In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:

```
fraud_check = pd.read_csv('Fraud_check.csv')
fraud_check
```

Out[3]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	NO	Single	68833	50047	10	YES
1	YES	Divorced	33700	134075	18	YES
2	NO	Married	36925	160205	30	YES
3	YES	Single	50190	193264	15	YES
4	NO	Married	81002	27533	28	NO
...	...	...	...	...	...	...
595	YES	Divorced	76340	39492	7	YES
596	YES	Divorced	69967	55369	2	YES
597	NO	Divorced	47334	154058	0	YES
598	YES	Married	98592	180083	17	NO
599	NO	Divorced	96519	158137	16	NO

600 rows × 6 columns

In [4]:

```
fraud_check.head()
```

Out[4]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	NO	Single	68833	50047	10	YES
1	YES	Divorced	33700	134075	18	YES
2	NO	Married	36925	160205	30	YES
3	YES	Single	50190	193264	15	YES
4	NO	Married	81002	27533	28	NO

In [5]:

```
fraud_check.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Undergrad        600 non-null    object
 1   Marital.Status   600 non-null    object
 2   Taxable.Income   600 non-null    int64 
 3   City.Population  600 non-null    int64 
 4   Work.Experience  600 non-null    int64 
 5   Urban            600 non-null    object
dtypes: int64(3), object(3)
memory usage: 28.2+ KB
```

In [7]:

```
fraud_check.shape
```

Out[7]:

```
(600, 6)
```

In [8]:

```
fraud_check.isna().sum()
```

Out[8]:

```
Undergrad          0
Marital.Status     0
Taxable.Income     0
City.Population    0
Work.Experience    0
Urban              0
dtype: int64
```

In [9]:

```
fraud_check.describe(include='all')
```

Out[9]:

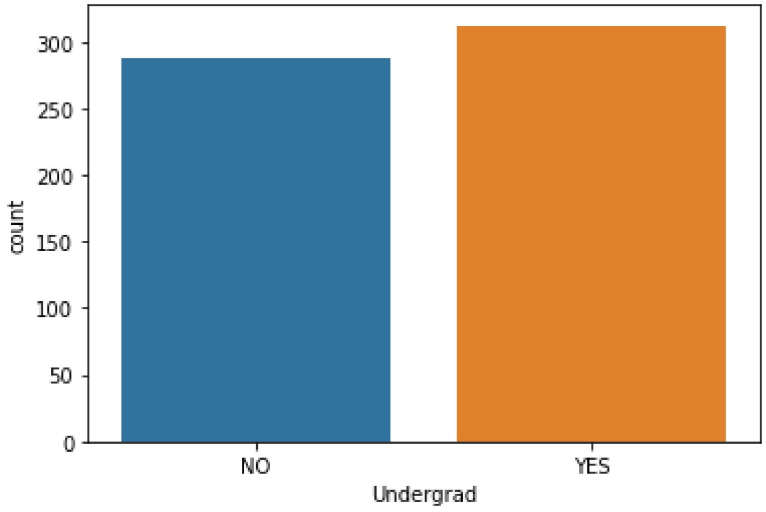
	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
count	600	600	600.000000	600.000000	600.000000	600
unique	2	3	NaN	NaN	NaN	2
top	YES	Single	NaN	NaN	NaN	YES
freq	312	217	NaN	NaN	NaN	302
mean	NaN	NaN	55208.375000	108747.368333	15.558333	NaN
std	NaN	NaN	26204.827597	49850.075134	8.842147	NaN
min	NaN	NaN	10003.000000	25779.000000	0.000000	NaN
25%	NaN	NaN	32871.500000	66966.750000	8.000000	NaN
50%	NaN	NaN	55074.500000	106493.500000	15.000000	NaN
75%	NaN	NaN	78611.750000	150114.250000	24.000000	NaN
max	NaN	NaN	99619.000000	199778.000000	30.000000	NaN

