In [1]:
```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import os
```

Reading dataset

In [2]:
```
df = pd.read_csv("C:/Users/Rog Strix/Desktop/Green Belt/fertilizer predictor/Fertilize
```

In [3]:
```
df.head()
```

Out[3]:

	Temparature	Humidity	Moisture	Soil Type	Crop Type	Nitrogen	Potassium	Phosphorous	Fertilizer Name
0	26	52	38	Sandy	Maize	37	0	0	Urea
1	29	52	45	Loamy	Sugarcane	12	0	36	DAP
2	34	65	62	Black	Cotton	7	9	30	14-35-14
3	32	62	34	Red	Tobacco	22	0	20	28-28
4	28	54	46	Clayey	Paddy	35	0	0	Urea

In [4]:
```
df.describe()
```

Out[4]:

	Temparature	Humidity	Moisture	Nitrogen	Potassium	Phosphorous
count	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000
mean	30.282828	59.151515	43.181818	18.909091	3.383838	18.606061
std	3.502304	5.840331	11.271568	11.599693	5.814667	13.476978
min	25.000000	50.000000	25.000000	4.000000	0.000000	0.000000
25%	28.000000	54.000000	34.000000	10.000000	0.000000	9.000000
50%	30.000000	60.000000	41.000000	13.000000	0.000000	19.000000
75%	33.000000	64.000000	50.500000	24.000000	7.500000	30.000000
max	38.000000	72.000000	65.000000	42.000000	19.000000	42.000000

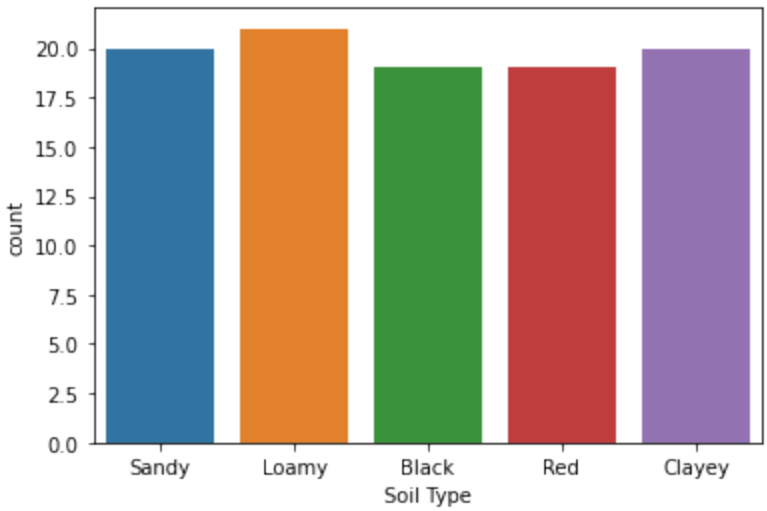
In [5]:
```
df['Soil Type'].unique()
```

Out[5]:
```
array(['Sandy', 'Loamy', 'Black', 'Red', 'Clayey'], dtype=object)
```

Visualizing Data

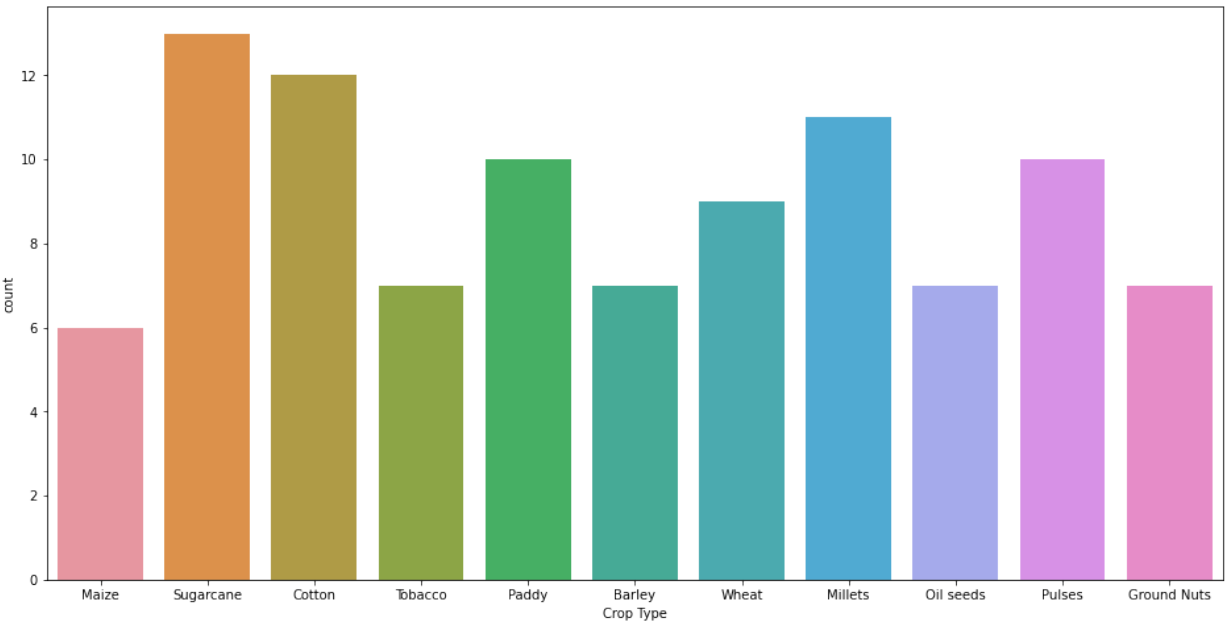
In [6]:
```
import seaborn as sns
sns.countplot(x='Soil Type', data = df)
```

