In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Fertilizer Prediction.csv')
df.head()
```

Out[2]:

	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 [3]:
```
df.columns
```

Out[3]:
```
Index(['Temparature', 'Humidity ', 'Moisture', 'Soil Type', 'Crop Type',
       'Nitrogen', 'Potassium', 'Phosphorous', 'Fertilizer Name'],
      dtype='object')
```

In [4]:
```
df.shape
```

Out[4]:
```
(99, 9)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Temparature      99 non-null     int64 
 1   Humidity         99 non-null     int64 
 2   Moisture         99 non-null     int64 
 3   Soil Type        99 non-null     object
 4   Crop Type        99 non-null     object
 5   Nitrogen         99 non-null     int64 
 6   Potassium        99 non-null     int64 
 7   Phosphorous      99 non-null     int64 
 8   Fertilizer Name  99 non-null     object
dtypes: int64(6), object(3)
memory usage: 7.1+ KB
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Temparature        0
Humidity           0
Moisture           0
Soil Type          0
Crop Type          0
Nitrogen           0
Potassium          0
Phosphorous        0
Fertilizer Name    0
dtype: int64
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Temparature', 'Humidity ', 'Moisture', 'Soil Type', 'Crop Type',
       'Nitrogen', 'Potassium', 'Phosphorous', 'Fertilizer Name'],
      dtype='object')
```

In [8]:
```
df.columns =  df.columns.str.lower()
```

In [9]:
```
df.columns = df.columns.str.replace(" ", "")
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['temparature', 'humidity', 'moisture', 'soiltype', 'croptype',
       'nitrogen', 'potassium', 'phosphorous', 'fertilizername'],
      dtype='object')
```

In [11]:
```
# Here the dataset donot have any missing value
```

In [12]:
```
df['temparature'].value_counts()
```

Out[12]:
```
30    14
29    13
34    11
26    10
28     9
25     7
27     7
36     6
32     5
33     5
31     5
35     3
37     2
38     2
Name: temparature, dtype: int64
```

In [13]:
```
df['temparature'].duplicated().sum()
```

Out[13]:
```
85
```

In [14]:
```
# Here we 85 duplicate value so we can drop the duplicate value
```

In [15]:
```
df['temparature'].drop_duplicates()
```

Out[15]:
```
0     26
1     29
2     34
3     32
4     28
6     25
7     33
8     30
10    27
11    31
22    35
28    37
33    36
45    38
Name: temparature, dtype: int64
```

In [16]:
```
df['temparature'].isnull().sum()
```

Out[16]:
```
0
```

In [17]:
```
df['humidity'].duplicated().sum()
```

Out[17]:
```
86
```

In [18]:
```
df['humidity'].drop_duplicates()
```

Out[18]:
```
0     52
2     65
3     62
4     54
6     50
7     64
8     60
9     58
22    68
28    70
44    67
55    53
96    72
Name: humidity, dtype: int64
```

In [19]:
```
df['humidity'].isnull().sum()
```

Out[19]:
```
0
```

In [20]:
```
df['moisture'].value_counts()
```

Out[20]:
```
34    6
38    5
43    4
32    4
30    4
48    4
41    3
47    3
40    3
63    3
44    3
39    3
31    3
37    3
65    3
35    3
33    3
42    3
59    2
57    2
62    2
51    2
26    2
49    2
27    2
61    2
36    2
64    2
50    2
45    2
28    2
58    1
60    1
46    1
29    1
53    1
54    1
56    1
```

```
52    1
55    1
25    1
Name: moisture, dtype: int64
```

In [21]:
```
df['moisture'].duplicated().sum()
```

Out[21]:
```
58
```

In [22]:
```
df['moisture'].drop_duplicates()
```

Out[22]:
```
0     38
1     45
2     62
3     34
4     46
5     35
6     64
7     50
8     42
9     33
10    28
11    48
12    65
13    41
14    31
15    49
17    39
19    52
20    44
21    53
23    37
26    63
27    30
28    32
29    36
30    40
31    27
36    61
38    26
41    58
43    60
47    43
48    29
49    51
53    47
56    54
58    56
66    57
67    55
77    59
93    25
Name: moisture, dtype: int64
```

In [23]:
```
df['moisture'].isnull().sum()
```

Out[23]:
```
0
```

In [24]:
```
df.head()
```

Out[24]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous	fertilizername
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 [25]:
```
df.columns
```

Out[25]:
```
Index(['temparature', 'humidity', 'moisture', 'soiltype', 'croptype',
       'nitrogen', 'potassium', 'phosphorous', 'fertilizername'],
      dtype='object')
```

In [26]:
```
df['soiltype'].value_counts()
```

Out[26]:
```
Loamy     21
Sandy     20
Clayey    20
Black     19
Red       19
Name: soiltype, dtype: int64
```

In [27]:
```
df['soiltype'].duplicated().sum()
```

Out[27]:
```
94
```

In [28]:
```
df['soiltype'].drop_duplicates()
```

Out[28]:
```
0     Sandy
1     Loamy
2     Black
3       Red
4    Clayey
Name: soiltype, dtype: object
```

In [29]:
```
df['soiltype'].isnull().sum()
```

Out[29]:
```
0
```

In [30]:
```
df.head()
```

Out[30]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous	fertilizername
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 [31]:
```
df['croptype'].value_counts()
```

Out[31]:
```
Sugarcane      13
Cotton         12
Millets        11
Paddy          10
Pulses         10
Wheat           9
Tobacco         7
Barley          7
Oil seeds       7
Ground Nuts     7
```

```
Maize           6
Name: croptype, dtype: int64
```

In [32]:
```
df['croptype'].duplicated().sum()
```

Out[32]:
```
88
```

In [33]:
```
df['croptype'].drop_duplicates()
```

Out[33]:
```
0           Maize
1       Sugarcane
2          Cotton
3         Tobacco
4           Paddy
5          Barley
7           Wheat
8         Millets
9       Oil seeds
10         Pulses
14    Ground Nuts
Name: croptype, dtype: object
```

In [34]:
```
df['croptype'].isnull().sum()
```

Out[34]:
```
0
```

In [35]:
```
df.head()
```

Out[35]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous	fertilizername
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 [36]:
```
df['nitrogen'].value_counts()
```

Out[36]:
```
12    10
13     8
10     7
8      6
9      6
23     5
11     5
24     5
14     5
41     5
15     5
21     4
39     4
36     3
38     3
35     3
22     3
7      3
6      2
37     2
5      2
40     1
42     1
```

```
4       1
Name: nitrogen, dtype: int64
```

In [37]:
```
df['nitrogen'].duplicated().sum()
```

Out[37]:
```
75
```

In [38]:
```
df['nitrogen'].drop_duplicates()
```

Out[38]:
```
0      37
1      12
2       7
3      22
4      35
6       9
7      41
8      21
10     13
11     14
12     36
13     24
15     10
16     38
18     39
22     11
25     23
36      8
38     15
49      5
52     40
68      6
73     42
95      4
Name: nitrogen, dtype: int64
```

In [39]:
```
df['nitrogen'].isnull().sum()
```

Out[39]:
```
0
```

In [40]:
```
df.head()
```

Out[40]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous	fertilizername
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 [41]:
```
df['potassium'].value_counts()
```

Out[41]:
```
0      71
9       5
10      4
7       3
8       3
13      2
14      2
18      2
19      2
17      2
```

```
15     1
12     1
16     1
Name: potassium, dtype: int64
```

In [42]:
```
df['potassium'].duplicated().sum()
```

Out[42]:
```
86
```

In [43]:
```
df['potassium'].drop_duplicates()
```

Out[43]:
```
0      0
2      9
5     10
9      7
11    15
15    13
21    14
45     8
54    12
63    18
68    19
78    16
86    17
Name: potassium, dtype: int64
```

In [44]:
```
df['potassium'].isnull().sum()
```

Out[44]:
```
0
```

In [45]:
```
df.head()
```

Out[45]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous	fertilizername
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 [46]:
```
df['phosphorous'].value_counts()
```

Out[46]:
```
0     22
19     5
15     5
30     5
20     4
13     4
10     3
40     3
41     3
14     3
9      3
37     3
29     3
21     3
38     2
16     2
24     2
28     2
35     2
```