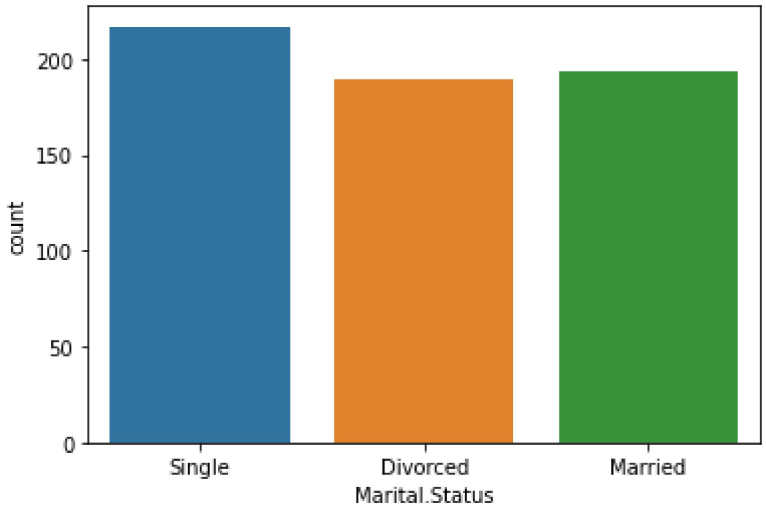
In [11]:

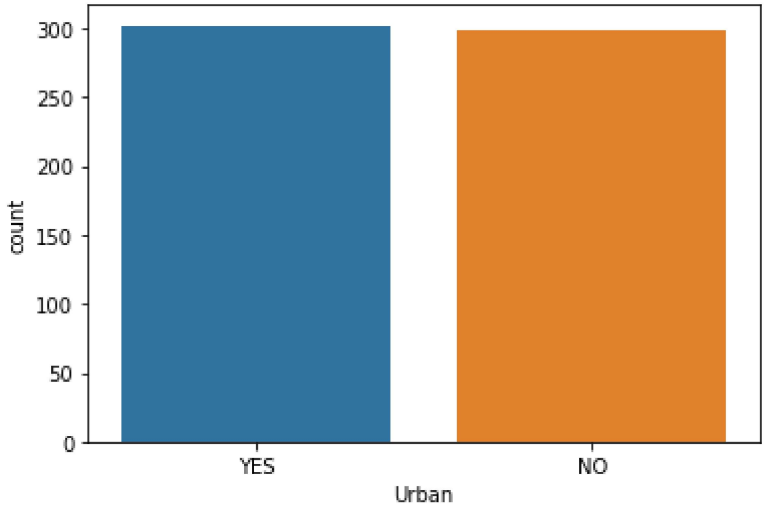
```
import warnings
warnings.filterwarnings('ignore')
```

In [12]:

```
# checking count of categories for categorial columns
import seaborn as sns

sns.countplot(fraud_check['Undergrad'])
plt.show()

sns.countplot(fraud_check['Marital.Status'])
plt.show()

sns.countplot(fraud_check['Urban'])
plt.show()
```