Out[6]:
```
<AxesSubplot:xlabel='Soil Type', ylabel='count'>
```



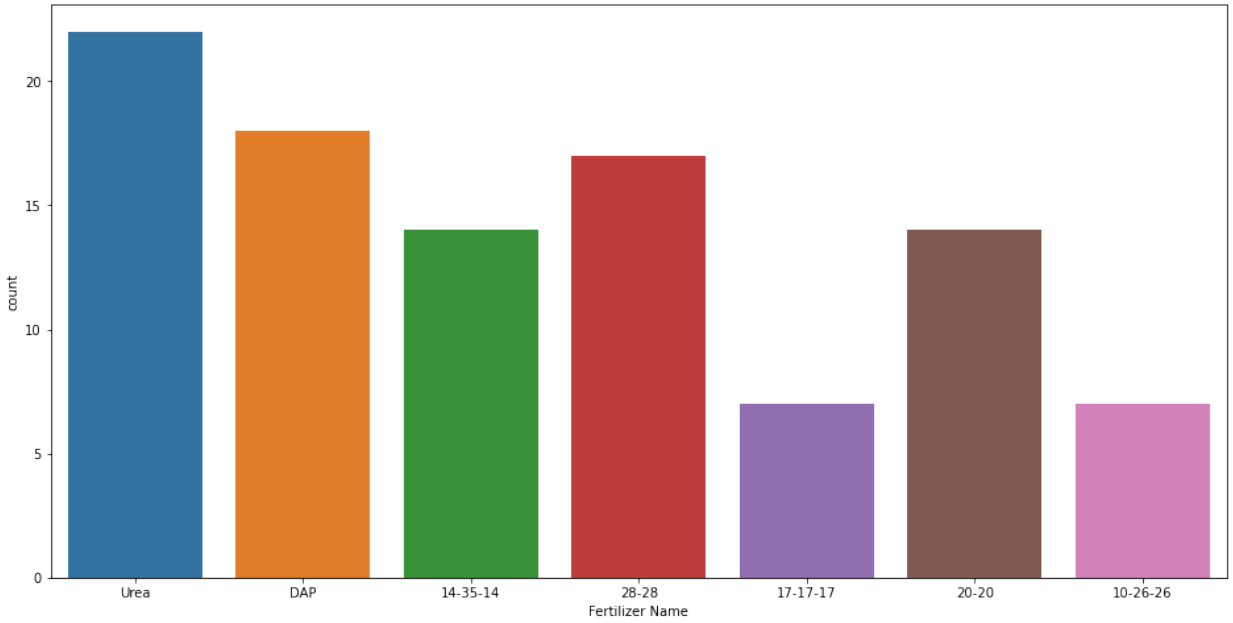
In [7]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
sns.countplot(x='Crop Type', data = df)
```

Out[7]:
```
<AxesSubplot:xlabel='Crop Type', ylabel='count'>
```



In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
sns.countplot(x='Fertilizer Name', data = df)
```

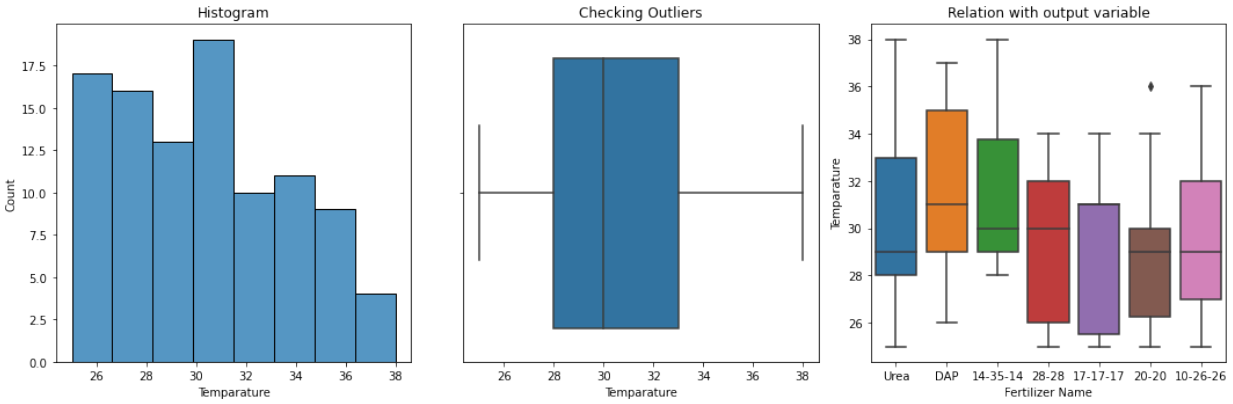
Out[8]:
```
<AxesSubplot:xlabel='Fertilizer Name', ylabel='count'>
```



