Crop Price Prediction

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
df = pd.read_csv('C:\\Users\\sowmy\\OneDrive\\Desktop\\archive (1)\\dataset.csv')
```

Explorative Data Analysis

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

Out[ ]:

	State	Crop	CostCultivation	CostCultivation2	Production	Yield	Temperature
0	Uttar Pradesh	ARHAR	9794.05	9800.25	1941.55	9.83	28.96
1	Karnataka	ARHAR	10593.15	10594.15	2172.46	7.47	29.22
2	Gujarat	ARHAR	13468.82	13469.82	1898.30	9.59	28.47
3	Andhra Pradesh	ARHAR	17051.66	17052.66	3670.54	6.42	28.49
4	Maharashtra	ARHAR	17130.55	17131.55	2775.80	8.72	28.30

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

Out[ ]:

	State	Crop	CostCultivation	CostCultivation2	Production	Yield	Tempera
44	Tamil Nadu	SUGARCANE	66335.06	66336.06	85.79	1015.45	
45	Madhya Pradesh	WHEAT	12464.40	12465.40	810.25	23.59	
46	Punjab	WHEAT	17945.58	17946.58	804.80	39.83	
47	Uttar Pradesh	WHEAT	18979.38	18980.38	769.84	34.99	
48	Rajasthan	WHEAT	19119.08	19120.08	683.58	37.19	

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

Out[ ]:
```
(49, 9)
```

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

Out[ ]:
```
Index(['State', 'Crop', 'CostCultivation', 'CostCultivation2', 'Production',
       'Yield', 'Temperature', 'RainFall Annual', 'Price'],
      dtype='object')
```

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
State              0
Crop               0
CostCultivation    0
CostCultivation2   0
Production         0
Yield              0
Temperature        0
RainFall Annual    0
Price              0
dtype: int64
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   State             49 non-null     object 
 1   Crop              49 non-null     object 
 2   CostCultivation   49 non-null     float64
 3   CostCultivation2  49 non-null     float64
 4   Production        49 non-null     float64
 5   Yield             49 non-null     float64
 6   Temperature       49 non-null     float64
 7   RainFall Annual   49 non-null     float64
 8   Price             49 non-null     float64
dtypes: float64(7), object(2)
memory usage: 3.6+ KB
```

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

Out[ ]:

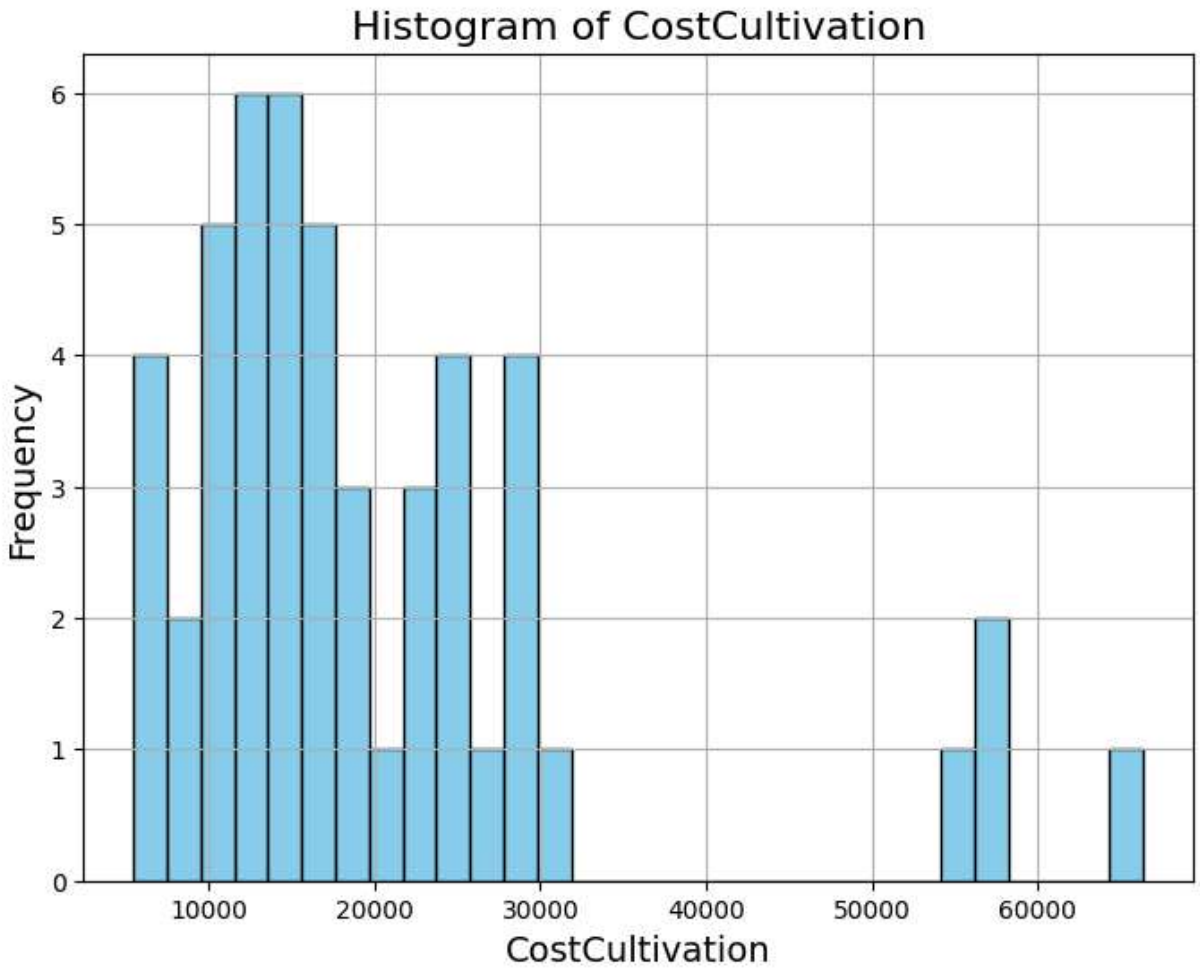
	CostCultivation	CostCultivation2	Production	Yield	Temperature	RainF Annu
count	49.000000	49.000000	49.000000	49.000000	49.000000	49.0000
mean	20363.537347	20364.643469	1620.537755	98.086735	28.780612	2951.7408
std	13561.435306	13561.350894	1104.990472	245.293123	0.246555	373.9649
min	5483.540000	5484.540000	85.790000	1.320000	28.110000	2352.1000
25%	12774.410000	12775.410000	732.620000	9.590000	28.660000	2687.2000
50%	17022.000000	17023.000000	1595.560000	13.700000	28.760000	2957.8000
75%	24731.060000	24732.060000	2228.970000	36.610000	28.890000	3264.4000
max	66335.060000	66336.060000	5777.480000	1015.450000	29.460000	3722.8000

