Importing the Libraries

In [1]:
```
# for manipulations
import numpy as np
import pandas as pd

# for data visualizations
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')

# for interactivity
import ipywidgets
from ipywidgets import interact
```

```
C:\Users\rbgir\anaconda3\lib\site-packages\numpy\_distributor_init.py:30: Us
erWarning: loaded more than 1 DLL from .libs:
C:\Users\rbgir\anaconda3\lib\site-packages\numpy\.libs\libopenblas.WCDJNK7YV
MPZQ2ME2ZZHJJRJ3JIKNDB7.gfortran-win_amd64.dll
C:\Users\rbgir\anaconda3\lib\site-packages\numpy\.libs\libopenblas64__v0.3.2
1-gcc_10_3_0.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
C:\Users\rbgir\anaconda3\lib\site-packages\scipy\__init__.py:138: UserWarnin
g: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciP
y (detected version 1.24.4)
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion} is
required for this version of "
```

Reading the Dataset

In [2]:
```
# lets read the dataset
data = pd.read_csv('Crop_recommendation.csv')

# lets check teh shape of the dataset
print("Shape of the Dataset :", data.shape)
```

```
Shape of the Dataset : (2200, 8)
```

In [3]:
```
# lets check the head of the dataset
data.head()
```

Out[3]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

```
## Description for each of the columns in the Dataset

N - ratio of Nitrogen content in soil
P - ratio of Phosphorous content in soil
K - ration of Potassium content in soil
temperature - temperature in degree Celsius
humidity - relative humidity in %
ph - ph value of the soil
rainfall - rainfall in mm
```

In [4]:
```
# lets check if there is any missing value present in the dataset
data.isnull().sum()
```

Out[4]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [5]:
```
# lets check the Crops present in this Dataset
data['label'].value_counts()
```

Out[5]:
```
label
rice           100
maize          100
jute           100
cotton         100
coconut        100
papaya         100
orange         100
apple          100
muskmelon      100
watermelon     100
grapes         100
mango          100
banana         100
pomegranate    100
lentil         100
blackgram      100
mungbean       100
mothbeans      100
pigeonpeas     100
kidneybeans    100
chickpea       100
coffee         100
Name: count, dtype: int64
```

Descriptive Statistics

In [6]:
```
# lets check the Summary for all the crops

print("Average Ratio of Nitrogen in the Soil : {0:.2f}".format(data['N'].mean(
print("Average Ratio of Phosphorous in the Soil : {0:.2f}".format(data['P'].me
print("Average Ratio of Potassium in the Soil : {0:.2f}".format(data['K'].mear
print("Average Tempature in Celsius : {0:.2f}".format(data['temperature'].mear
print("Average Relative Humidity in % : {0:.2f}".format(data['humidity'].mean(
print("Average PH Value of the soil : {0:.2f}".format(data['ph'].mean()))
print("Average Rainfall in mm : {0:.2f}".format(data['rainfall'].mean()))
```

```
Average Ratio of Nitrogen in the Soil : 50.55
Average Ratio of Phosphorous in the Soil : 53.36
Average Ratio of Potassium in the Soil : 48.15
Average Tempature in Celsius : 25.62
Average Relative Humidity in % : 71.48
Average PH Value of the soil : 6.47
Average Rainfall in mm : 103.46
```

In [7]:

```python
# lets check the Summary Statistics for each of the Crops

@interact
def summary(crops = list(data['label'].value_counts().index)):
    x = data[data['label'] == crops]
    print("---------------------------------------------")
    print("Statistics for Nitrogen")
    print("Minimum Nitrigen required :", x['N'].min())
    print("Average Nitrogen required :", x['N'].mean())
    print("Maximum Nitrogen required :", x['N'].max())
    print("---------------------------------------------")
    print("Statistics for Phosphorous")
    print("Minimum Phosphorous required :", x['P'].min())
    print("Average Phosphorous required :", x['P'].mean())
    print("Maximum Phosphorous required :", x['P'].max())
    print("---------------------------------------------")
    print("Statistics for Potassium")
    print("Minimum Potassium required :", x['K'].min())
    print("Average Potassium required :", x['K'].mean())
    print("Maximum Potassium required :", x['K'].max())
    print("---------------------------------------------")
    print("Statistics for Temperature")
    print("Minimum Temperature required : {0:.2f}".format(x['temperature'].min
    print("Average Temperature required : {0:.2f}".format(x['temperature'].mea
    print("Maximum Temperature required : {0:.2f}".format(x['temperature'].max
    print("---------------------------------------------")
    print("Statistics for Humidity")
    print("Minimum Humidity required : {0:.2f}".format(x['humidity'].min()))
    print("Average Humidity required : {0:.2f}".format(x['humidity'].mean()))
    print("Maximum Humidity required : {0:.2f}".format(x['humidity'].max()))
    print("---------------------------------------------")
    print("Statistics for PH")
    print("Minimum PH required : {0:.2f}".format(x['ph'].min()))
    print("Average PH required : {0:.2f}".format(x['ph'].mean()))
    print("Maximum PH required : {0:.2f}".format(x['ph'].max()))
    print("---------------------------------------------")
    print("Statistics for Rainfall")
    print("Minimum Rainfall required : {0:.2f}".format(x['rainfall'].min()))
    print("Average Rainfall required : {0:.2f}".format(x['rainfall'].mean()))
    print("Maximum Rainfall required : {0:.2f}".format(x['rainfall'].max()))
```

crops maize

```
---------------------------------------------
Statistics for Nitrogen
Minimum Nitrigen required : 60
Average Nitrogen required : 77.76
Maximum Nitrogen required : 100
---------------------------------------------
Statistics for Phosphorous
Minimum Phosphorous required : 35
Average Phosphorous required : 48.44
Maximum Phosphorous required : 60
---------------------------------------------
Statistics for Potassium
Minimum Potassium required : 15
Average Potassium required : 19.79
Maximum Potassium required : 25
---------------------------------------------
Statistics for Temperature
Minimum Temperature required : 18.04
Average Temperature required : 22.39
Maximum Temperature required : 26.55
---------------------------------------------
Statistics for Humidity
Minimum Humidity required : 55.28
Average Humidity required : 65.09
Maximum Humidity required : 74.83
---------------------------------------------
Statistics for PH
Minimum PH required : 5.51
Average PH required : 6.25
Maximum PH required : 7.00
---------------------------------------------
Statistics for Rainfall
Minimum Rainfall required : 60.65
Average Rainfall required : 84.77
Maximum Rainfall required : 109.75
```