```
39     2
31     2
23     2
36     2
22     2
12     2
18     2
32     1
11     1
8      1
42     1
33     1
17     1
Name: phosphorous, dtype: int64
```

In [47]:
```
df['phosphorous'].duplicated().sum()
```

Out[47]:
```
67
```

In [48]:
```
df['phosphorous'].drop_duplicates()
```

Out[48]:
```
0      0
1     36
2     30
3     20
5     13
6     10
8     18
10    40
11    12
13    22
14    41
15    14
17    19
20     9
22    37
26    29
28    39
30    23
36    31
38    11
41    32
42    24
44    35
45    28
46     8
50    21
53    42
54    15
67    33
68    16
79    38
95    17
Name: phosphorous, dtype: int64
```

In [49]:
```
df['phosphorous'].isnull().sum()
```

Out[49]:
```
0
```

In [50]:
```
df.head()
```

Out[50]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous	fertilizername
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 [51]:
```
df['fertilizername'].value_counts()
```

Out[51]:
```
Urea         22
DAP          18
28-28        17
14-35-14     14
20-20        14
17-17-17      7
10-26-26      7
Name: fertilizername, dtype: int64
```

In [52]:
```
df['fertilizername'].duplicated().sum()
```

Out[52]:
```
92
```

In [53]:
```
df['fertilizername'].drop_duplicates()
```

Out[53]:
```
0          Urea
1           DAP
2     14-35-14
3        28-28
5     17-17-17
6        20-20
63    10-26-26
Name: fertilizername, dtype: object
```

Details for the type of fertilizer and also about the use of that fertilizer

1 .14-35-14 is an ideal complex particularly for Rice, Cotton, groundnut, chillies, soyabean, potato and other commercial crops which require high Phosphate initially. However for chlorine sensitive crops like grapes, application of 14-35-14 is not advisable.

2 .28-28 Can be used as basal or top-dressing fertiliser. It is an ideal complex fertiliser for all crops for basal application.

3.17-17-17 The compound fertilizer contains nitrogen (N) and phosphorus (P_2O_5) in a balanced manner. It should be used for balanced fertilization, especially in phosphorus-poor soils. It is used as base fertilizer (subsoil) in all plants considering the root depth of the plant.

4.20-20 Water-soluble fertilizer NPK 20-20-20 is well balanced in nitrogen (N), phosphorus (P) and potassium (K) in the ratio of 20:20:20 . It contains magnesium and trace elements . It is formulated with high quality nutrients thereby obtaining a high solubility and perfect assimilation.

5.10-26-26 Used as a complex fertilizer for supply of all three major nutrient like Nitrogen, Phosphorus and Potash. Used in basal application in crops like Wheat, Paddy, Maize, Pulses, Sugarcane, Vegetables etc.

6.Urea is a source of Nitrogen, an essential nutrient crucial for crop growth and development. Urea is the most important nitrogenous fertilizer in the country because of its high N content (46%N). It also has industrial applications such as the production of plastics and as a nutritional supplement for cattle.

7.DAP fertilizer Di-ammonium Phosphate popularly known as DAP is a preferred fertilizer in India because it contains both Nitrogen and Phosphorus which are primary macro-nutrients and part of 18 essential plant nutrients. DAP (NH4)2HPO4: Fertilizer grade DAP Contains 18% Nitrogen and 46% Phosphorus (P2O5)

Univariante analysis

In [54]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [55]:
```
num_cols = [fea for fea in df.columns if df[fea].dtype!='O']
cat_cols = [fea for fea in df.columns if df[fea].dtype=='O']
```

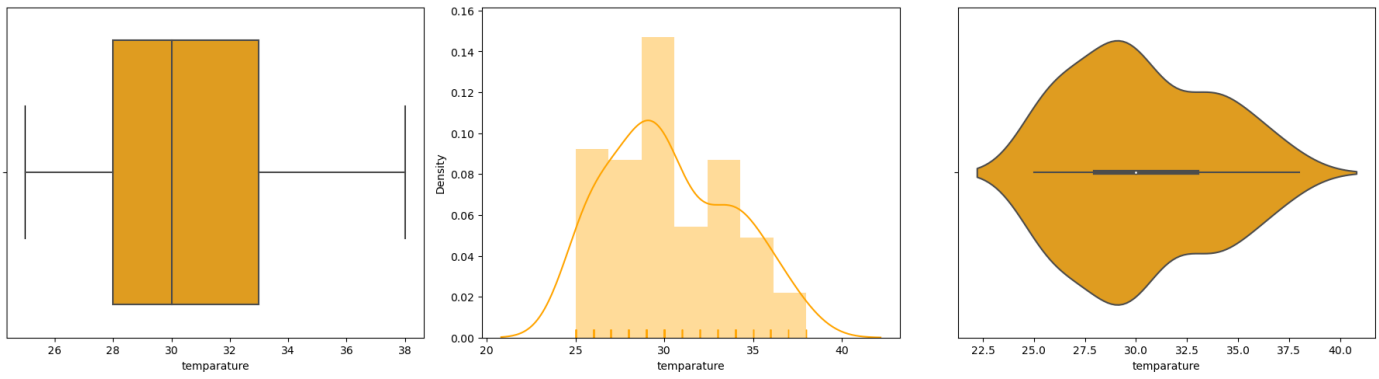
In [56]:
```
num_cols
```

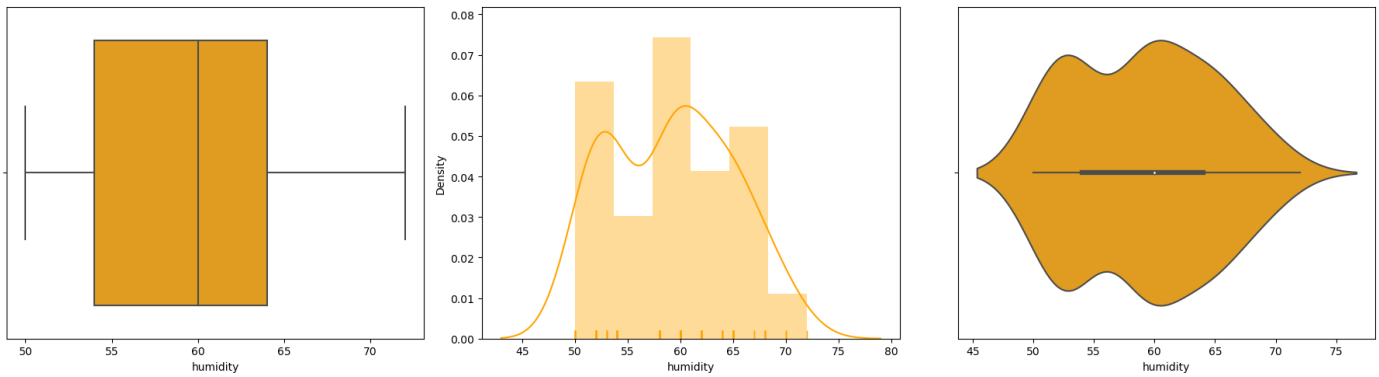
Out[56]:
```
['temparature', 'humidity', 'moisture', 'nitrogen', 'potassium', 'phosphorous']
```

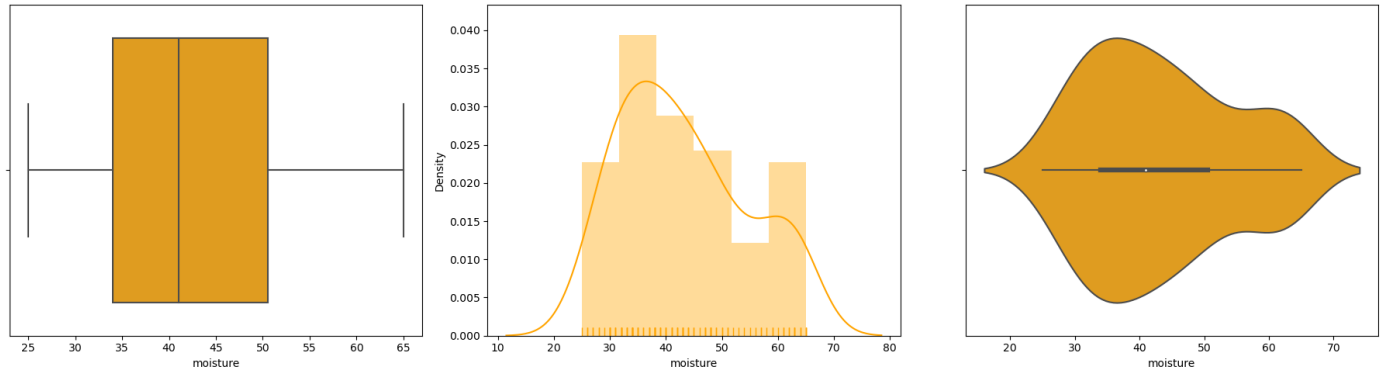
In [57]:
```
cat_cols
```

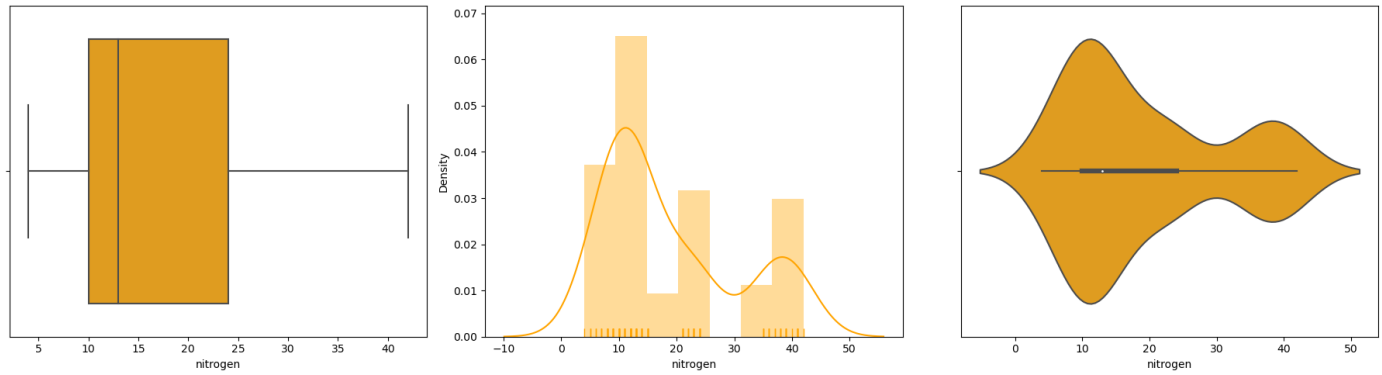
Out[57]:
```
['soiltype', 'croptype', 'fertilizername']
```

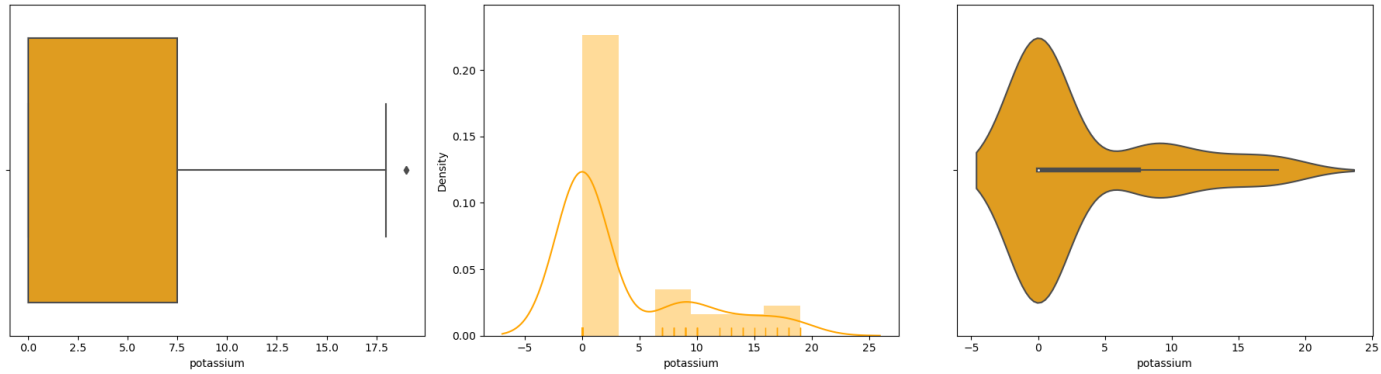
In [58]:
```
num = df[num_cols]
for i in enumerate(num_cols):
    f = plt.figure(figsize=(18,5))
    ax = f.add_subplot(131)
    sns.boxplot(num[i[1]],color = "orange")

    ax1 = f.add_subplot(132)
    sns.distplot(num[i[1]], rug = True,  color= 'orange',kde=True)

    ax2 = f.add_subplot(133)
    sns.violinplot(num[i[1]], orient= 'vertical',  color= 'orange')
    plt.tight_layout()
    plt.show()
```

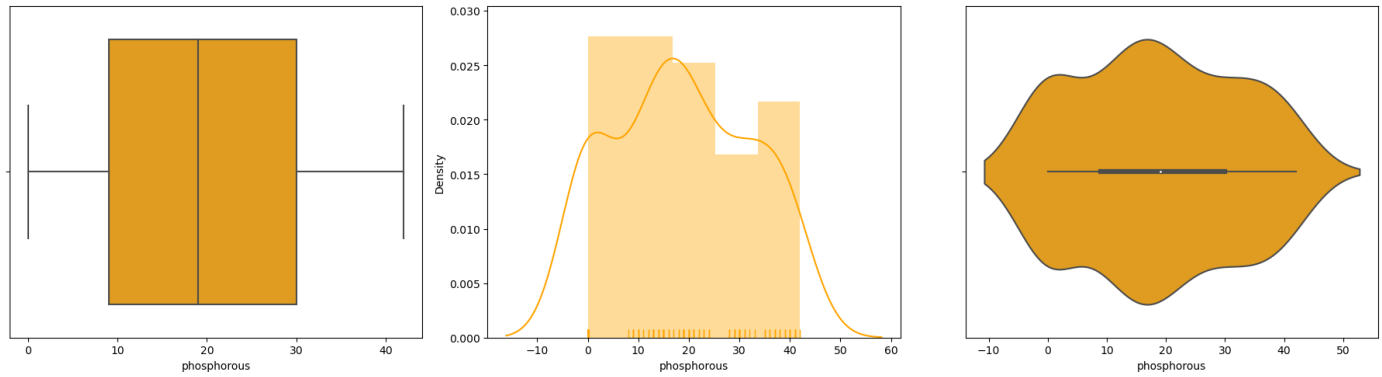






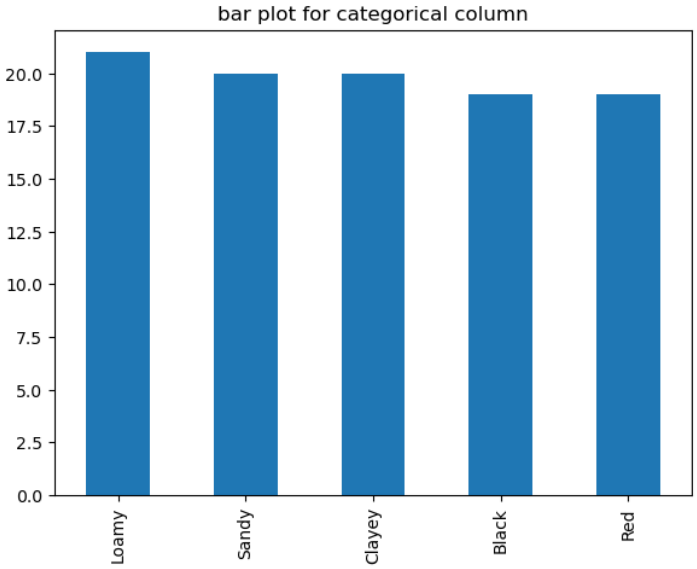


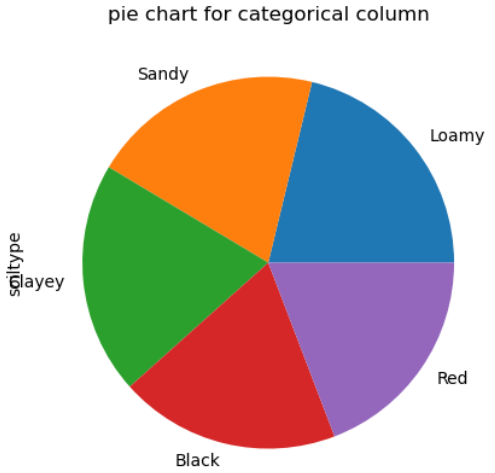


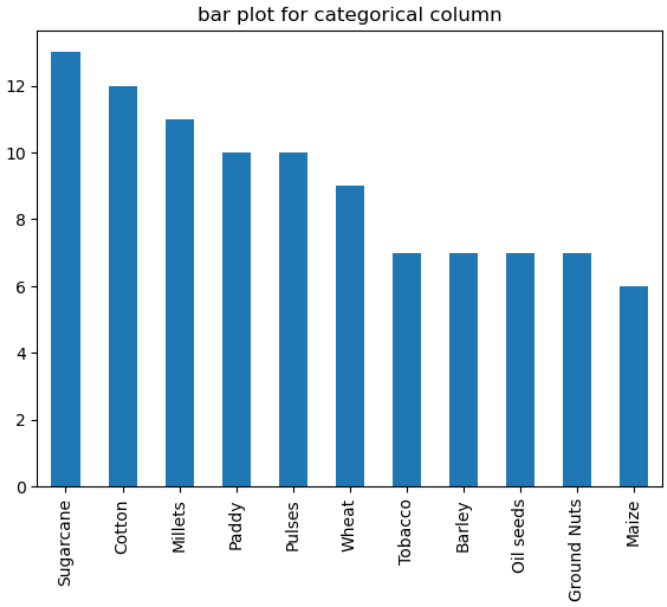


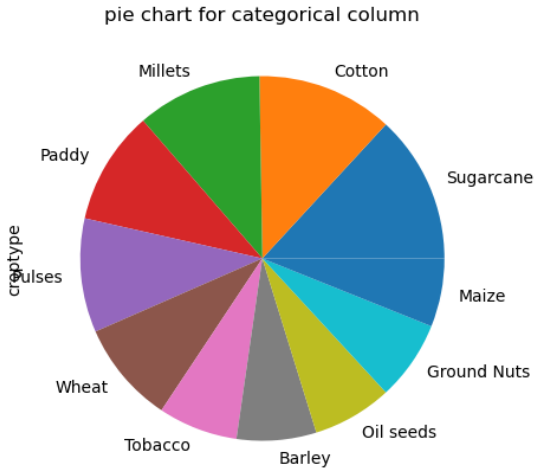
In [59]:
```
def plot(df,col_name):
    fig=plt.figure(figsize=(15,5))
    ax0=fig.add_subplot(121)
    df.value_counts().plot.bar()
    plt.title("bar plot for {} column".format(col_name))

