANCE,PERCENTILES)

```
[21] df["JAN"].std(ddof=1)
```

```
21.082806458323486
```

```
[22] df["FEB"].var(ddof=1)
```

```
768.9304749902496
```

```
[23] import numpy as np
     x=df["ANNUAL"]
     np.percentile(x,[0,25,50,75,100])
```

```
array([  94.6,  830.4, 1116.2, 1530.9, 7229.3])
```

DATA PRE-PR0CESSING AND FEATURE ENGINEERING

```
df.describe()
```

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.00
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.033697	291.152262	194.609048	90.44
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.364643	152.647325	99.830540	74.99
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.600000	14.100000	8.600000	3.10
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.400000	194.600000	128.800000	34.30
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.700000	284.800000	181.300000	62.60
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.800000	358.100000	234.100000	130.20
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.900000	1522.100000	826.300000	517.70

DATA PRE-PR0CESSING AND FEATURE ENGINEERING

```
df.isnull().sum()  #find the null
```

```
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
```

```
from sklearn.impute import SimpleImputer
# Specify the column to exclude (string type)

column_to_exclude = 'SUBDIVISION'

# Create a list of numerical columns
numerical_columns = [col for col in df.columns if col != column_to_exclude and
df[col].dtype != object]

# Create the imputer, excluding the string column
imputer = SimpleImputer(strategy='mean', copy=False)  # avoid unnecessary data
copy
imputer.fit(df[numerical_columns])

# Transform the data, excluding the string column
df[numerical_columns] = imputer.transform(df[numerical_columns])
```

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

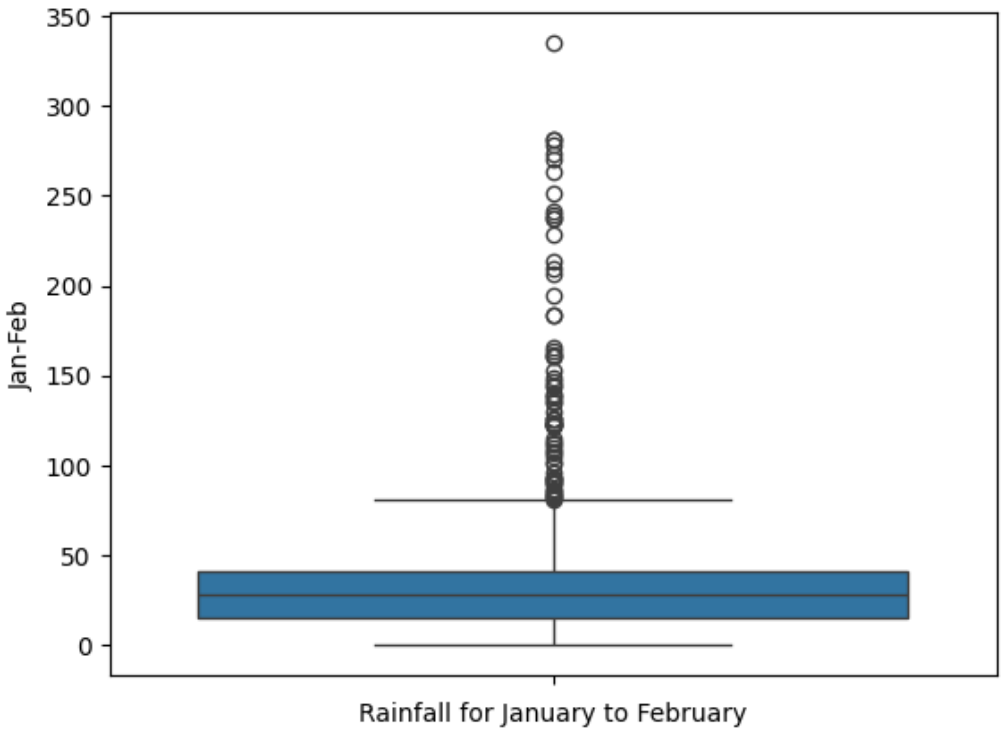
```
SUBDIVISION    0
YEAR           0
JAN            0
FEB            0
MAR            0
APR            0
MAY            0
JUN            0
JUL            0
AUG            0
SEP            0
OCT            0
NOV            0
DEC            0
ANNUAL         0
Jan-Feb        0
Mar-May        0
Jun-Sep        0
Oct-Dec        0
dtype: int64
```

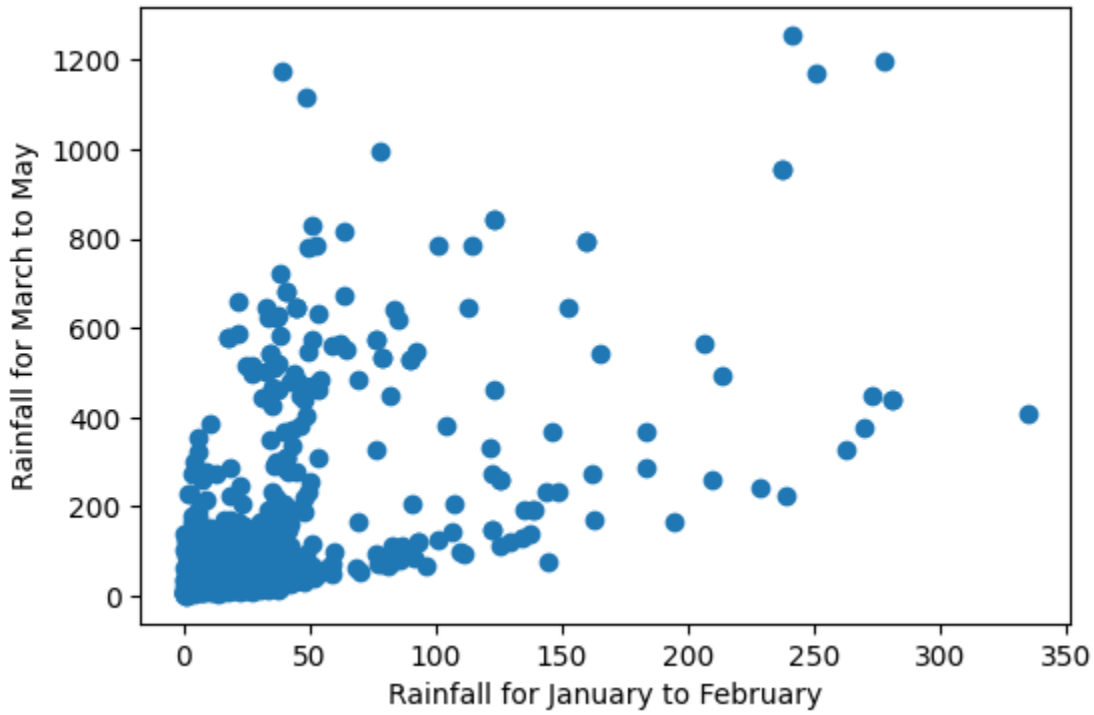
```
df={
    'Jan-Feb':[1,3,55],
    'Mar-May':[2,3.0,51]
}
df=pd.DataFrame(df)
df.corr()
```

	Jan-Feb	Mar-May
Jan-Feb	1.00000	0.99989
Mar-May	0.99989	1.00000

OUTLIER ANALYSIS AND VISUALIZATION USING BOX PLOT AND SCATTER PLOT AND DATA VISUALIZATION

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(df["Jan-Feb"])
plt.xlabel("Rainfall for January to February")
plt.show()
# Create a scatter plot using Matplotlib
fig, ax = plt.subplots(figsize=(6, 4))
ax.scatter(df['Jan-Feb'], df['Mar-May'])
ax.set_xlabel("Rainfall for January to February")
ax.set_ylabel("Rainfall for March to May")
plt.show()
```

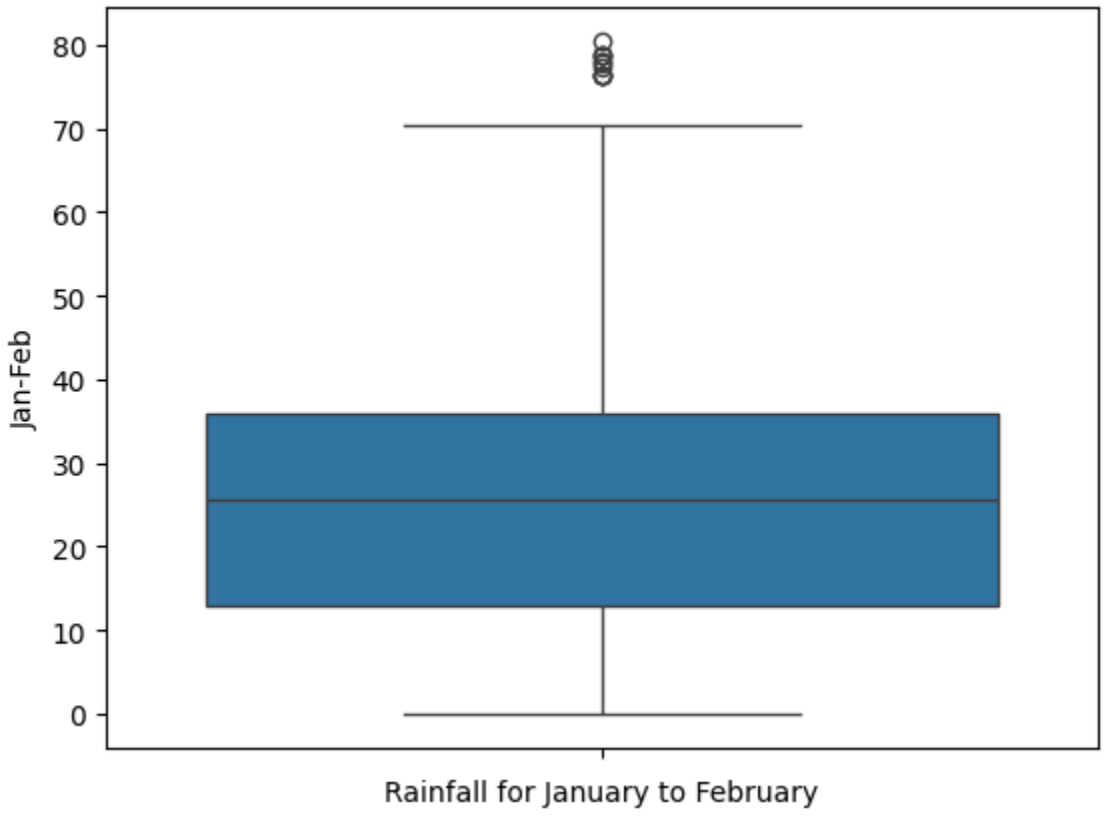


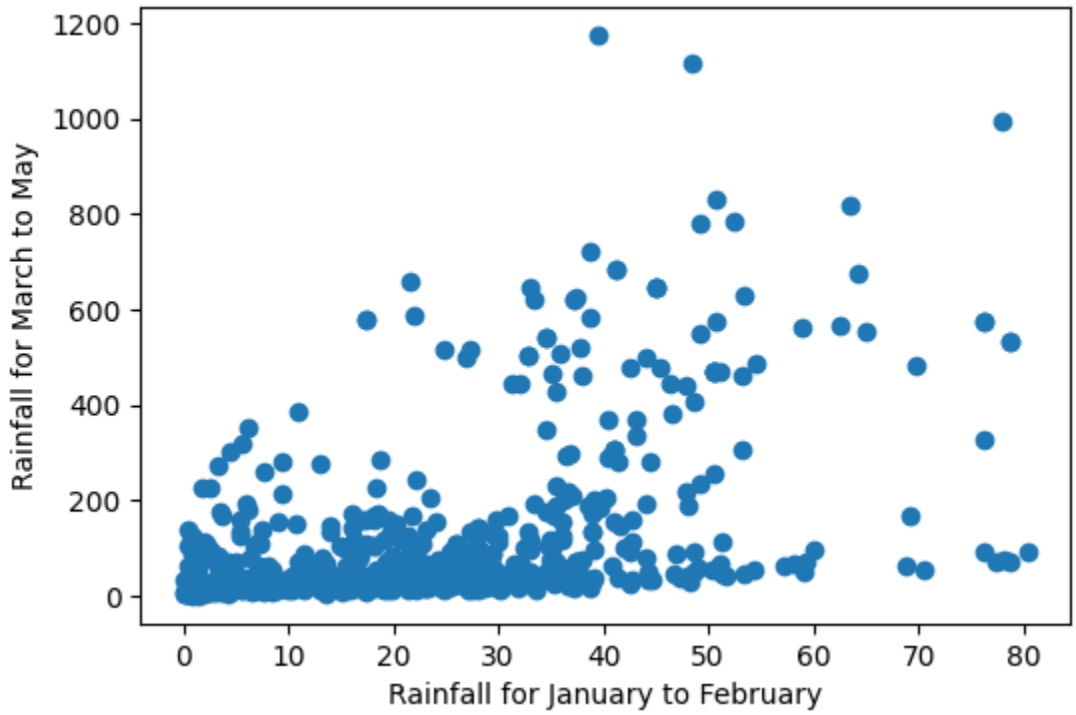


REMOVE OUTLIERS

```
import seaborn as sns
import matplotlib.pyplot as plt
# Function to remove outliers using IQR method
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_filtered = df[(df[column] >= lower_bound) & (df[column] <=
upper_bound)]
    return df_filtered
```

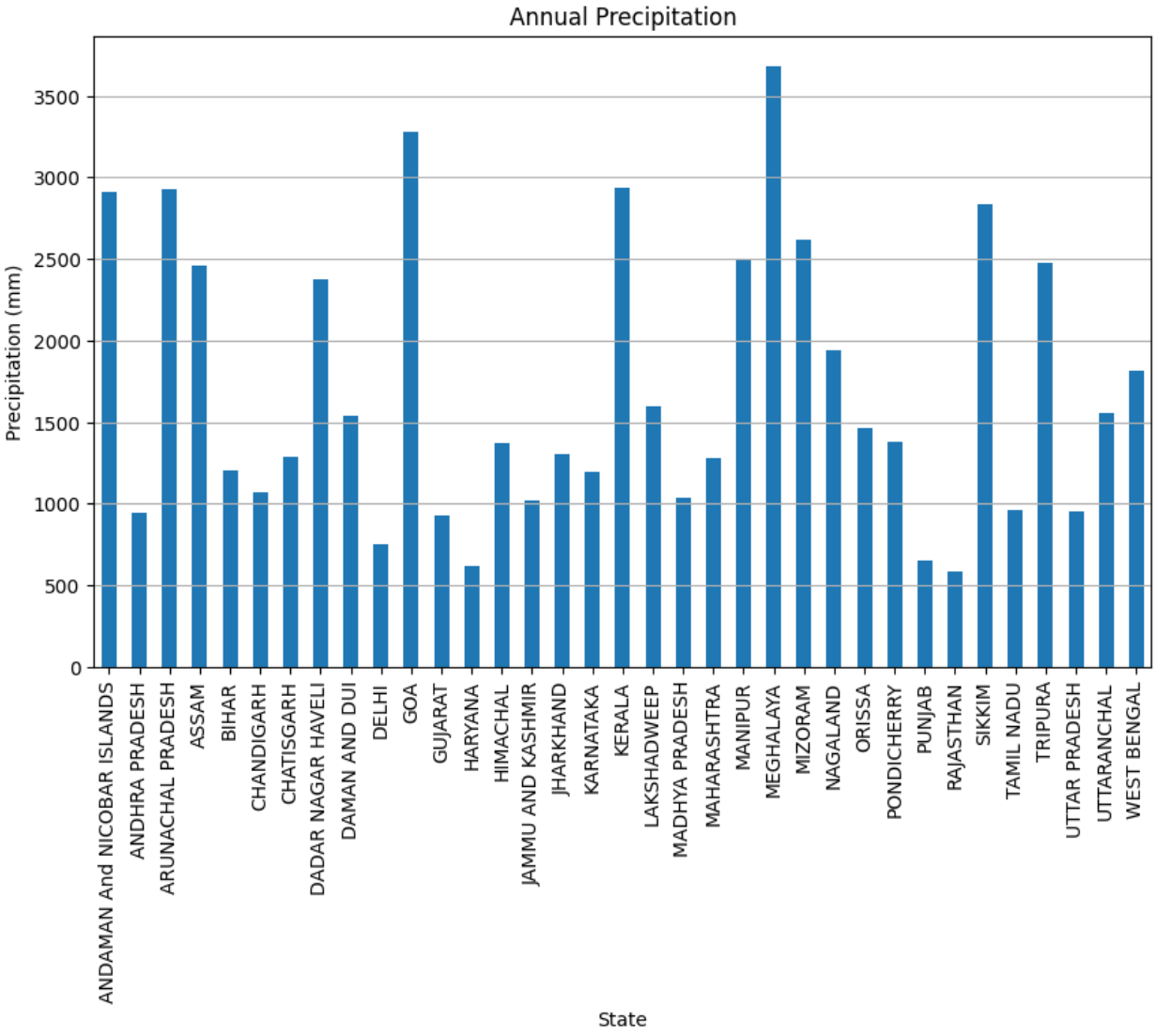
```
# Remove outliers from "Jan-Feb" column
df_filtered = remove_outliers(df, "Jan-Feb")
# Create a boxplot using Seaborn after removing outliers
sns.boxplot(df_filtered["Jan-Feb"])
plt.xlabel("Rainfall for January to February")
plt.show()
# Create a scatter plot using Matplotlib after removing outliers
fig, ax = plt.subplots(figsize=(6, 4))
ax.scatter(df_filtered['Jan-Feb'], df_filtered['Mar-May'])
ax.set_xlabel("Rainfall for January to February")
ax.set_ylabel("Rainfall for March to May")
plt.show()
```





VISUALIZATION

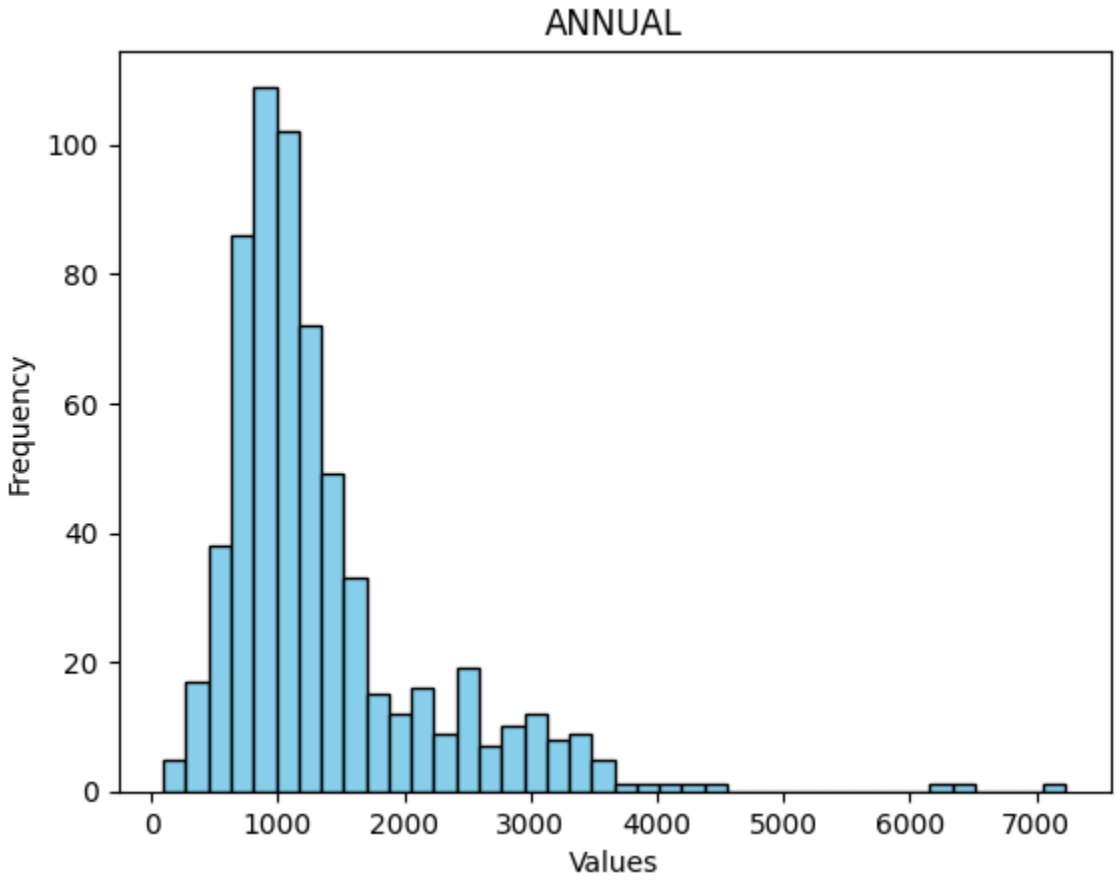
```
annual_data =
df.groupby('STATE_UT_NAME')['ANNUAL'].mean()

# Plotting
plt.figure(figsize=(10, 6))
annual_data.plot(kind='bar')
plt.title('Annual Precipitation')
plt.xlabel('State')
plt.ylabel('Precipitation (mm)')
plt.grid(axis='y')
plt.show()
```



It is inferred that Meghalaya has received the highest annual rainfall.

```
data=np.array(df['ANNUAL'])
plt.hist(data,bins=40,color='skyblue',edgecolo
r='black')
plt.xlabel("Values")
```

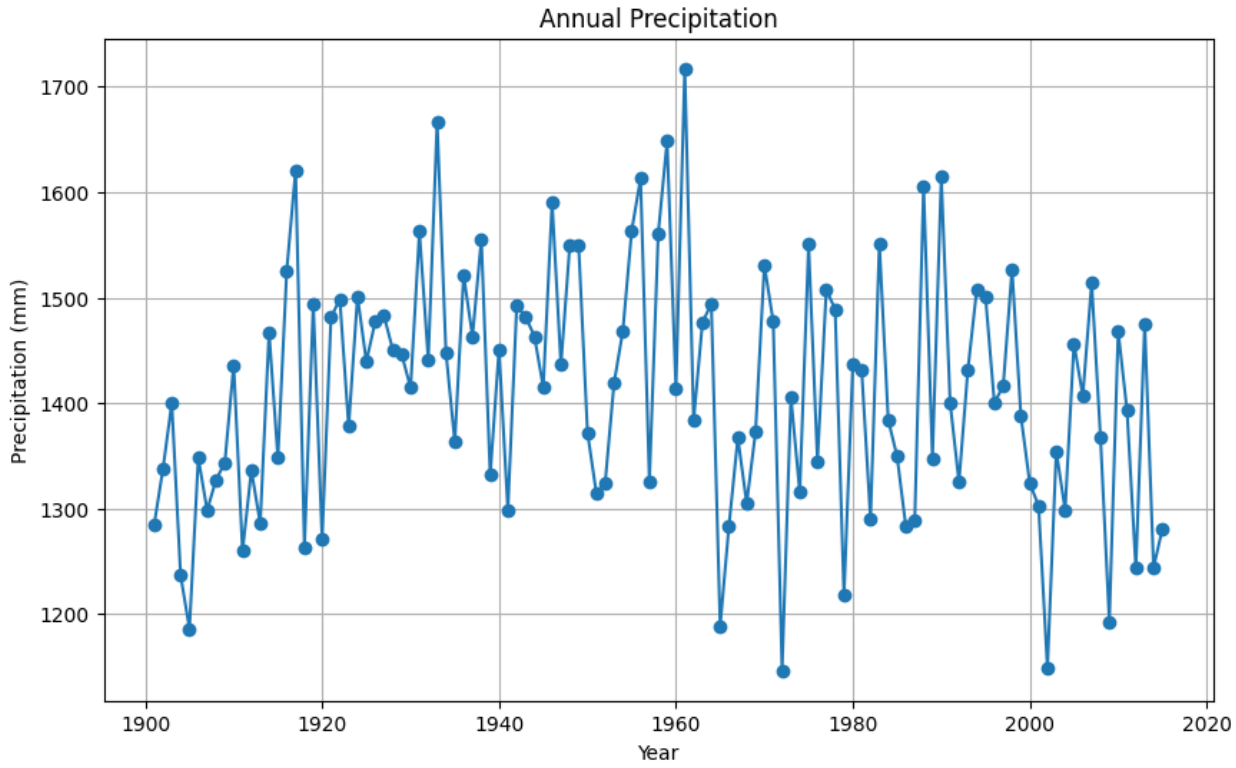
```
plt.ylabel("Frequency")
plt.title("ANNUAL RAIN FALL DISTRIBUTION")
plt.show()
```



It shows the right skewness in the frequency distribution of annual rainfall

```
rainfall=pd.read_csv("../RAIN DATASET/rainfall
in india 1901-2015.csv")
# Assuming your DataFrame is named df
annual_data =
rainfall.groupby('YEAR')['ANNUAL'].mean()
```

```
# Plotting
plt.figure(figsize=(10, 6))
plt.plot(annual_data.index,
annual_data.values, marker='o', linestyle='-')
plt.title('Annual Precipitation')
plt.xlabel('Year')
plt.ylabel('Precipitation (mm)')
plt.grid(True)
plt.show()
```



It is inferred that years between 1920 to 1965 have highest average rainfalls.