In [8]:
```python
## Lets compare the Average Requirement for each crops with average conditions

@interact
def compare(conditions = ['N','P','K','temperature','ph','humidity','rainfall'
    print("Average Value for", conditions,"is {0:.2f}".format(data[conditions]
    print("---------------------------------------------")
    print("Rice : {0:.2f}".format(data[(data['label'] == 'rice')][conditions].
    print("Black Grams : {0:.2f}".format(data[data['label'] == 'blackgram'][cc
    print("Banana : {0:.2f}".format(data[(data['label'] == 'banana')][conditic
    print("Jute : {0:.2f}".format(data[data['label'] == 'jute'][conditions].me
    print("Coconut : {0:.2f}".format(data[(data['label'] == 'coconut')][condit
    print("Apple : {0:.2f}".format(data[data['label'] == 'apple'][conditions].
    print("Papaya : {0:.2f}".format(data[(data['label'] == 'papaya')][conditic
    print("Muskmelon : {0:.2f}".format(data[data['label'] == 'muskmelon'][conc
    print("Grapes : {0:.2f}".format(data[(data['label'] == 'grapes')][conditic
    print("Watermelon : {0:.2f}".format(data[data['label'] == 'watermelon'][cc
    print("Kidney Beans: {0:.2f}".format(data[(data['label'] == 'kidneybeans')
    print("Mung Beans : {0:.2f}".format(data[data['label'] == 'mungbean'][conc
    print("Oranges : {0:.2f}".format(data[(data['label'] == 'orange')][conditi
    print("Chick Peas : {0:.2f}".format(data[data['label'] == 'chickpea'][conc
    print("Lentils : {0:.2f}".format(data[(data['label'] == 'lentil')][conditi
    print("Cotton : {0:.2f}".format(data[data['label'] == 'cotton'][conditions
    print("Maize : {0:.2f}".format(data[(data['label'] == 'maize')][conditions
    print("Moth Beans : {0:.2f}".format(data[data['label'] == 'mothbeans'][cor
    print("Pigeon Peas : {0:.2f}".format(data[(data['label'] == 'pigeonpeas')]
    print("Mango : {0:.2f}".format(data[data['label'] == 'mango'][conditions].
    print("Pomegranate : {0:.2f}".format(data[(data['label'] == 'pomegranate')
    print("Coffee : {0:.2f}".format(data[data['label'] == 'coffee'][conditions
```

conditions N

```
Average Value for N is 50.55
---------------------------------------------
Rice : 79.89
Black Grams : 40.02
Banana : 100.23
Jute : 78.40
Coconut : 21.98
Apple : 20.80
Papaya : 49.88
Muskmelon : 100.32
Grapes : 23.18
Watermelon : 99.42
Kidney Beans: 20.75
Mung Beans : 20.99
Oranges : 19.58
Chick Peas : 40.09
Lentils : 18.77
Cotton : 117.77
Maize : 77.76
Moth Beans : 21.44
Pigeon Peas : 20.73
Mango : 20.07
Pomegranate : 18.87
Coffee : 101.20
```

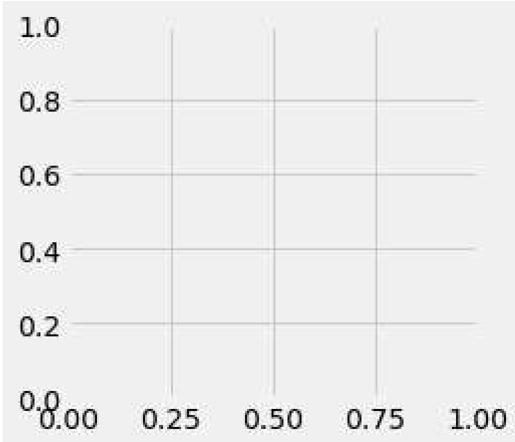
In [9]:
```python
# lets make this funtion more Intuitive

@interact
def compare(conditions = ['N','P','K','temperature','ph','humidity','rainfall'
    print("Crops which require greater than average", conditions,'\n')
    print(data[data[conditions] > data[conditions].mean()]['label'].unique())
    print("---------------------------------------------")
    print("Crops which require less than average", conditions,'\n')
    print(data[data[conditions] <= data[conditions].mean()]['label'].unique())
```