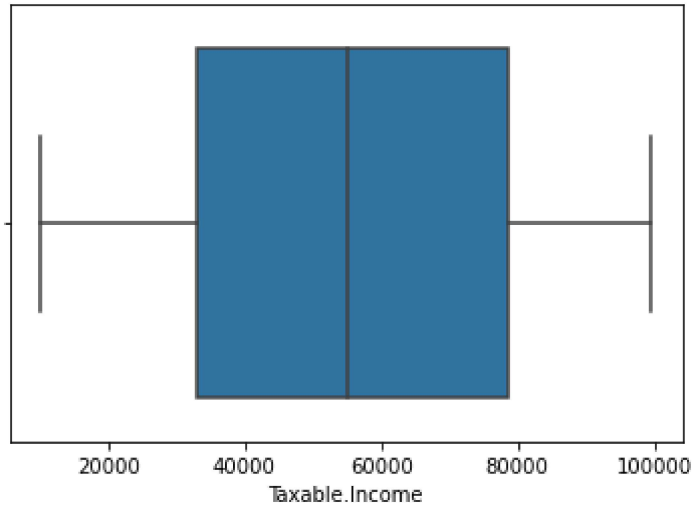


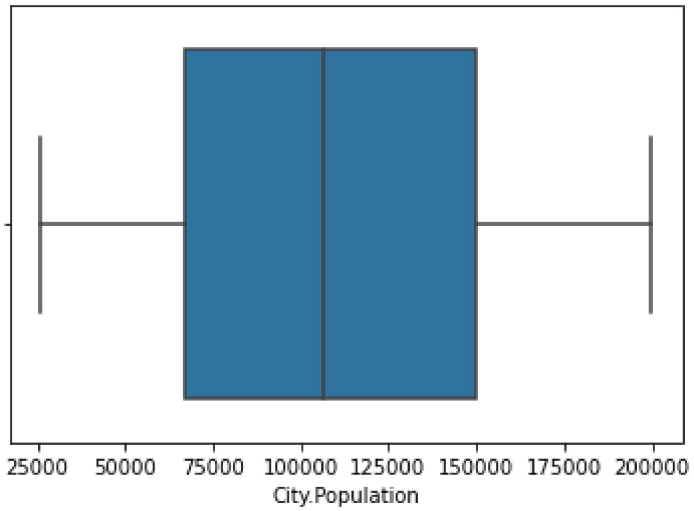


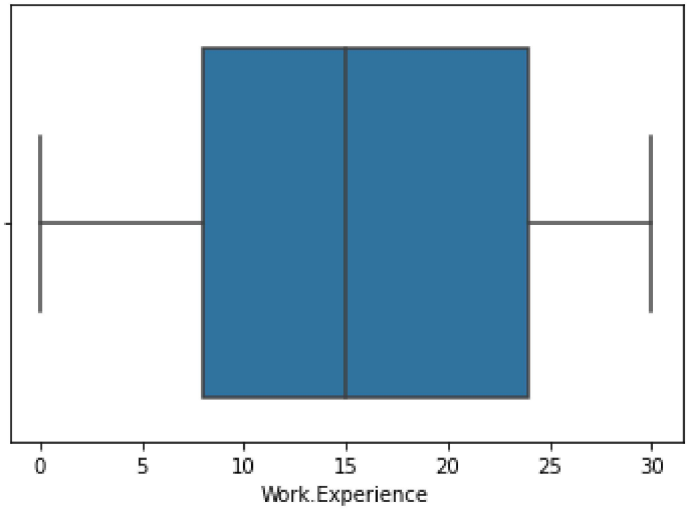

300
250
200
150
100
50
0
count
YES
NO
Urban

In [13]:

```
#checking for outliers in numerical data

sns.boxplot(fraud_check['Taxable.Income'])
plt.show()

sns.boxplot(fraud_check['City.Population'])
plt.show()

sns.boxplot(fraud_check['Work.Experience'])
plt.show()
```







In [15]:

```
round(fraud_check.corr(),4)
```

Out[15]:

	Taxable.Income	City.Population	Work.Experience
Taxable.Income	1.0000	-0.0644	-0.0018
City.Population	-0.0644	1.0000	0.0131
Work.Experience	-0.0018	0.0131	1.0000

In [16]:

```
#converting taxable income <= 30000 as 'Risky' and others are 'Good'.
fraud_check['Taxable.Income'] = pd.cut(x=fraud_check['Taxable.Income'], bins =[10002,30000,
fraud_check
```

Out[16]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	NO	Single	Good	50047	10	YES
1	YES	Divorced	Good	134075	18	YES
2	NO	Married	Good	160205	30	YES
3	YES	Single	Good	193264	15	YES
4	NO	Married	Good	27533	28	NO
...	...	...	...	...	...	...
595	YES	Divorced	Good	39492	7	YES
596	YES	Divorced	Good	55369	2	YES
597	NO	Divorced	Good	154058	0	YES
598	YES	Married	Good	180083	17	NO
599	NO	Divorced	Good	158137	16	NO

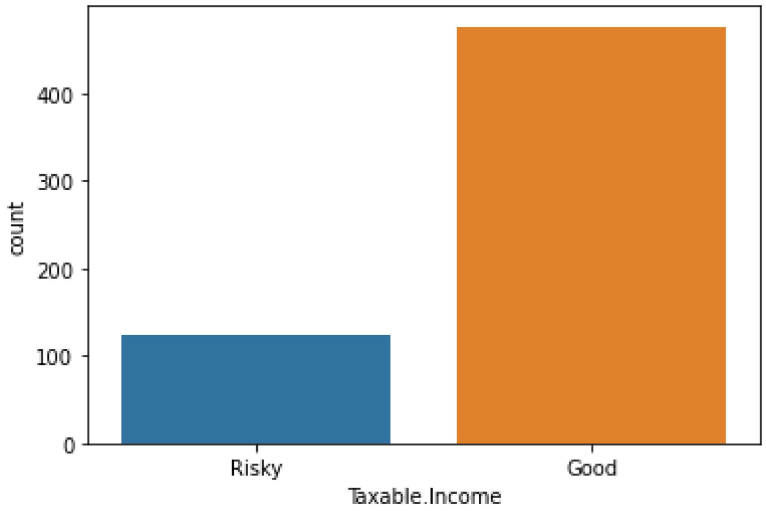
600 rows × 6 columns

In [17]:

```
sns.countplot(fraud_check['Taxable.Income'])
fraud_check['Taxable.Income'].value_counts()
```

Out[17]:

```
Good     476
Risky    124
Name: Taxable.Income, dtype: int64
```



In [21]:

```
# encoding categorial data
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
fraud_check['Undergrad'] = le.fit_transform(fraud_check['Undergrad'])
fraud_check['Marital.Status'] = le.fit_transform(fraud_check['Marital.Status'])
fraud_check['Taxable.Income'] = le.fit_transform(fraud_check['Taxable.Income'])
fraud_check['Work.Experience'] = le.fit_transform(fraud_check['Work.Experience'])
fraud_check['Urban'] = le.fit_transform(fraud_check['Urban'])
fraud_check['City.Population'] = le.fit_transform(fraud_check['City.Population'])

fraud_check
```

Out[21]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	0	2	0	84	10	1
1	1	0	0	398	18	1
2	0	1	0	481	30	1
3	1	2	0	574	15	1
4	0	1	0	4	28	0
...	...	...	...	...	...	...
595	1	0	0	55	7	1
596	1	0	0	107	2	1
597	0	0	0	459	0	1
598	1	1	0	533	17	0
599	0	0	0	477	16	0

600 rows × 6 columns

In [22]:

```
X = fraud_check.drop(labels = 'Taxable.Income', axis=1)
y = fraud_check[['Taxable.Income']]
```

In [23]:

```
x
```

Out[23]:

	Undergrad	Marital.Status	City.Population	Work.Experience	Urban
0	0	2	84	10	1
1	1	0	398	18	1
2	0	1	481	30	1
3	1	2	574	15	1
4	0	1	4	28	0
...	...	...	...	...	...
595	1	0	55	7	1
596	1	0	107	2	1
597	0	0	459	0	1
598	1	1	533	17	0
599	0	0	477	16	0

600 rows × 5 columns

In [24]:

```
y
```

Out[24]:

	Taxable.Income
0	0
1	0
2	0
3	0
4	0
...	...
595	0
596	0
597	0
598	0
599	0

600 rows × 1 columns

In [25]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20, random_state=12)
```

In [26]:

```
X_train.shape, y_train.shape
```

Out[26]:

```
((480, 5), (480, 1))
```

In [27]:

```
X_test.shape , y_test.shape
```

Out[27]:

```
((120, 5), (120, 1))
```

Building model based on C5.0 Algorithm

In [41]:

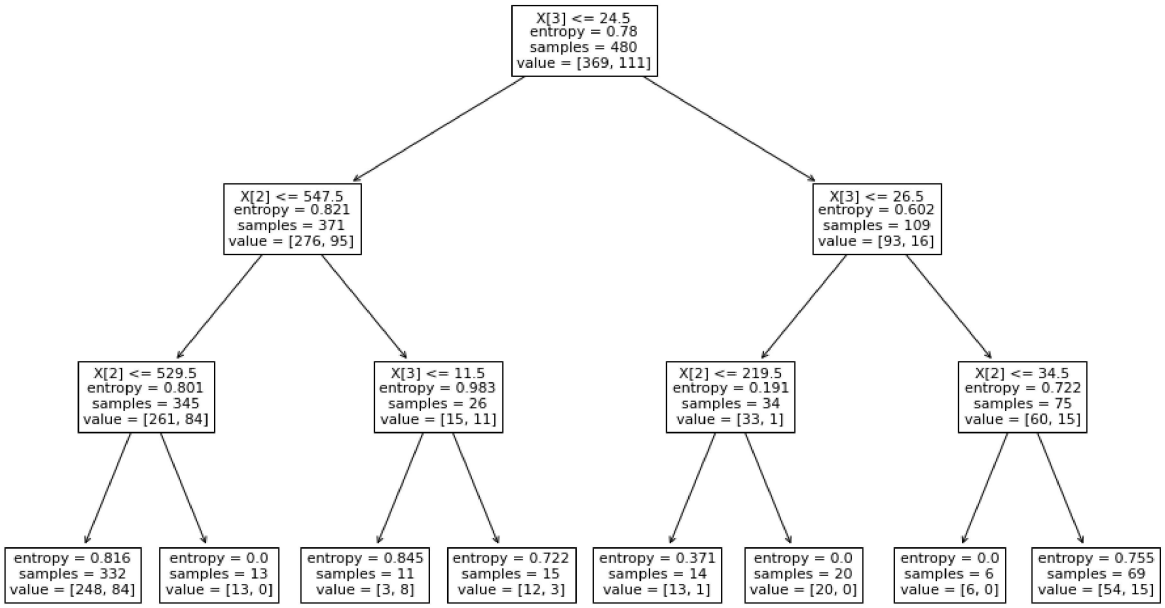
```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(criterion = 'entropy' , max_depth =3)
dt_model.fit(X_train,y_train)
```

Out[41]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [42]:

```
from sklearn import tree
plt.figure(figsize=(16,10))
tree.plot_tree(dt_model);
```



In [43]:

```
y_train_pred = dt_model.predict(X_train)
y_train_pred
pd.Series(y_train_pred).value_counts()
```

Out[43]:

```
0    469
1     11
dtype: int64
```

In [44]:

```
y_test_pred = dt_model.predict(X_test)
y_test_pred
pd.Series(y_test_pred).value_counts()
```

Out[44]:

```
0    115
1      5
dtype: int64
```

In [45]:

```
dt_model.score(X_test,y_test)
```

Out[45]:

```
0.85
```

In [ ]:

```
```

Building model based on CART Algorithm

In [46]:

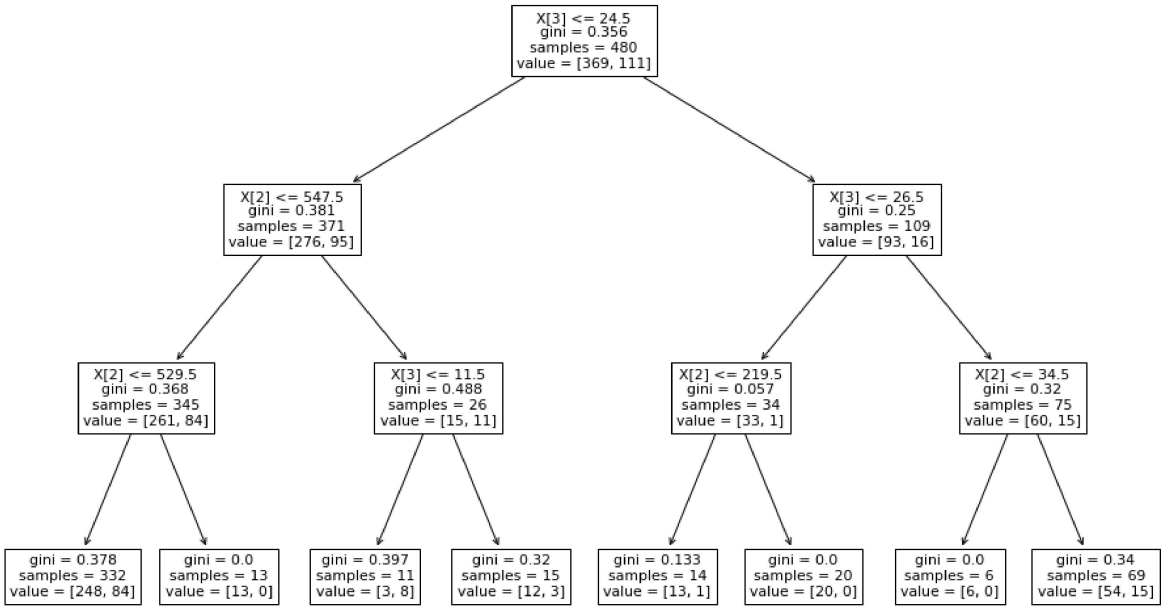
```
dt_model_cart = DecisionTreeClassifier(criterion = 'gini' , max_depth =3)
dt_model_cart.fit(X_train,y_train)
```

Out[46]:

```
DecisionTreeClassifier(max_depth=3)
```

In [47]:

```
from sklearn import tree
plt.figure(figsize=(16,10))

tree.plot_tree(dt_model_cart);
```



In [48]:

```
y_train_pred = dt_model_cart.predict(X_train)
y_train_pred
pd.Series(y_train_pred).value_counts()
```

Out[48]:

```
0    469
1     11
dtype: int64
```

In [49]:

```
y_test_pred = dt_model_cart.predict(X_test)
y_test_pred
pd.Series(y_test_pred).value_counts()
```

Out[49]:

```
0    115
1      5
dtype: int64
```

In [50]:

```
dt_model_cart.score(X_test,y_test)
```

Out[50]:

```
0.85
```

In [ ]:

```

```