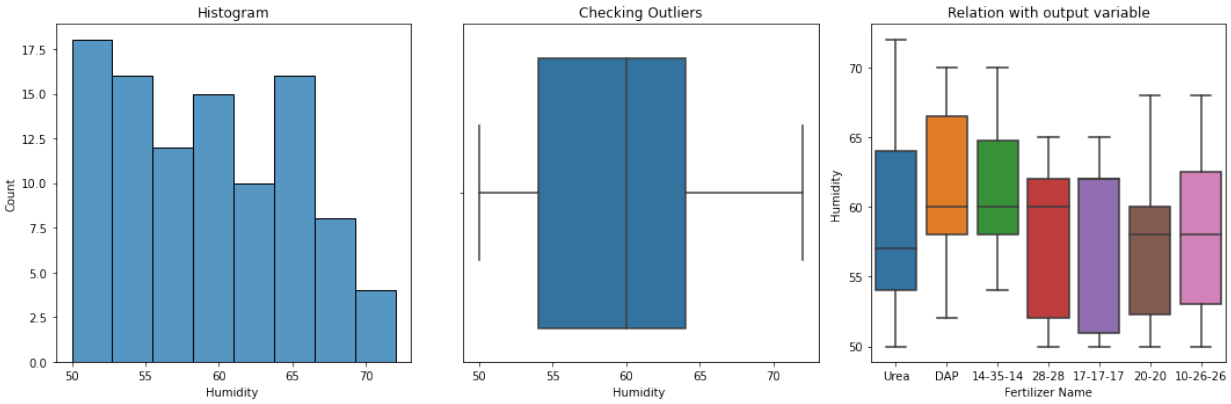
In [9]:
```
#Defining function for Continuous and catogorical variable
def plot_conti(x):
    fig, axes = plt.subplots(nrows=1,ncols=3,figsize=(15,5),tight_layout=True)
    axes[0].set_title('Histogram')
    sns.histplot(x,ax=axes[0])
    axes[1].set_title('Checking Outliers')
    sns.boxplot(x,ax=axes[1])
    axes[2].set_title('Relation with output variable')
    sns.boxplot(y = x,x = df['Fertilizer Name'])

def plot_cato(x):
    fig, axes = plt.subplots(nrows=1,ncols=2,figsize=(15,5),tight_layout=True)
    axes[0].set_title('Count Plot')
    sns.countplot(x,ax=axes[0])
    axes[1].set_title('Relation with output variable')
    sns.countplot(x = x,hue = df['Fertilizer Name'], ax=axes[1])
```

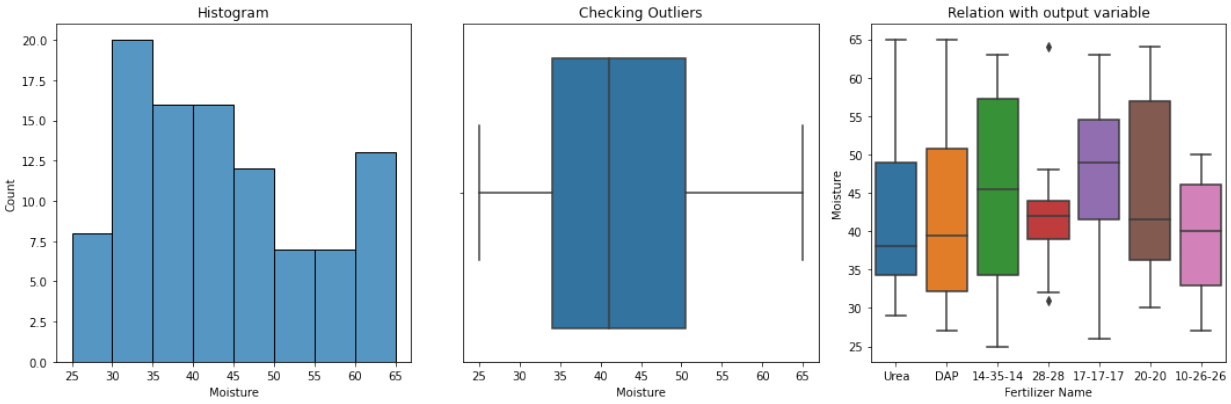
In [10]:
```
#EDA - Temparature variable
plot_conti(df['Temparature'])
```



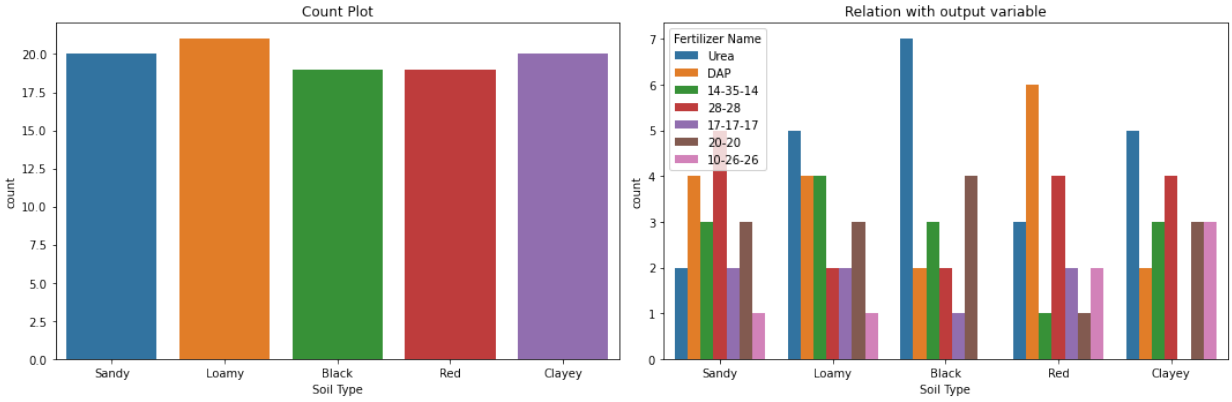
In [11]:
```
#EDA - Humidity variable
plot_conti(df['Humidity '])
```