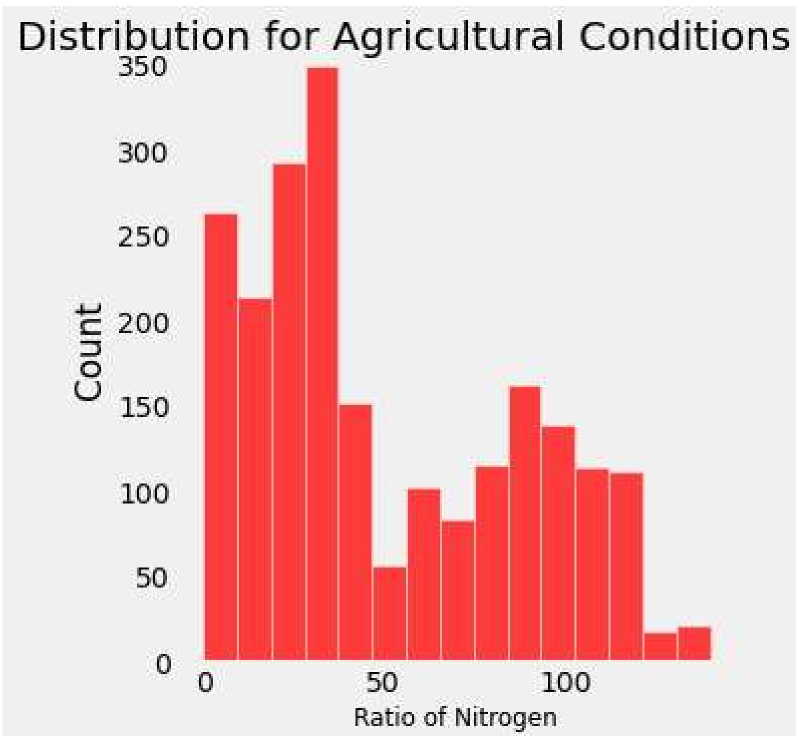
conditions N

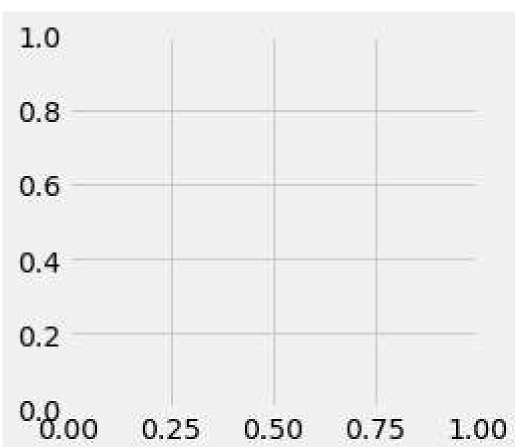
```
Crops which require greater than average N

['rice' 'maize' 'chickpea' 'blackgram' 'banana' 'watermelon' 'muskmelon'
 'papaya' 'cotton' 'jute' 'coffee']
---------------------------------------------
Crops which require less than average N

['chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean' 'blackgram'
 'lentil' 'pomegranate' 'mango' 'grapes' 'apple' 'orange' 'papaya'
 'coconut']
```

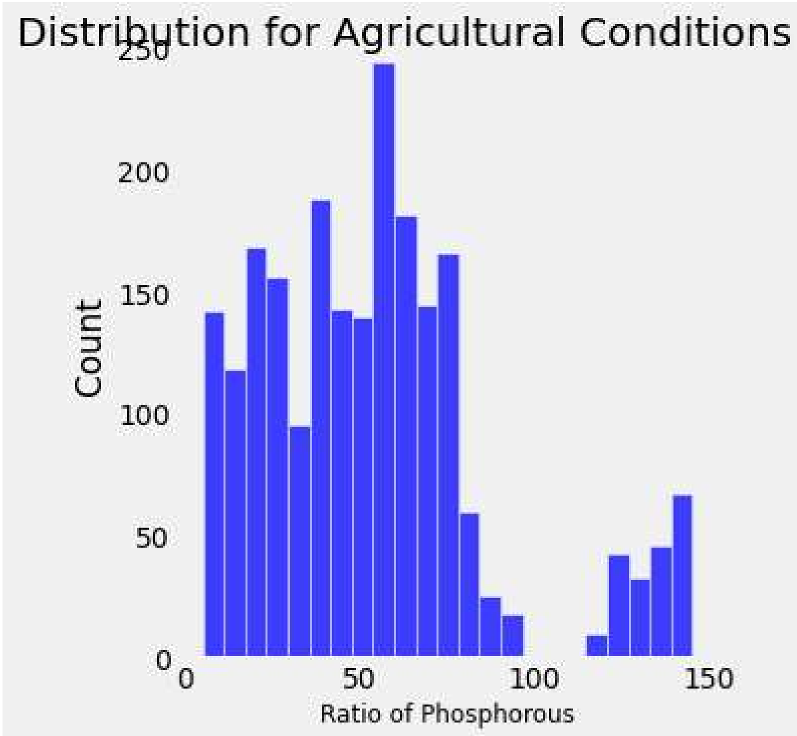
Analyzing Agricultural Conditions

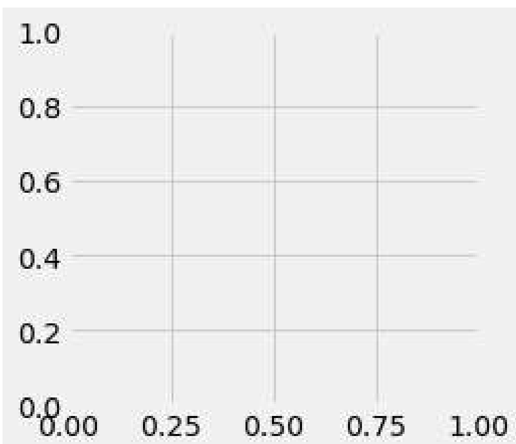
In [10]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (15, 7)

plt.subplot(2, 4, 1)
sns.displot(data['N'].values, color='red')  # Convert to numpy array with .val
plt.xlabel('Ratio of Nitrogen', fontsize=12)
plt.grid()
plt.suptitle('Distribution for Agricultural Conditions', fontsize=20)
plt.show()


plt.subplot(2, 4, 2)
sns.displot(data['P'].values, color='blue')
plt.xlabel('Ratio of Phosphorous', fontsize=12)
plt.grid()
plt.suptitle('Distribution for Agricultural Conditions', fontsize=20)
plt.show()

plt.subplot(2, 4, 3)
sns.displot(data['K'].values, color='darkblue')
plt.xlabel('Ratio of Potassium', fontsize=12)
plt.grid()
plt.suptitle('Distribution for Agricultural Conditions', fontsize=20)
plt.show()

plt.subplot(2, 4, 4)
sns.displot(data['temperature'].values, color='black')
plt.xlabel('Temperature', fontsize=12)
plt.grid()
plt.suptitle('Distribution for Agricultural Conditions', fontsize=20)
plt.show()

plt.subplot(2, 4, 5)
sns.displot(data['rainfall'].values, color='purple')
plt.xlabel('Rainfall', fontsize=12)
plt.grid()
plt.suptitle('Distribution for Agricultural Conditions', fontsize=20)
plt.show()

plt.subplot(2, 4, 6)
sns.displot(data['humidity'].values, color='lightgreen')
plt.xlabel('Humidity', fontsize=12)
plt.grid()
plt.suptitle('Distribution for Agricultural Conditions', fontsize=20)
plt.show()

plt.subplot(2, 4, 7)
sns.displot(data['ph'].values, color='darkgreen')
plt.xlabel('pH Level', fontsize=12)
plt.grid()

