In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv(r"C:\Users\vyshn\OneDrive\Desktop\FertilizerPrediction.csv")
```

In [3]:

```
df.head()
```

Out[3]:

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

In [8]:

```
df['Fertilizer Name'].value_counts()
```

Out[8]:

```
Urea        22
DAP         18
28-28       17
14-35-14    14
20-20       14
17-17-17     7
10-26-26     7
Name: Fertilizer Name, dtype: int64
```

In [4]:
```
df
```

Out[4]:

	Temparature	Humidity	Moisture	Soil Type	Crop Type	Nitrogen	Potassium	Phosphorous	Fer I
0	26	52	38	Sandy	Maize	37	0	0	
1	29	52	45	Loamy	Sugarcane	12	0	36	
2	34	65	62	Black	Cotton	7	9	30	14-
3	32	62	34	Red	Tobacco	22	0	20	
4	28	54	46	Clayey	Paddy	35	0	0	
...	...	...	...	...	...	...	...	...	
94	25	50	32	Clayey	Pulses	24	0	19	
95	30	60	27	Red	Tobacco	4	17	17	10-
96	38	72	51	Loamy	Wheat	39	0	0	
97	36	60	43	Sandy	Millets	15	0	41	
98	29	58	57	Black	Sugarcane	12	0	10	

99 rows × 9 columns

In [5]:
```
df.describe()
```

Out[5]:

	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 [6]:

```
df.corr() #correlation
```

```
C:\Users\vyshn\AppData\Local\Temp\ipykernel_7812\1412503361.py:1: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a f
uture version, it will default to False. Select only valid columns or specify
the value of numeric_only to silence this warning.
  df.corr() #correlation
```

Out[6]:

	Temparature	Humidity	Moisture	Nitrogen	Potassium	Phosphorous
Temparature	1.000000	0.973164	0.091222	-0.033771	-0.023424	0.207545
Humidity	0.973164	1.000000	0.091342	-0.060646	-0.003833	0.204044
Moisture	0.091222	0.091342	1.000000	-0.095945	0.027727	0.009276
Nitrogen	-0.033771	-0.060646	-0.095945	1.000000	-0.500087	-0.686971
Potassium	-0.023424	-0.003833	0.027727	-0.500087	1.000000	0.089192
Phosphorous	0.207545	0.204044	0.009276	-0.686971	0.089192	1.000000

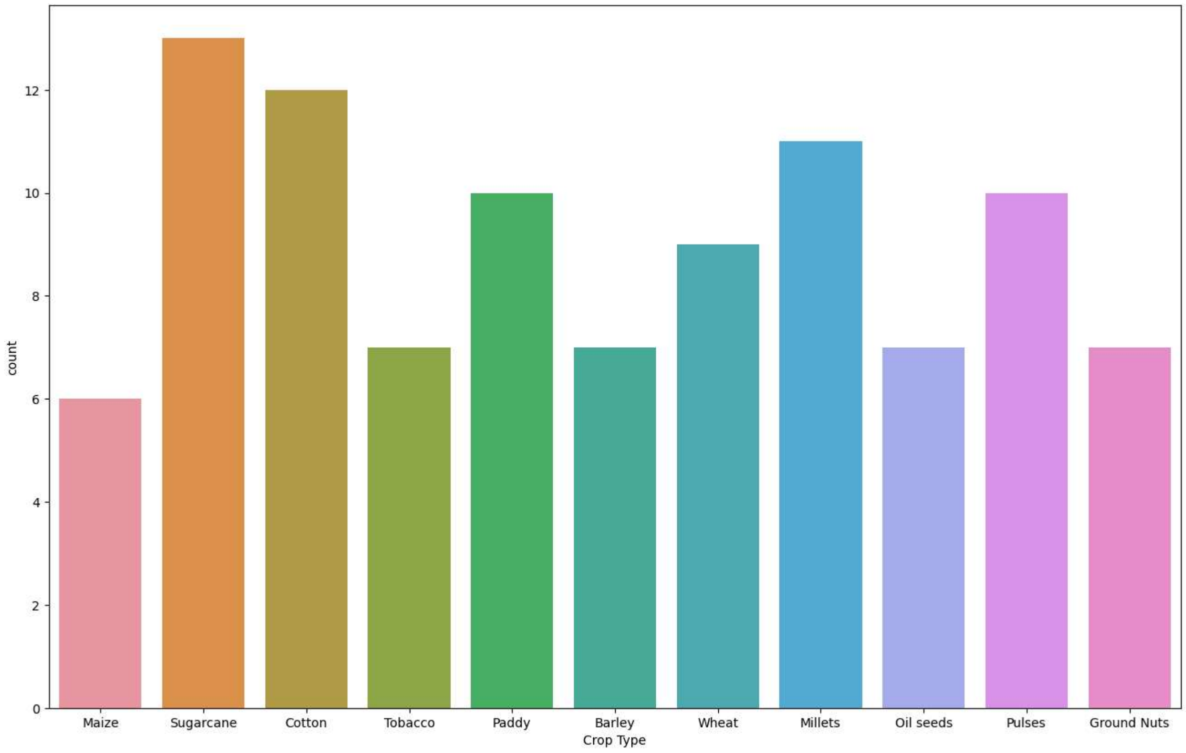
In [7]:

```
# count the no of crops in the given dataser

plt.figure(figsize=(16,10))
sns.countplot(x='Crop Type',data=df)
```

Out[7]:

```
<Axes: xlabel='Crop Type', ylabel='count'>
```



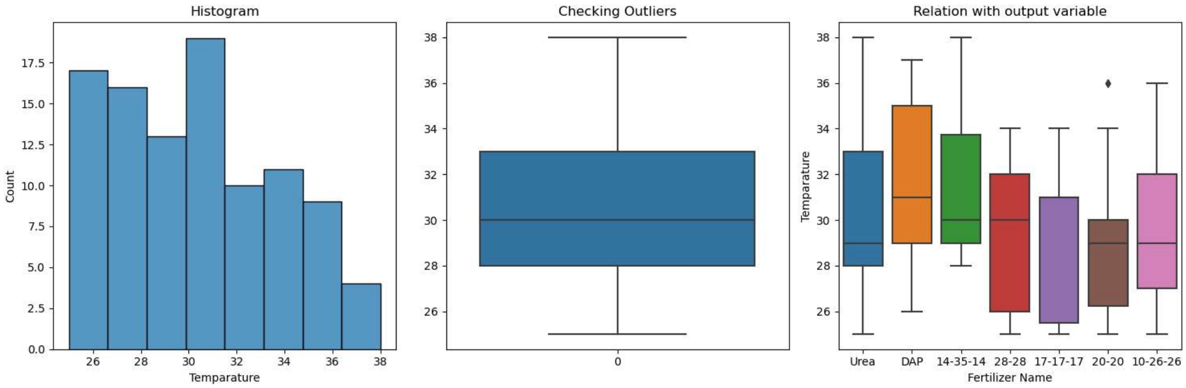
In [9]:

```
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])
```

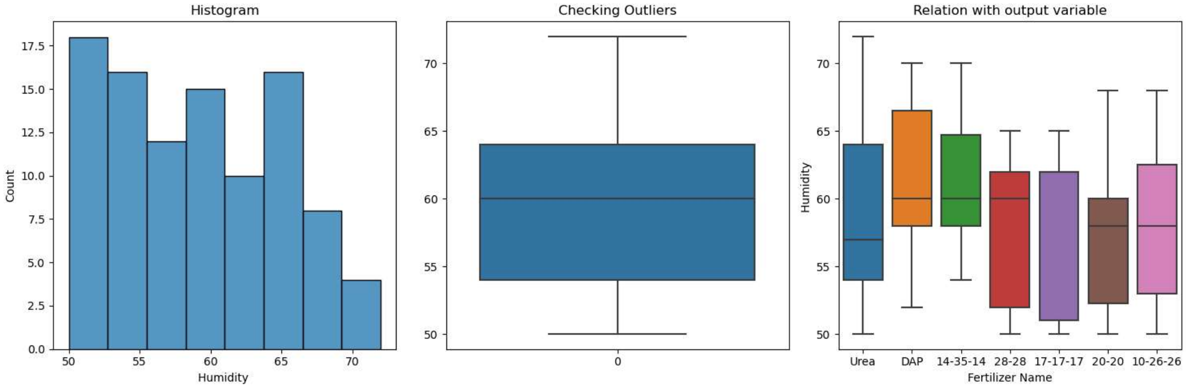
In [10]:

```
plot_conti(df['Temparature'])
```

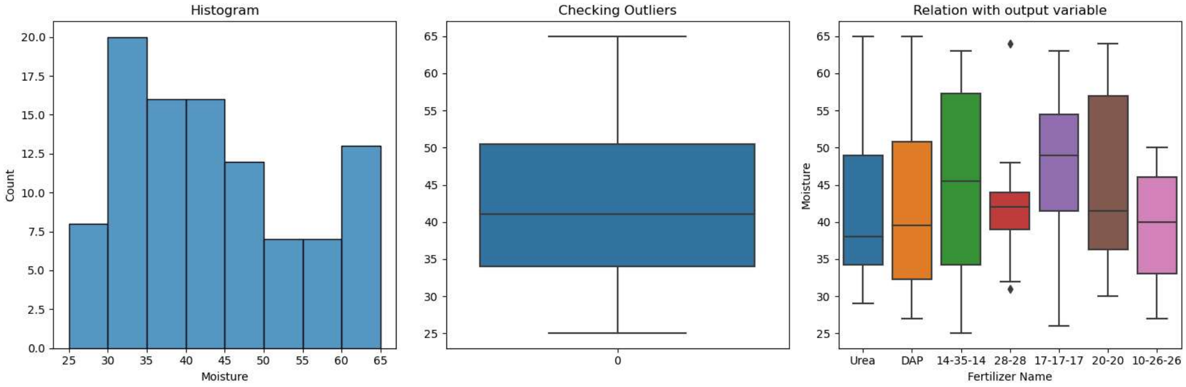


In [11]:

```
plot_conti(df['Humidity '])
```

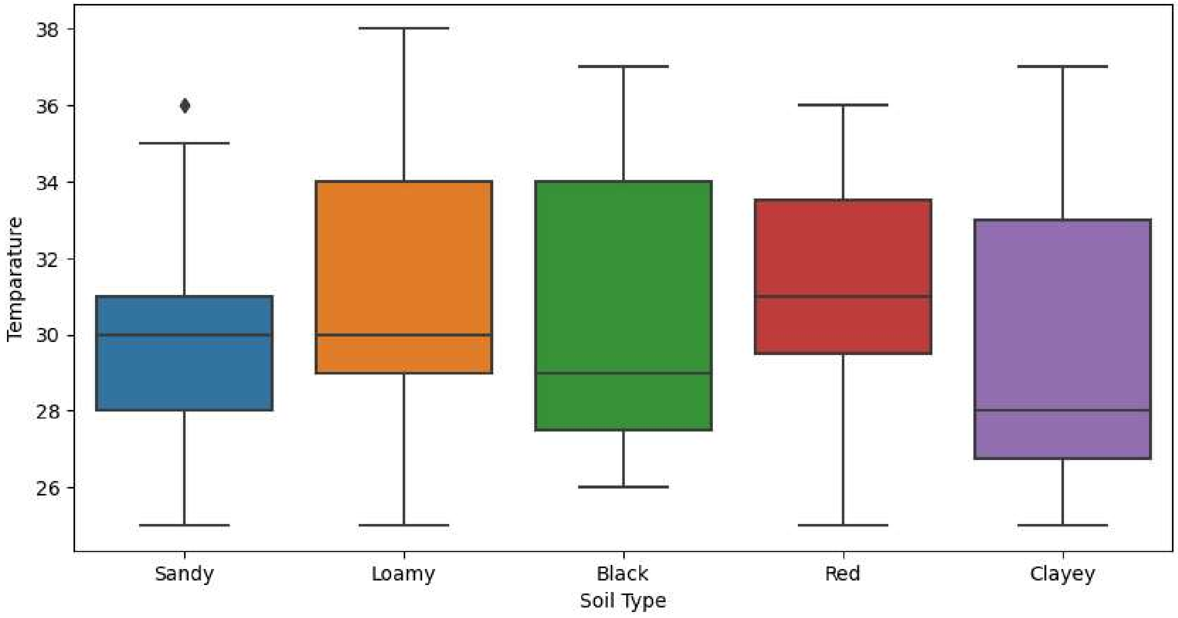


In [12]:
```
plot_conti(df['Moisture'])
```



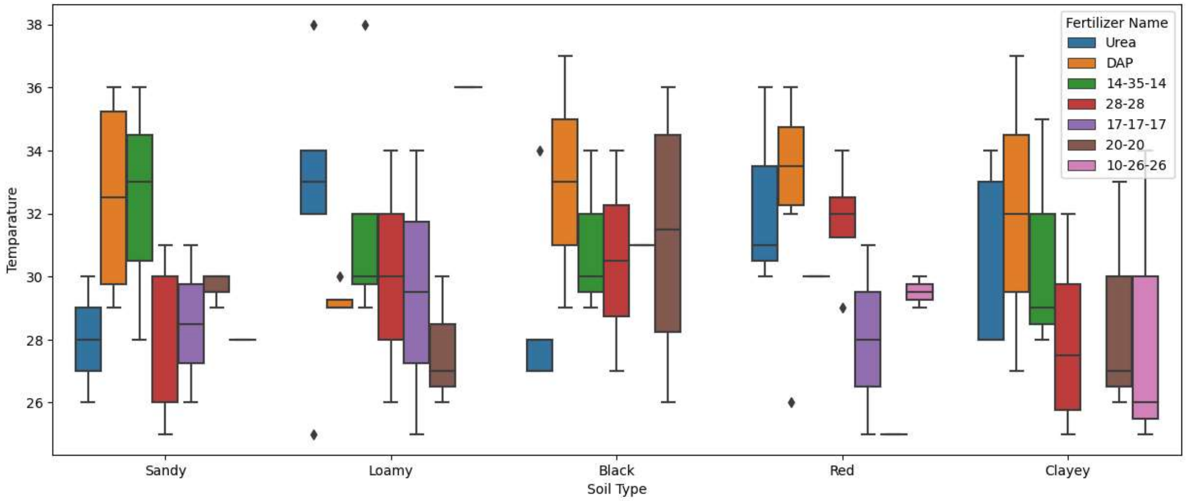
In [13]:
```
#relation of soil type with Temperature
plt.figure(figsize=(10,5))
sns.boxplot(x=df['Soil Type'],y=df['Temparature'])
```

Out[13]:
```
<Axes: xlabel='Soil Type', ylabel='Temparature'>
```



In [14]:
```
plt.figure(figsize=(15,6))
sns.boxplot(x=df['Soil Type'],y=df['Temparature'],hue=df['Fertilizer Name'])
```

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



In [15]:
```
from sklearn.preprocessing import LabelEncoder
```

In [16]:
```
#lets convert the categorial data int labels


LE= LabelEncoder()

df['Soil Type'] = LE.fit_transform(df['Soil Type'])
Soil_Type = pd.DataFrame(zip(LE.classes_,LE.transform(LE.classes_)),columns=['
Soil_Type = Soil_Type.set_index('Original')
Soil_Type
```

Out[16]:

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

In [17]:

```
LE1 =  LabelEncoder()
df['Crop Type'] = LE1.fit_transform(df['Crop Type'])

#creating the DataFrame
Crop_Type = pd.DataFrame(zip(LE1.classes_,LE1.transform(LE1.classes_)),columns
Crop_Type = Crop_Type.set_index('Original')
Crop_Type
```

Out[17]:

	Encoded
Original	
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 [18]:

```
from sklearn.preprocessing import LabelEncoder
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(enc
Fertilizer = Fertilizer.set_index('Original')
Fertilizer
```

Out[18]:

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

In [19]:

```
#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',
```

In [20]:

```
from sklearn.ensemble import RandomForestClassifier
rand = RandomForestClassifier(random_state = 32)
rand.fit(x_train,y_train)
```

Out[20]:

```
RandomForestClassifier(random_state=32)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [21]:

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score
# making predictions on the set
pred_rand = rand.predict(x_test)
pred_rand
# calculate accuracy on the test set
y_pred = rand.predict(x_test)
acc = accuracy_score(y_test,y_pred)
print(acc)
# calculate accuracy on the train set
train_pred = rand.predict(x_train)
train_acc = accuracy_score(y_train, train_pred)
print(train_acc)
```

```
0.9
1.0
```

In [22]:

```
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 :  0.9876543209876543
Best params :  {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 300}
```

In [23]:

```
from sklearn.ensemble import RandomForestClassifier
rand = RandomForestClassifier(n_estimators=300,min_samples_split=2,max_depth=5
rand.fit(x_train,y_train)
```

Out[23]:

```
RandomForestClassifier(max_depth=5, n_estimators=300, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:

```
y_pred = rand.predict(x_test)
acc = accuracy_score(y_test,y_pred)
print(acc)
train_pred = rand.predict(x_train)
train_acc = accuracy_score(y_train, train_pred)
print(train_acc)
```

```
0.9
1.0
```

In [25]:

```
df.head()
```

Out[25]:

	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 [31]:

```
import numpy as np
import warnings
warnings.simplefilter('ignore')
prediction = rand.predict((np.array([[34,65,62,0,1,7,9,30]])))
print("Fertilizer:", prediction)
```

```
Fertilizer: [1]
```

In [ ]:

In [ ]: