EXP 4:

AIM:

To solve the following data science equations , visualize the plots and data distributions for your dataset using the python libraries.

PROCEDURE:

1) **Solving the equation :**

```
from sympy import symbols, Eq, solve
a, b, c, d, e = symbols('a b c d e')
eq_1 = Eq(3.5 * a + 4 * b + 4.5 * c - 2 * d - 2.5 * e, 0.83)
eq_2 = Eq(4.25 * a + 3 * b + 1.75 * c - 2 * d - 0.75 * e, 0.33)
eq_3 = Eq(a + b + c, d + e)
print('eq_1 :',eq_1)
print('eq_2 :',eq_2)
print('eq_3 :',eq_3)
solution = solve((eq_1, eq_2, eq_3), (a, b, c, d, e))
print(solution)
```

```
In [6]: runfile('C:/Users/csemtslab/.spyder-py3/temp.py', wdir='C:/Users/csemtslab/.spyder-py3')
eq_1 : Eq(3.5*a + 4*b + 4.5*c - 2*d - 2.5*e, 0.83)
eq_2 : Eq(4.25*a + 3*b + 1.75*c - 2*d - 0.75*e, 0.33)
eq_3 : Eq(a + b + c - d - e, 0)
{a: c - e - 0.0566666666666667, b: -2.0*c + e + 0.4575, d: 0.400833333333333 - e}
```

```
from sympy import symbols,Eq,solve
y=symbols('y')
eq_1=Eq(((3*y+4)/(y-1)),(2+(7/(y-1))))
print('eq_1 :',eq_1)
sol=solve((eq_1),(y))
print(sol)
```

```
In [13]: runfile('C:/Users/csemtslab/.spyder-py3/te
eq_1 : Eq((3*y + 4)/(y - 1), 2 + 7/(y - 1))
[]
```

2.Perform all types of distribution (discrete and continuous) and find appropriate distribution for your project data set. Write in comments why and what is the need of distribution.

NORMAL DISTRIBUTION:

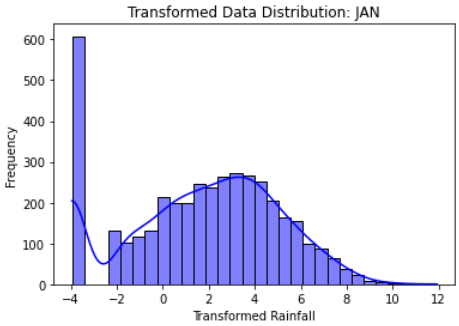
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import boxcox
import seaborn as sns

data=pd.read_csv(r'C:\Users\Roja Traders\OneDrive\Desktop\rainfall in india 1901-2015.csv')
print(data.head())

data.dropna(inplace=True)

rainfall_attributes = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb', 'Mar-May', 'Jun-Sep', 'Oct-Dec']

from scipy.stats import shapiro

for attribute in rainfall_attributes:
    stat, p = shapiro(data[attribute])
    significance_level = 0.05
    if p < significance_level:
        print(f"Attribute: {attribute}, Shapiro-Wilk Test Statistic: {stat:.4f}, p-value: {p:.4f}")
        print("Reject the null hypothesis. The data is not normally distributed.")
    else:
        print(f"Attribute: {attribute}, Shapiro-Wilk Test Statistic: {stat:.4f}, p-value: {p:.4f}")
        print("Fail to reject the null hypothesis. The data may be normally distributed.")

for attribute in rainfall_attributes:
    mean = np.mean(data[attribute])
    median = np.median(data[attribute])
    std_dev = np.std(data[attribute])
    print(f"Attribute: {attribute}, Mean: {mean:.2f}, Median: {median:.2f}, Standard Deviation: {std_dev:.2f}")
```

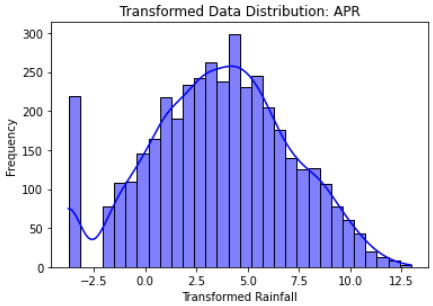
```
Attribute: JAN, Mean: 18.82, Median: 5.90, Standard Deviation: 33.52
Attribute: FEB, Mean: 21.64, Median: 6.60, Standard Deviation: 35.76
Attribute: MAR, Mean: 27.25, Median: 7.80, Standard Deviation: 46.82
Attribute: APR, Mean: 42.71, Median: 15.50, Standard Deviation: 67.26
Attribute: MAY, Mean: 84.87, Median: 36.05, Standard Deviation: 122.54
Attribute: JUN, Mean: 228.93, Median: 138.45, Standard Deviation: 233.51
Attribute: JUL, Mean: 346.50, Median: 284.30, Standard Deviation: 269.32
Attribute: AUG, Mean: 289.90, Median: 259.50, Standard Deviation: 187.68
Attribute: SEP, Mean: 197.00, Median: 173.60, Standard Deviation: 135.25
Attribute: OCT, Mean: 95.14, Median: 64.65, Standard Deviation: 99.31
Attribute: NOV, Mean: 39.55, Median: 9.50, Standard Deviation: 68.27
Attribute: DEC, Mean: 18.70, Median: 3.00, Standard Deviation: 42.18
Attribute: ANNUAL, Mean: 1411.01, Median: 1121.30, Standard Deviation: 903.74
Attribute: Jan-Feb, Mean: 40.46, Median: 19.00, Standard Deviation: 59.13
Attribute: Mar-May, Mean: 154.84, Median: 74.25, Standard Deviation: 200.29
Attribute: Jun-Sep, Mean: 1062.33, Median: 880.20, Standard Deviation: 705.88
Attribute: Oct-Dec, Mean: 153.38, Median: 97.55, Standard Deviation: 166.39
```

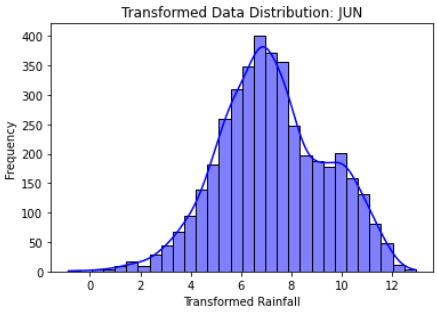
```
constant = 0.001
data_positive = data[attribute] + constant

transformed_data = pd.DataFrame()
for attribute in rainfall_attributes:
    constant = 0.001
    data_positive = data[attribute] + constant
    transformed_attribute, _ = boxcox(data_positive)
    transformed_data[attribute] = transformed_attribute

for attribute in rainfall_attributes:
    sns.histplot(transformed_data[attribute], kde=True, color='blue', bins=30)

    plt.title(f"Transformed Data Distribution: {attribute}")
    plt.xlabel('Transformed Rainfall')
    plt.ylabel('Frequency')
    plt.show()
```

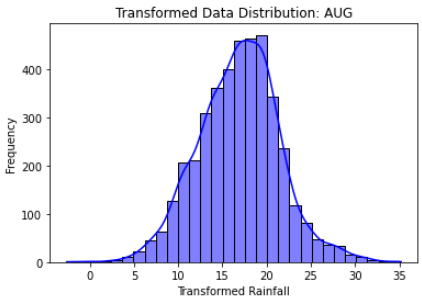
By using the wilk shapiro test , we can determine if the data of the attributes are normally distributed or not . If the p_value from the test is less than 0.05 , then the data is not normally distributed . From our dataset , every attributes have failed the wilk shapiro test . Hence we transformed the data using boxcox transformation to visualize the normally distributed data.

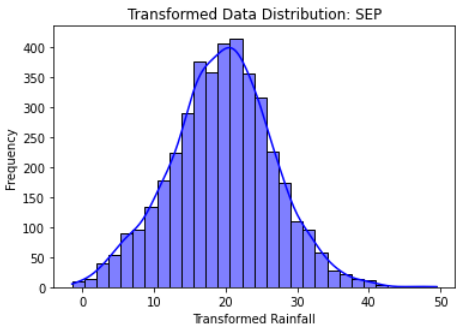
The plots of the attributes after the transformation:

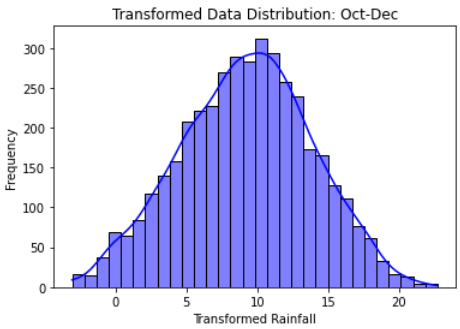








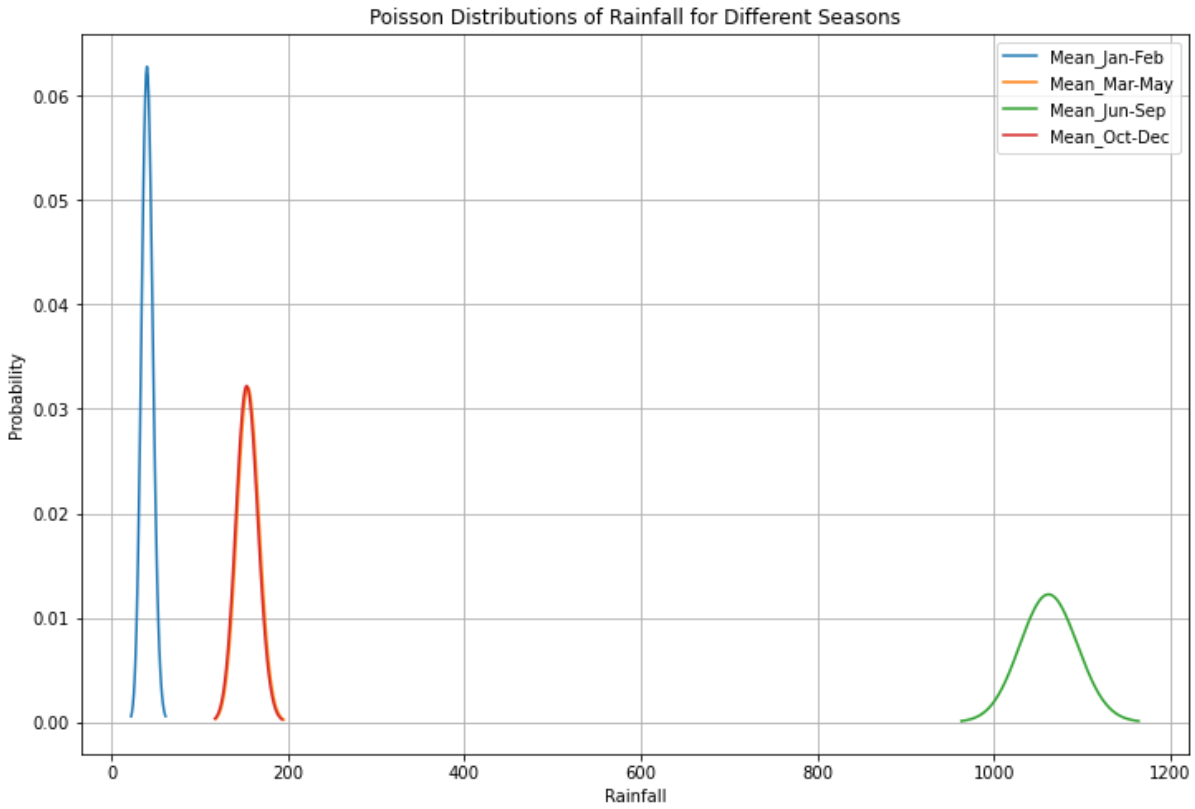




POISSON DISTRIBUTION:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import boxcox,poisson
import seaborn as sns

df=pd.read_csv(r'C:\Users\Roja Traders\OneDrive\Desktop\rainfall in india 1901-2015.csv')
df.dropna(inplace=True)

df['Mean_Jan-Feb'] = df['Jan-Feb']
df['Mean_Mar-May'] = df['Mar-May']
df['Mean_Jun-Sep'] = df['Jun-Sep']
df['Mean_Oct-Dec'] = df['Oct-Dec']

poisson_dists = {}
for season in ['Mean_Jan-Feb', 'Mean_Mar-May', 'Mean_Jun-Sep', 'Mean_Oct-Dec']:
    mu = df[season].mean()
```

```
    poisson_dists[season] = poisson(mu)

plt.figure(figsize=(12, 8))
for season, poisson_dist in poisson_dists.items():
    x = range(int(poisson_dist.ppf(0.001)), int(poisson_dist.ppf(0.999)) + 1)
    plt.plot(x, poisson_dist.pmf(x), label=season)

plt.title('Poisson Distributions of Rainfall for Different Seasons')
plt.xlabel('Rainfall')
plt.ylabel('Probability')
plt.legend()
plt.grid(True)
plt.show()
```

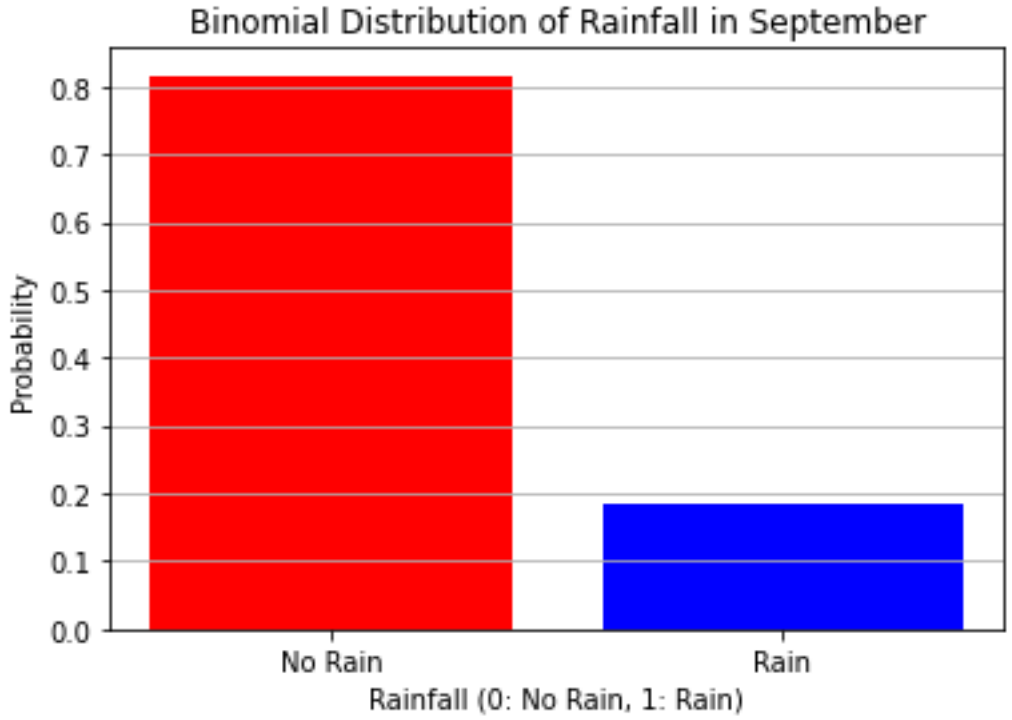


From this distribution, we can make inferences about the likelihood of different levels of rainfall occurring in each season.

BINOMIAL DISTRIBUTION:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import boxcox,poisson,expon,binom
import seaborn as sns
```

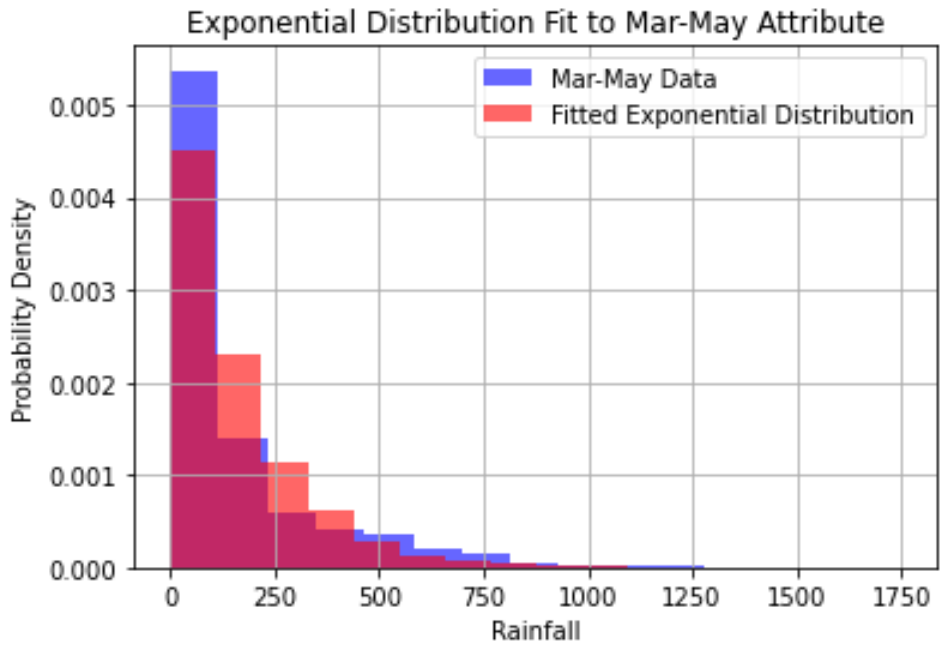
```
df=pd.read_csv(r'C:\Users\Roja Traders\OneDrive\Desktop\rainfall in india 1901-2015.csv')
df.dropna(inplace=True)
threshold = 300

df['SEP_binary'] = (df['SEP'] > threshold).astype(int)

p_success = df['SEP_binary'].mean()
p_failure = 1 - p_success
n_trials = len(df)
binom_dist = binom(n=n_trials, p=p_success)

x = [0, 1]
probabilities = [p_failure, p_success]
plt.bar(x, probabilities, color=['red', 'blue'])
plt.title('Binomial Distribution of Rainfall in September')
plt.xlabel('Rainfall (0: No Rain, 1: Rain)')
plt.ylabel('Probability')
plt.xticks(x, ['No Rain', 'Rain'])
plt.grid(axis='y')
plt.show()
```



We can infer from this distribution the probability of the rainfall in the September month.

EXPONENTIAL DISTRIBUTION:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import boxcox,poisson,expon,binom
import seaborn as sns

df=pd.read_csv(r'C:\Users\Roja Traders\OneDrive\Desktop\rainfall in india 1901-2015.csv')

df.dropna(inplace=True)
loc, scale = expon.fit(df['Mar-May'])
samples = expon.rvs(loc=loc, scale=scale, size=len(df))

plt.hist(df['Mar-May'], bins=15, density=True, alpha=0.6, color='b', label='Mar-May Data')
plt.hist(samples, bins=15, density=True, alpha=0.6, color='r', label='Fitted Exponential
Distribution')

plt.title('Exponential Distribution Fit to Mar-May Attribute')
plt.xlabel('Rainfall')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```

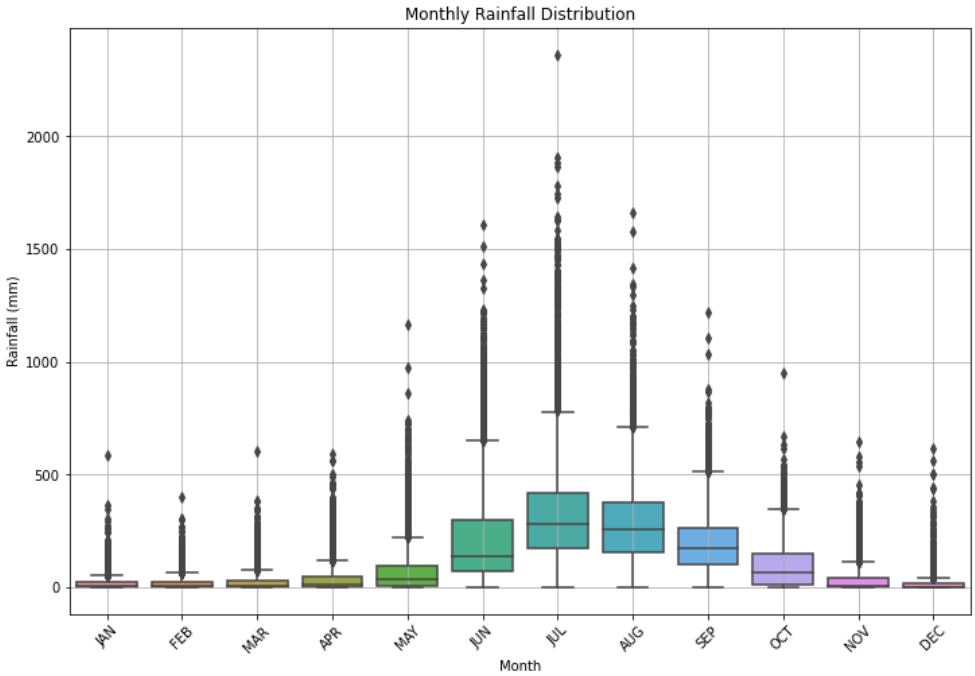


It provides a framework for analyzing the occurrence and timing of rainfall events between the month March and May.

3. Perform all types of visualization. Any 10 Visualization line plot, box plot, histogram, heat map etc

BOXPLOT:

```
months = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC']
import seaborn as sns

plt.figure(figsize=(12, 8))
sns.boxplot(data=data[months])
plt.title('Monthly Rainfall Distribution')
plt.xlabel('Month')
plt.ylabel('Rainfall (mm)')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

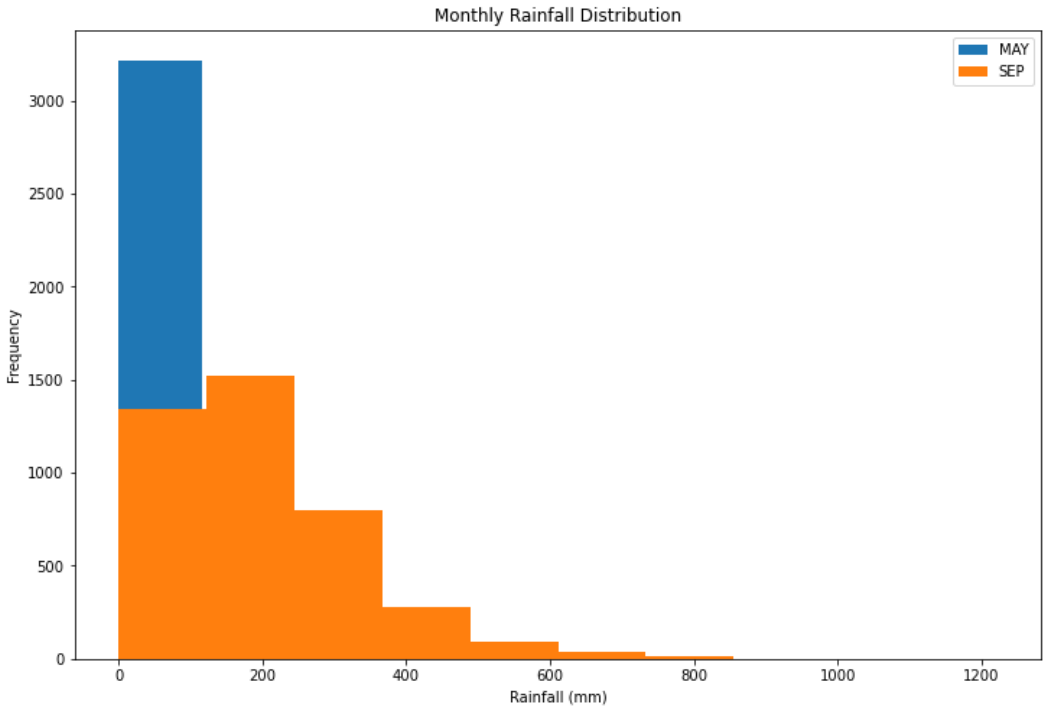


INFERENCE : From this visualization of the boxplot , we can infer high rainfall is common in those months which have longer upper whiskers than the lower whiskers.

HISTOGRAM:

```
months = ['MAY', 'SEP']
plt.figure(figsize=(12, 8))
for month in months:
    plt.hist(data[month], label=month)

plt.title('Monthly Rainfall Distribution')
plt.xlabel('Rainfall (mm)')
```

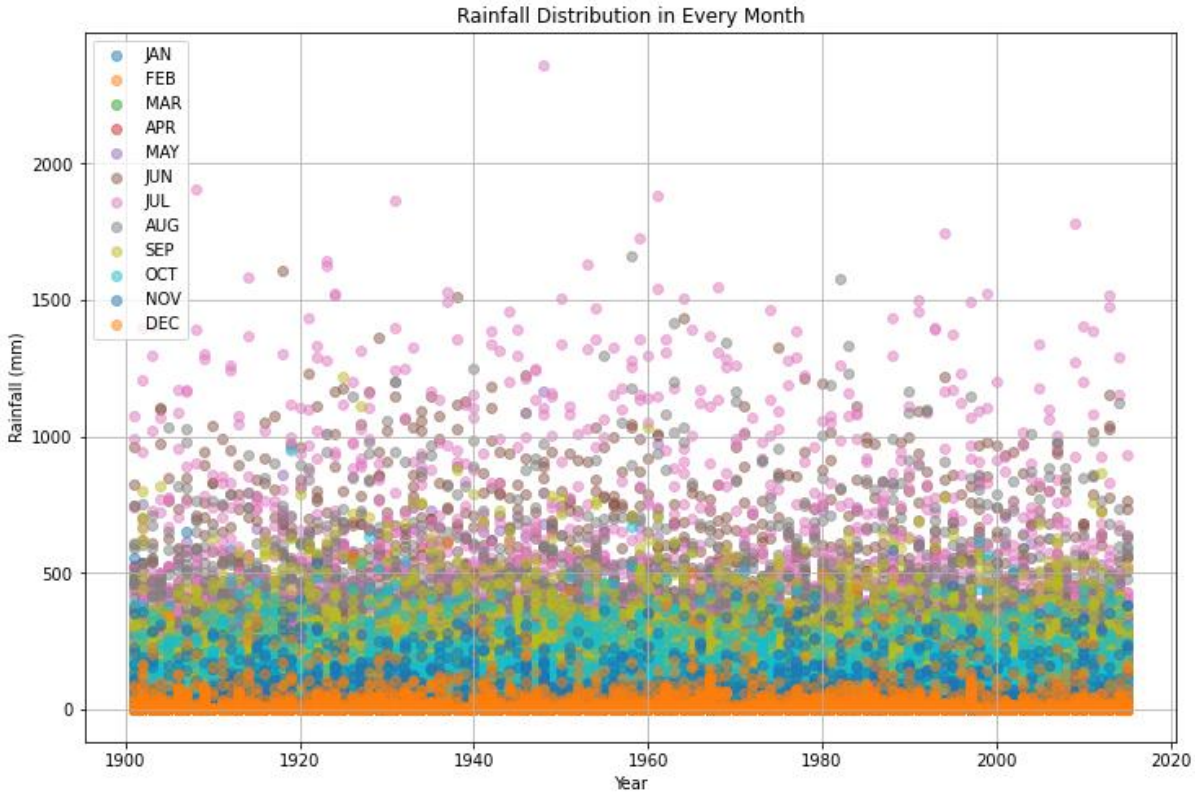
```
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



INFERENCE: We can how the rainfall pattern has been distributed in different months by the width of the histogram. Here September is wider than May. Hence there has been higher rainfall in September compared to May.

SCATTERPLOT:

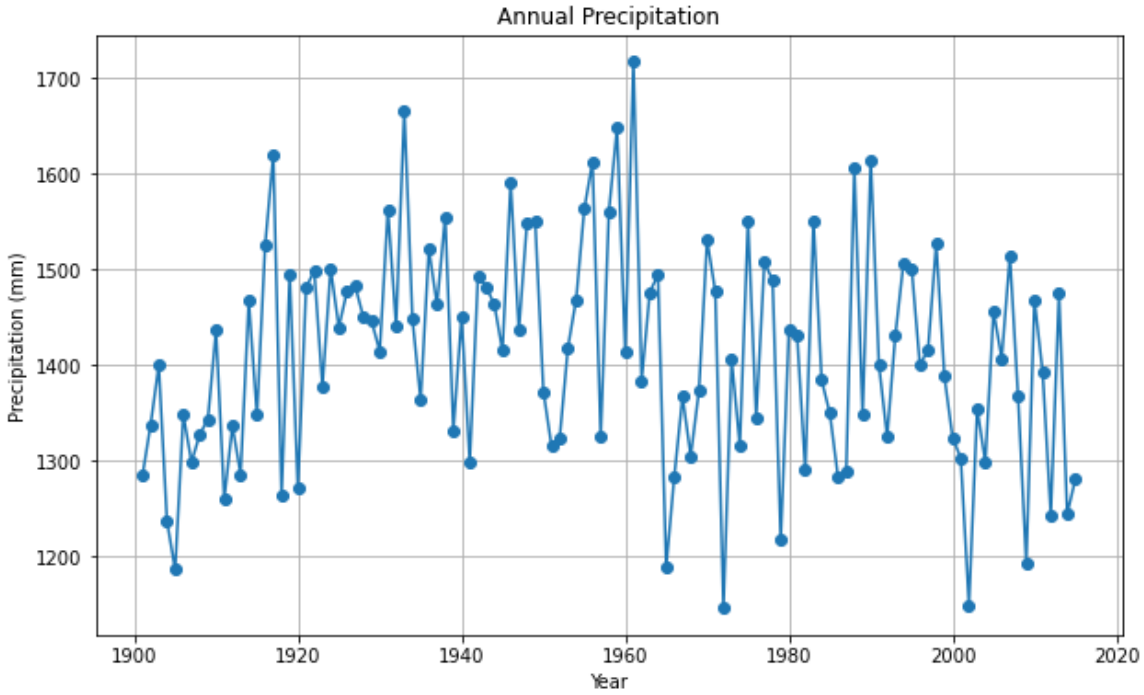
```
months = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC']

plt.figure(figsize=(12, 8))
for month in months:
    plt.scatter(data['YEAR'], data[month], alpha=0.5, label=month)

plt.title('Rainfall Distribution in Every Month')
plt.xlabel('Year')
plt.ylabel('Rainfall (mm)')
plt.legend()
plt.grid(True)

plt.show()
```



INFERENCE: From this scatter plot we can infer that clustered data points have been highly distributed for the months JULY , AUG and SEP which means they have high rainfall over the years.

LINE PLOT:

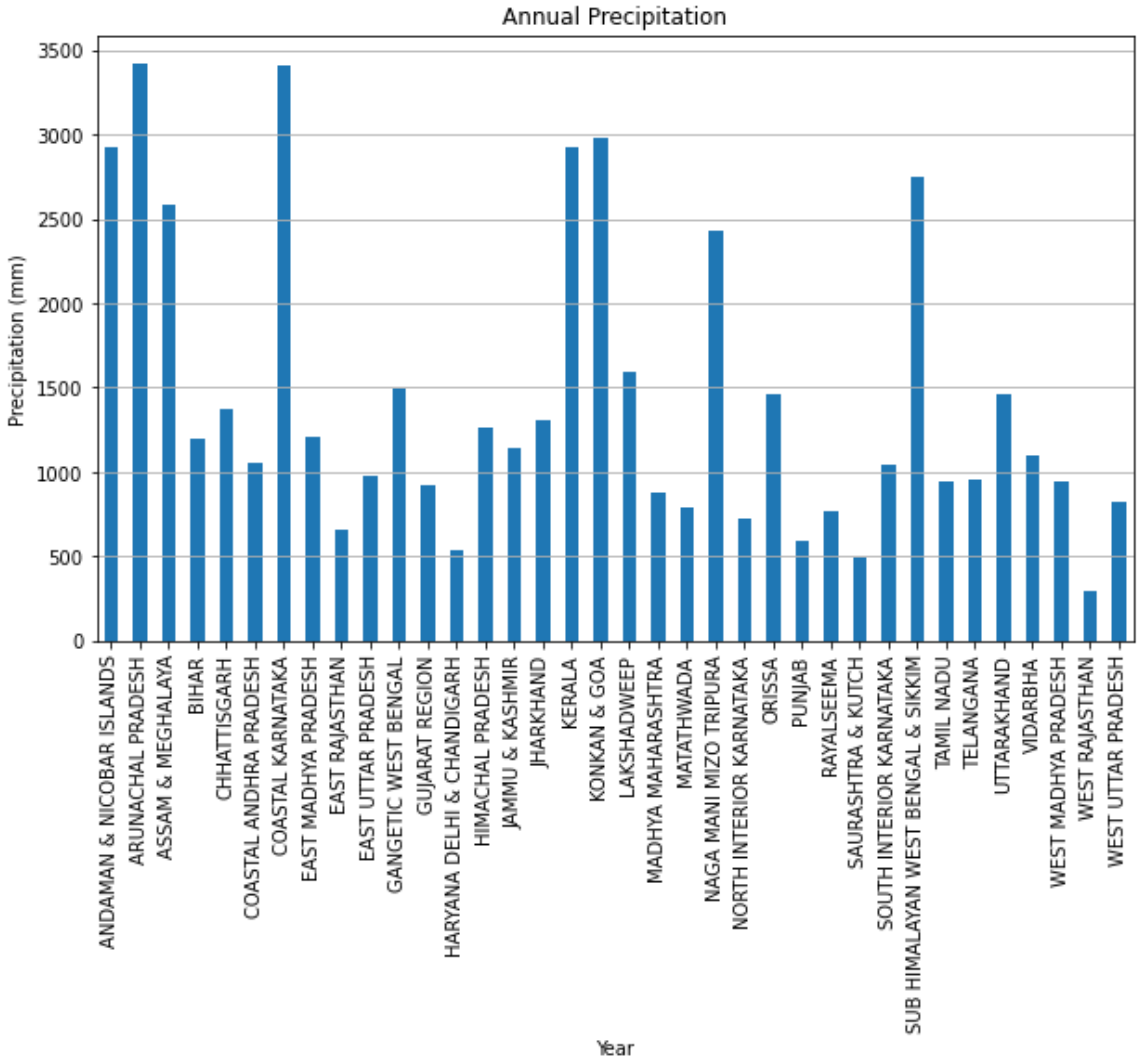
```
annual_data = data.groupby('YEAR')['ANNUAL'].mean()
plt.figure(figsize=(10, 6))
plt.plot(annual_data.index, annual_data.values, marker='o', linestyle='-')
plt.title('Annual Precipitation')
plt.xlabel('Year')
plt.ylabel('Precipitation (mm)')
plt.grid(True)
plt.show()
```



INFERENCE: We can infer that the Sharp spikes plot represent extreme precipitation events, such as heavy rainfall years or drought years.

BAR PLOT:

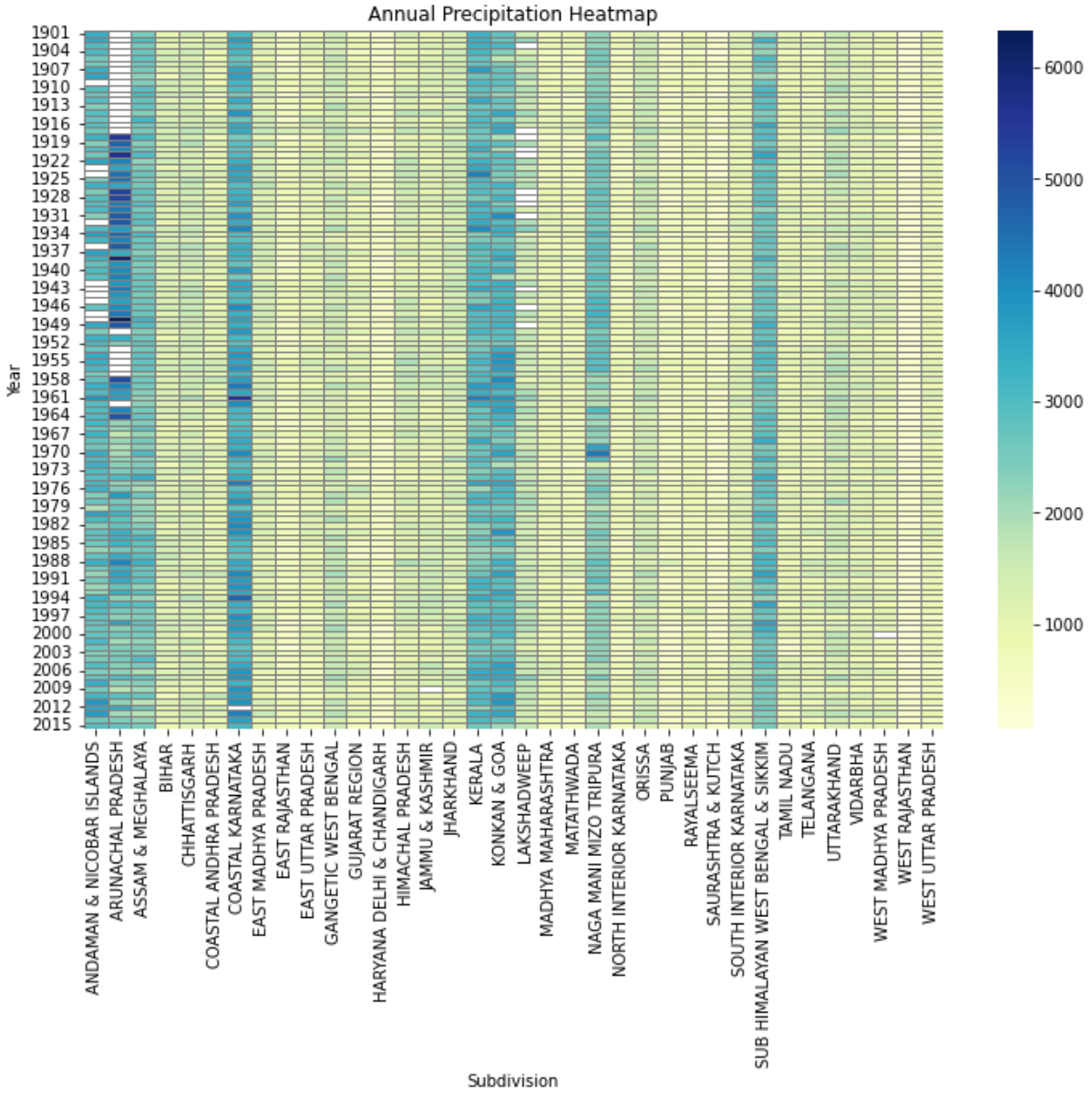
```
annual_data = data.groupby('SUBDIVISION')['ANNUAL'].mean()
plt.figure(figsize=(10, 6))
annual_data.plot(kind='bar')
plt.title('Annual Precipitation')
plt.xlabel('Year')
plt.ylabel('Precipitation (mm)')
plt.grid(axis='y')
plt.show()
```



INFERENCE: We can infer the level of rainfall in different subdivisions of India. We are able to understand that East Madhya Pradesh has the highest distribution of rainfall using this bar plots

HEATMAP:

```
pivot_table = data.pivot_table(index='YEAR', columns='SUBDIVISION', values='ANNUAL',
aggfunc='mean')
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, cmap='YlGnBu', linewidths=0.5, linecolor='grey')
plt.title('Annual Precipitation Heatmap')
plt.xlabel('Subdivision')
plt.ylabel('Year')

plt.show()
```



INFERENCE : We can infer that darker shades indicate higher levels of precipitation, while lighter shades indicate lower levels.

4.Implement all types of matrices square, diagonal, identity symmetric orthogonal and find determinant ,inverse , Eigen value and Eigen vector

```
import numpy as np

squr_mat=np.array([[1 ,2 ,3],[3 ,4,5],[3 ,7,8]])

print(squr_mat)
```

```
[[1 2 3]
 [3 4 5]
 [3 7 8]]
```

```
dia=squr_mat.diagonal()

print(dia)
```

```
[1 4 8]
```

```
idem=np.eye(squr_mat.shape[0])

print(idem)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
orthogonal_matrix = np.array([[0, 1, 0],

                              [1, 0, 0],

                              [0, 0, -1]])

print(orthogonal_matrix)
```

```
[[ 0  1  0]
 [ 1  0  0]
 [ 0  0 -1]]
```

```
det=np.linalg.det(squr_mat)

print(det)
```

```
6.000000000000003
```

```
invers=np.linalg.inv(squr_mat)

print(invers)
```

```
[[-0.5         0.83333333 -0.33333333]
 [-1.5        -0.16666667  0.66666667]
 [ 1.5        -0.16666667 -0.33333333]]
```

```
m,v=np.linalg.eig(squr_mat)

print("The eigen value is")

print(m)

print("the eigen vector is")

print(v)
```

```
The eigen value is
[13.478192+0.j         -0.239096+0.62289379j -0.239096-0.62289379j]
the eigen vector is
[[ 0.2775324 +0.j          -0.06679751+0.50833064j -0.06679751-0.50833064j]
 [ 0.5155073 +0.j           0.65157427+0.j          0.65157427-0.j        ]
 [ 0.81069599+0.j          -0.51233867-0.22382607j -0.51233867+0.22382607j]]
```

5. Perform correlation analysis and covariance analysis and explain the reduced dataset.

```
reduced_data = data.drop(columns=['SUBDIVISION', 'YEAR'])
correlation_matrix = reduced_data.corr()
covariance_matrix = reduced_data.cov()

print("Correlation Matrix:")
print(correlation_matrix)
print("\nCovariance Matrix:")
print(covariance_matrix)
```

Correlation Matrix:

	JAN	FEB	MAR	...	Mar-May	Jun-Sep	Oct-Dec
JAN	1.000000	0.456459	0.399812	...	0.241217	-0.024268	0.087145
FEB	0.456459	1.000000	0.579576	...	0.379886	0.047948	0.018598
MAR	0.399812	0.579576	1.000000	...	0.641823	0.161403	0.087602
APR	0.208090	0.361791	0.554360	...	0.863684	0.390712	0.320196
MAY	0.127298	0.200900	0.362696	...	0.915210	0.494495	0.522896
JUN	-0.037414	0.027404	0.164650	...	0.535912	0.893636	0.406475
JUL	-0.052095	0.013354	0.096497	...	0.313516	0.907881	0.190529
AUG	0.013644	0.070252	0.136355	...	0.317500	0.840761	0.157282
SEP	0.022724	0.078842	0.176751	...	0.470525	0.701744	0.319255
OCT	0.009575	-0.008786	0.083714	...	0.466106	0.415959	0.862828
NOV	0.063049	-0.023225	0.008610	...	0.272978	0.125444	0.808548
DEC	0.219185	0.131614	0.134524	...	0.227635	0.041042	0.604685
ANNUAL	0.105696	0.181563	0.322199	...	0.696318	0.943661	0.529596
Jan-Feb	0.842896	0.863497	0.577116	...	0.366450	0.015241	0.060643
Mar-May	0.241217	0.379886	0.641823	...	1.000000	0.471468	0.447902
Jun-Sep	-0.024268	0.047948	0.161403	...	0.471468	1.000000	0.310142
Oct-Dec	0.087145	0.018598	0.087602	...	0.447902	0.310142	1.000000

Covariance Matrix:

	JAN	FEB	...	Jun-Sep	Oct-Dec
JAN	1123.705635	547.204789	...	-574.302234	486.139705
FEB	547.204789	1278.921373	...	1210.544447	110.685226
MAR	627.623246	970.619991	...	5335.975603	682.688669
APR	469.208110	870.297027	...	18553.718114	3584.213292
MAY	522.979603	880.521813	...	42784.434730	10664.553435
JUN	-292.896828	228.873633	...	147333.087256	15797.114707
JUL	-470.373744	128.636125	...	172638.012225	8540.293631
AUG	85.847772	471.574347	...	111411.148032	4912.897325
SEP	103.037654	381.391856	...	67012.549594	7186.529135
OCT	31.881228	-31.210244	...	29167.462379	14261.843388
NOV	144.301154	-56.708006	...	6046.449706	9186.727746
DEC	309.957169	198.558753	...	1222.299455	4245.051426
ANNUAL	3202.434778	5868.717564	...	602139.287508	79658.063865
Jan-Feb	1670.913802	1826.150370	...	636.294983	596.794971
Mar-May	1619.753013	2721.387812	...	66673.568824	14930.988830
Jun-Sep	-574.302234	1210.544447	...	498393.925538	36436.582955
Oct-Dec	486.139705	110.685226	...	36436.582955	27693.655177

```
correlation_matrix = reduced_data.corr()
covariance_matrix = reduced_data.cov()

# Plot correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
```

```
plt.title('Correlation Matrix')
plt.show()

# Plot covariance matrix
plt.figure(figsize=(10, 8))
sns.heatmap(covariance_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Covariance Matrix')
plt.show()
```