plt.suptitle('Distribution for Agricultural Conditions', fontsize=20)
plt.show()
```

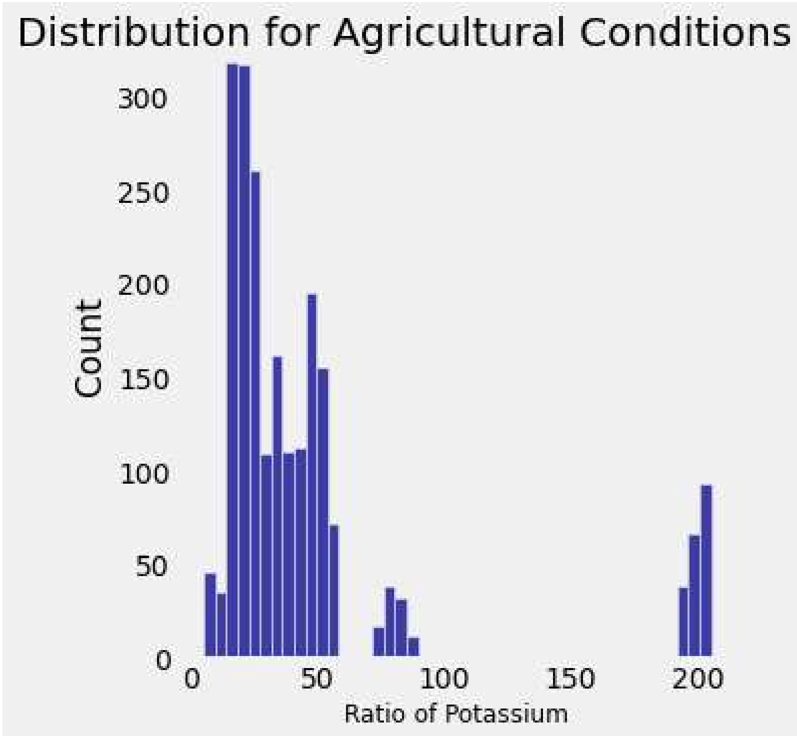

1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00

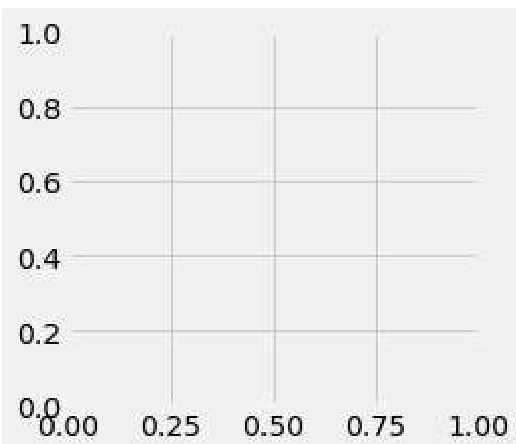

Distribution for Agricultural Conditions
Count
350
300
250
200
150
100
50
0
0
50
100
Ratio of Nitrogen

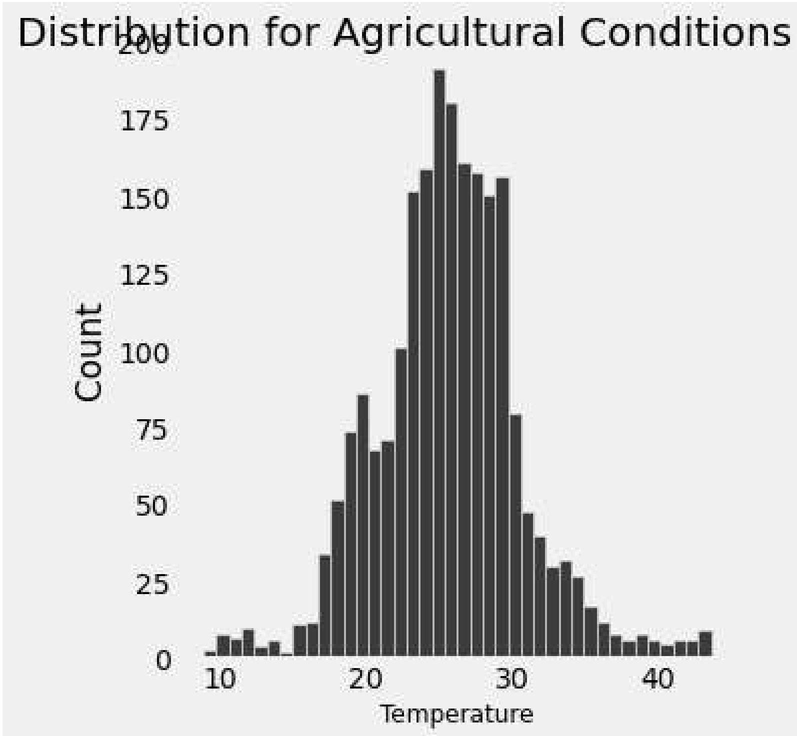

1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00

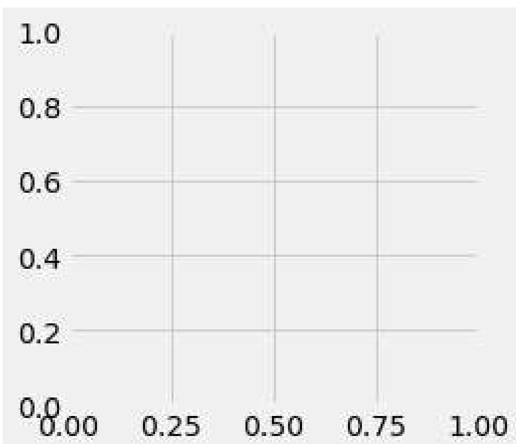

Distribution for Agricultural Conditions
Count
Ratio of Phosphorous
250
200
150
100
50
0
0
50
100
150

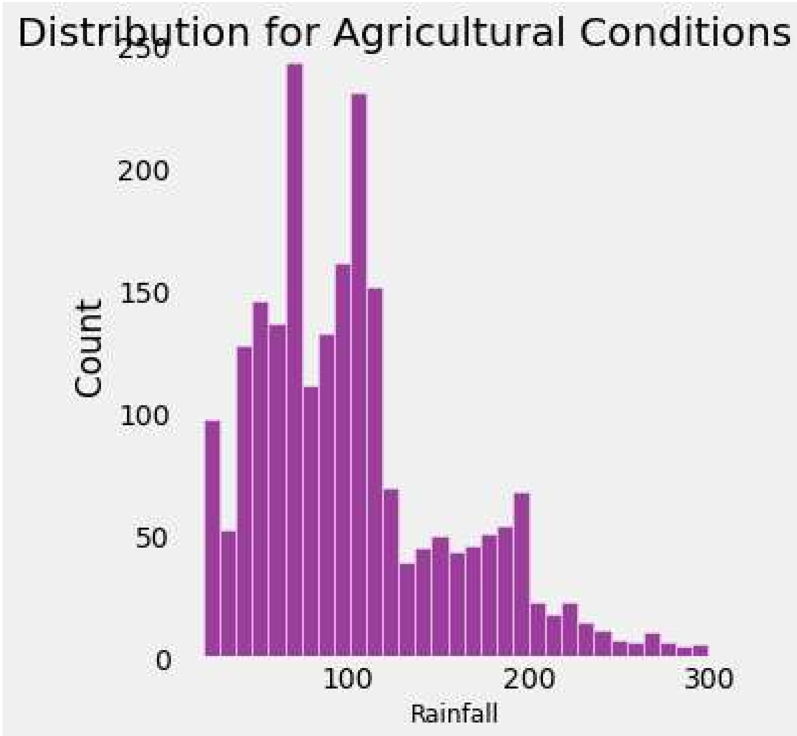

1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00

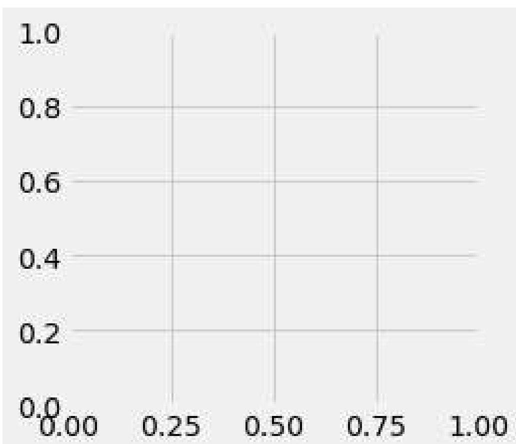

Distribution for Agricultural Conditions
Count