In [12]:
```
#EDA - Moisture variable
plot_conti(df['Moisture'])
```

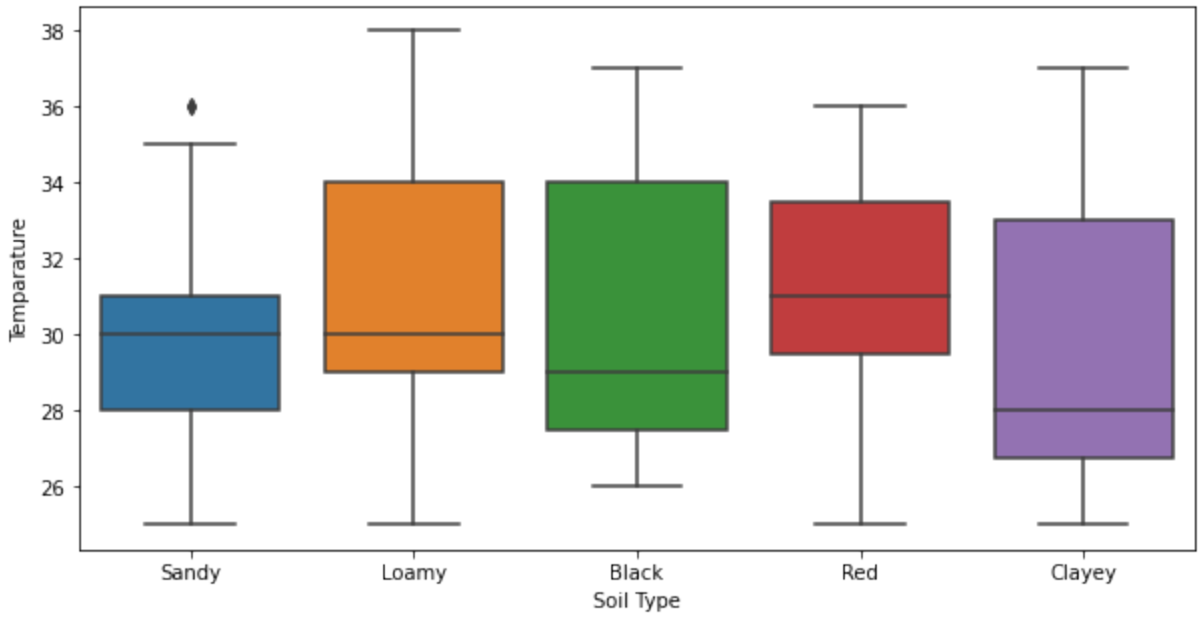


In [13]:
```
plot_cato(df['Soil Type'])
```



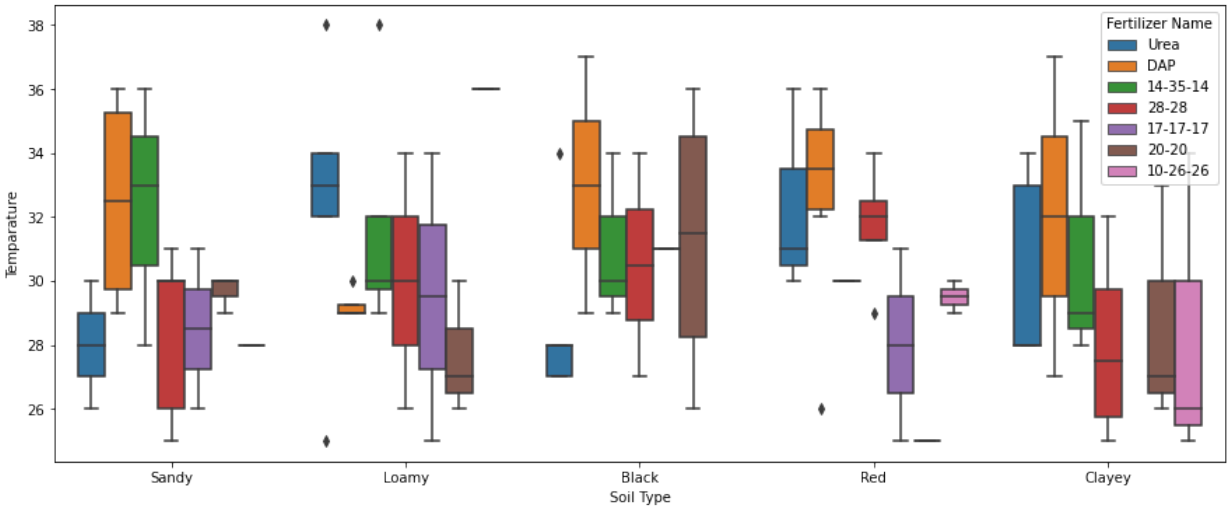
In [14]:
```
#relation of soil type with Temperature
plt.figure(figsize=(10,5))
sns.boxplot(x=df['Soil Type'],y=df['Temparature'])
```

Out[14]:
```
<AxesSubplot:xlabel='Soil Type', ylabel='Temparature'>
```

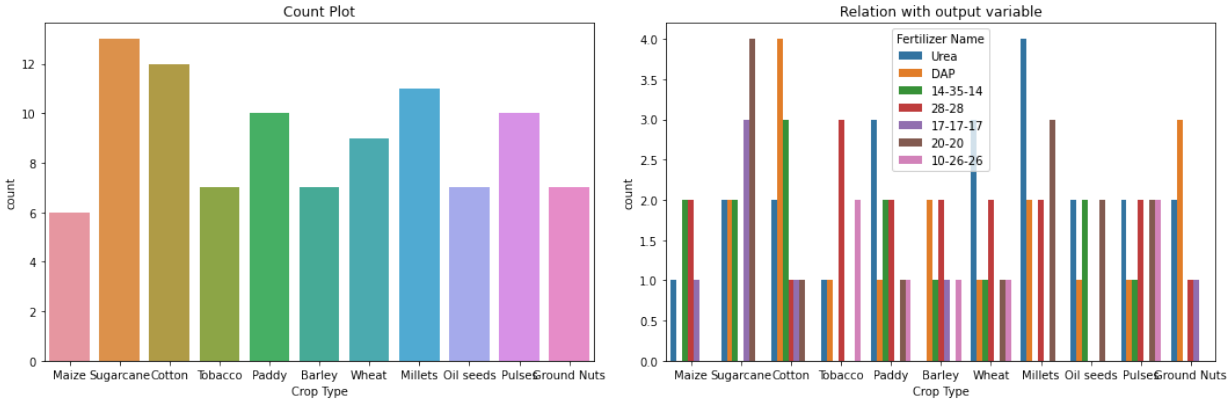


In [15]:
```
#relation of soil type and Temperature with output variable
plt.figure(figsize=(15,6))
sns.boxplot(x=df['Soil Type'],y=df['Temparature'],hue=df['Fertilizer Name'])
```

Out[15]:
```
<AxesSubplot:xlabel='Soil Type', ylabel='Temparature'>
```

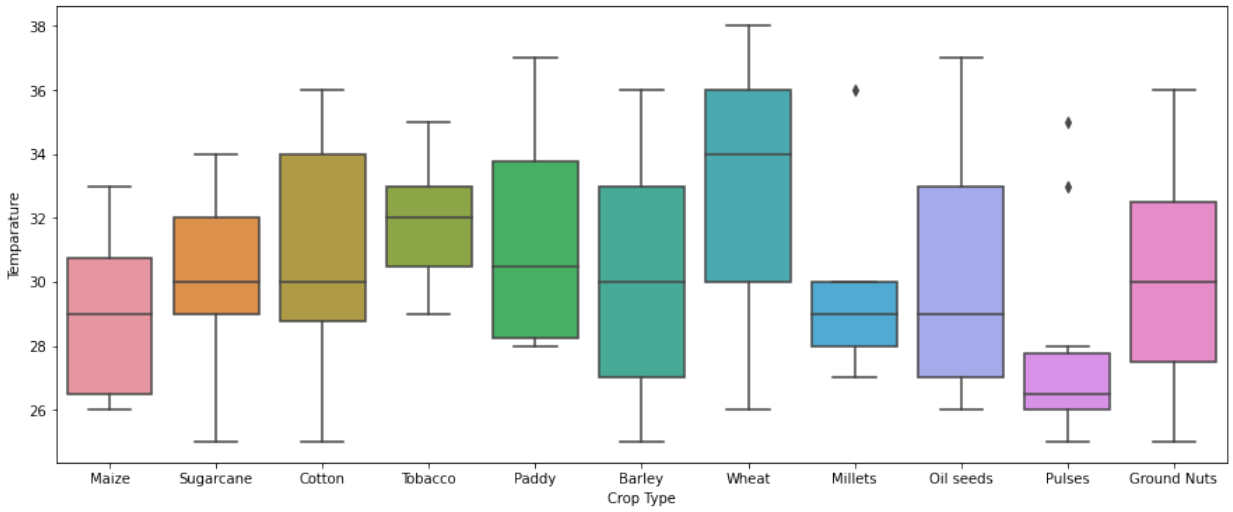


In [16]:
```
#EDA - Crop_Type variable
plot_cato(df['Crop Type'])
```



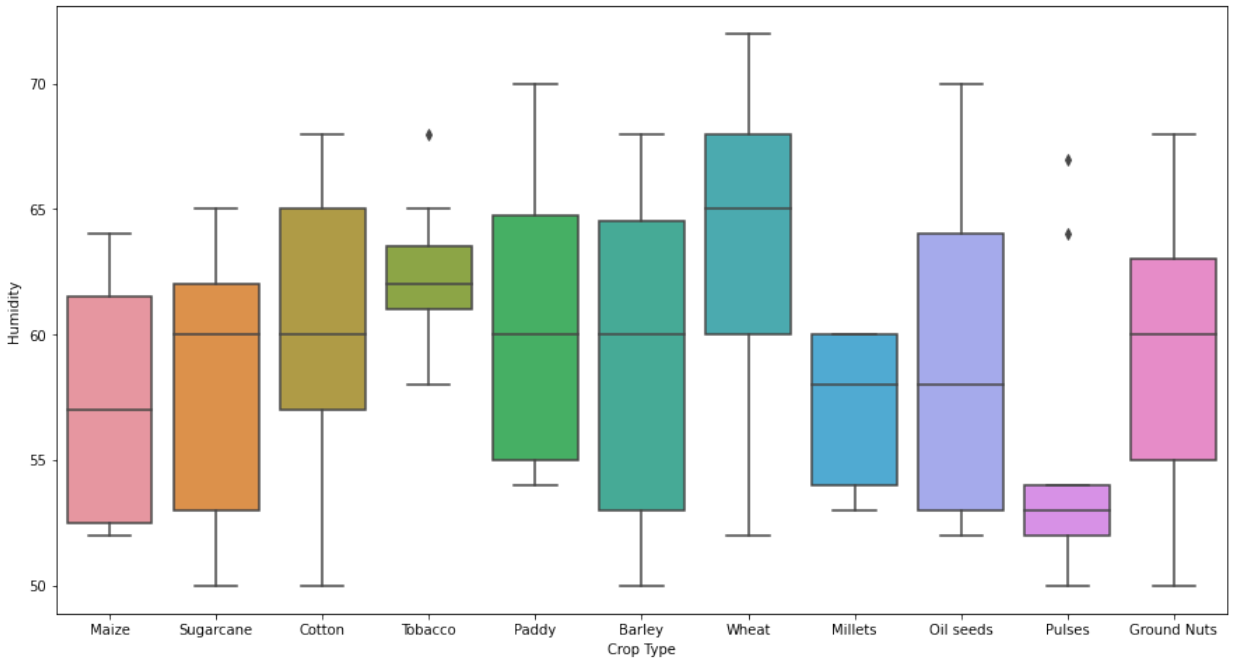
In [17]:
```
#relation of crop type with temperature
plt.figure(figsize=(15,6))
sns.boxplot(x=df['Crop Type'],y=df['Temparature'])
```