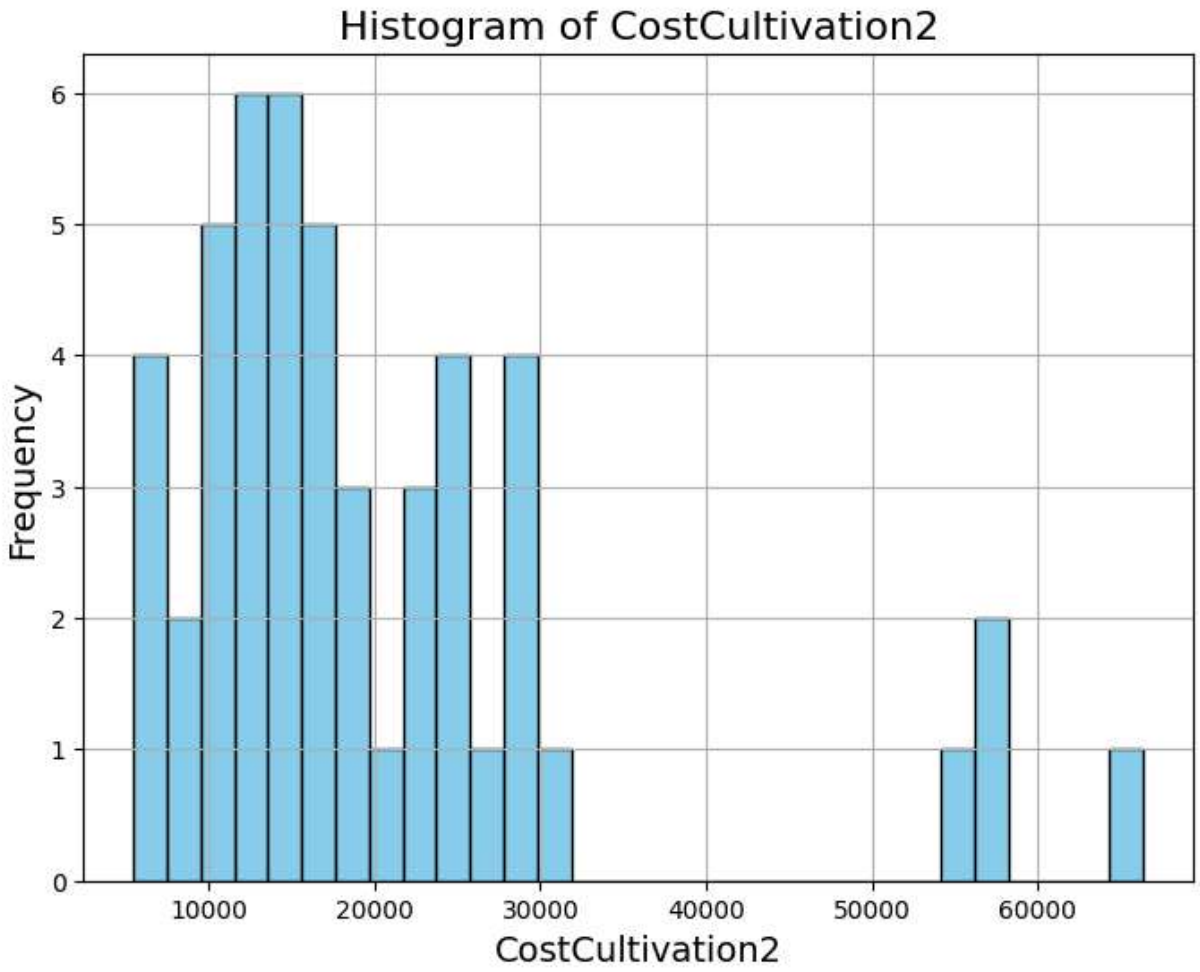
In [ ]:
```
df.nunique()
```

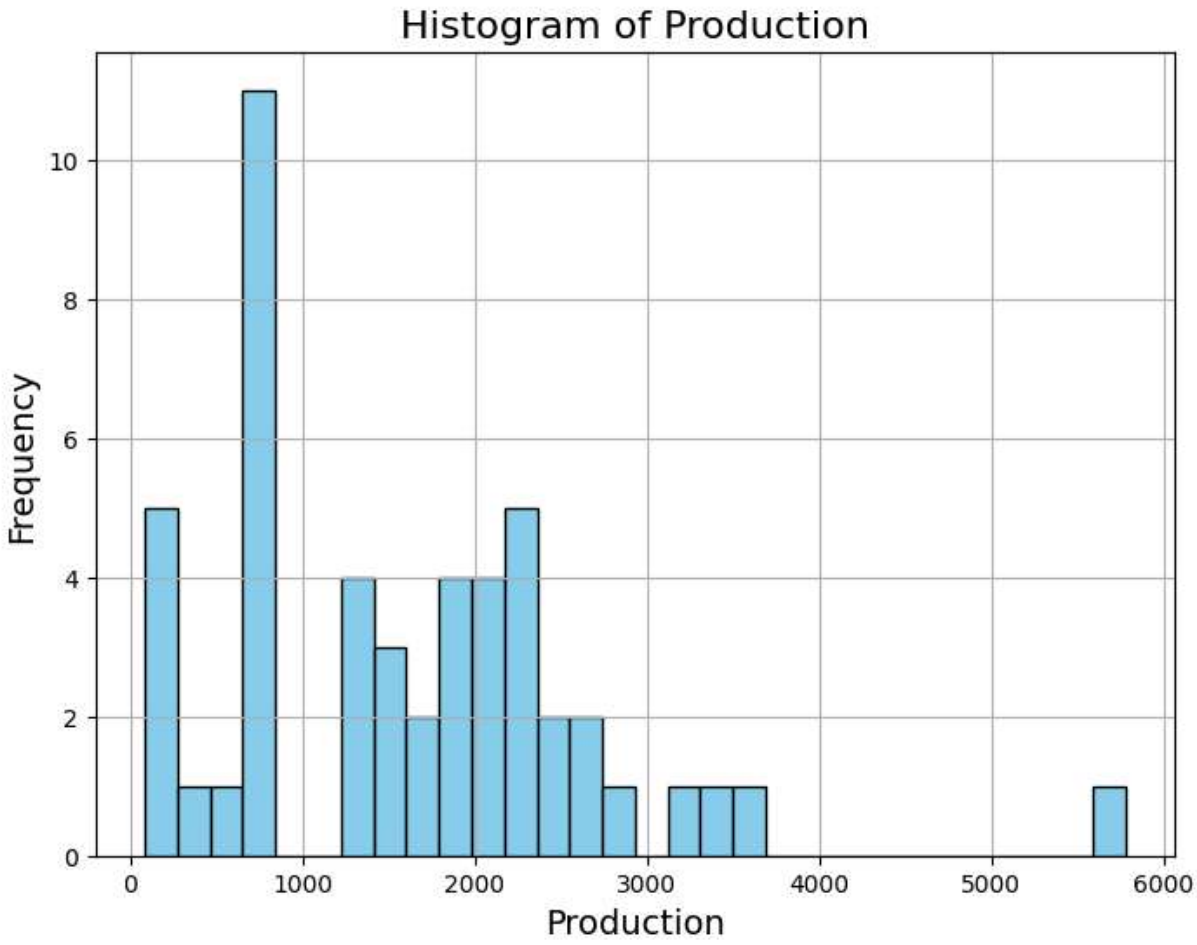
Out[ ]:
```
State              13
Crop               10
CostCultivation    49
CostCultivation2   49
Production         49
Yield              49
Temperature        35
RainFall Annual    43
Price              49
dtype: int64
```

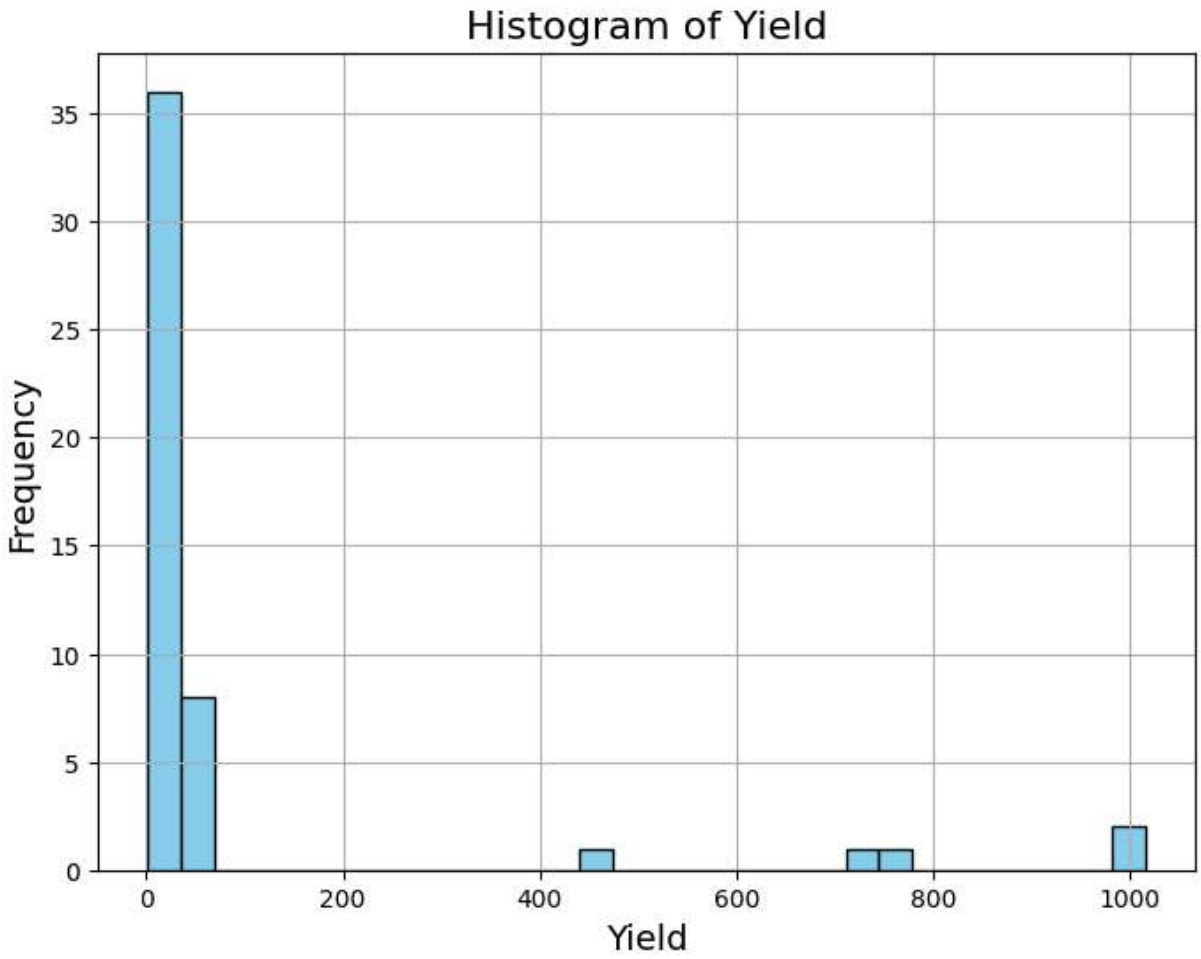
In [ ]:
```
columns_of_interest = ['CostCultivation', 'CostCultivation2', 'Production', 'Yield'

for column in columns_of_interest:
    plt.figure(figsize=(8, 6))
```

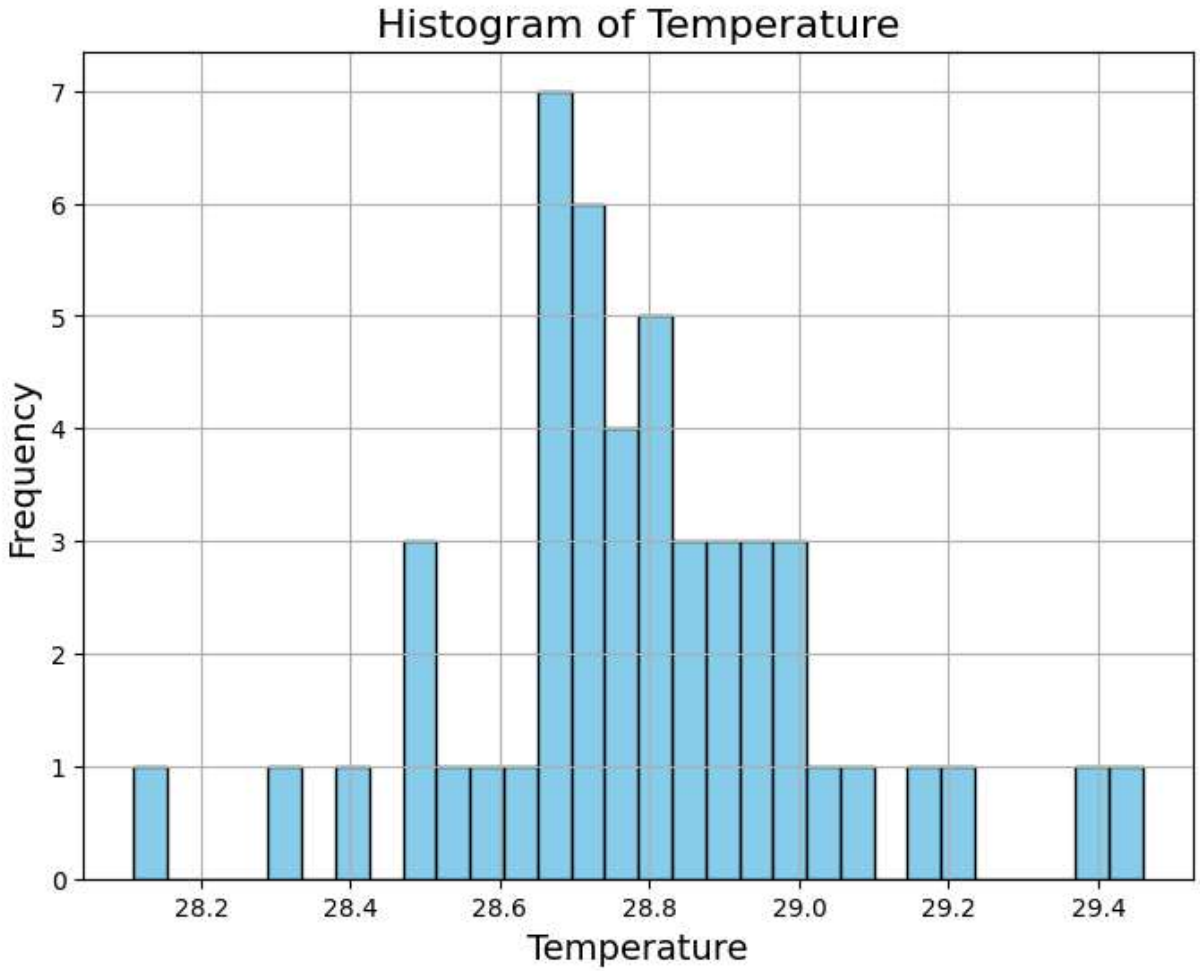
```
    plt.hist(df[column], bins=30, color='skyblue', edgecolor='black')
    plt.title(f'Histogram of {column}', fontsize=16)
    plt.xlabel(column, fontsize=14)
    plt.ylabel('Frequency', fontsize=14)
    plt.grid(True)
    plt.show()
```