Ratio of Potassium
0
50
100
150
200
250
300

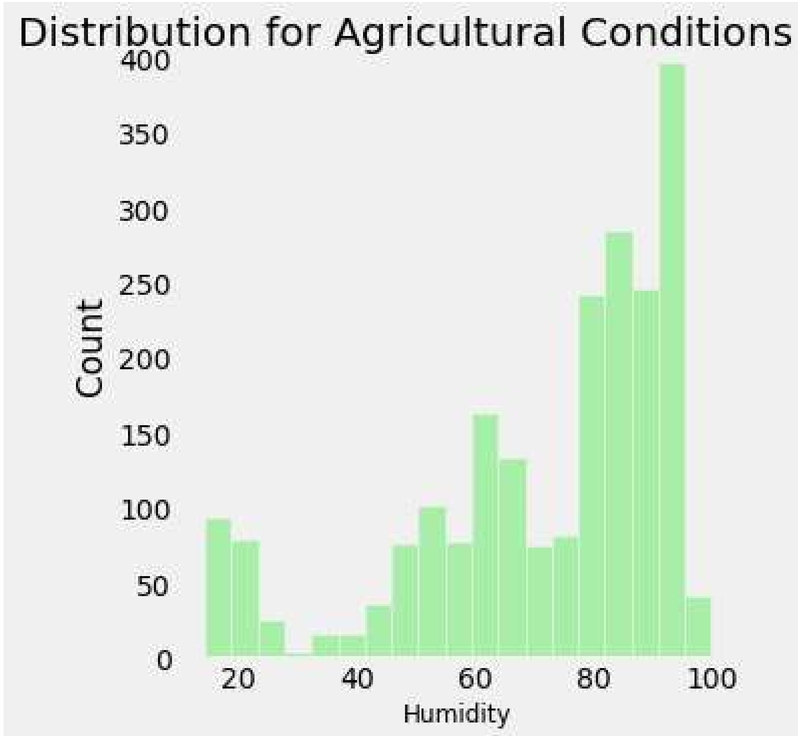

0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00


Distribution for Agricultural Conditions
Count
Temperature
0
25
50
75
100
125
150
175
200
10
20
30
40

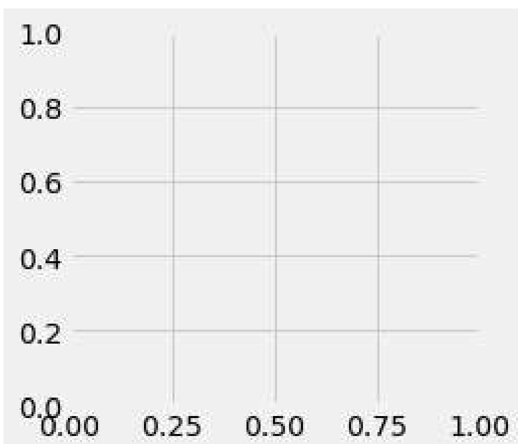

0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00


Distribution for Agricultural Conditions
Count
0
50
100
150
200
250
100
200
300
Rainfall

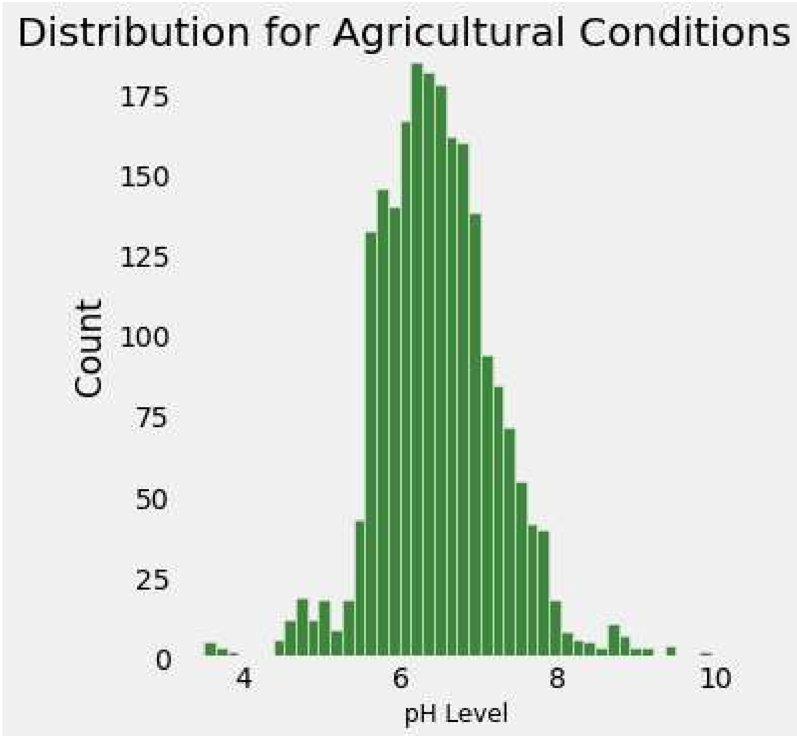

0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00



Distribution for Agricultural Conditions
Count
Humidity
0
50
100
150
200
250
300
350
400
20
40
60
80
100



0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00



In [11]:
```
## Lets find out some Interesting Facts

print("Some Interesting Patterns")
print("---------------------------------")
print("Crops which requires very High Ratio of Nitrogen Content in Soil:", dat
print("Crops which requires very High Ratio of Phosphorous Content in Soil:",
print("Crops which requires very High Ratio of Potassium Content in Soil:", da
print("Crops which requires very High Rainfall:", data[data['rainfall'] > 200]
print("Crops which requires very Low Temperature :", data[data['temperature']
print("Crops which requires very High Temperature :", data[data['temperature']
print("Crops which requires very Low Humidity:", data[data['humidity'] < 20]['
print("Crops which requires very Low pH:", data[data['ph'] < 4]['label'].uniqu
print("Crops which requires very High pH:", data[data['ph'] > 9]['label'].unic
```

```
Some Interesting Patterns
---------------------------------
Crops which requires very High Ratio of Nitrogen Content in Soil: ['cotton']
Crops which requires very High Ratio of Phosphorous Content in Soil: ['grape
s' 'apple']
Crops which requires very High Ratio of Potassium Content in Soil: ['grapes'
'apple']
Crops which requires very High Rainfall: ['rice' 'papaya' 'coconut']
Crops which requires very Low Temperature : ['grapes']
Crops which requires very High Temperature : ['grapes' 'papaya']
Crops which requires very Low Humidity: ['chickpea' 'kidneybeans']
Crops which requires very Low pH: ['mothbeans']
Crops which requires very High pH: ['mothbeans']
```

In [12]:
```
### Lets understand which crops can only be Grown in Summer Season, Winter Sea

print("Summer Crops")
print(data[(data['temperature'] > 30) & (data['humidity'] > 50)]['label'].unic
print("-----------------------------------")
print("Winter Crops")
print(data[(data['temperature'] < 20) & (data['humidity'] > 30)]['label'].unic
print("-----------------------------------")
print("Rainy Crops")
print(data[(data['rainfall'] > 200) & (data['humidity'] > 30)]['label'].unique
```

```
Summer Crops
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
-----------------------------------
Winter Crops
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']
-----------------------------------
Rainy Crops
['rice' 'papaya' 'coconut']
```

Clustering Similar Crops

In [13]:
```
### Lets try to Cluster these Crops

# lets import the warnings library so that we can avoid warnings
import warnings
warnings.filterwarnings('ignore')

# Lets select the Spending score, and Annual Income Columns from the Data
x = data.loc[:, ['N','P','K','temperature','ph','humidity','rainfall']].values

# let's check the shape of x
print(x.shape)

# lets convert this data into a dataframe
x_data  = pd.DataFrame(x)
x_data.head()
```

```
(2200, 7)
```

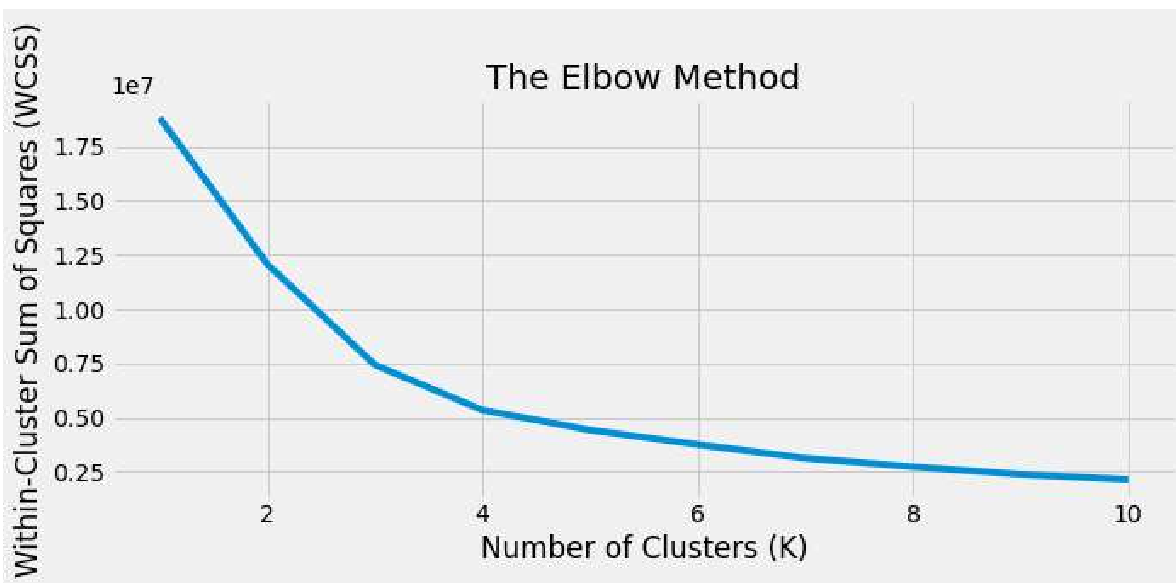
Out[13]:

	0	1	2	3	4	5	6
0	90.0	42.0	43.0	20.879744	6.502985	82.002744	202.935536
1	85.0	58.0	41.0	21.770462	7.038096	80.319644	226.655537
2	60.0	55.0	44.0	23.004459	7.840207	82.320763	263.964248
3	74.0	35.0	40.0	26.491096	6.980401	80.158363	242.864034
4	78.0	42.0	42.0	20.130175	7.628473	81.604873	262.717340

In [14]:
```
!pip install --upgrade threadpoolctl
```

```
Requirement already satisfied: threadpoolctl in c:\users\rbgir\anaconda3\lib
\site-packages (3.2.0)
```

In [16]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (10, 4)

wcss = []  # List to store the within-cluster sum of squares (WCSS) for differ

for i in range(1, 11):
    # Create a KMeans instance with i clusters, 'k-means++' initialization, an
    km = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, rando

    # Fit the KMeans model to the data and calculate the WCSS
    km.fit(x)  # Assuming 'x' is your dataset
    wcss.append(km.inertia_)
# Plot the results
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method', fontsize=20)
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.show()
```