Out[17]:
```
<AxesSubplot:xlabel='Crop Type', ylabel='Temparature'>
```

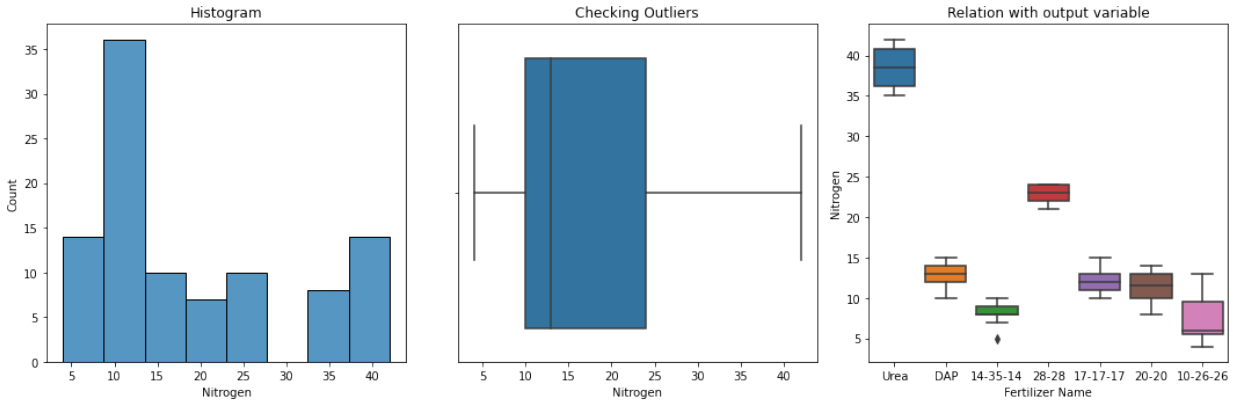


In [18]:
```
#relation of crop type with Humidity
plt.figure(figsize=(15,8))
sns.boxplot(x=df['Crop Type'],y=df['Humidity '])
```

Out[18]:
```
<AxesSubplot:xlabel='Crop Type', ylabel='Humidity '>
```

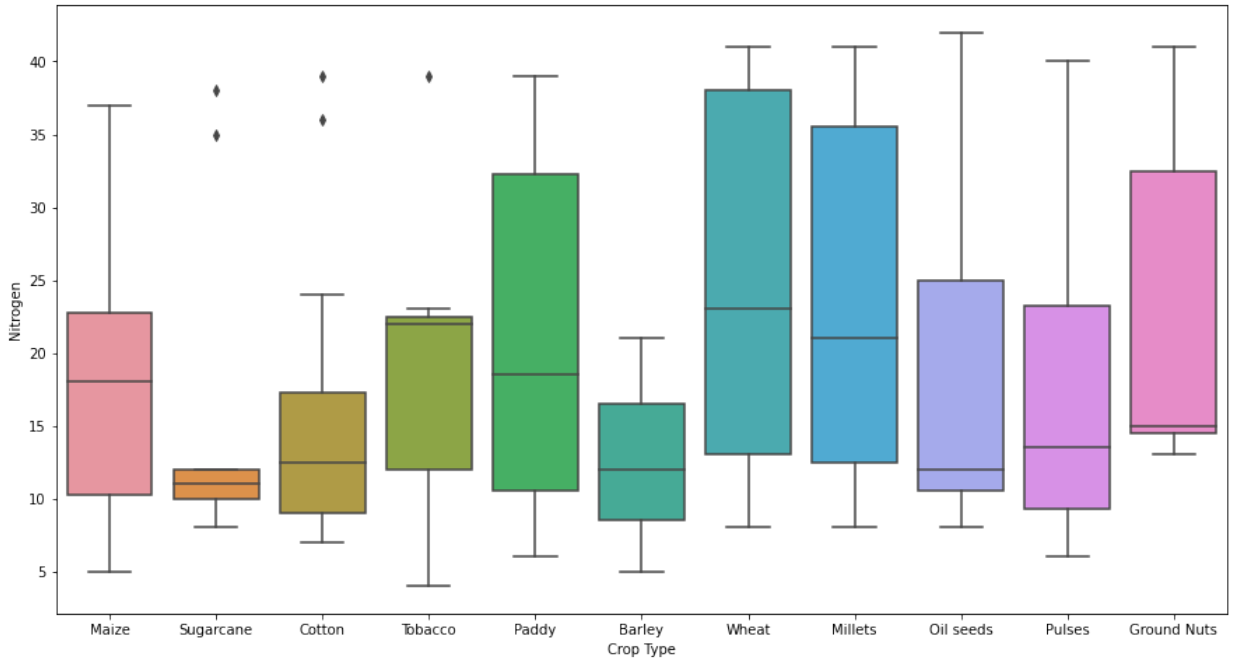


In [19]:
```
#EDA - Nitrogen variable
plot_conti(df['Nitrogen'])
```

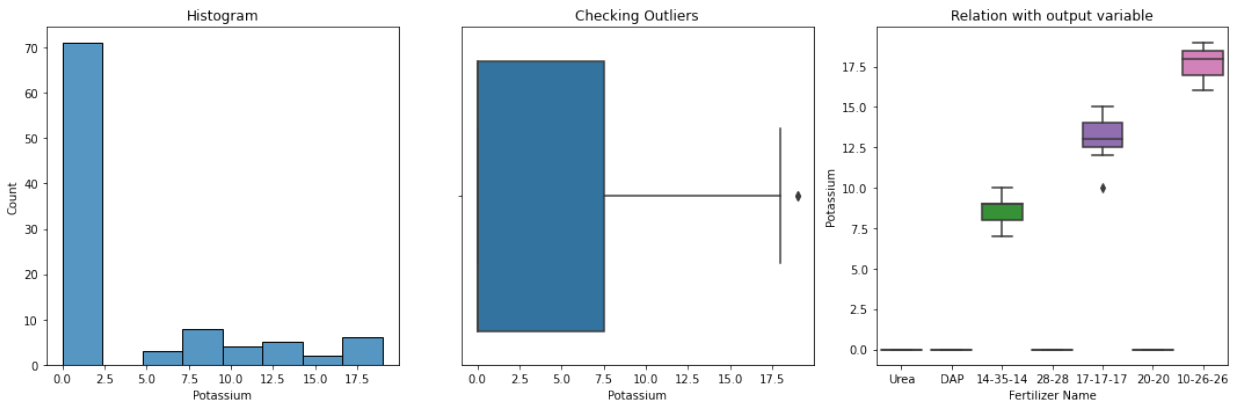


In [20]:
```
#relation of nitrogen wrt to crop type
plt.figure(figsize=(15,8))
sns.boxplot(x=df['Crop Type'],y=df['Nitrogen'])
```

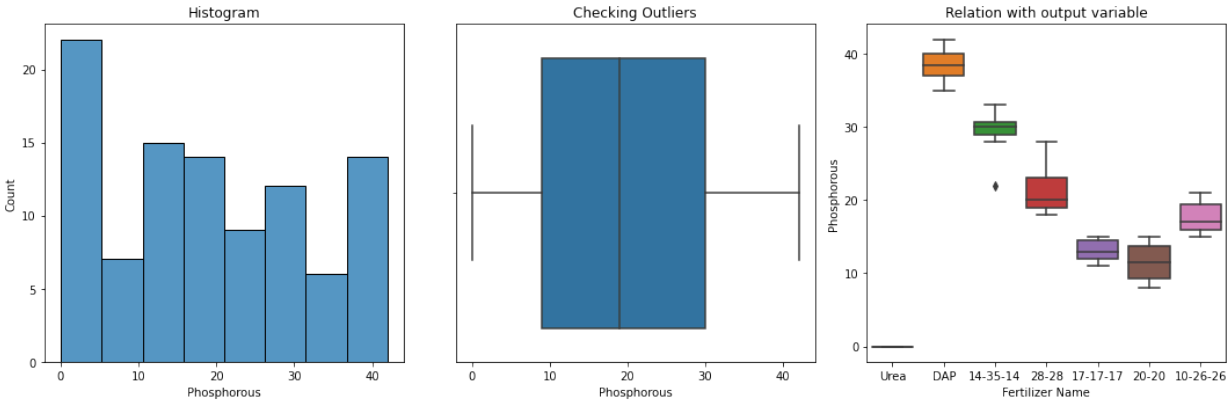
Out[20]:
```
<AxesSubplot:xlabel='Crop Type', ylabel='Nitrogen'>
```



In [21]:
```
#EDA - Potassium variable
plot_conti(df['Potassium'])
```



In [22]:
```
#EDA - Phosphorous variable
plot_conti(df['Phosphorous'])
```



Preprocessing using One-Hot Encoder

In [23]:
```
y = df['Fertilizer Name'].copy()
X = df.drop('Fertilizer Name', axis=1).copy()
```

In [24]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3,4])], remainder='
X = np.array(ct.fit_transform(X))
```

In [25]:
```
X[0]
```

Out[25]:
```
array([ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,
        0.,  0.,  0., 26., 52., 38., 37.,  0.,  0.])
```

Train-test split

In [26]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=True
```

Feature Scaling

In [27]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [28]:
```
X_train[0]
```

Out[28]:
```
array([-0.48181206,  1.98206242, -0.59408853, -0.50452498, -0.41169348,
       -0.24806947, -0.33601075, -0.33601075, -0.17277369, -0.36214298,
       -0.24806947, -0.36214298,  3.24037035, -0.43549417, -0.24806947,
       -0.36214298, -1.50323411, -1.54242294, -1.00710689,  0.46072126,
       -0.57643157,  0.09092764])
```