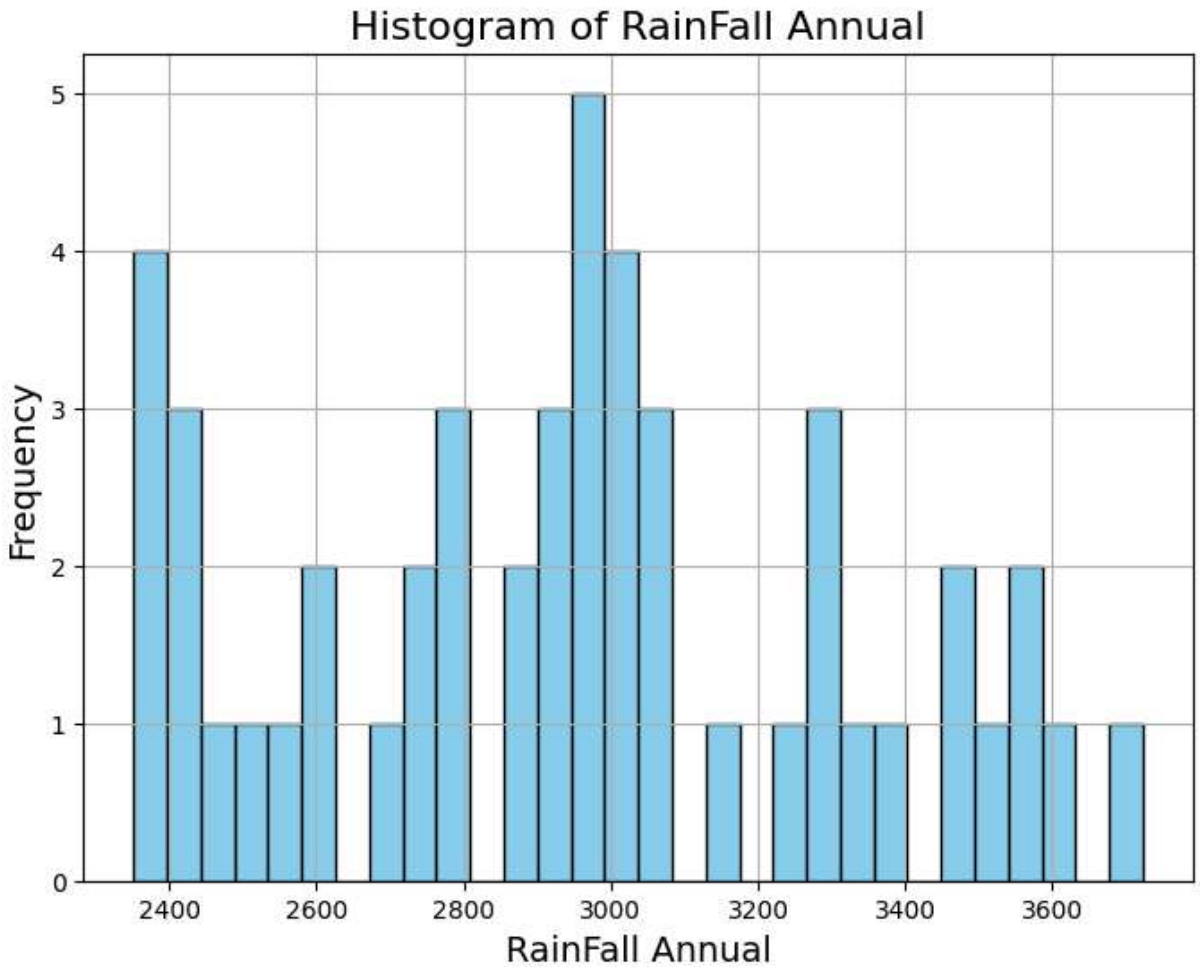


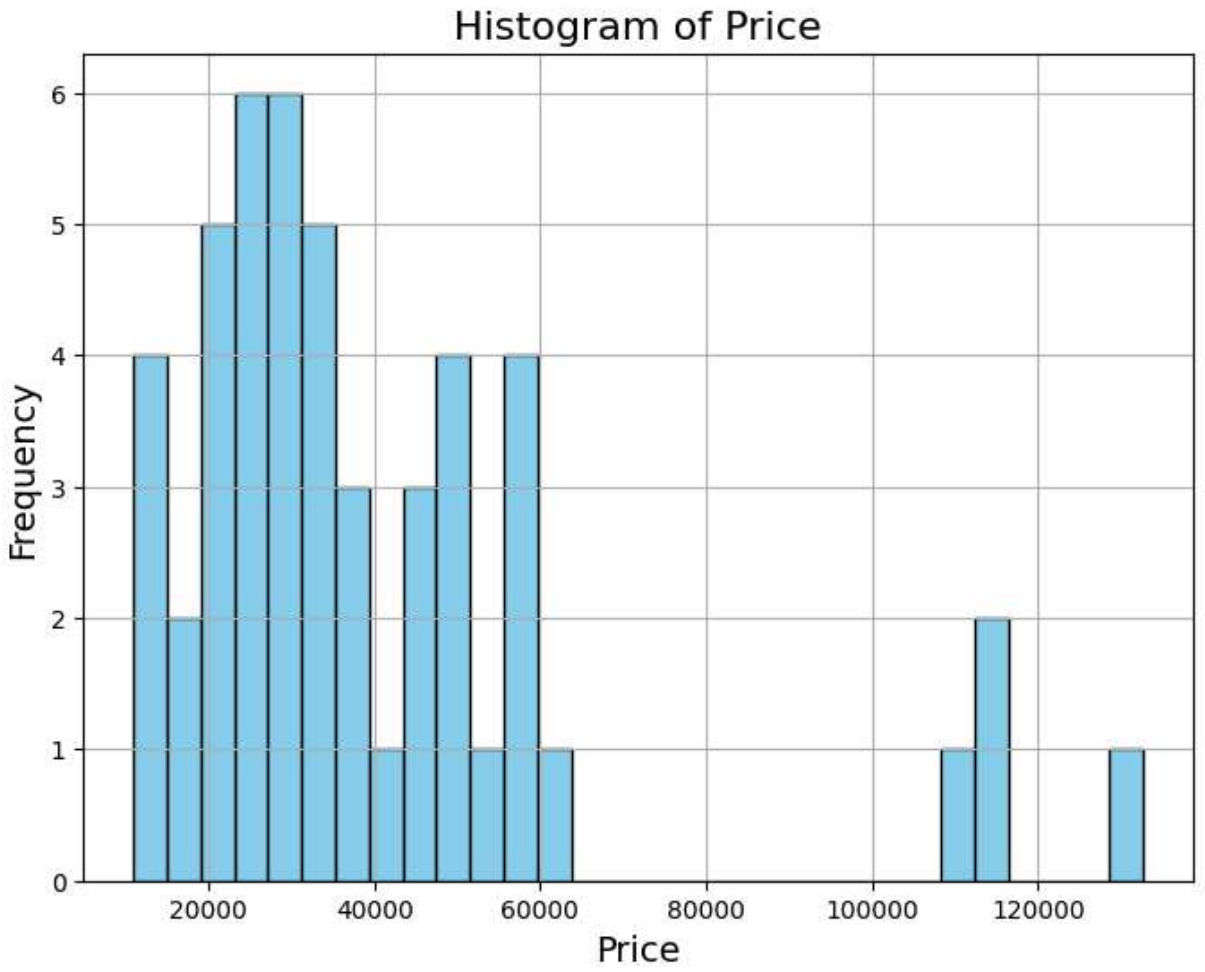

Histogram of CostCultivation2
Frequency
CostCultivation2
0
1
2
3
4
5
6
10000
20000
30000
40000
50000
60000


Histogram of Production
Frequency
Production
0
2
4
6
8
10
0
1000
2000
3000
4000
5000
6000


Histogram of Yield
Frequency
Yield
0
5
10
15
20
25
30
35
0
200
400
600
800
1000


Histogram of Temperature
Frequency
Temperature
0
1
2
3
4
5
6
7
28.2
28.4
28.6
28.8
29.0
29.2
29.4

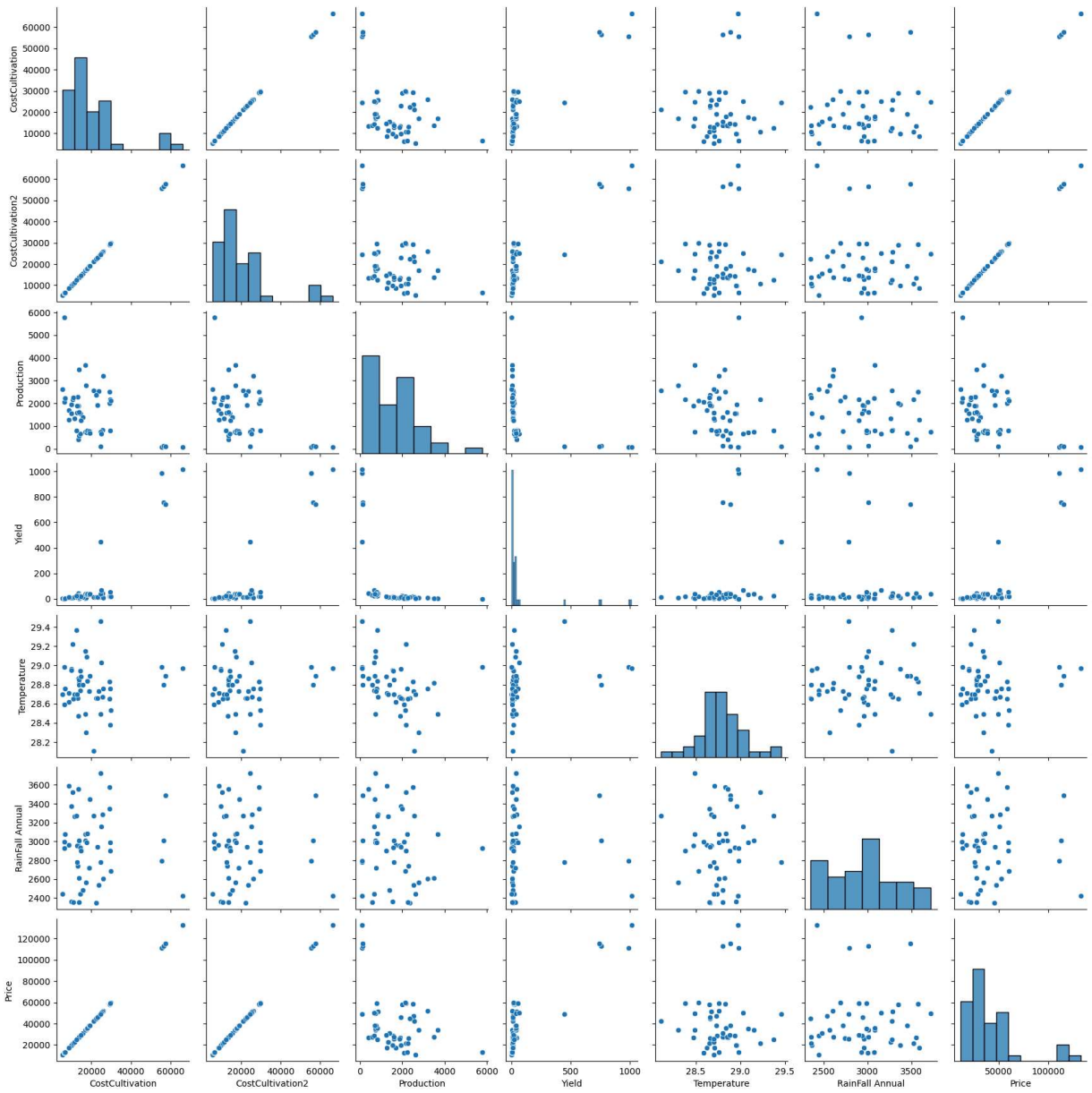

Histogram of RainFall Annual
Frequency
RainFall Annual
0
1
2
3
4
5
2400
2600
2800
3000
3200
3400
3600



In [ ]:
```
print("DataFrame Columns:", df.columns)
print("Columns of Interest:", columns_of_interest)
```

```
DataFrame Columns: Index(['State', 'Crop', 'CostCultivation', 'CostCultivation2', 'P
roduction',
       'Yield', 'Temperature', 'RainFall Annual', 'Price'],
      dtype='object')
Columns of Interest: ['CostCultivation', 'CostCultivation2', 'Production', 'Yield',
'Temperature', 'RainFall Annual', 'Price']
```