    ax1=fig.add_subplot(122)
    df.value_counts().plot.pie()
    plt.title("pie chart for {} column".format(col_name))
    plt.show()
```

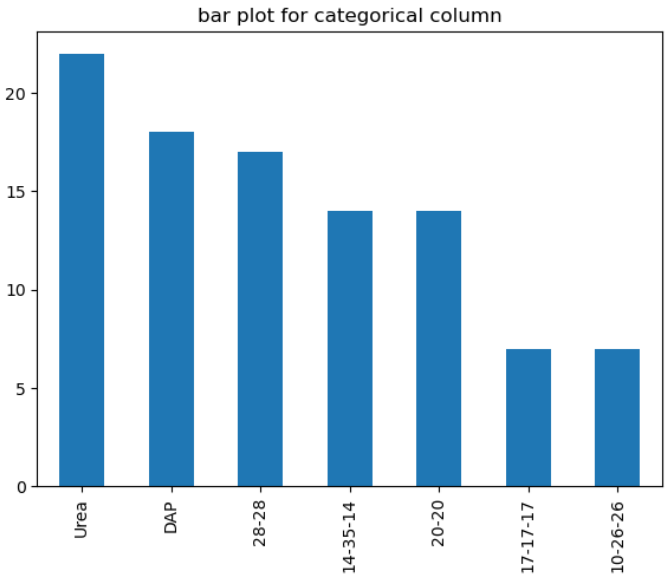
In [60]:
```
cat=df[cat_cols]
for i in enumerate(cat_cols):
    plot(cat[i[1]],"categorical")
```

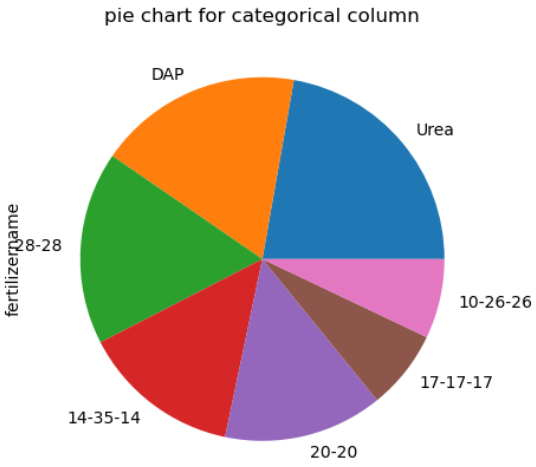








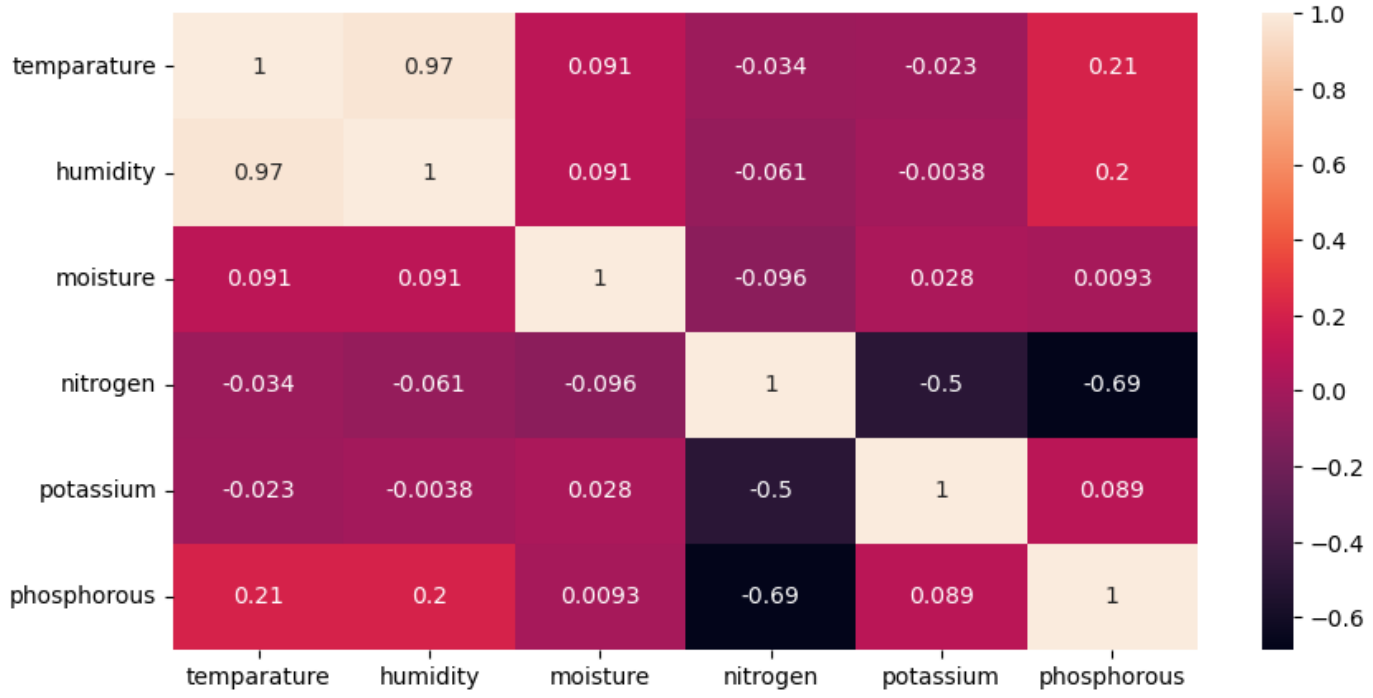




Multivariant-Analysis

In [61]:
```
plt.figure(figsize = (10,5))
sns.heatmap(df.corr(),annot = True)
```

Out[61]:
```
<AxesSubplot:>
```



In [62]:
```
## Feature Engineering
```

In [63]:
```
cat_cols
```

Out[63]:
```
['soiltype', 'croptype', 'fertilizername']
```

In [64]:
```
df["soiltype"].value_counts()
```

Out[64]:
```
Loamy     21
Sandy     20
Clayey    20
Black     19
Red       19
Name: soiltype, dtype: int64
```

In [65]:
```
df["croptype"].value_counts()
```

Out[65]:
```
Sugarcane      13
Cotton         12
Millets        11
Paddy          10
Pulses         10
Wheat           9
Tobacco         7
Barley          7
Oil seeds       7
Ground Nuts     7
Maize           6
Name: croptype, dtype: int64
```

In [66]:
```
df["fertilizername"].value_counts()
```

Out[66]:
```
Urea        22
DAP         18
28-28       17
14-35-14    14
20-20       14
17-17-17     7
10-26-26     7
Name: fertilizername, dtype: int64
```

In [67]:
```
from sklearn.preprocessing import LabelEncoder
```

In [68]:
```
soil_type_label_encoder = LabelEncoder()
df["soiltype"] = soil_type_label_encoder.fit_transform(df["soiltype"])
```

In [69]:
```
crop_type_label_encoder = LabelEncoder()
df["croptype"] = crop_type_label_encoder.fit_transform(df["croptype"])
```

In [70]:
```
croptype_dict = {}
for i in range(len(df["croptype"].unique())):
    croptype_dict[i] = crop_type_label_encoder.inverse_transform([i])[0]
print(croptype_dict)

soiltype_dict = {}
for i in range(len(df["soiltype"].unique())):
    soiltype_dict[i] = soil_type_label_encoder.inverse_transform([i])[0]
print(soiltype_dict)
```

```
{0: 'Barley', 1: 'Cotton', 2: 'Ground Nuts', 3: 'Maize', 4: 'Millets', 5: 'Oil seeds',
6: 'Paddy', 7: 'Pulses', 8: 'Sugarcane', 9: 'Tobacco', 10: 'Wheat'}
{0: 'Black', 1: 'Clayey', 2: 'Loamy', 3: 'Red', 4: 'Sandy'}
```

In [71]:
```
fertname_label_encoder = LabelEncoder()
df["fertilizername"] = fertname_label_encoder.fit_transform(df["fertilizername"])
```

In [72]:
```
fertname_dict = {}
for i in range(len(df["fertilizername"].unique())):
    fertname_dict[i] = fertname_label_encoder.inverse_transform([i])[0]
print(fertname_dict)
```

```
{0: '10-26-26', 1: '14-35-14', 2: '17-17-17', 3: '20-20', 4: '28-28', 5: 'DAP', 6: 'Ure
a'}
```

In [73]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [74]:
```
df.head()
```

Out[74]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous	fertilizername
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 [75]:
```
X = df.drop(['fertilizername'] , axis = 1)
y = df['fertilizername']
```

In [76]:
```
X
```

Out[76]:

	temparature	humidity	moisture	soiltype	croptype	nitrogen	potassium	phosphorous
0	26	52	38	4	3	37	0	0
1	29	52	45	2	8	12	0	36
2	34	65	62	0	1	7	9	30
3	32	62	34	3	9	22	0	20
4	28	54	46	1	6	35	0	0

...	...	...	...	...	...	...	...	...
94	25	50	32	1	7	24	0	19
95	30	60	27	3	9	4	17	17
96	38	72	51	2	10	39	0	0
97	36	60	43	4	4	15	0	41
98	29	58	57	0	8	12	0	10

99 rows × 8 columns

In [77]:
```
y
```

Out[77]:
```
0     6
1     5
2     1
3     4
4     6
     ..
94    4
95    0
96    6
97    5
98    3
Name: fertilizername, Length: 99, dtype: int32
```

In [78]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

In [79]:
```
import imblearn
from imblearn.over_sampling import SMOTE
from collections import Counter

from sklearn.pipeline import make_pipeline

from sklearn.neighbors import  KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import xgboost
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, confusion_matrix
import pickle
```

In [80]:
```
counter = Counter(y)
counter
```

Out[80]:
```
Counter({6: 22, 5: 18, 1: 14, 4: 17, 2: 7, 3: 14, 0: 7})
```

In [81]:
```
upsample = SMOTE()
X, y = upsample.fit_resample(X, y)
counter = Counter(y)
print(counter)
```

```
Counter({6: 22, 5: 22, 1: 22, 4: 22, 2: 22, 3: 22, 0: 22})
```

In [82]:
```
X_train, X_test, y_train, y_test = train_test_split(X.values, y, test_size = 0.2, random
print(f"Train Data: {X_train.shape}, {y_train.shape}")
print(f"Train Data: {X_test.shape}, {y_test.shape}")
```

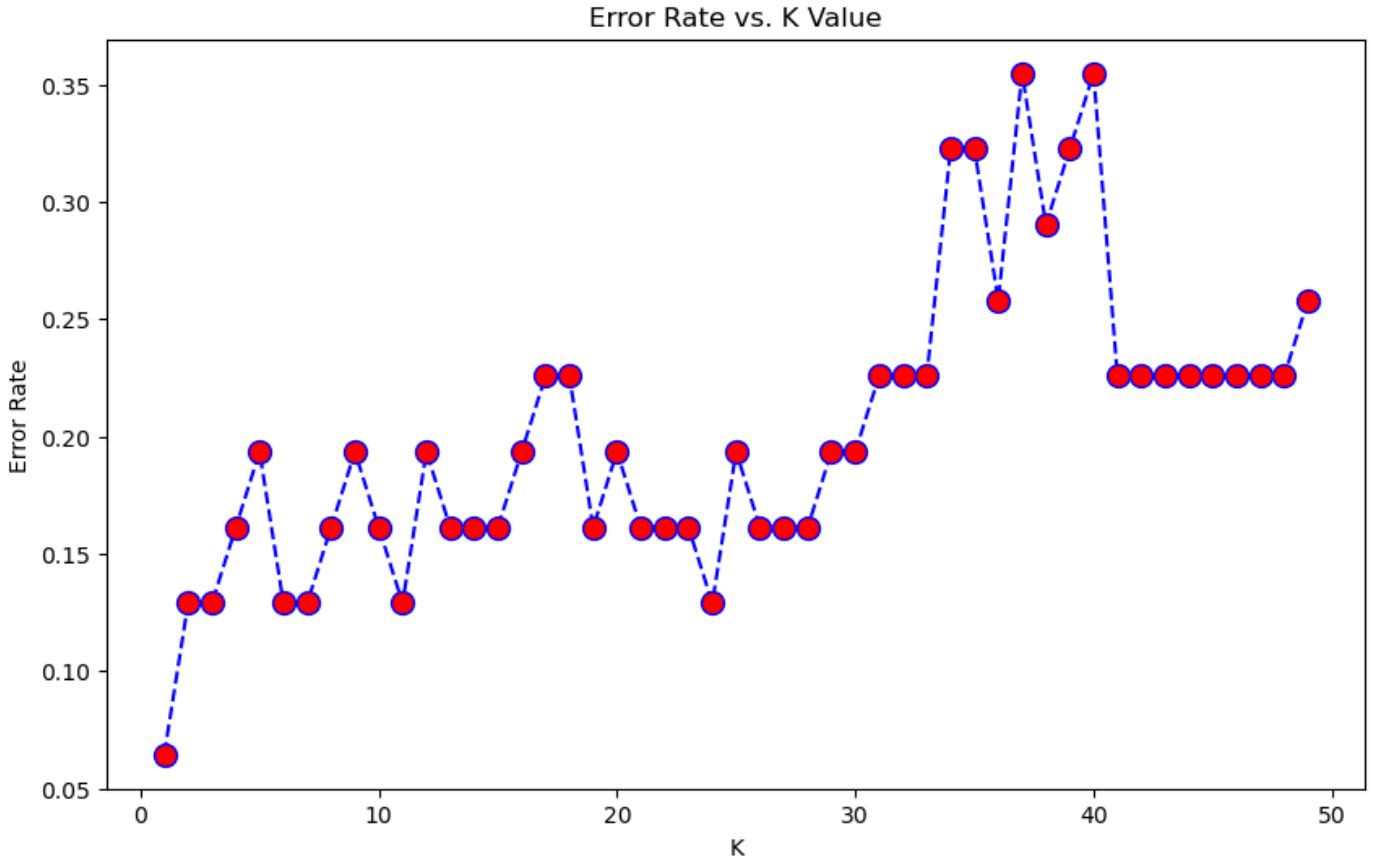
```
Train Data: (123, 8), (123,)
Train Data: (31, 8), (31,)
```

KNN Classifier

In [83]:
```
error_rate = []
for i in range(1, 50):
    pipeline = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors = i))
    pipeline.fit(X_train, y_train)
    predictions = pipeline.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    print(f"Accuracy at k = {i} is {accuracy}")
    error_rate.append(np.mean(predictions != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,50),error_rate,color='blue', linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
print("Minimum error:-",min(error_rate),"at K =",error_rate.index(min(error_rate))+1)
```

```
Accuracy at k = 1 is 0.9354838709677419
Accuracy at k = 2 is 0.8709677419354839
Accuracy at k = 3 is 0.8709677419354839
Accuracy at k = 4 is 0.8387096774193549
Accuracy at k = 5 is 0.8064516129032258
Accuracy at k = 6 is 0.8709677419354839
Accuracy at k = 7 is 0.8709677419354839
Accuracy at k = 8 is 0.8387096774193549
Accuracy at k = 9 is 0.8064516129032258
Accuracy at k = 10 is 0.8387096774193549
Accuracy at k = 11 is 0.8709677419354839
Accuracy at k = 12 is 0.8064516129032258
Accuracy at k = 13 is 0.8387096774193549
Accuracy at k = 14 is 0.8387096774193549
Accuracy at k = 15 is 0.8387096774193549
Accuracy at k = 16 is 0.8064516129032258
Accuracy at k = 17 is 0.7741935483870968
Accuracy at k = 18 is 0.7741935483870968
Accuracy at k = 19 is 0.8387096774193549
Accuracy at k = 20 is 0.8064516129032258
Accuracy at k = 21 is 0.8387096774193549
Accuracy at k = 22 is 0.8387096774193549
Accuracy at k = 23 is 0.8387096774193549
Accuracy at k = 24 is 0.8709677419354839
Accuracy at k = 25 is 0.8064516129032258
Accuracy at k = 26 is 0.8387096774193549
Accuracy at k = 27 is 0.8387096774193549
Accuracy at k = 28 is 0.8387096774193549
Accuracy at k = 29 is 0.8064516129032258
Accuracy at k = 30 is 0.8064516129032258
Accuracy at k = 31 is 0.7741935483870968
Accuracy at k = 32 is 0.7741935483870968
Accuracy at k = 33 is 0.7741935483870968
Accuracy at k = 34 is 0.6774193548387096
Accuracy at k = 35 is 0.6774193548387096
Accuracy at k = 36 is 0.7419354838709677
Accuracy at k = 37 is 0.6451612903225806
Accuracy at k = 38 is 0.7096774193548387
Accuracy at k = 39 is 0.6774193548387096
Accuracy at k = 40 is 0.6451612903225806
Accuracy at k = 41 is 0.7741935483870968
Accuracy at k = 42 is 0.7741935483870968
Accuracy at k = 43 is 0.7741935483870968
Accuracy at k = 44 is 0.7741935483870968
Accuracy at k = 45 is 0.7741935483870968
Accuracy at k = 46 is 0.7741935483870968
```

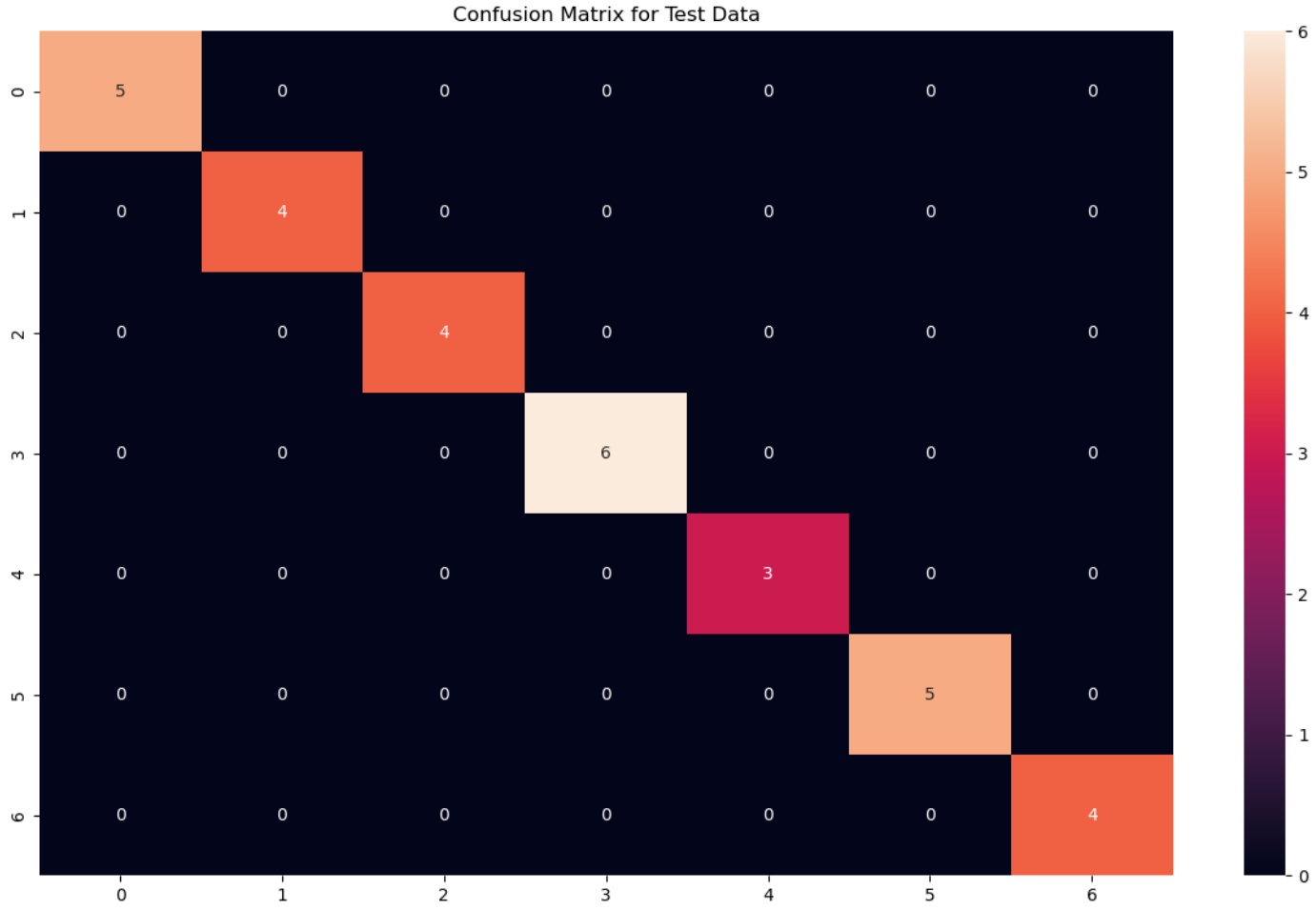
```
Accuracy at k = 47 is 0.7741935483870968
Accuracy at k = 48 is 0.7741935483870968
Accuracy at k = 49 is 0.7419354838709677
Minimum error:- 0.06451612903225806 at K = 1
```



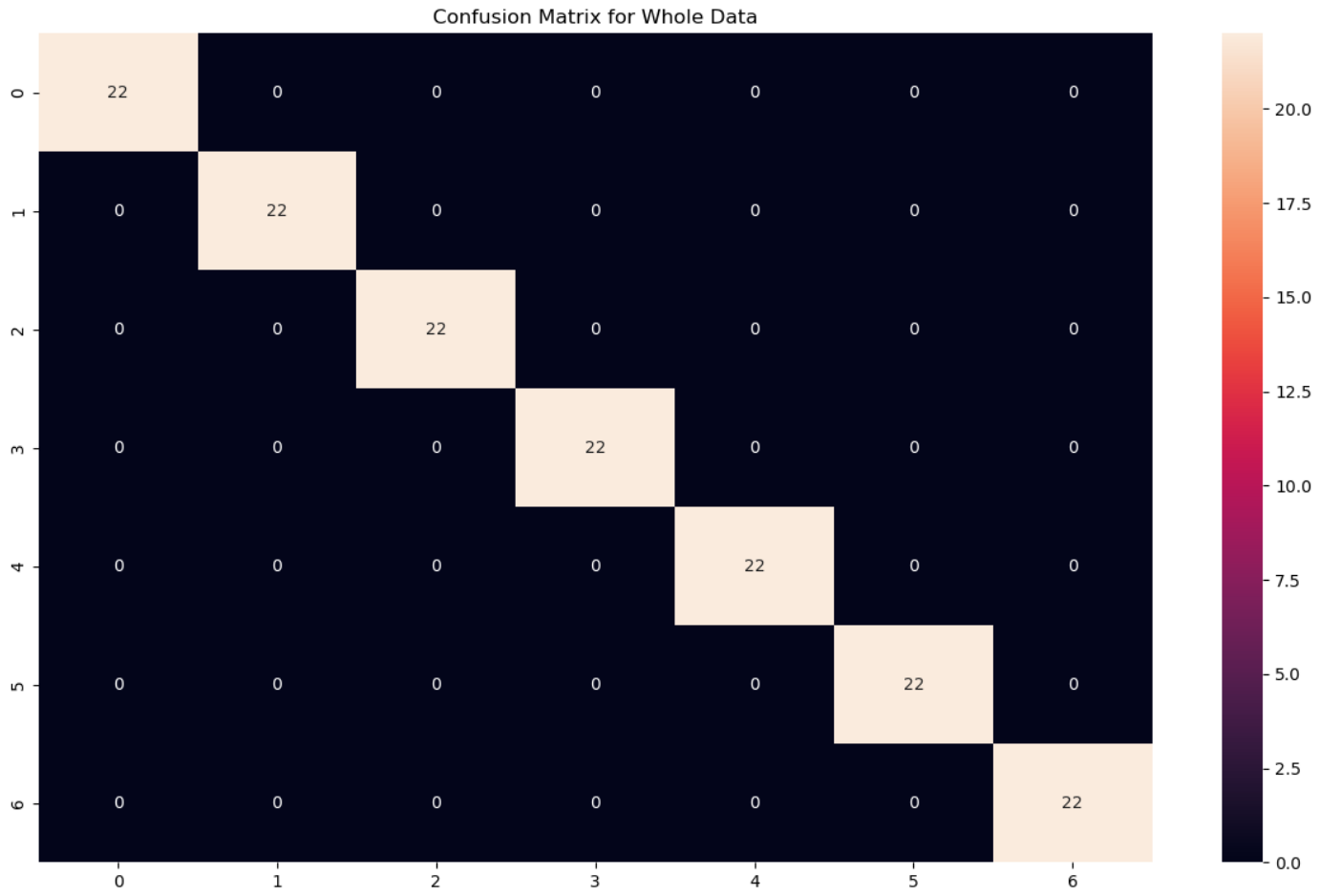
SVM Classifier

In [84]:
```
svm_pipeline = make_pipeline(StandardScaler(), SVC(probability=True))
svm_pipeline.fit(X_train, y_train)

# Accuray On Test Data
predictions = svm_pipeline.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy on Test Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y_test, predictions), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()

print()

# Accuray On Whole Data
predictions = svm_pipeline.predict(X.values)
accuracy = accuracy_score(y, predictions)
print(f"Accuracy on Whole Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y, predictions), annot = True)
plt.title("Confusion Matrix for Whole Data")
plt.show()
```

```
Accuracy on Test Data: 100.0%
```



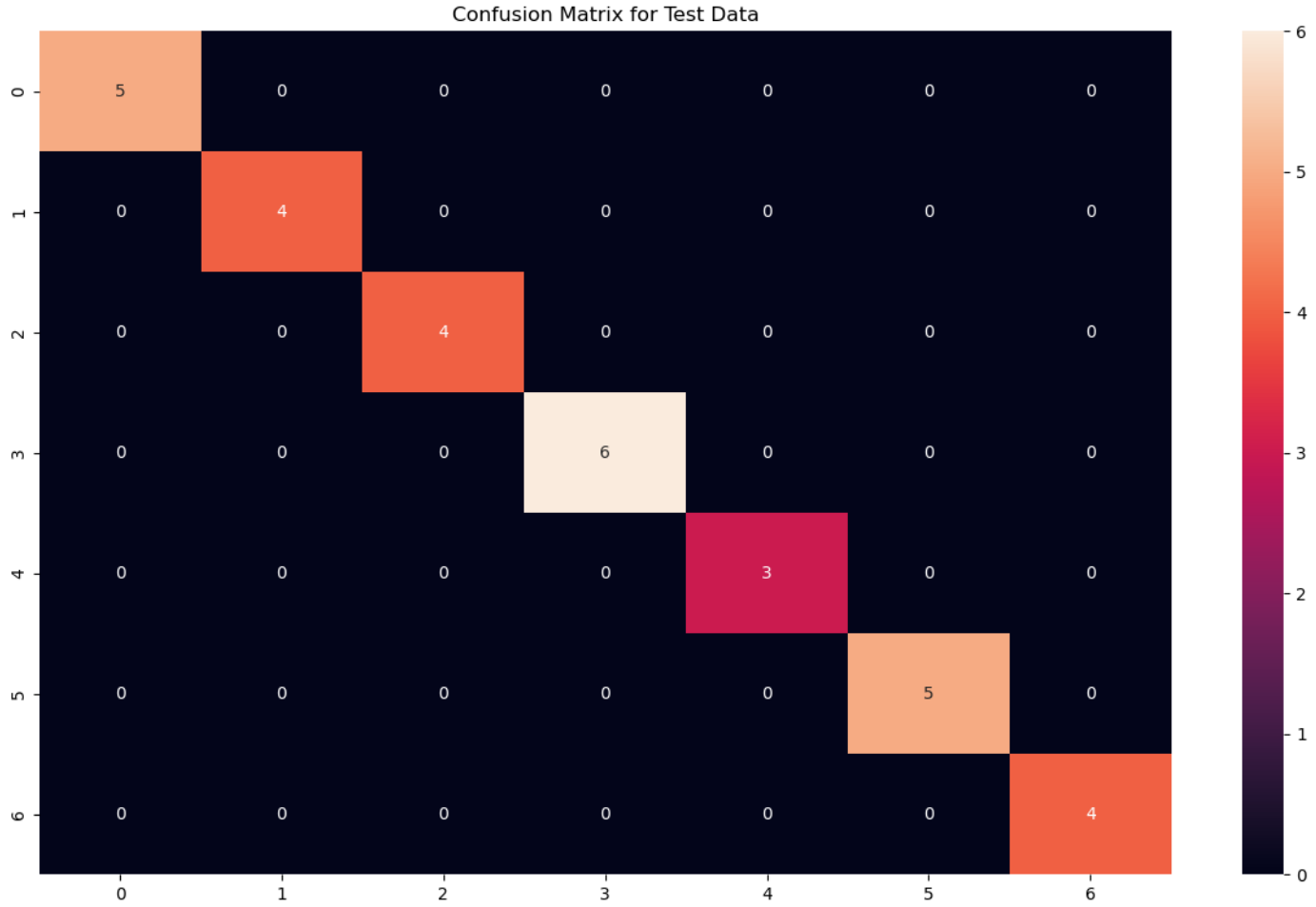

```
Accuracy on Whole Data: 100.0%
```



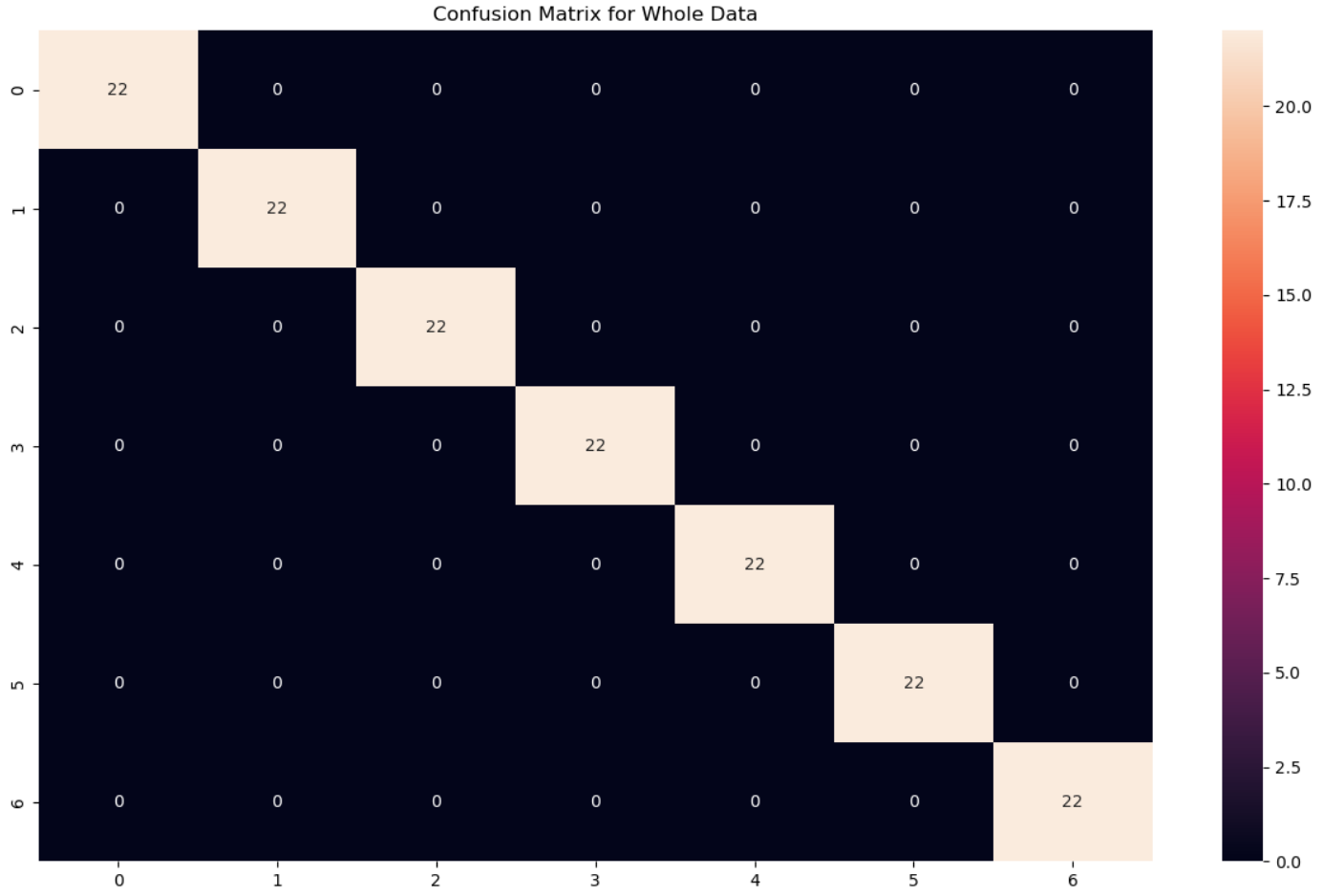
Random_Forest_Classifier

In [85]:
```
rf_pipeline = make_pipeline(StandardScaler(), RandomForestClassifier(random_state = 18))
rf_pipeline.fit(X_train, y_train)

# Accuray On Test Data
predictions = rf_pipeline.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy on Test Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y_test, predictions), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()

print()

# Accuray On Whole Data
predictions = rf_pipeline.predict(X.values)
accuracy = accuracy_score(y, predictions)
print(f"Accuracy on Whole Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y, predictions), annot = True)
plt.title("Confusion Matrix for Whole Data")
plt.show()
```