In [17]:

```
# lets implement the K Means algorithm to perform Clustering analysis
km = KMeans(n_clusters = 4, init = 'k-means++', max_iter = 300, n_init = 10, r
y_means = km.fit_predict(x)

# lets find out the Results
a = data['label']
y_means = pd.DataFrame(y_means)
z = pd.concat([y_means, a], axis = 1)
z = z.rename(columns = {0: 'cluster'})

# lets check the Clusters of each Crops
print("Lets check the Results After Applying the K Means Clustering Analysis \
print("Crops in First Cluster:", z[z['cluster'] == 0]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Second Cluster:", z[z['cluster'] == 1]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Third Cluster:", z[z['cluster'] == 2]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Forth Cluster:", z[z['cluster'] == 3]['label'].unique())
```

```
Lets check the Results After Applying the K Means Clustering Analysis

Crops in First Cluster: ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'moth
beans' 'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'mango' 'orange' 'papaya' 'coconut']
---------------------------------------------------------------
Crops in Second Cluster: ['maize' 'banana' 'watermelon' 'muskmelon' 'papaya'
'cotton' 'coffee']
---------------------------------------------------------------
Crops in Third Cluster: ['grapes' 'apple']
---------------------------------------------------------------
Crops in Forth Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coff
ee']
```

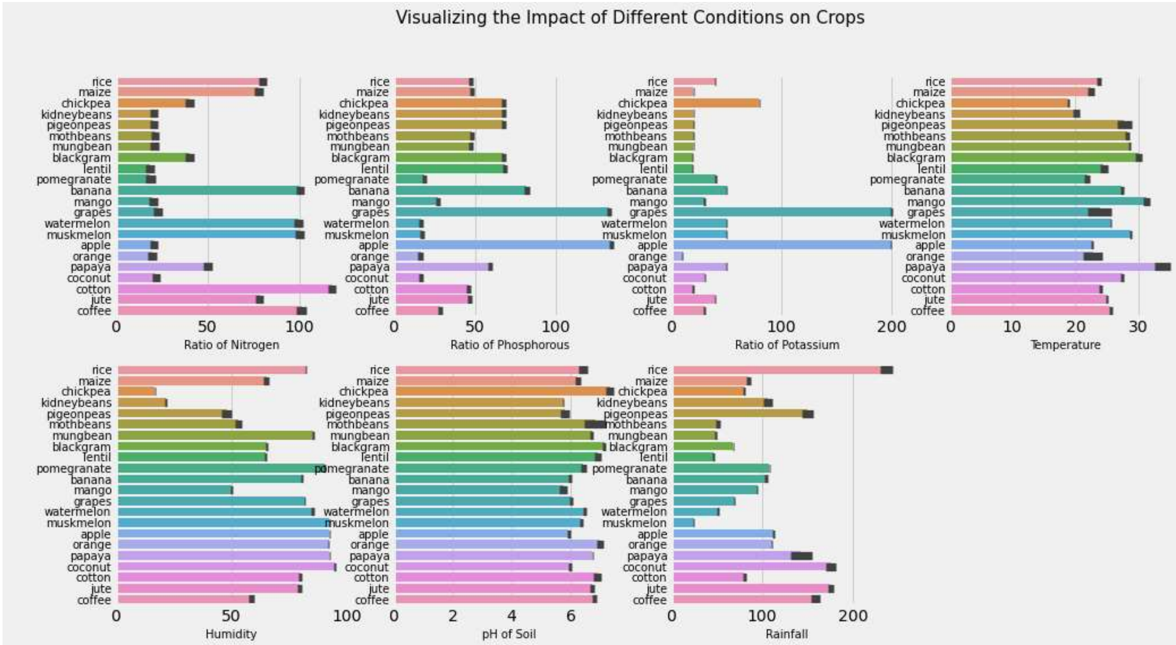
In [18]:

```python
# Hard Clustering

print("Results for Hard Clustering\n")
counts = z[z['cluster'] == 0]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
print("Crops in Cluster 1:", list(d.index))
print("---------------------------------------------------")
counts = z[z['cluster'] == 1]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
print("Crops in Cluster 2:", list(d.index))
print("---------------------------------------------------")
counts = z[z['cluster'] == 2]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
print("Crops in Cluster 3:", list(d.index))
print("---------------------------------------------------")
counts = z[z['cluster'] == 3]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
print("Crops in Cluster 4:", list(d.index))
```

```
Results for Hard Clustering

Crops in Cluster 1: ['chickpea', 'kidneybeans', 'mothbeans', 'mungbean', 'bl
ackgram', 'lentil', 'pomegranate', 'mango', 'orange']
---------------------------------------------------
Crops in Cluster 2: ['maize', 'banana', 'watermelon', 'muskmelon', 'cotton']
---------------------------------------------------
Crops in Cluster 3: ['grapes', 'apple']
---------------------------------------------------
Crops in Cluster 4: ['rice', 'pigeonpeas', 'papaya', 'coconut', 'jute', 'cof
fee']
```

visualizing the Hidden Patterns

In [19]:
```
### Data Visualizations

plt.rcParams['figure.figsize'] = (15, 8)

plt.subplot(2, 4, 1)
sns.barplot(data['N'], data['label'])
plt.ylabel(' ')
plt.xlabel('Ratio of Nitrogen', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 2)
sns.barplot(data['P'], data['label'])
plt.ylabel(' ')
plt.xlabel('Ratio of Phosphorous', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 3)
sns.barplot(data['K'], data['label'])
plt.ylabel(' ')
plt.xlabel('Ratio of Potassium', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 4)
sns.barplot(data['temperature'], data['label'])
plt.ylabel(' ')
plt.xlabel('Temperature', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 5)
sns.barplot(data['humidity'], data['label'])
plt.ylabel(' ')
plt.xlabel('Humidity', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 6)
sns.barplot(data['ph'], data['label'])
plt.ylabel(' ')
plt.xlabel('pH of Soil', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 7)
sns.barplot(data['rainfall'], data['label'])
plt.ylabel(' ')
plt.xlabel('Rainfall', fontsize = 10)
plt.yticks(fontsize = 10)

plt.suptitle('Visualizing the Impact of Different Conditions on Crops', fontsi
plt.show()
```



Predictive Modelling

In [21]:
```
# lets split the Dataset for Predictive Modelling

y = data['label']
x = data.drop(['label'], axis = 1)

print("Shape of x:", x.shape)
print("Shape of y:", y.shape)
```

```
Shape of x: (2200, 7)
Shape of y: (2200,)
```

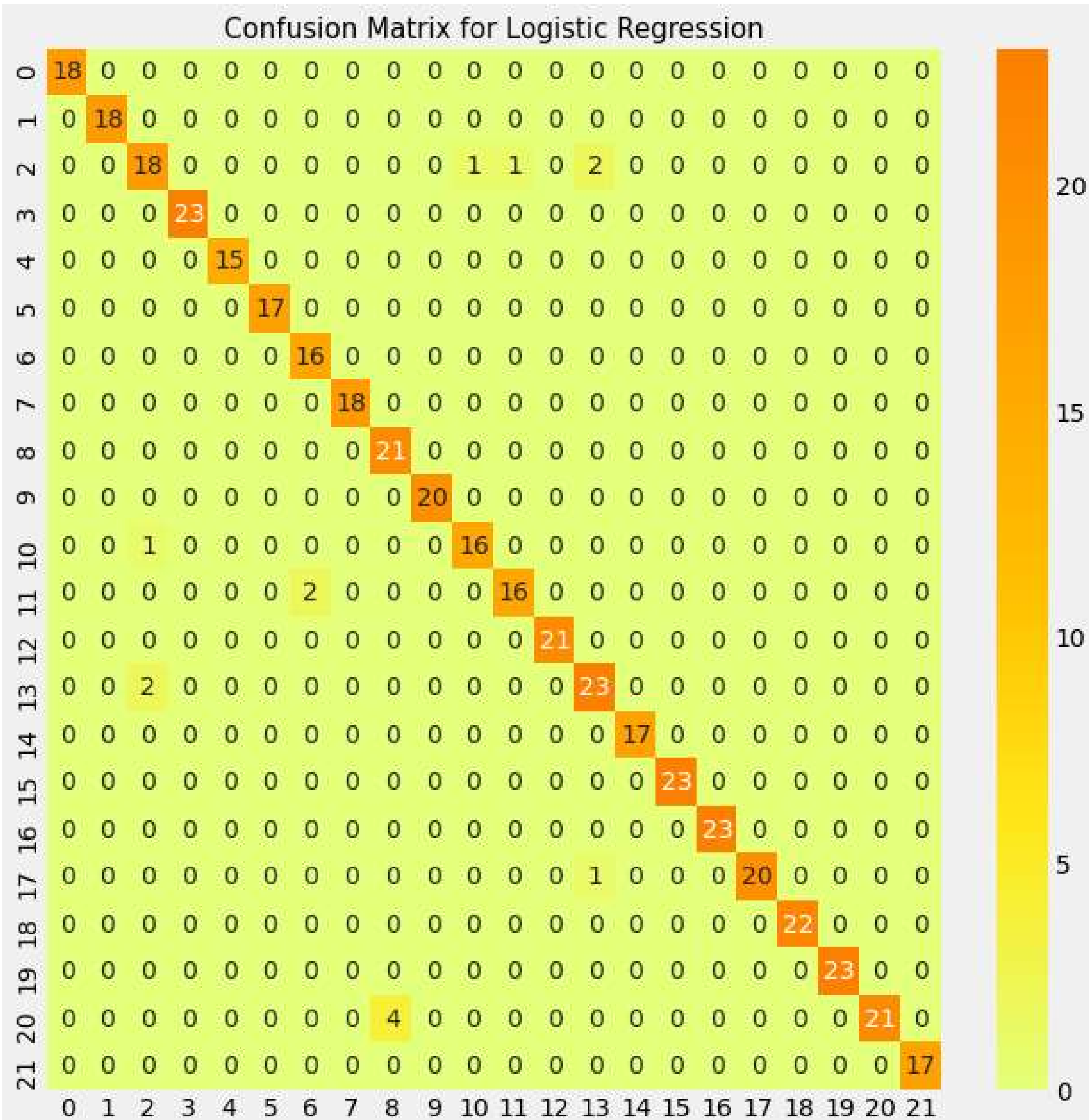
In [22]:
```
# lets create Training and Testing Sets for Validation of Results
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, rar

print("The Shape of x train:", x_train.shape)
print("The Shape of x test:", x_test.shape)
print("The Shape of y train:", y_train.shape)
print("The Shape of y test:", y_test.shape)
```

```
The Shape of x train: (1760, 7)
The Shape of x test: (440, 7)
The Shape of y train: (1760,)
The Shape of y test: (440,)
```

In [23]:

```
# lets create a Predictive Model

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

In [24]:
```
# lets evaluate the Model Performance
from sklearn.metrics import classification_report, confusion_matrix

# lets print the Confusion matrix first
plt.rcParams['figure.figsize'] = (10, 10)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot = True, cmap = 'Wistia')
plt.title('Confusion Matrix for Logistic Regression', fontsize = 15)
plt.show()

# lets print the Classification Report also
cr = classification_report(y_test, y_pred)
print(cr)
```



```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        18
      banana       1.00      1.00      1.00        18
   blackgram       0.86      0.82      0.84        22
    chickpea       1.00      1.00      1.00        23
     coconut       1.00      1.00      1.00        15
      coffee       1.00      1.00      1.00        17
      cotton       0.89      1.00      0.94        16
      grapes       1.00      1.00      1.00        18
        jute       0.84      1.00      0.91        21
 kidneybeans       1.00      1.00      1.00        20
      lentil       0.94      0.94      0.94        17
       maize       0.94      0.89      0.91        18
       mango       1.00      1.00      1.00        21
   mothbeans       0.88      0.92      0.90        25
    mungbean       1.00      1.00      1.00        17
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        23
      papaya       1.00      0.95      0.98        21
  pigeonpeas       1.00      1.00      1.00        22
 pomegranate       1.00      1.00      1.00        23
        rice       1.00      0.84      0.91        25
  watermelon       1.00      1.00      1.00        17

    accuracy                           0.97       440
   macro avg       0.97      0.97      0.97       440
weighted avg       0.97      0.97      0.97       440
```

Calculate Accuracy for Logistic Regression in this model:

In [25]:
```
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression  # Assuming you're using L

# Create and train your model
model = LogisticRegression()  # Replace with your actual model initialization
model.fit(x_train, y_train)

# Make predictions on the test set
y_pred = model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.97
```

It has 97 % Accuracy in Crop prediction .

Real time Predictions

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

In [ ]:
```
prediction = model.predict((np.array([[90,40,40,20,80,7,200]])))

print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

In [ ]:
```
# lets check the Model for Oranges also
data[data['label'] == 'orange'].head()
```

In [ ]:
```
# lets do some Real time Predictions
prediction = model.predict((np.array([[20,30,10,15,90,7.5,100]])))
print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

In [ ]:
```
prediction = model.predict((np.array([[2,20,29,20,70,6,300]])))

print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

In [ ]:
```
prediction = model.predict((np.array([[5,31,60,40,35,9,112]])))

print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

In [ ]:
```

```