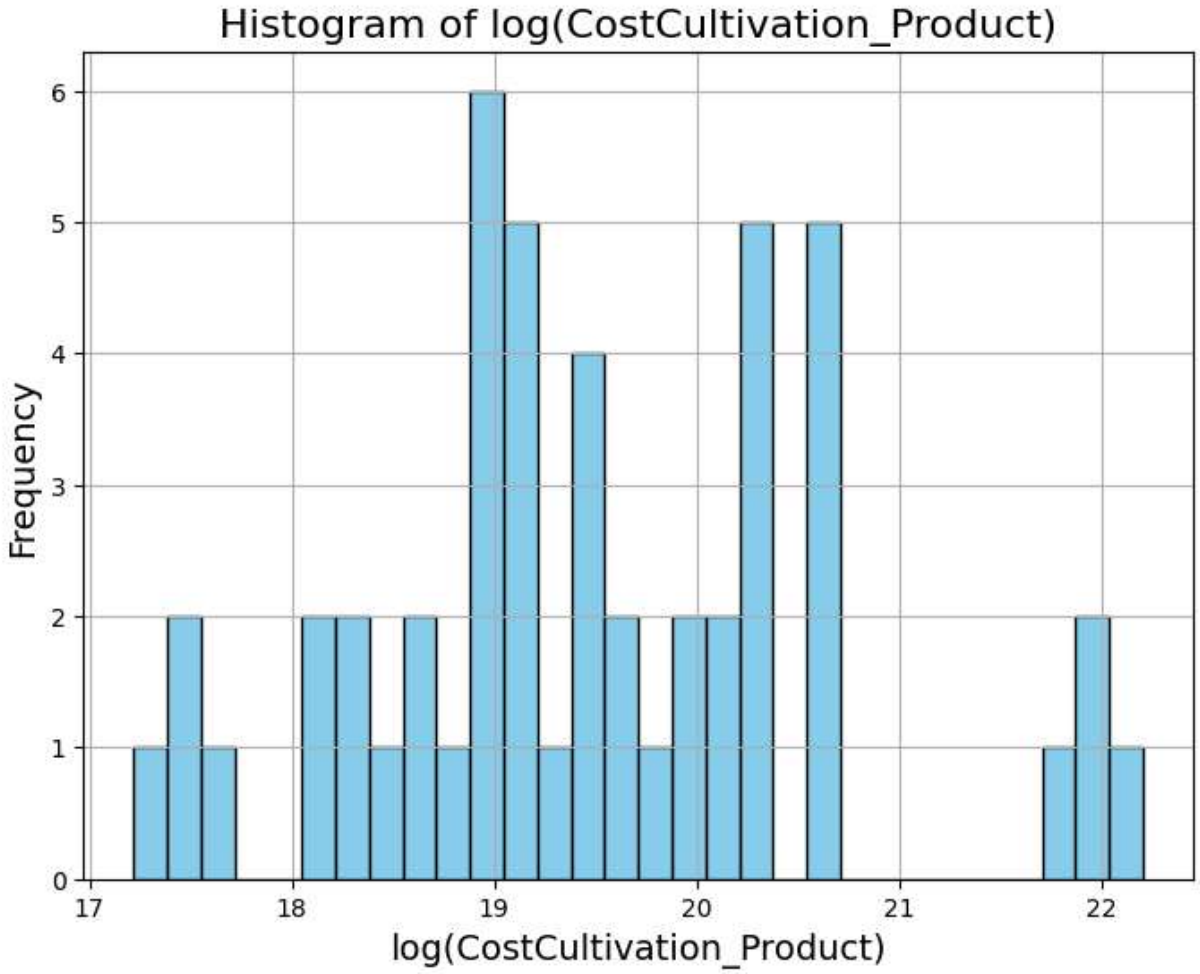
In [ ]:
```
import seaborn as sns

# Remove the "_log" suffixes from the DataFrame column names
df.columns = df.columns.str.replace('_log', '')

# Specify the columns of interest without the "_log" suffix
columns_of_interest = ['CostCultivation', 'CostCultivation2', 'Production', 'Yield'

# Create pairplot to visualize relationships between all features
sns.pairplot(df[columns_of_interest], kind='scatter', diag_kind='auto')
plt.show()
```

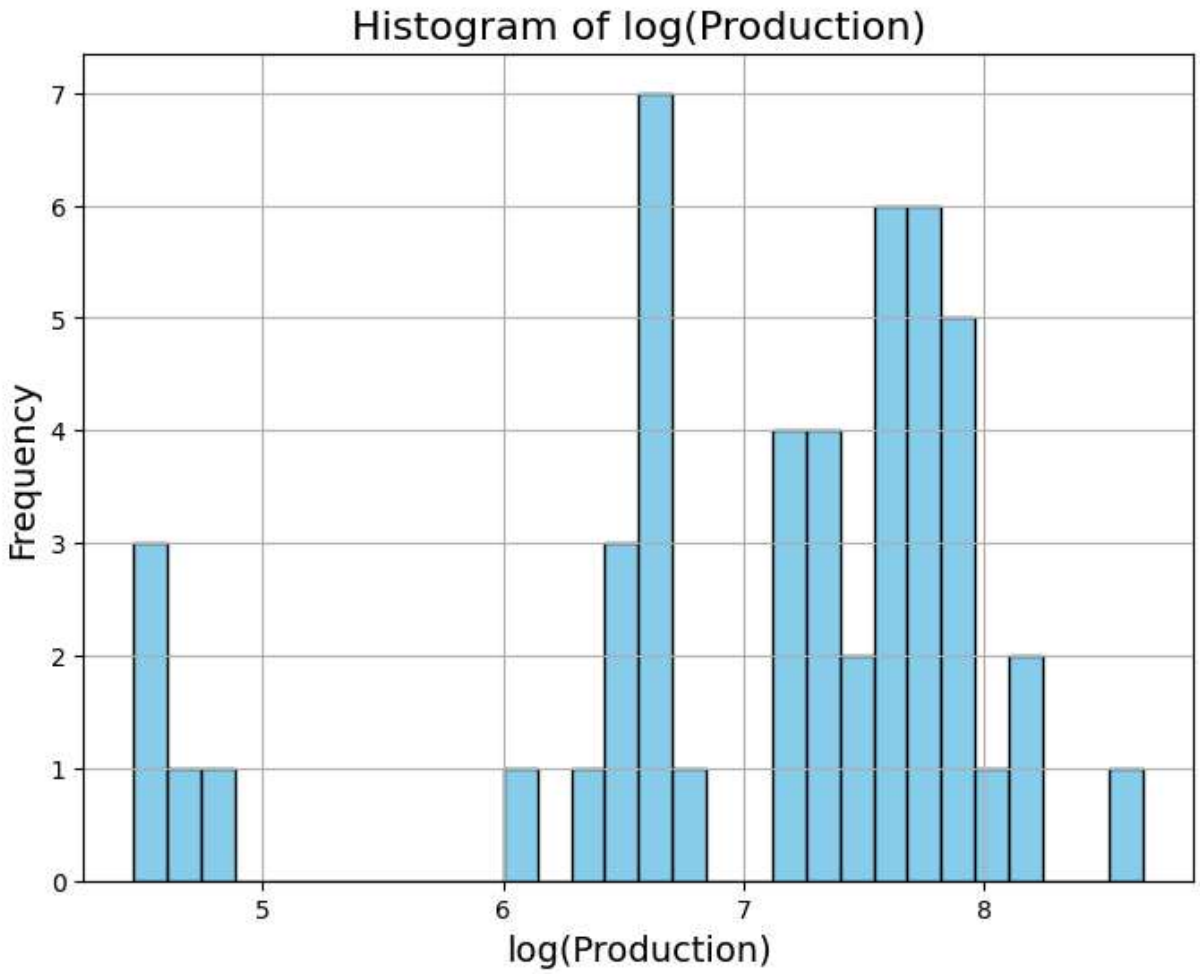


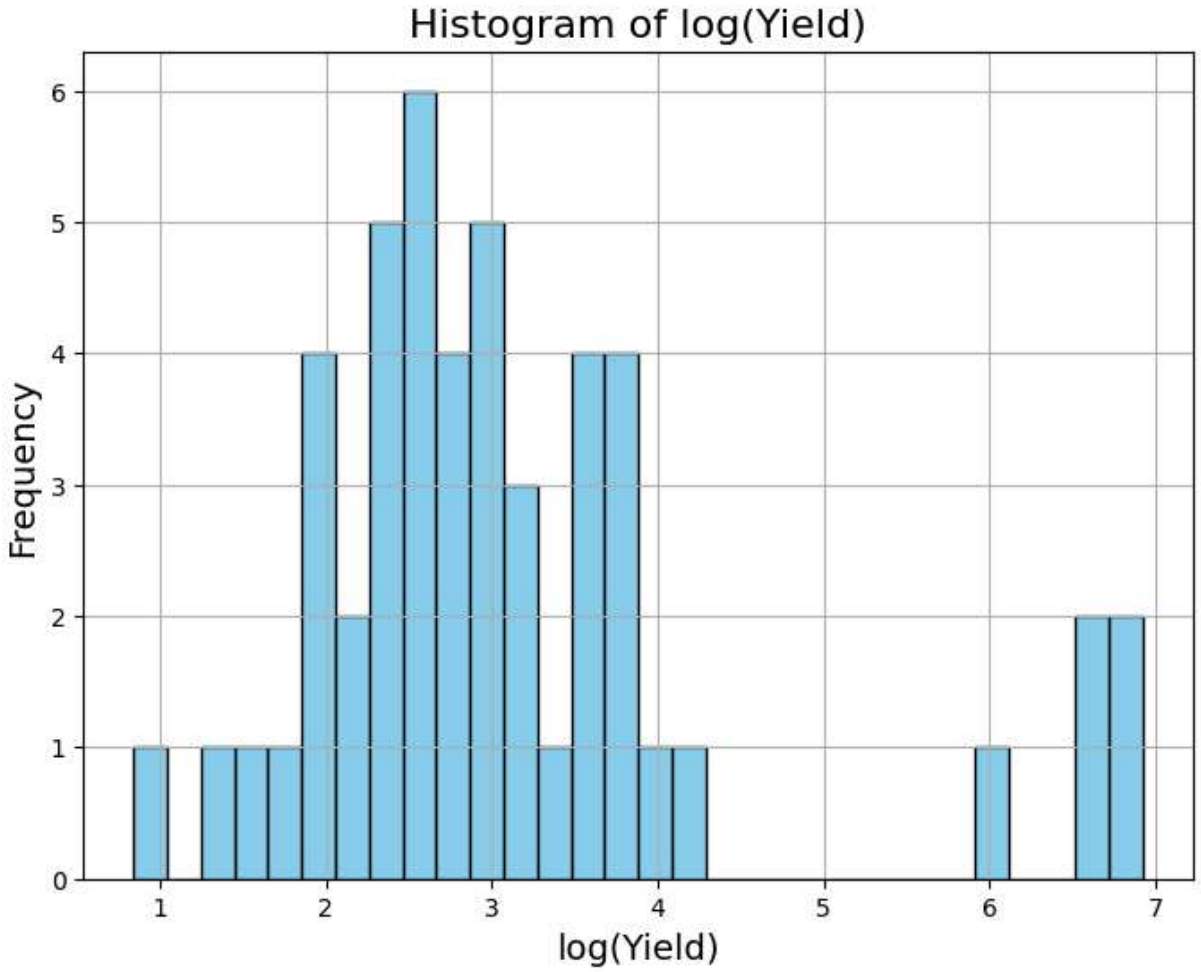
In [ ]:
```
# Calculate the product of "CostCultivation" and "CostCultivation2"
df['CostCultivation_Product'] = df['CostCultivation'] * df['CostCultivation2']

df = df.drop(columns=['CostCultivation', 'CostCultivation2'])

# Display the first few rows of the DataFrame with the new feature
print(df.head())
```

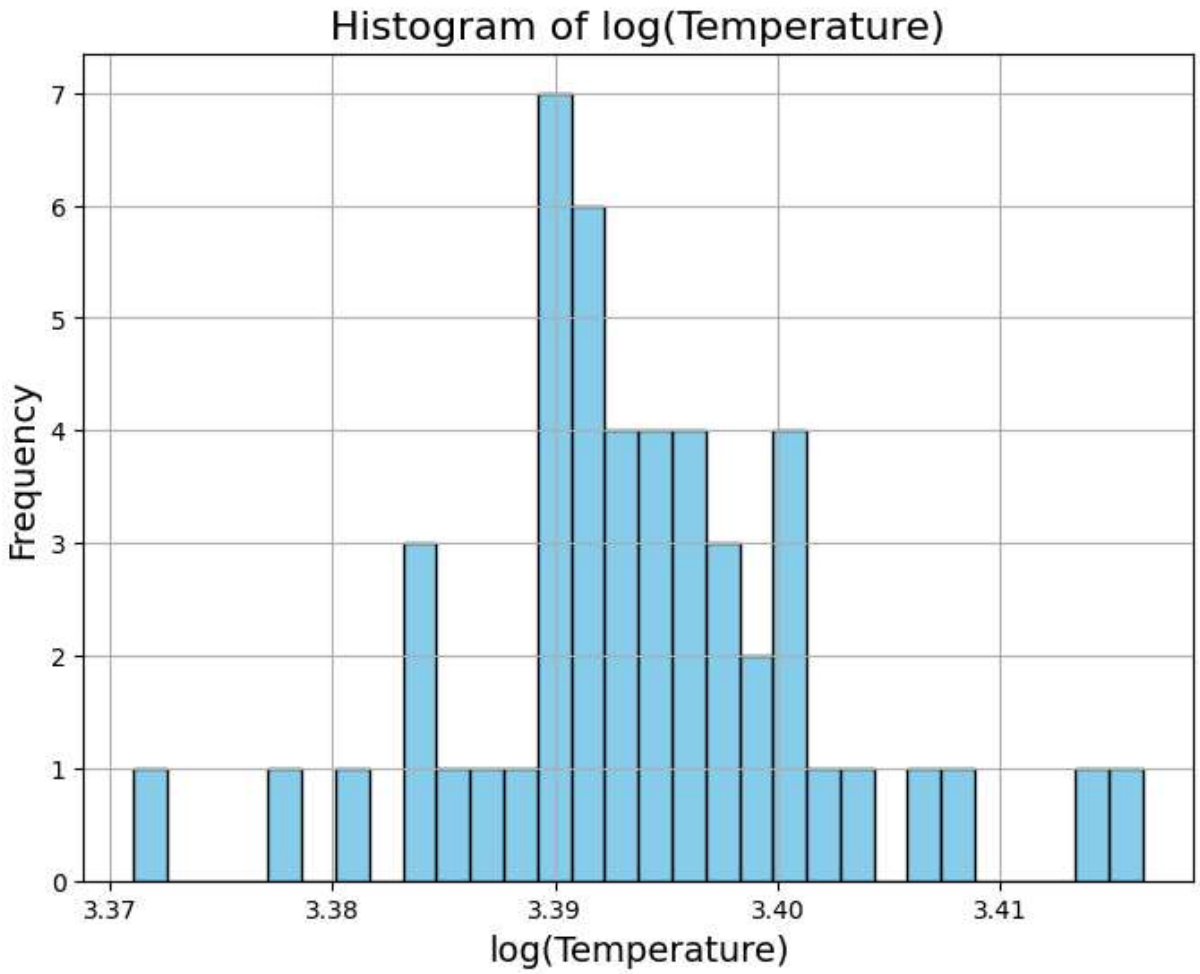
```
            State   Crop  Production  Yield  Temperature  RainFall Annual  \
0   Uttar Pradesh  ARHAR     1941.55   9.83        28.96           3373.2
1       Karnataka  ARHAR     2172.46   7.47        29.22           3520.7
2         Gujarat  ARHAR     1898.30   9.59        28.47           2957.4
3  Andhra Pradesh  ARHAR     3670.54   6.42        28.49           3079.6
4     Maharashtra  ARHAR     2775.80   8.72        28.30           2566.7

      Price  CostCultivation_Product
0  19589.10             9.598414e+07
1  21187.30             1.122254e+08
2  26938.64             1.814226e+08
3  34104.32             2.907762e+08
4  34262.10             2.934729e+08
```

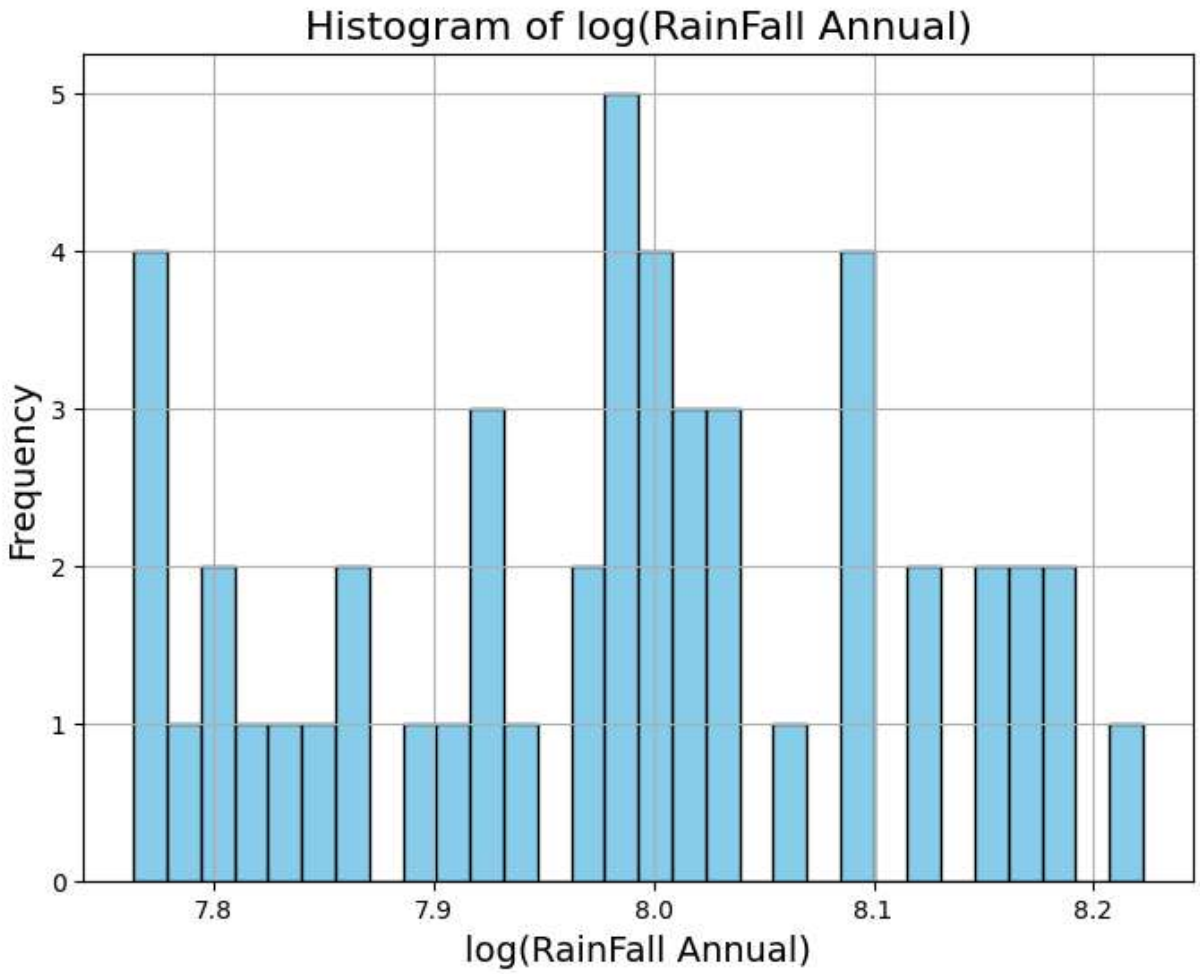
In [ ]:
```
columns_of_interest = ['CostCultivation_Product', 'Production', 'Yield', 'Temperatu

# Check if the columns of interest exist in the DataFrame
for column in columns_of_interest:
    if column not in df.columns:
        print(f"Column '{column}' does not exist in the DataFrame.")

# Apply logarithmic transformation to each column of interest and create new column
for column in columns_of_interest:
    if column in df.columns:
        df[column + '_log'] = np.log(df[column] + 1)  # Adding 1 to avoid log(0) if

# Plot histograms of the log-transformed columns
for column in columns_of_interest:
    if column + '_log' in df.columns:
        plt.figure(figsize=(8, 6))
        plt.hist(df[column + '_log'], bins=30, color='skyblue', edgecolor='black')
        plt.title(f'Histogram of log({column})', fontsize=16)
        plt.xlabel(f'log({column})', fontsize=14)
        plt.ylabel('Frequency', fontsize=14)
        plt.grid(True)
        plt.show()
```

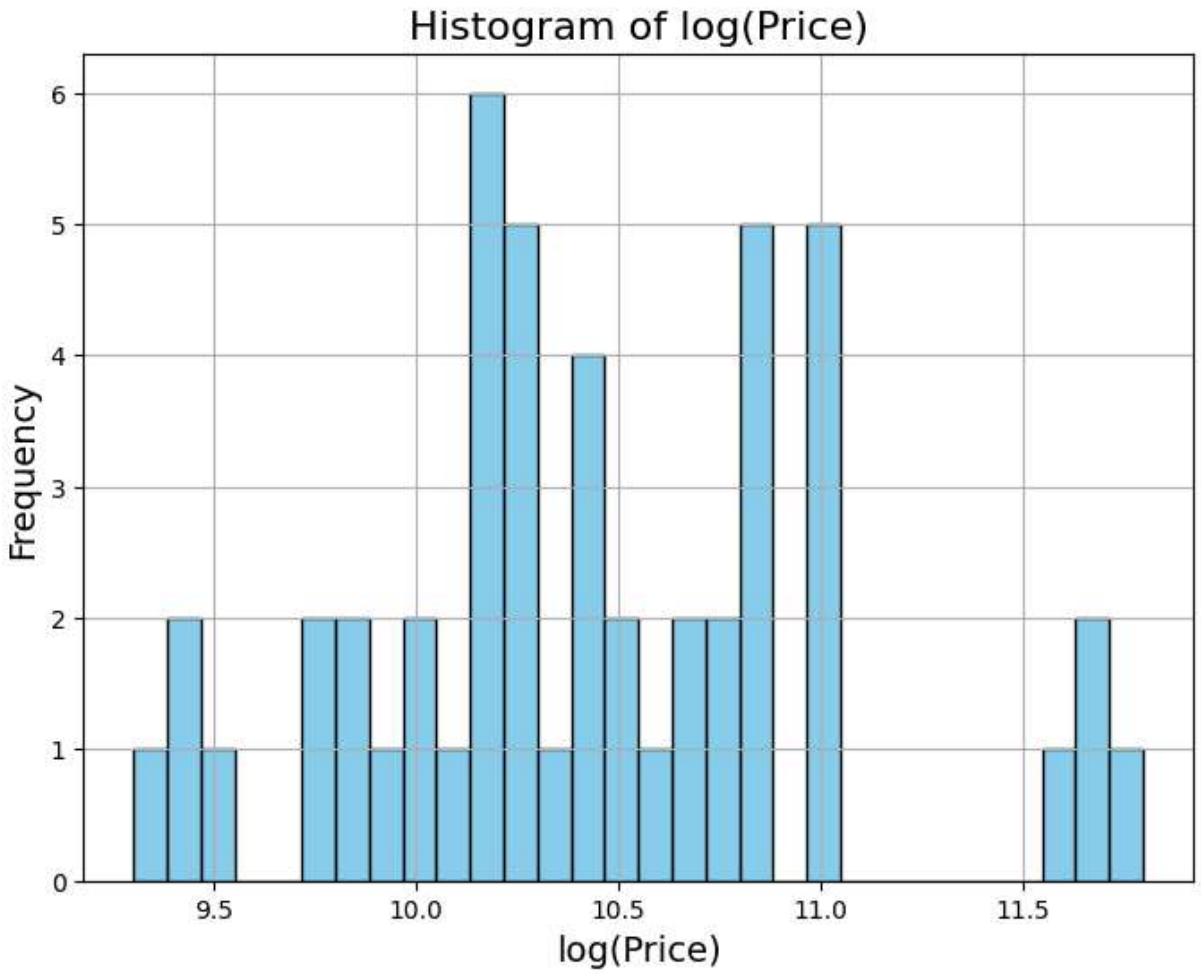

Histogram of log(CostCultivation_Product)
Frequency
log(CostCultivation_Product)
0
1
2
3
4
5
6
17
18
19
20
21
22


Histogram of log(Production)
Frequency
log(Production)
0
1
2
3
4
5
6
7
5
6
7
8

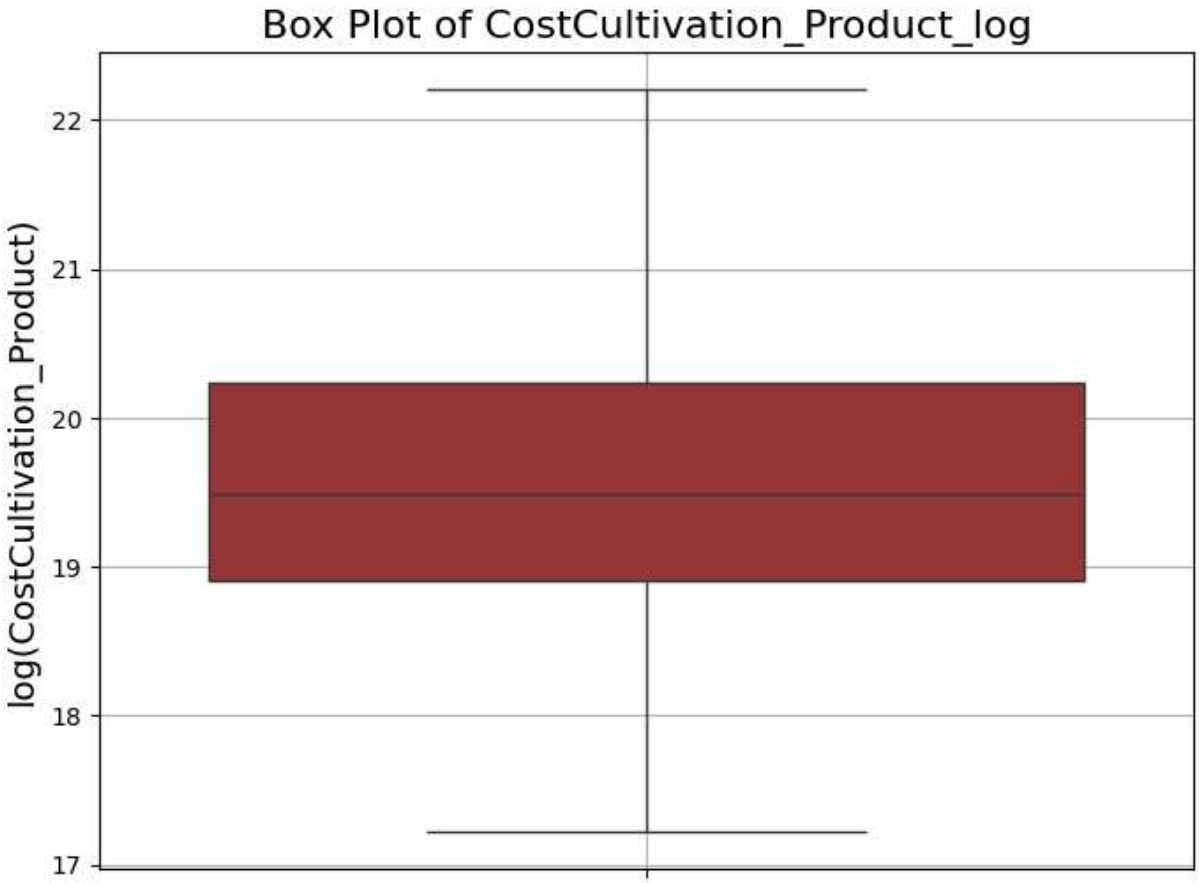

Histogram of log(Yield)
Frequency
log(Yield)
0
1
2
3
4
5
6
7

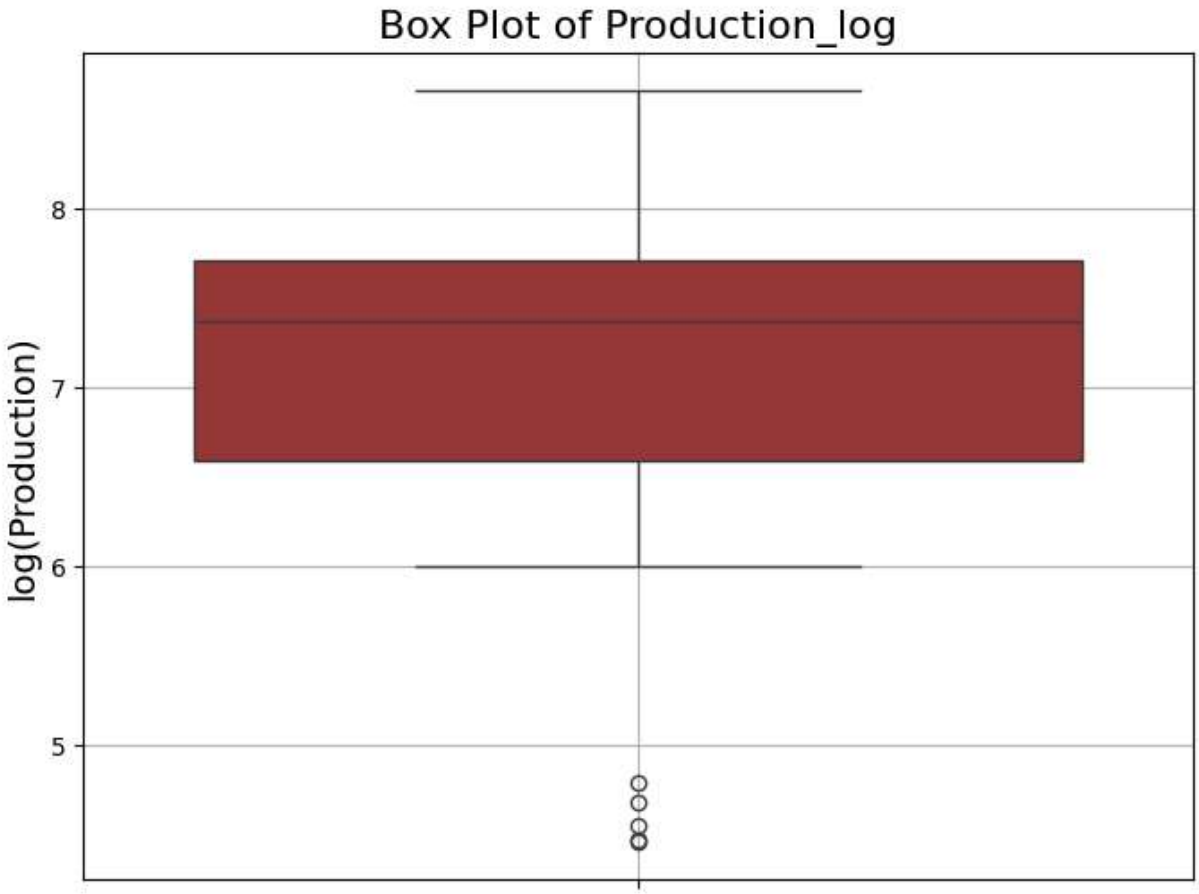

Histogram of log(Temperature)
Frequency
log(Temperature)
0
1
2
3
4
5
6
7
3.37
3.38
3.39
3.40
3.41

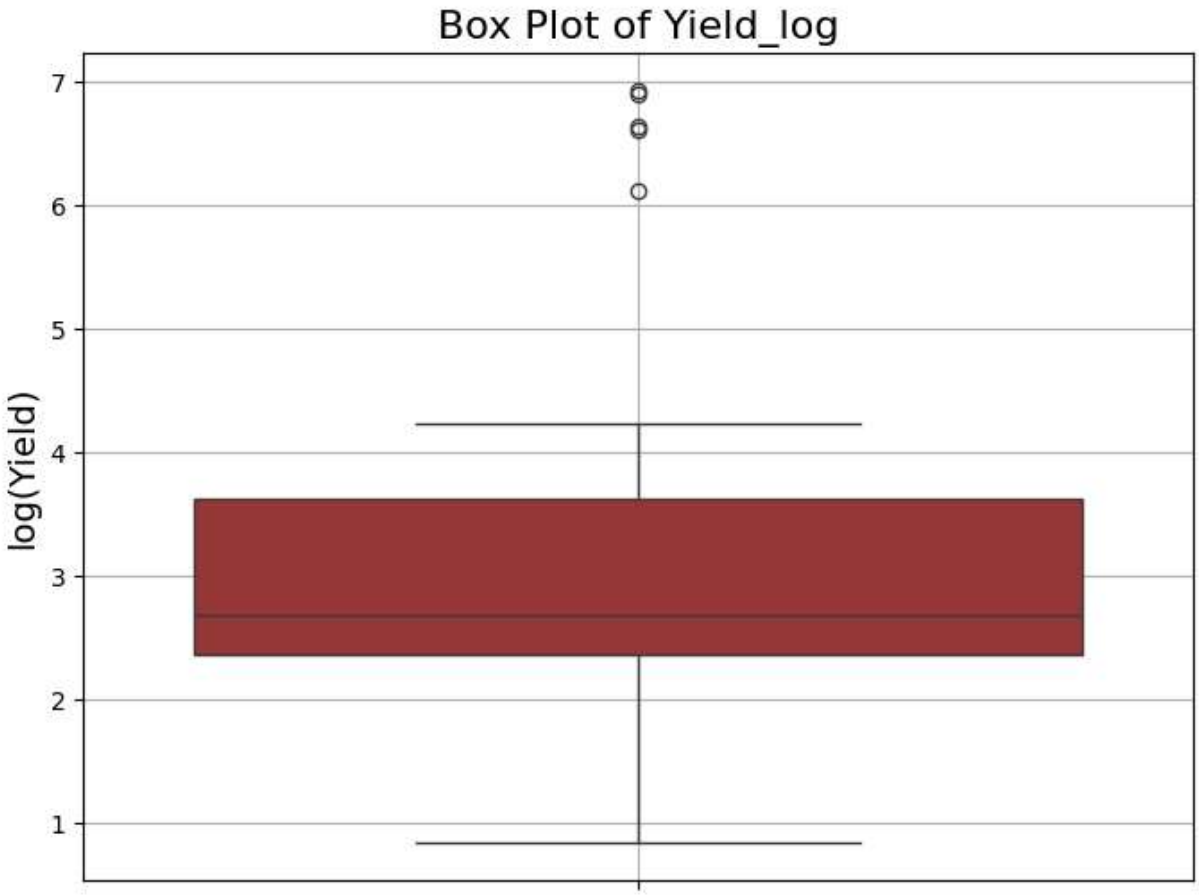

Histogram of log(RainFall Annual)
Frequency
log(RainFall Annual)
0
1
2
3
4
5
7.8
7.9
8.0
8.1
8.2

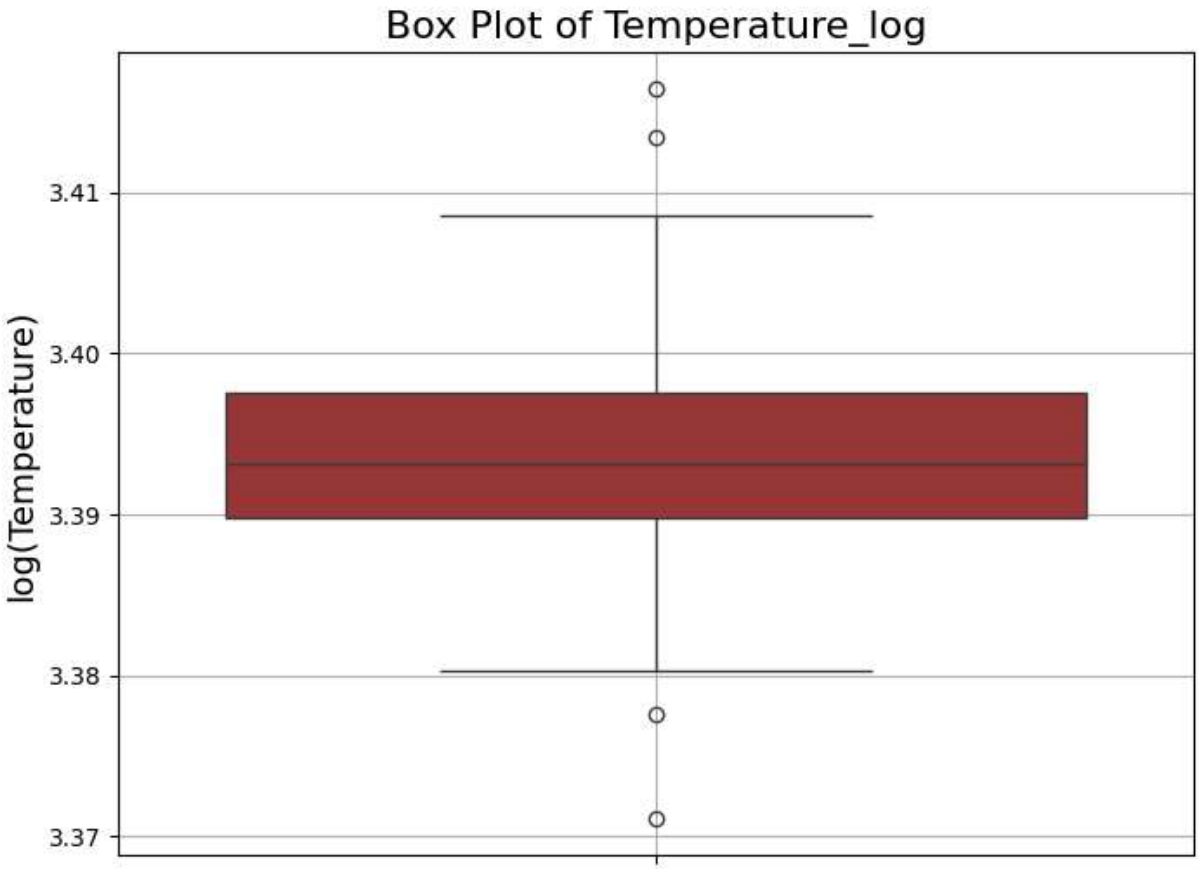


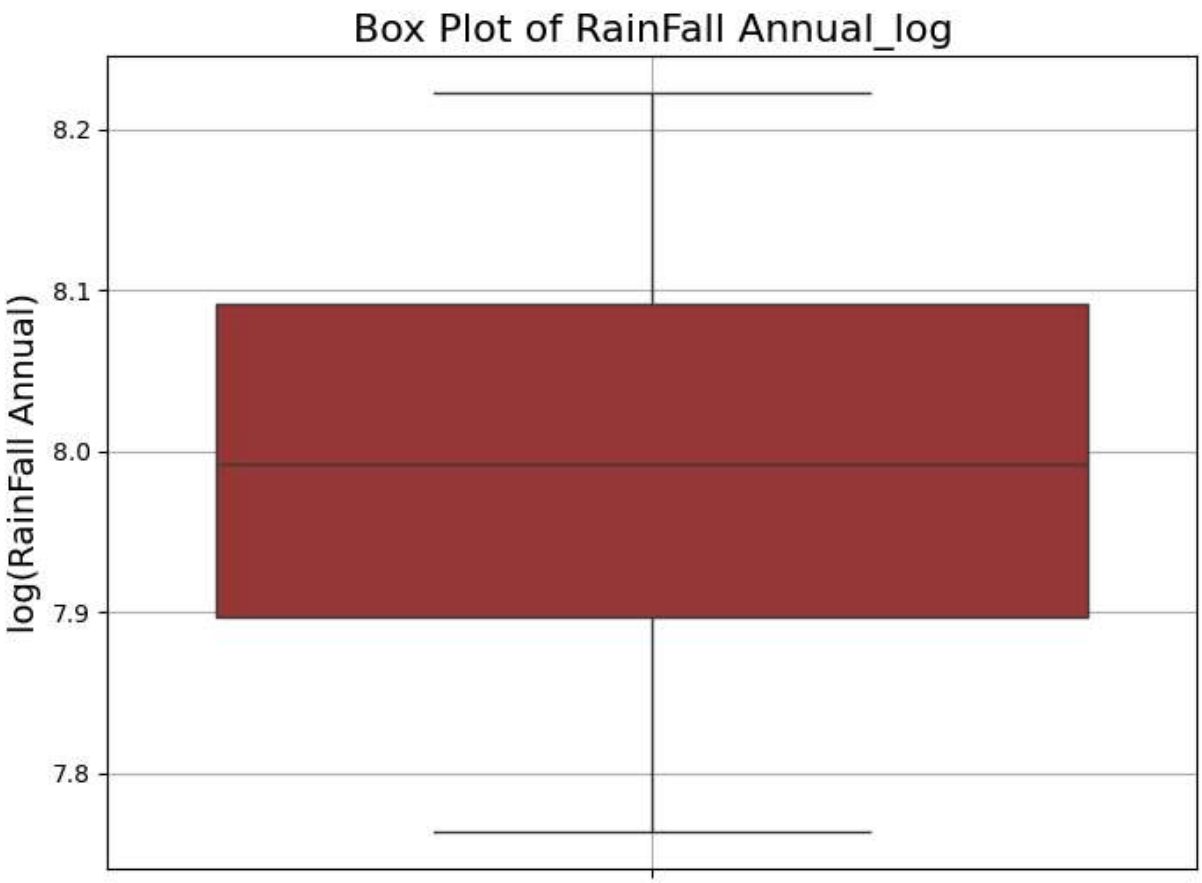
In [ ]:
```
for column in columns_of_interest:
    log_column = column + '_log'
    if log_column in df.columns:
        plt.figure(figsize=(8, 6))
        sns.boxplot(data=df, y=log_column, color='brown', orient='v')
        plt.title(f'Box Plot of {log_column}', fontsize=16)
        plt.ylabel(f'log({column})', fontsize=14)
        plt.grid(True)
        plt.show()
```


Box Plot of CostCultivation_Product_log
log(CostCultivation_Product)
22
21
20
19
18
17


Box Plot of Production_log
log(Production)
8
7
6
5


Box Plot of Yield_log
log(Yield)
7
6
5
4
3
2
1


Box Plot of Temperature_log
log(Temperature)
3.41
3.40
3.39
3.38
3.37


Box Plot of RainFall Annual_log
log(RainFall Annual)
8.2
8.1
8.0
7.9
7.8


Box Plot of Price_log
log(Price)
11.5
11.0
10.5
10.0
9.5

In [ ]:
```
df
```

Out[ ]:

	State	Crop	Production	Yield	Temperature	RainFall Annual	Price	Cost
0	Uttar Pradesh	ARHAR	1941.55	9.83	28.96	3373.2	19589.10	
1	Karnataka	ARHAR	2172.46	7.47	29.22	3520.7	21187.30	
2	Gujarat	ARHAR	1898.30	9.59	28.47	2957.4	26938.64	
3	Andhra Pradesh	ARHAR	3670.54	6.42	28.49	3079.6	34104.32	
4	Maharashtra	ARHAR	2775.80	8.72	28.30	2566.7	34262.10	
5	Maharashtra	COTTON	2539.47	12.69	28.73	2534.4	47423.88	
6	Punjab	COTTON	2003.76	24.39	28.65	3347.9	58095.20	
7	Andhra Pradesh	COTTON	2509.99	17.83	28.83	3576.4	58282.54	
8	Gujarat	COTTON	2179.26	19.05	28.38	2899.4	59233.18	
9	Haryana	COTTON	2127.35	19.90	28.53	2687.2	59838.94	
10	Rajasthan	GRAM	1691.66	6.83	28.62	2960.5	17106.38	
11	Madhya Pradesh	GRAM	1551.94	10.29	28.95	2365.8	19608.78	
12	Uttar Pradesh	GRAM	1882.68	10.93	28.67	2957.8	25667.08	
13	Maharashtra	GRAM	2277.68	8.05	28.66	2741.3	25972.90	
14	Andhra Pradesh	GRAM	1559.04	16.69	28.94	2937.5	28844.96	
15	Karnataka	GROUNDNUT	3484.01	4.71	28.82	2612.4	27295.20	
16	Andhra Pradesh	GROUNDNUT	2554.91	11.97	28.11	3275.0	42459.02	
17	Tamil Nadu	GROUNDNUT	2358.00	11.98	28.66	2352.1	45016.72	
18	Gujarat	GROUNDNUT	1918.92	13.45	28.66	2943.2	45903.56	
19	Maharashtra	GROUNDNUT	3207.35	9.33	28.76	2606.4	52158.32	
20	Bihar	MAIZE	404.43	42.95	28.86	3554.2	27028.84	
21	Karnataka	MAIZE	581.69	31.10	28.80	2357.7	27586.70	
22	Rajasthan	MAIZE	658.77	23.56	28.74	2442.9	28843.92	
23	Uttar Pradesh	MAIZE	1387.36	13.70	28.80	2480.5	31271.86	
24	Andhra Pradesh	MAIZE	840.58	42.68	28.67	3282.2	51375.18	

	State	Crop	Production	Yield	Temperature	RainFall Annual	Price	Cost
25	Orissa	MOONG	2614.14	3.01	28.70	2442.9	10968.08	
26	Rajasthan	MOONG	2068.67	4.05	28.59	2998.3	12409.46	
27	Karnataka	MOONG	5777.48	1.32	28.98	2926.6	12882.28	
28	Andhra Pradesh	MOONG	2228.97	5.90	28.76	3075.1	13369.36	
29	Maharashtra	MOONG	2261.24	6.70	28.65	2357.7	21562.52	
30	Uttar Pradesh	PADDY	732.62	36.61	29.15	3007.5	34045.00	
31	Orissa	PADDY	715.04	32.42	29.09	2987.5	34957.10	
32	West Bengal	PADDY	731.25	39.04	28.49	3722.8	49463.12	
33	Punjab	PADDY	669.86	67.41	29.03	3154.0	50310.50	
34	Andhra Pradesh	PADDY	789.90	56.00	28.76	2987.5	59330.68	
35	Madhya Pradesh	MUSTARD	1279.60	12.94	28.71	3591.1	17373.86	
36	Rajasthan	MUSTARD	1341.29	13.54	28.70	3264.4	22772.40	
37	Uttar Pradesh	MUSTARD	1595.56	13.57	28.70	2782.5	25549.82	
38	Gujarat	MUSTARD	1610.40	11.61	28.85	3007.5	27482.28	
39	Haryana	MUSTARD	1251.12	19.94	28.88	2898.2	29431.54	
40	Uttar Pradesh	SUGARCANE	93.64	448.89	29.46	2782.5	49077.64	
41	Karnataka	SUGARCANE	86.53	986.21	28.98	2791.6	111311.88	
42	Andhra Pradesh	SUGARCANE	119.72	757.92	28.80	3007.5	113243.32	
43	Maharashtra	SUGARCANE	107.56	744.01	28.89	3489.6	115348.20	
44	Tamil Nadu	SUGARCANE	85.79	1015.45	28.97	2422.2	132671.12	
45	Madhya Pradesh	WHEAT	810.25	23.59	29.37	3275.1	24929.80	
46	Punjab	WHEAT	804.80	39.83	28.84	3079.9	35892.16	
47	Uttar Pradesh	WHEAT	769.84	34.99	28.73	2721.9	37959.76	
48	Rajasthan	WHEAT	683.58	37.19	28.89	3449.0	38239.16	

In [ ]:
```
df = df.drop(columns=columns_of_interest)
```

In [ ]:
```
df
```

Out[ ]:

	State	Crop	CostCultivation_Product_log	Production_log	Yield_log	Tem
0	Uttar Pradesh	ARHAR	18.379694	7.571757	2.382320	
1	Karnataka	ARHAR	18.536020	7.684076	2.136531	
2	Gujarat	ARHAR	19.016340	7.549241	2.359910	
3	Andhra Pradesh	ARHAR	19.488064	8.208366	2.004179	
4	Maharashtra	ARHAR	19.497296	7.929054	2.274186	
5	Maharashtra	COTTON	20.147468	7.840104	2.616666	
6	Punjab	COTTON	20.553382	7.603280	3.234355	
7	Andhra Pradesh	COTTON	20.559821	7.828432	2.935451	
8	Gujarat	COTTON	20.592180	7.687199	2.998229	
9	Haryana	COTTON	20.612529	7.663102	3.039749	
10	Rajasthan	GRAM	18.108119	7.434057	2.057963	
11	Madhya Pradesh	GRAM	18.381171	7.347905	2.423917	
12	Uttar Pradesh	GRAM	18.919635	7.540983	2.479056	
13	Maharashtra	GRAM	18.943324	7.731352	2.202765	
14	Andhra Pradesh	GRAM	19.153087	7.352467	2.873000	
15	Karnataka	GROUNDNUT	19.042638	8.156226	1.742219	
16	Andhra Pradesh	GROUNDNUT	19.926295	7.846164	2.562639	
17	Tamil Nadu	GROUNDNUT	20.043284	7.765993	2.563410	
18	Gujarat	GROUNDNUT	20.082302	7.560039	2.670694	
19	Maharashtra	GROUNDNUT	20.337784	8.073512	2.335052	
20	Bihar	MAIZE	19.023025	6.004948	3.783053	
21	Karnataka	MAIZE	19.063884	6.367655	3.468856	
22	Rajasthan	MAIZE	19.153015	6.491891	3.201119	
23	Uttar Pradesh	MAIZE	19.314654	7.235878	2.687847	
24	Andhra Pradesh	MAIZE	20.307527	6.735281	3.776890	

	State	Crop	CostCultivation_Product_log	Production_log	Yield_log	Tem
25	Orissa	MOONG	17.219195	7.869073	1.388791	
26	Rajasthan	MOONG	17.466134	7.635144	1.619388	
27	Karnataka	MOONG	17.540922	8.661896	0.841567	
28	Andhra Pradesh	MOONG	17.615147	7.709743	1.931521	
29	Maharashtra	MOONG	18.571129	7.724111	2.041220	
30	Uttar Pradesh	PADDY	19.484583	6.597991	3.627270	
31	Orissa	PADDY	19.537459	6.573736	3.509155	
32	West Bengal	PADDY	20.231671	6.596122	3.689879	
33	Punjab	PADDY	20.265644	6.508560	4.225519	
34	Andhra Pradesh	PADDY	20.595469	6.673172	4.043051	
35	Madhya Pradesh	MUSTARD	18.139150	7.155084	2.634762	
36	Rajasthan	MUSTARD	18.680315	7.202132	2.676903	
37	Uttar Pradesh	MUSTARD	18.910477	7.375607	2.678965	
38	Gujarat	MUSTARD	19.056299	7.384859	2.534490	
39	Haryana	MUSTARD	19.193350	7.132593	3.041661	
40	Uttar Pradesh	SUGARCANE	20.216023	4.550080	6.109003	
41	Karnataka	SUGARCANE	21.853888	4.471982	6.894883	
42	Andhra Pradesh	SUGARCANE	21.888294	4.793474	6.631896	
43	Maharashtra	SUGARCANE	21.925127	4.687303	6.613398	
44	Tamil Nadu	SUGARCANE	22.204963	4.463491	6.924071	
45	Madhya Pradesh	WHEAT	18.861344	6.698576	3.202340	
46	Punjab	WHEAT	19.590254	6.691836	3.709417	
47	Uttar Pradesh	WHEAT	19.702270	6.647481	3.583241	
48	Rajasthan	WHEAT	19.716936	6.528806	3.642574	

In [ ]:
```
df = pd.get_dummies(df, columns=['State', 'Crop'])
```

In [ ]:
```
df
```

Out[ ]:

	CostCultivation_Product_log	Production_log	Yield_log	Temperature_log	RainFall Annual_log	I
0	18.379694	7.571757	2.382320	3.399863	8.123914	
1	18.536020	7.684076	2.136531	3.408504	8.166699	
2	19.016340	7.549241	2.359910	3.383373	7.992404	1
3	19.488064	8.208366	2.004179	3.384051	8.032880	1
4	19.497296	7.929054	2.274186	3.377588	7.850766	1
5	20.147468	7.840104	2.616666	3.392157	7.838107	1
6	20.553382	7.603280	3.234355	3.389462	8.116387	1
7	20.559821	7.828432	2.935451	3.395515	8.182392	1
8	20.592180	7.687199	2.998229	3.380314	7.972604	1
9	20.612529	7.663102	3.039749	3.385407	7.896627	1
10	18.108119	7.434057	2.057963	3.388450	7.993451	
11	18.381171	7.347905	2.423917	3.399529	7.769294	
12	18.919635	7.540983	2.479056	3.390136	7.992539	1
13	18.943324	7.731352	2.202765	3.389799	7.916552	1
14	19.153087	7.352467	2.873000	3.399195	7.985655	1
15	19.042638	8.156226	1.742219	3.395179	7.868407	1
16	19.926295	7.846164	2.562639	3.371082	8.094378	1
17	20.043284	7.765993	2.563410	3.389799	7.763489	1
18	20.082302	7.560039	2.670694	3.389799	7.987592	1
19	20.337784	8.073512	2.335052	3.393165	7.866109	1
20	19.023025	6.004948	3.783053	3.396520	8.176167	1
21	19.063884	6.367655	3.468856	3.394508	7.765866	1
22	19.153015	6.491891	3.201119	3.392493	7.801350	1
23	19.314654	7.235878	2.687847	3.394508	7.816618	1
24	20.307527	6.735281	3.776890	3.390136	8.096574	1
25	17.219195	7.869073	1.388791	3.391147	7.801350	
26	17.466134	7.635144	1.619388	3.387436	8.006134	
27	17.540922	8.661896	0.841567	3.400530	7.981938	
28	17.615147	7.709743	1.931521	3.393165	8.031418	
29	18.571129	7.724111	2.041220	3.389462	7.765866	

	CostCultivation_Product_log	Production_log	Yield_log	Temperature_log	RainFall Annual_log	I
30	19.484583	6.597991	3.627270	3.406185	8.009197	1
31	19.537459	6.573736	3.509155	3.404193	8.002527	1
32	20.231671	6.596122	3.689879	3.384051	8.222500	1
33	20.265644	6.508560	4.225519	3.402197	8.056744	1
34	20.595469	6.673172	4.043051	3.393165	8.002527	1
35	18.139150	7.155084	2.634762	3.391484	8.186492	
36	18.680315	7.202132	2.676903	3.391147	8.091138	1
37	18.910477	7.375607	2.678965	3.391147	7.931464	1
38	19.056299	7.384859	2.534490	3.396185	8.009197	1
39	19.193350	7.132593	3.041661	3.397189	7.972190	1
40	20.216023	4.550080	6.109003	3.416414	7.931464	1
41	21.853888	4.471982	6.894883	3.400530	7.934728	1
42	21.888294	4.793474	6.631896	3.394508	8.009197	1
43	21.925127	4.687303	6.613398	3.397524	8.157829	1
44	22.204963	4.463491	6.924071	3.400197	7.792844	1
45	18.861344	6.698576	3.202340	3.413455	8.094409	1
46	19.590254	6.691836	3.709417	3.395850	8.032977	1
47	19.702270	6.647481	3.583241	3.392157	7.909453	1
48	19.716936	6.528806	3.642574	3.397524	8.146130	1

49 rows × 29 columns

Linear Regression

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [ ]:
```
X = df.drop(columns=['Price_log'])
y = df['Price_log']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [ ]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[ ]:
```
LinearRegression
LinearRegression()
```

In [ ]:
```
y_pred = model.predict(X_test)
```

In [ ]:
```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 8.930501346563236e-10
```

In [ ]:
```
from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_pred)
print("R^2 Score:", r2)
```

```
R^2 Score: 0.9999999983198382
```

Support Vector Regression

In [ ]:
```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
```

In [ ]:
```
svr_model = SVR()
svr_model.fit(X_train, y_train)
```

Out[ ]:
```
SVR
SVR()
```

In [ ]:
```
y_pred_svr = svr_model.predict(X_test)
```

In [ ]:
```
mse_svr = mean_squared_error(y_test, y_pred_svr)
print("Mean Squared Error:", mse_svr)
```

```
Mean Squared Error: 0.1872095175013046
```

In [ ]:
```
svr_r2 = r2_score(y_test, y_pred)
print("R^2 Score:", svr_r2)
```

```
R^2 Score: 0.9999999983198382
```

Decision Tree Regression

In [ ]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [ ]:
```
tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)
```

Out[ ]:
```
DecisionTreeRegressor
DecisionTreeRegressor()
```

In [ ]:
```
y_pred_tree = tree_model.predict(X_test)
```

In [ ]:
```
mse_tree = mean_squared_error(y_test, y_pred_tree)
print("Mean Squared Error:", mse_tree)
```

```
Mean Squared Error: 0.00974855795805352
```

In [ ]:
```
r2_tree = r2_score(y_test, y_pred_tree)
print("R^2 Score:", r2_tree)
```

```
R^2 Score: 0.9816593109967473
```