```
Accuracy on Test Data: 100.0%
```



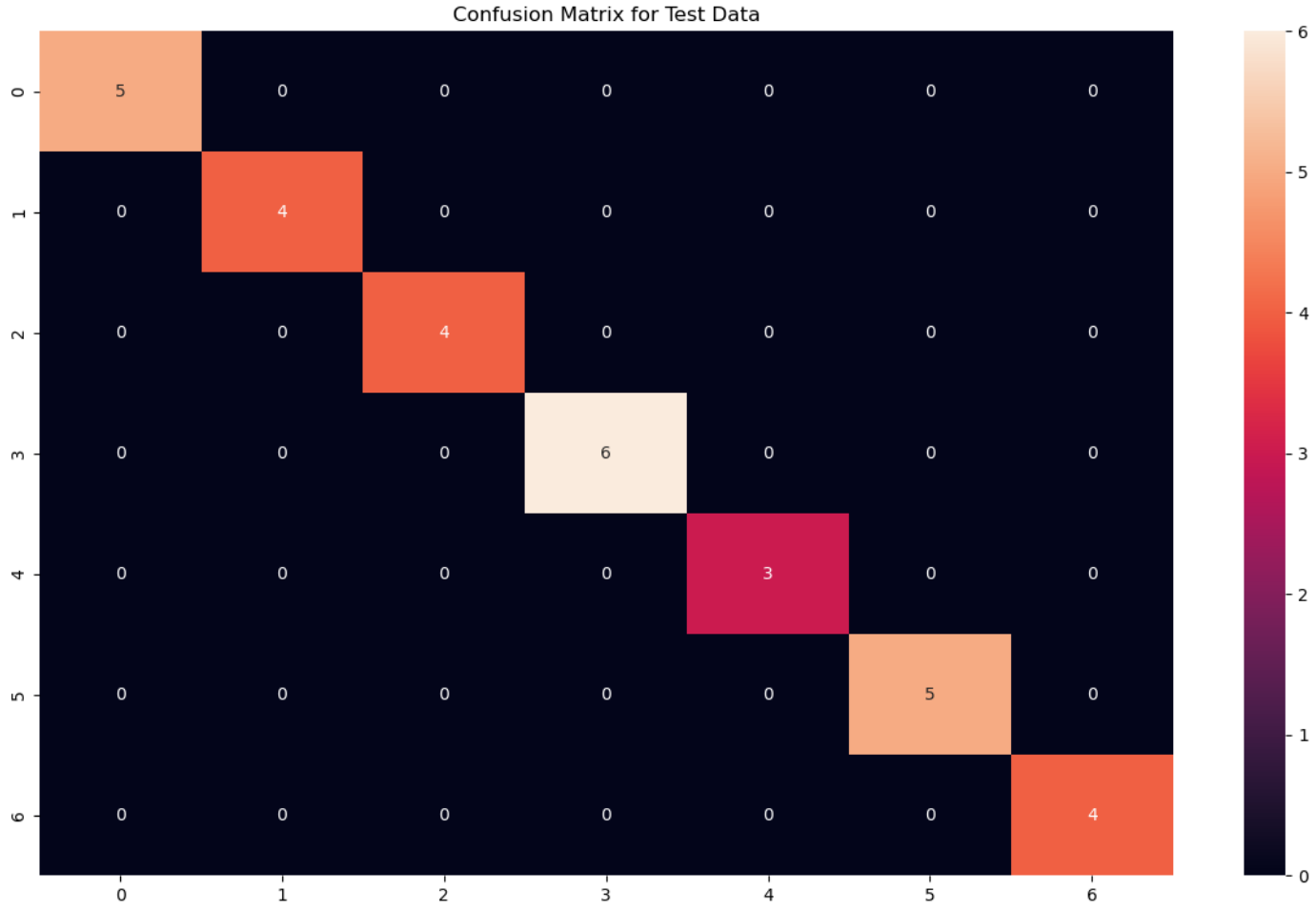

```
Accuracy on Whole Data: 100.0%
```



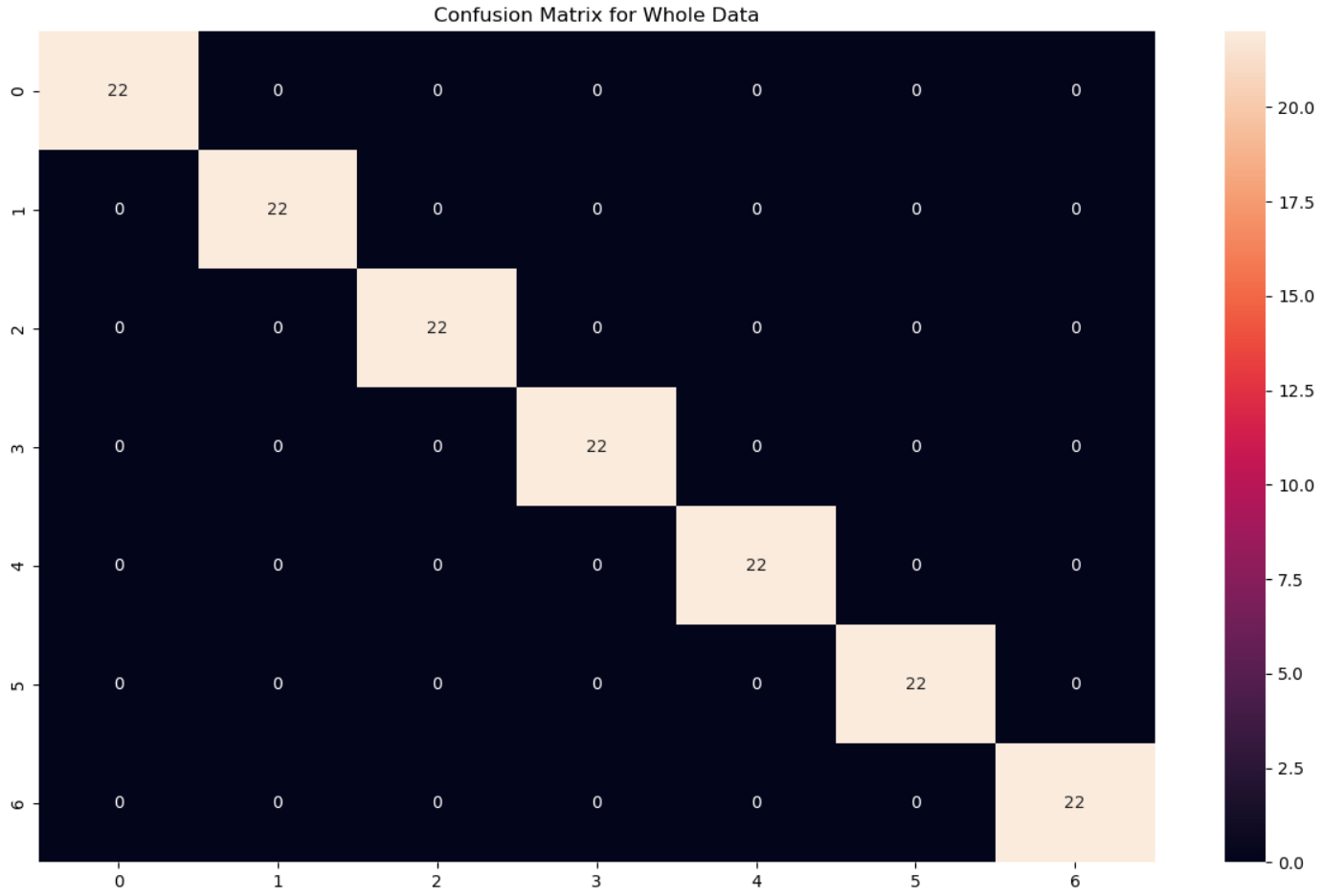
XG_Boost Classifier

In [86]:
```
xgb_pipeline = make_pipeline(StandardScaler(), XGBClassifier(random_state = 18))
xgb_pipeline.fit(X_train, y_train)

# Accuray On Test Data
predictions = xgb_pipeline.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy on Test Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y_test, predictions), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()

print()

# Accuray On Whole Data
predictions = xgb_pipeline.predict(X.values)
accuracy = accuracy_score(y, predictions)
print(f"Accuracy on Whole Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y, predictions), annot = True)
plt.title("Confusion Matrix for Whole Data")
plt.show()
```

```
Accuracy on Test Data: 100.0%
```




```
Accuracy on Whole Data: 100.0%
```



Final Pickling the model

In [87]:
```
pickle.dump(rf_pipeline, open('model.pkl', 'wb'))
```

In [88]:
```
pickled_model = pickle.load(open('model.pkl', 'rb'))
```

In [89]:
```
# Final use the randomforest classifier for the prediction
```

In [ ]:
```

```