Random Forest Classifier

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators= 100, criterion = 'gini' , random_sta
classifier.fit(X_train, y_train)
```

Out[29]:
```
RandomForestClassifier(random_state=42)
```

In [30]:
```
y_pred = classifier.predict(X_test)
```

Creating confusion matrix

In [31]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Out[31]:
```
[[2 0 1 0 0 0 0]
 [0 4 0 0 0 0 0]
 [0 0 2 0 0 0 0]
 [0 0 0 1 0 0 0]
 [0 0 0 0 6 0 0]
 [0 0 0 0 0 7 0]
 [0 0 0 0 0 0 7]]
0.9666666666666667
```

In [32]:
```
classifier.score(X_test, y_test)
```

Out[32]:
```
0.9666666666666667
```

Test accuracy = 96.67%

Preprocessing using Label Encoder

In [33]:
```
#encoding the labels for categorical variables
from sklearn.preprocessing import LabelEncoder
```

In [34]:
```
#encoding Soil Type variable
encode_soil = LabelEncoder()
df['Soil Type'] = encode_soil.fit_transform(df['Soil Type'])

#creating the DataFrame
Soil_Type = pd.DataFrame(zip(encode_soil.classes_,encode_soil.transform(encode_soil.cl
Soil_Type = Soil_Type.set_index('Original')
Soil_Type
```

Out[34]:

Original	Encoded
Black	0
Clayey	1
Loamy	2
Red	3
Sandy	4

In [35]:
```
encode_crop =  LabelEncoder()
df['Crop Type'] = encode_crop.fit_transform(df['Crop Type'])

#creating the DataFrame
Crop_Type = pd.DataFrame(zip(encode_crop.classes_,encode_crop.transform(encode_crop.cl
Crop_Type = Crop_Type.set_index('Original')
Crop_Type
```

Out[35]:

Original	Encoded
Barley	0
Cotton	1
Ground Nuts	2
Maize	3
Millets	4
Oil seeds	5
Paddy	6
Pulses	7
Sugarcane	8
Tobacco	9
Wheat	10

In [36]:
```
encode_ferti = LabelEncoder()
df['Fertilizer Name'] = encode_ferti.fit_transform(df['Fertilizer Name'])

#creating the DataFrame
Fertilizer = pd.DataFrame(zip(encode_ferti.classes_,encode_ferti.transform(encode_fert
Fertilizer = Fertilizer.set_index('Original')
Fertilizer
```

Out[36]:

Original	Encoded
10-26-26	0
14-35-14	1
17-17-17	2
20-20	3
28-28	4
DAP	5
Urea	6

In [37]:
```
#splitting the data into train and test
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df.drop('Fertilizer Name',axis=1),
print('Shape of Splitting :')
print('x_train = {}, y_train = {}, x_test = {}, y_test = {}'.format(x_train.shape,y_tr
```

```
Shape of Splitting :
x_train = (79, 8), y_train = (79,), x_test = (20, 8), y_test = (20,)
```

In [38]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 79 entries, 2 to 37
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temparature  79 non-null     int64
 1   Humidity     79 non-null     int64
 2   Moisture     79 non-null     int64
 3   Soil Type    79 non-null     int32
 4   Crop Type    79 non-null     int32
 5   Nitrogen     79 non-null     int64
 6   Potassium    79 non-null     int64
 7   Phosphorous  79 non-null     int64
dtypes: int32(2), int64(6)
memory usage: 4.9 KB
```

Random Forest Classifier

In [39]:
```
rand = RandomForestClassifier(random_state = 42)
rand.fit(x_train,y_train)
```

Out[39]:
```
RandomForestClassifier(random_state=42)
```

In [40]:
```
pred_rand = rand.predict(x_test)
```

Hyperparameter tuning with GridSearchCV

In [41]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report

params = {
    'n_estimators':[300,400,500],
    'max_depth':[5,10,15],
    'min_samples_split':[2,5,8]
}
grid_rand = GridSearchCV(rand,params,cv=3,verbose=3,n_jobs=-1)

grid_rand.fit(x_train,y_train)

pred_rand = grid_rand.predict(x_test)

print(classification_report(y_test,pred_rand))

print('Best score : ',grid_rand.best_score_)
print('Best params : ',grid_rand.best_params_)
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
              precision    recall  f1-score   support

           0       1.00      0.33      0.50         3
           1       0.75      1.00      0.86         3
           2       0.67      1.00      0.80         2
           3       1.00      1.00      1.00         2
           4       1.00      1.00      1.00         2
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         6

    accuracy                           0.90        20
   macro avg       0.92      0.90      0.88        20
weighted avg       0.93      0.90      0.88        20

Best score :  0.9748338081671415
Best params :  {'max_depth': 5, 'min_samples_split': 2, 'n_estimators': 300}
```

In [42]:
```
y_train[2]
```

Out[42]:
```
1
```

In [43]:
```
#pickling the file
import pickle
pickle_out = open('classifier.pkl','wb')
pickle.dump(grid_rand,pickle_out)
pickle_out.close()
```

In [44]:
```
df.head()
```

Out[44]:

	Temparature	Humidity	Moisture	Soil Type	Crop Type	Nitrogen	Potassium	Phosphorous	Fertilizer Name
0	26	52	38	4	3	37	0	0	6
1	29	52	45	2	8	12	0	36	5
2	34	65	62	0	1	7	9	30	1
3	32	62	34	3	9	22	0	20	4
4	28	54	46	1	6	35	0	0	6

In [45]:

```
model = pickle.load(open('classifier.pkl','rb'))
ans = model.predict([[34,65,62  ,0,     1,      7,      9,      30]])
if ans[0] == 0:
    print("10-26-26")
elif ans[0] ==1:
    print("14-35-14")
elif ans[0] == 2:
    print("17-17-17     ")
elif ans[0] == 3:
    print("20-20")
elif ans[0] == 4:
    print("28-28")
elif ans[0] == 5:
    print("DAP")
else:
    print("Urea")
```

```
14-35-14
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: