In [1]:
```
import numpy as np #used for numerical calculations
import pandas as pd #reading datasets and manupulating datasets
import matplotlib.pyplot as plt #for visualizing the data
import seaborn as sns #for visualizing the data
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
dataset = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
dataset.head()
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Rel
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

In [3]:
```
dataset.shape
```

Out[3]:
```
(1470, 35)
```

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

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

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

Out[5]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvol
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.

8 rows × 26 columns

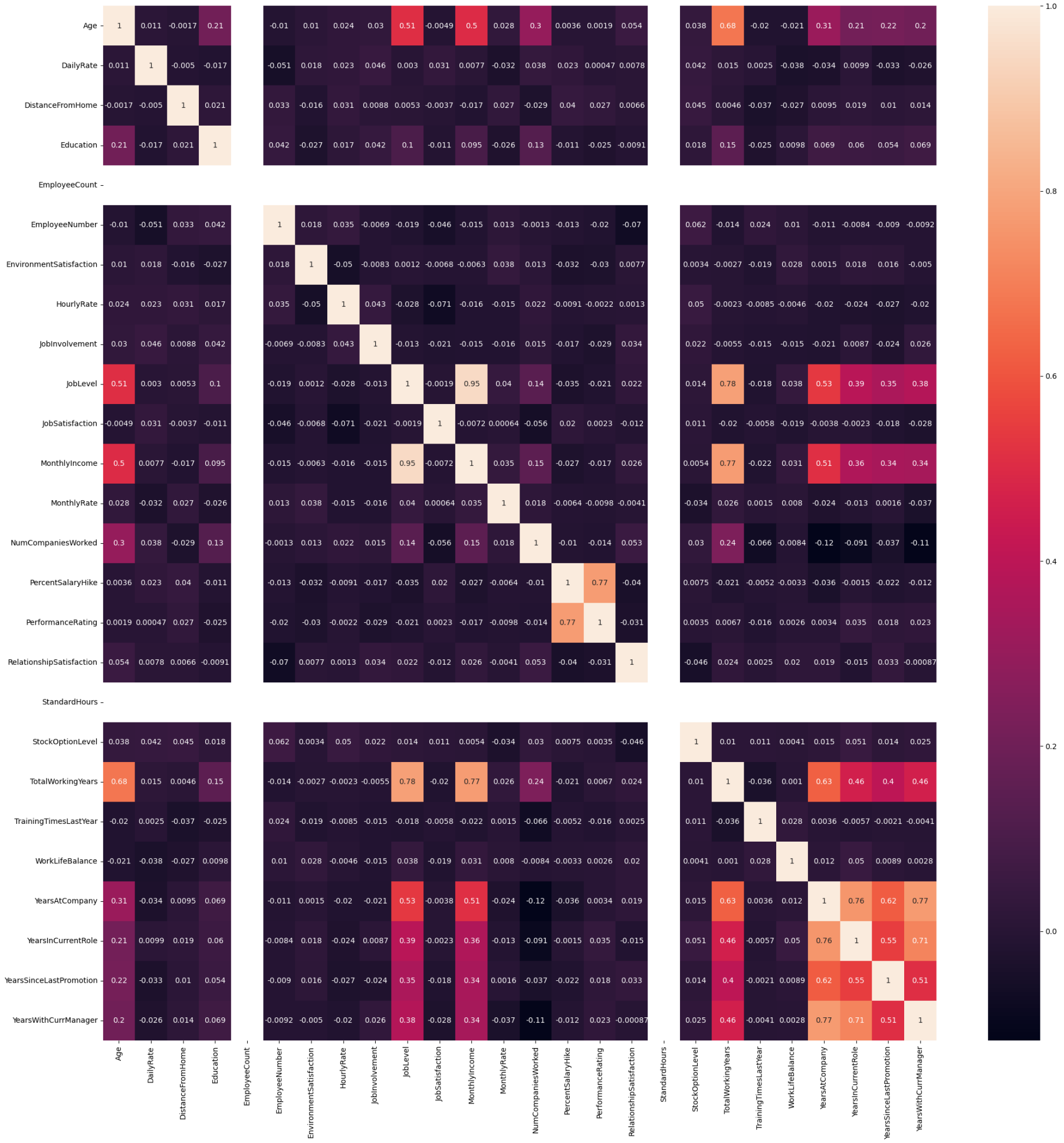
In [6]:
```
corr = dataset.corr()
```

```
C:\Users\ABILASH\AppData\Local\Temp\ipykernel_28756\350376347.py:1: FutureWarning: The default value of numeric_only in
DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the v
alue of numeric_only to silence this warning.
  corr = dataset.corr()
```

In [7]:
```
plt.subplots(figsize=(25,25))
```

In [7]:
```
plt.subplots(figsize=(25,25))
sns.heatmap(corr,annot=True)
```

Out[7]: <Axes: >



In [8]:
```
dataset.Attrition.value_counts()
```

Out[8]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

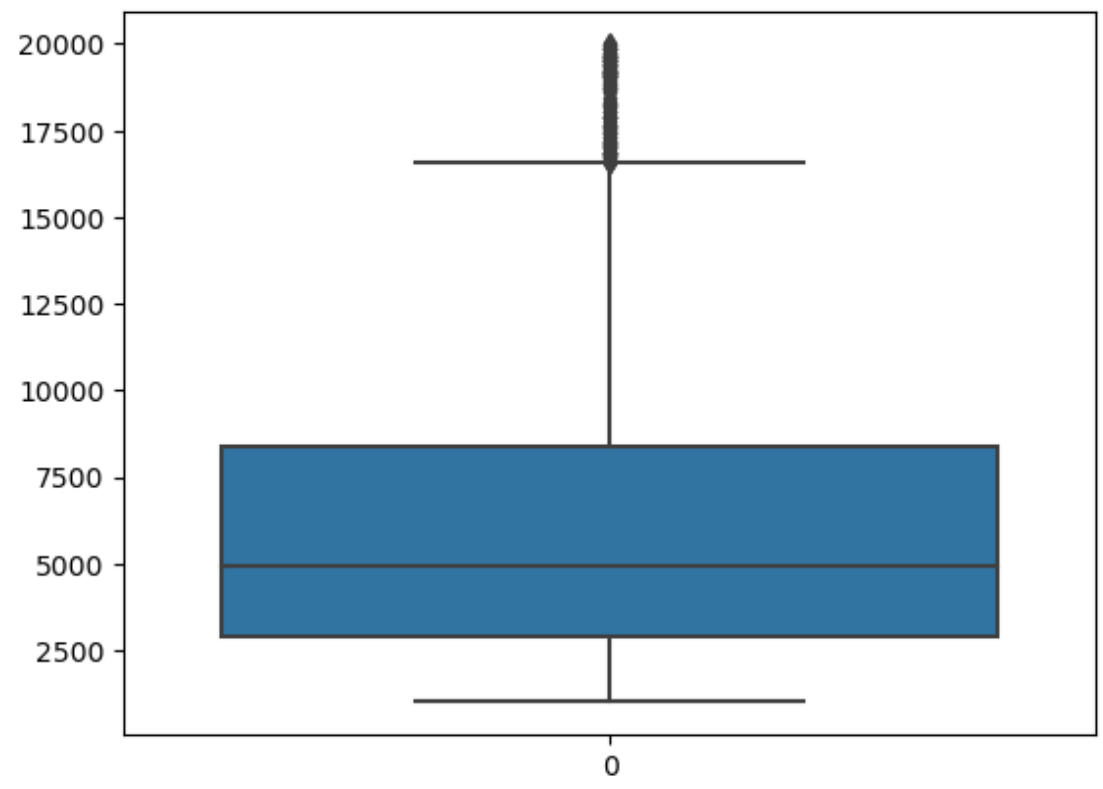
In [9]:
```
dataset.isnull().any()
```

In [9]:
```
dataset.isnull().any()
```

Out[9]:
```
Age                       False
Attrition                 False
BusinessTravel            False
DailyRate                 False
Department                False
DistanceFromHome          False
Education                 False
EducationField            False
EmployeeCount             False
EmployeeNumber            False
EnvironmentSatisfaction   False
Gender                    False
HourlyRate                False
JobInvolvement            False
JobLevel                  False
JobRole                   False
JobSatisfaction           False
MaritalStatus             False
MonthlyIncome             False
MonthlyRate               False
NumCompaniesWorked        False
Over18                    False
OverTime                  False
PercentSalaryHike         False
PerformanceRating         False
RelationshipSatisfaction  False
StandardHours             False
StockOptionLevel          False
TotalWorkingYears         False
TrainingTimesLastYear     False
WorkLifeBalance           False
YearsAtCompany            False
YearsInCurrentRole        False
YearsSinceLastPromotion   False
YearsWithCurrManager      False
dtype: bool
```

In [10]:
```
sns.boxplot(dataset.MonthlyIncome)
```

Out[10]:
```
<Axes: >
```



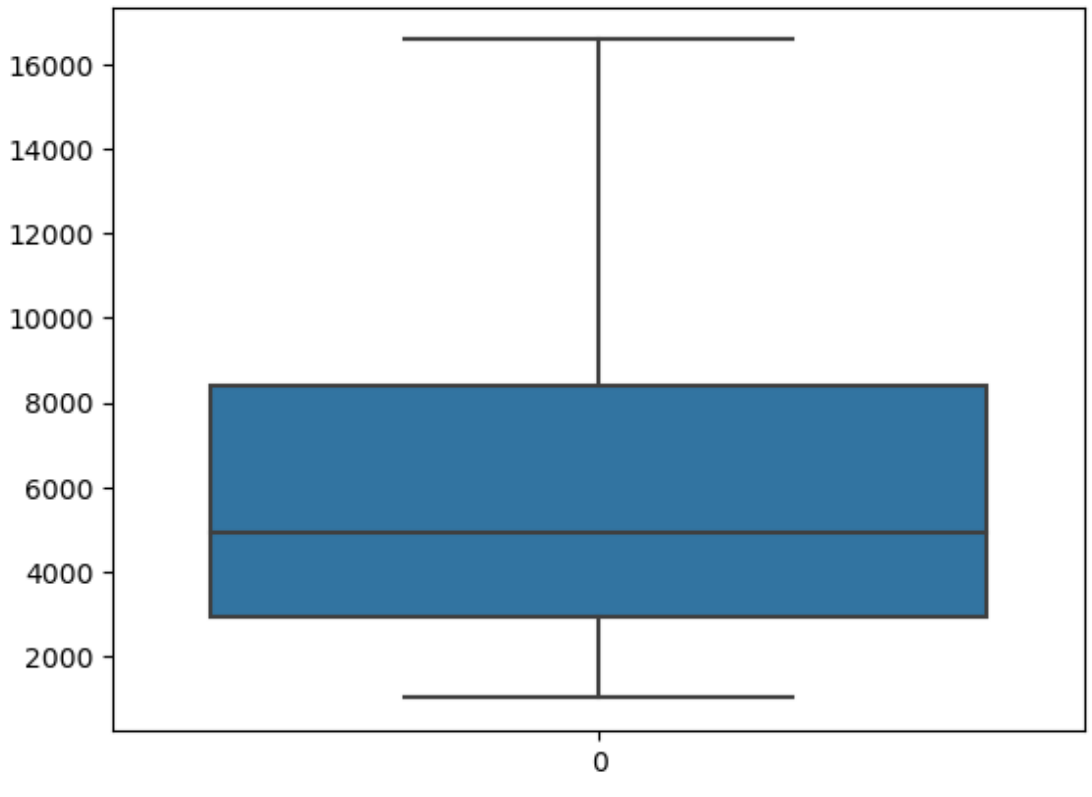
In [11]:
```
q1 = dataset.MonthlyIncome.quantile(0.25)
q3 = dataset.MonthlyIncome.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [12]:
```
dataset['MonthlyIncome'] = np.where(dataset['MonthlyIncome'] > upperlimit , upperlimit,dataset['MonthlyIncome'])
dataset['MonthlyIncome'] = np.where(dataset['MonthlyIncome'] < lowerlimit, lowerlimit, dataset['MonthlyIncome'])
```

In [13]:
```
sns.boxplot(dataset.MonthlyIncome)
```

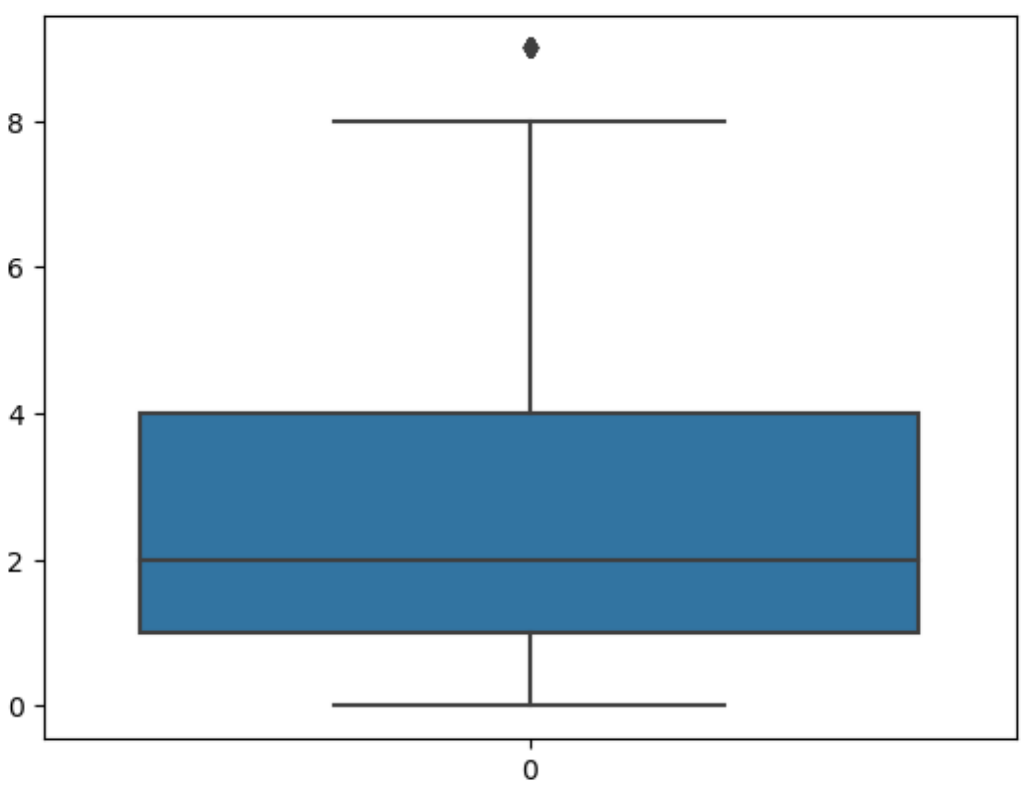
In [13]:
```
sns.boxplot(dataset.MonthlyIncome)
```

Out[13]: <Axes: >



In [14]:
```
sns.boxplot(dataset.NumCompaniesWorked)
```

Out[14]: <Axes: >



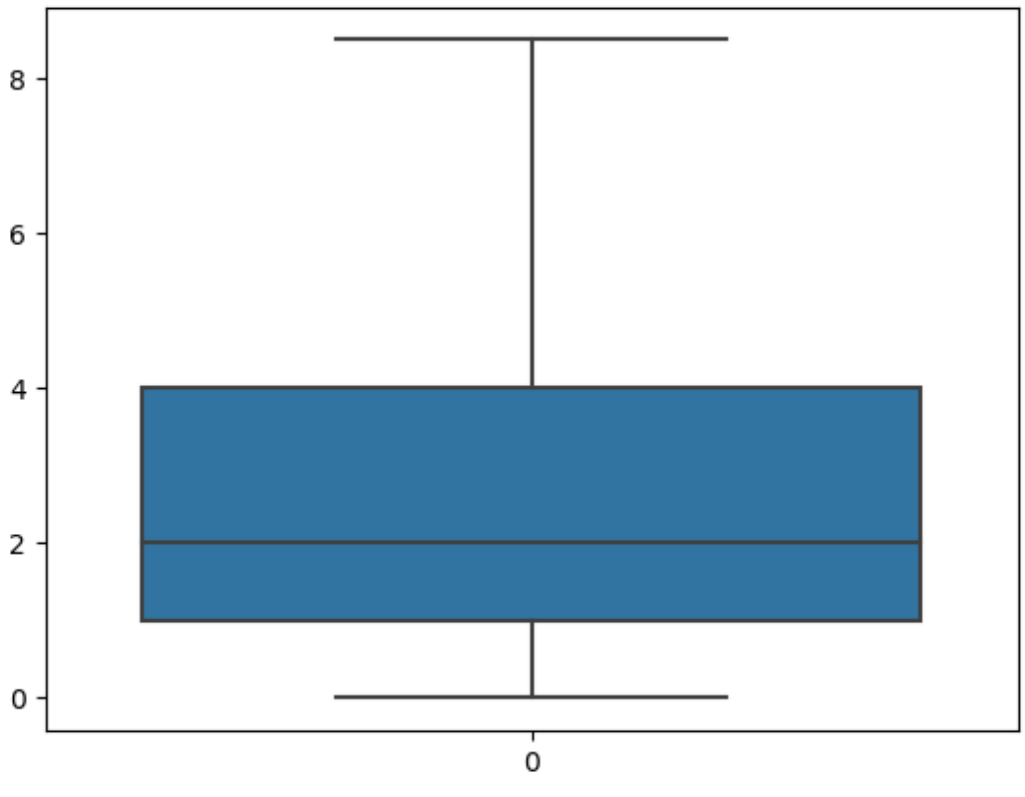
In [15]:
```
q1 = dataset.NumCompaniesWorked.quantile(0.25)
q3 = dataset.NumCompaniesWorked.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [16]:
```
dataset['NumCompaniesWorked'] = np.where(dataset['NumCompaniesWorked'] > upperlimit , upperlimit,dataset['NumCompaniesWo
dataset['NumCompaniesWorked'] = np.where(dataset['NumCompaniesWorked'] < lowerlimit, lowerlimit, dataset['NumCompaniesWo
```

In [17]:
```
sns.boxplot(dataset.NumCompaniesWorked)
```

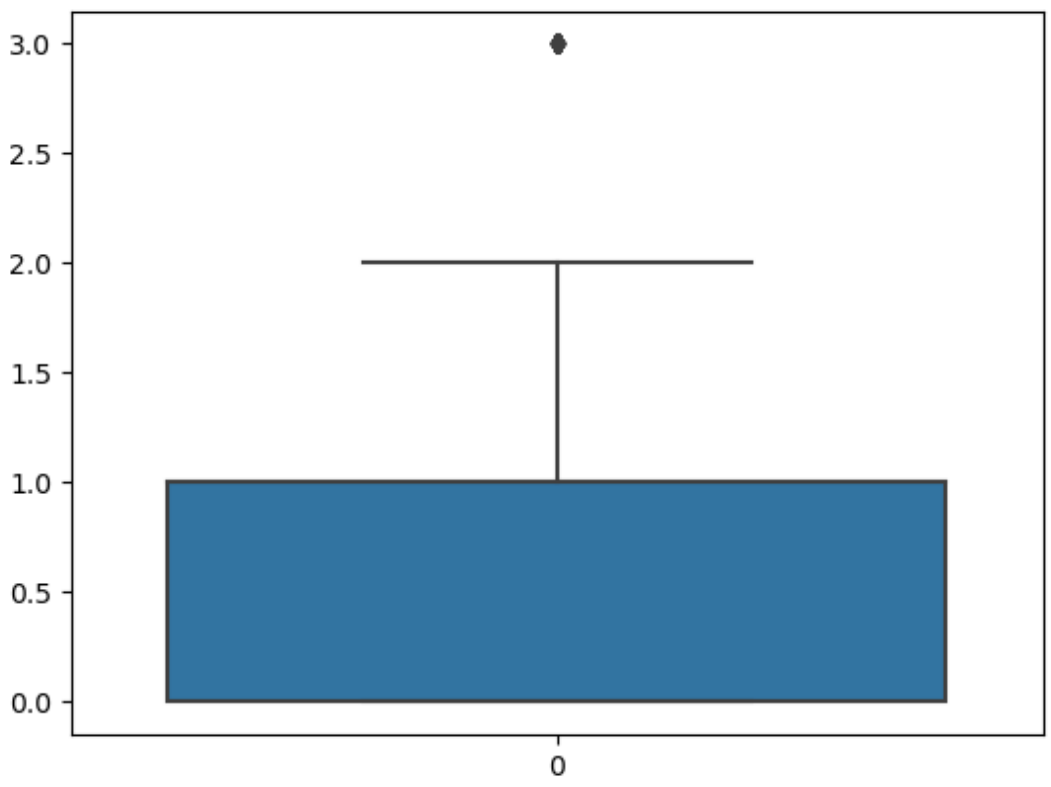
In [17]:
```
sns.boxplot(dataset.NumCompaniesWorked)
```

Out[17]: <Axes: >



In [18]:
```
sns.boxplot(dataset.StockOptionLevel)
```

Out[18]: <Axes: >



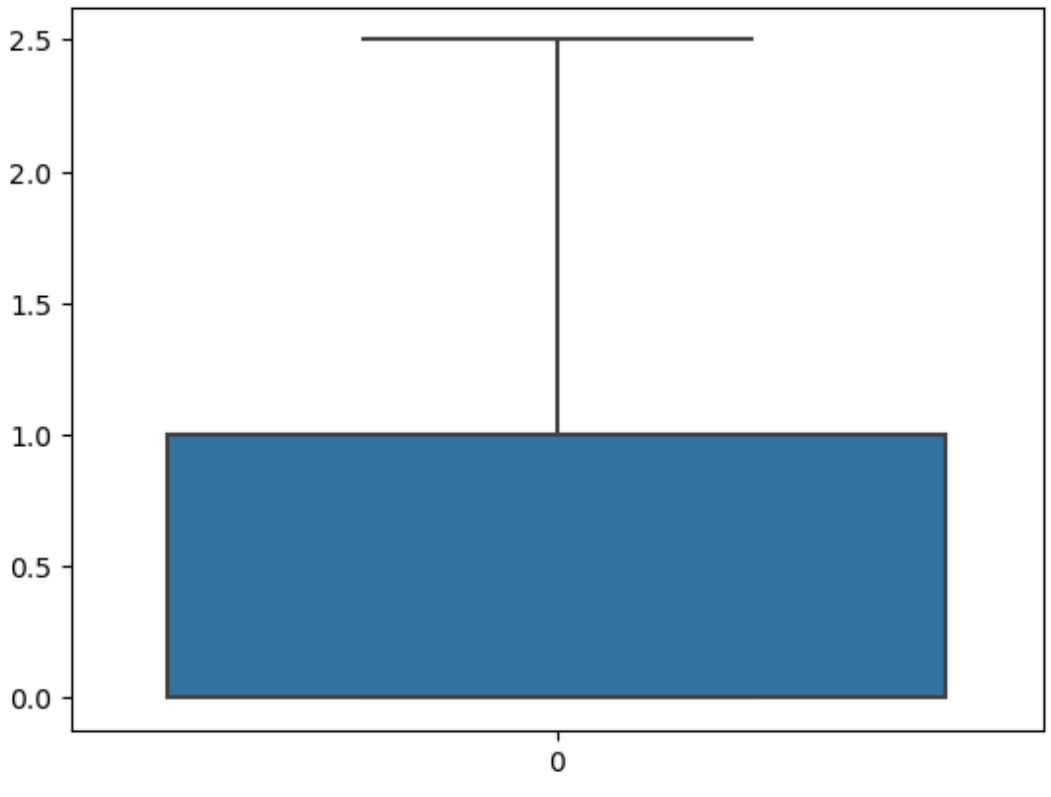
In [19]:
```
q1 = dataset.StockOptionLevel.quantile(0.25)
q3 = dataset.StockOptionLevel.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [20]:
```
dataset['StockOptionLevel'] = np.where(dataset['StockOptionLevel'] > upperlimit , upperlimit,dataset['StockOptionLevel'])
dataset['StockOptionLevel'] = np.where(dataset['StockOptionLevel'] < lowerlimit, lowerlimit, dataset['StockOptionLevel'])
```

In [21]:
```
sns.boxplot(dataset.StockOptionLevel)
```

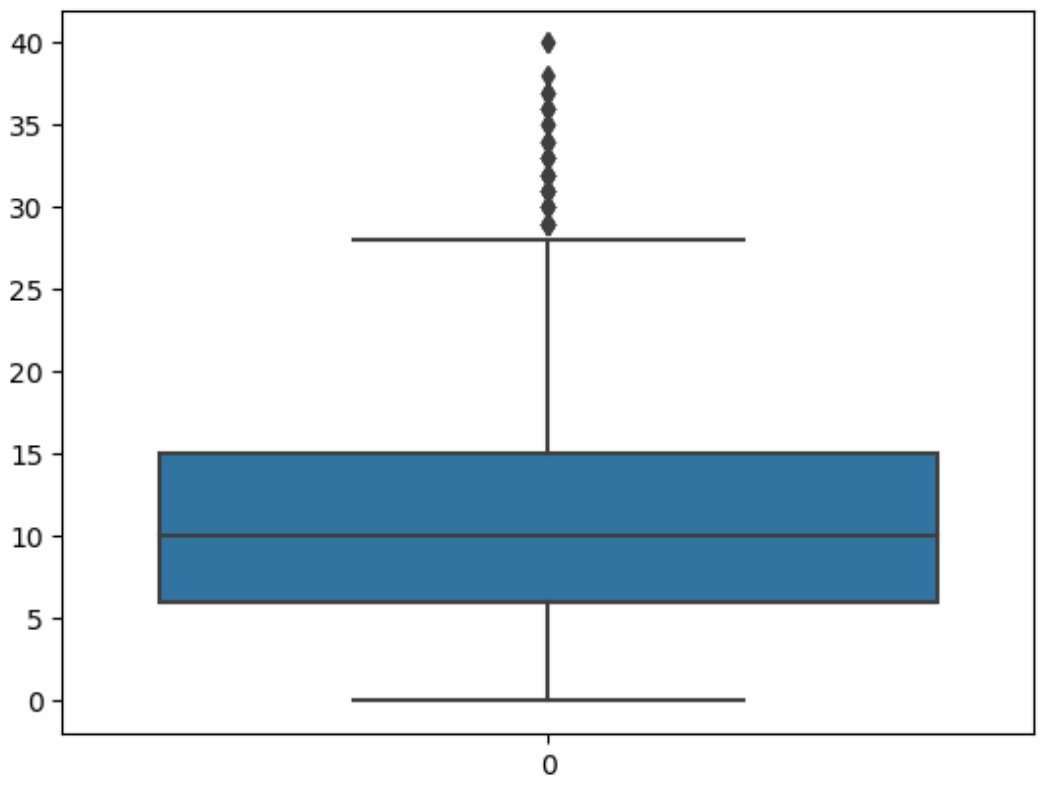
In [21]:
```
sns.boxplot(dataset.StockOptionLevel)
```

Out[21]: <Axes: >



In [22]:
```
sns.boxplot(dataset.TotalWorkingYears)
```

Out[22]: <Axes: >



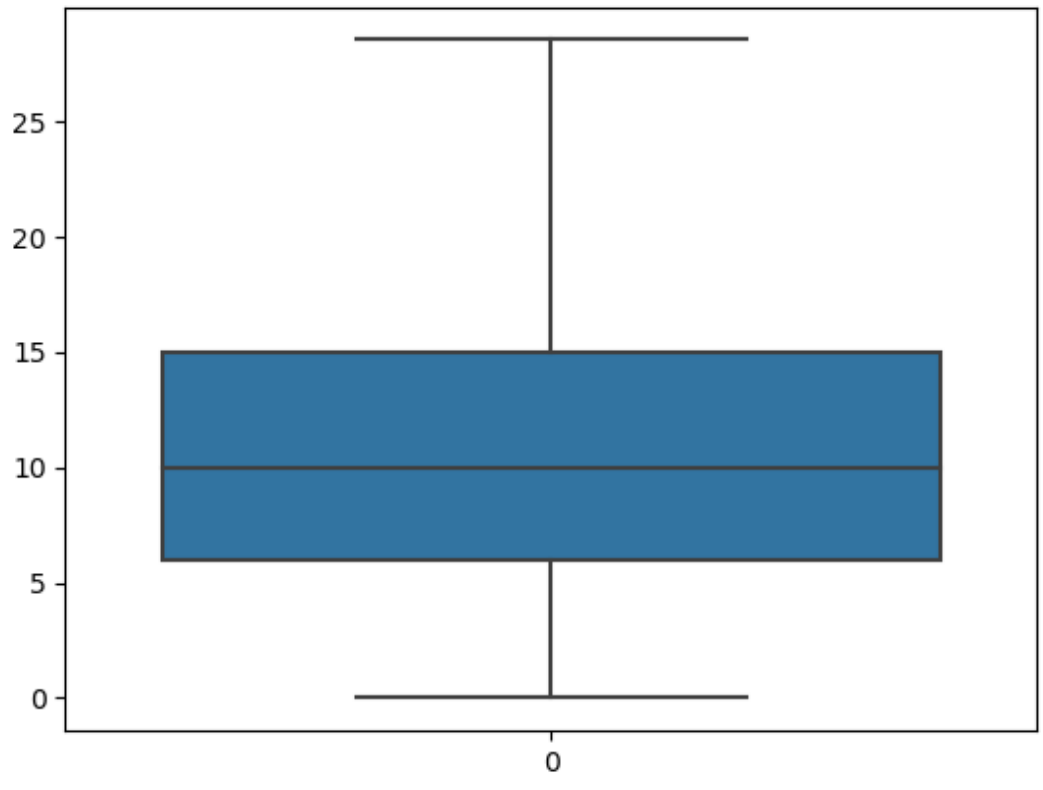
In [23]:
```
q1 = dataset.TotalWorkingYears.quantile(0.25)
q3 = dataset.TotalWorkingYears.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [24]:
```
dataset['TotalWorkingYears'] = np.where(dataset['TotalWorkingYears'] > upperlimit , upperlimit,dataset['TotalWorkingYears
dataset['TotalWorkingYears'] = np.where(dataset['TotalWorkingYears'] < lowerlimit, lowerlimit, dataset['TotalWorkingYears
```

In [25]:
```
sns.boxplot(dataset.TotalWorkingYears)
```

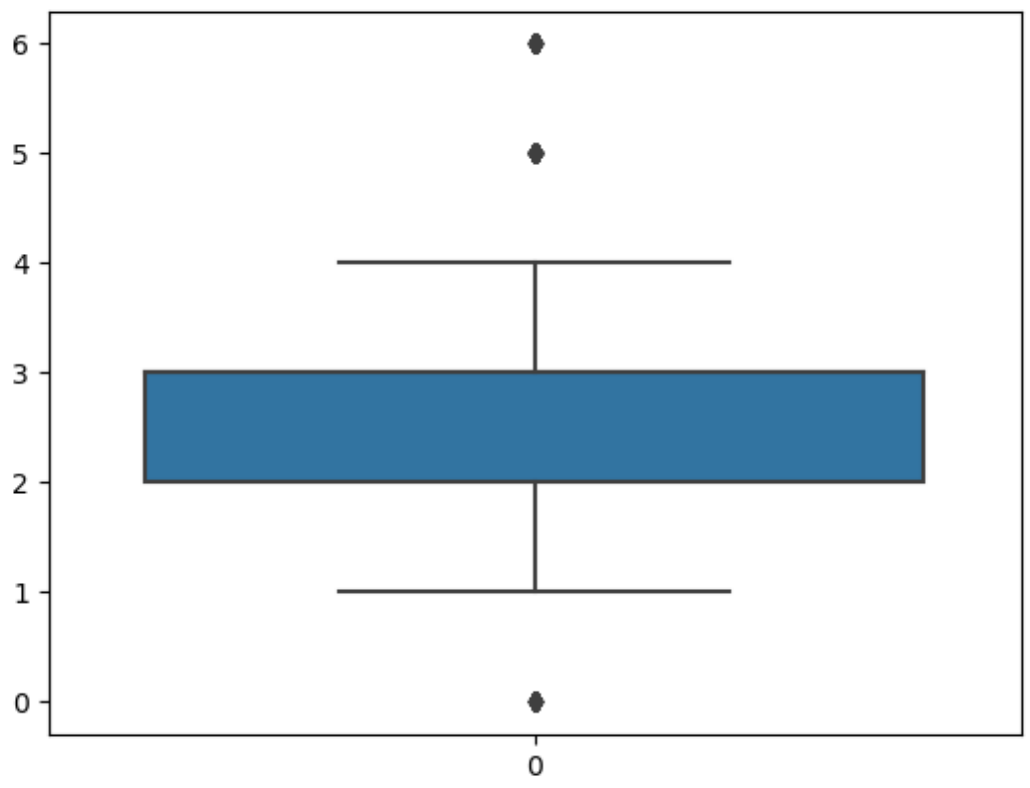
In [25]:
```
sns.boxplot(dataset.TotalWorkingYears)
```

Out[25]: <Axes: >



In [26]:
```
sns.boxplot(dataset.TrainingTimesLastYear)
```

Out[26]: <Axes: >



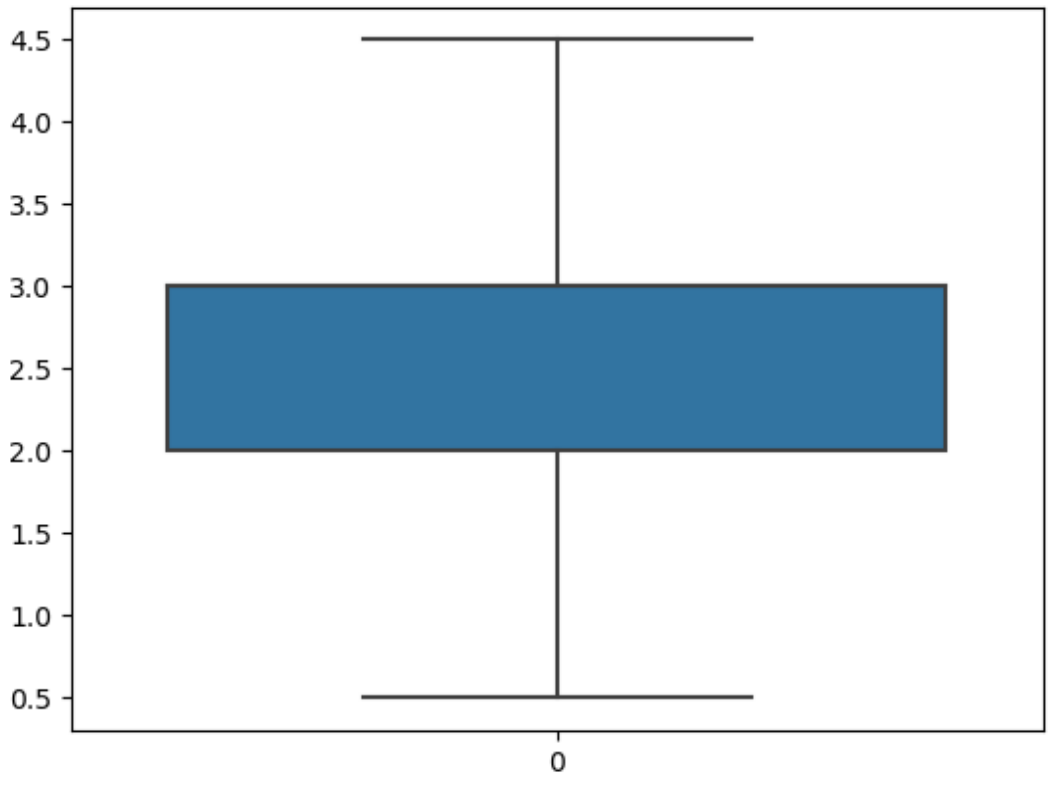
In [27]:
```
q1 = dataset.TrainingTimesLastYear.quantile(0.25)
q3 = dataset.TrainingTimesLastYear.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [28]:
```
et['TrainingTimesLastYear'] = np.where(dataset['TrainingTimesLastYear'] > upperlimit , upperlimit,dataset['TrainingTimesL
et['TrainingTimesLastYear'] = np.where(dataset['TrainingTimesLastYear'] < lowerlimit, lowerlimit, dataset['TrainingTimesL
```

In [29]:
```
sns.boxplot(dataset.TrainingTimesLastYear)
```

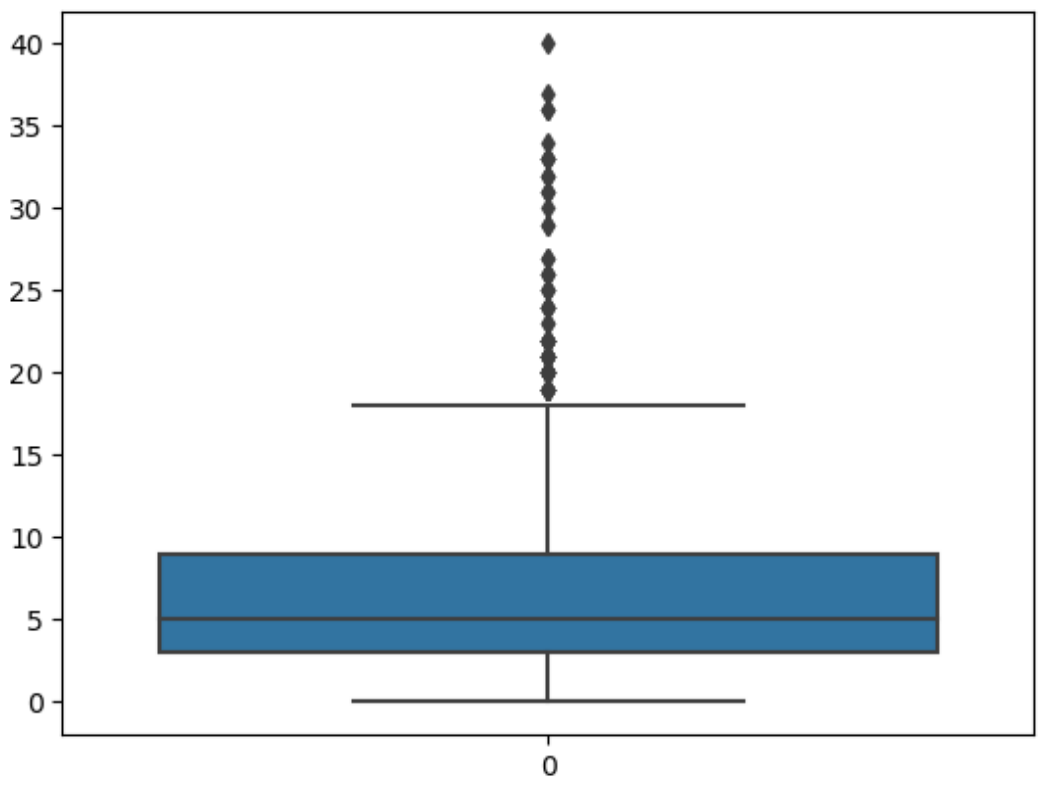
In [29]:
```
sns.boxplot(dataset.TrainingTimesLastYear)
```

Out[29]: <Axes: >



In [30]:
```
sns.boxplot(dataset.YearsAtCompany)
```

Out[30]: <Axes: >



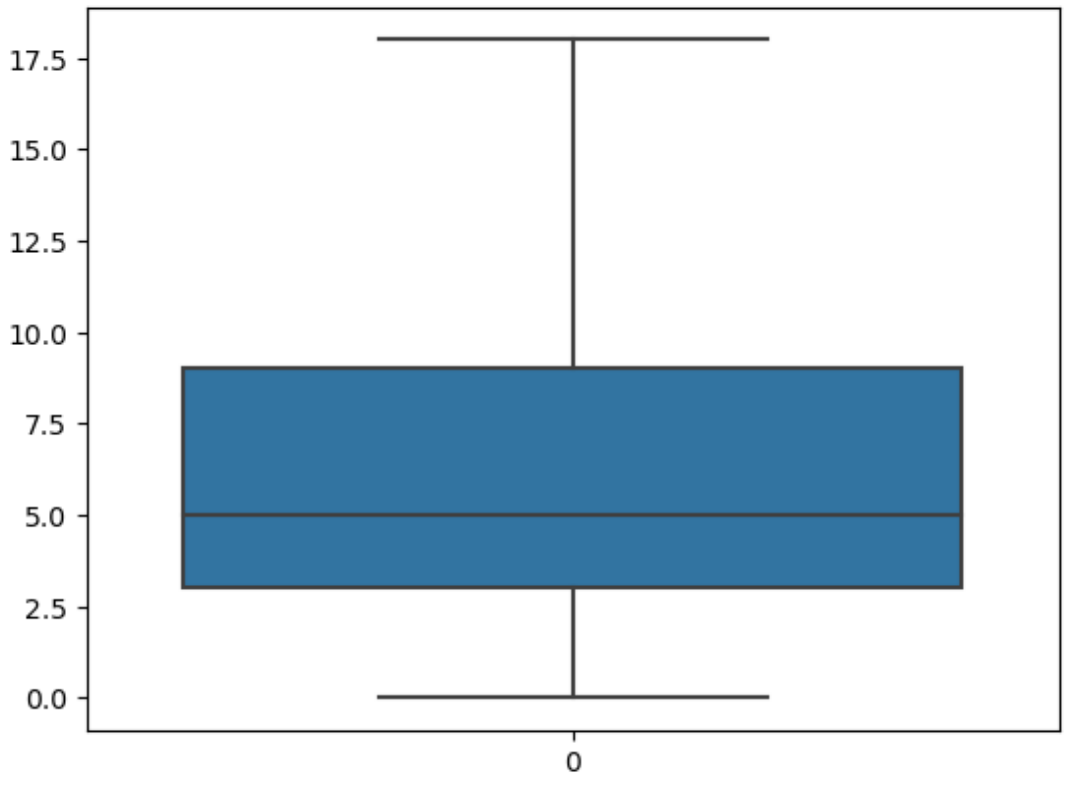
In [31]:
```
q1 = dataset.YearsAtCompany.quantile(0.25)
q3 = dataset.YearsAtCompany.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [32]:
```
dataset['YearsAtCompany'] = np.where(dataset['YearsAtCompany'] > upperlimit , upperlimit,dataset['YearsAtCompany'])
dataset['YearsAtCompany'] = np.where(dataset['YearsAtCompany'] < lowerlimit, lowerlimit, dataset['YearsAtCompany'])
```

In [33]:
```
sns.boxplot(dataset.YearsAtCompany)
```

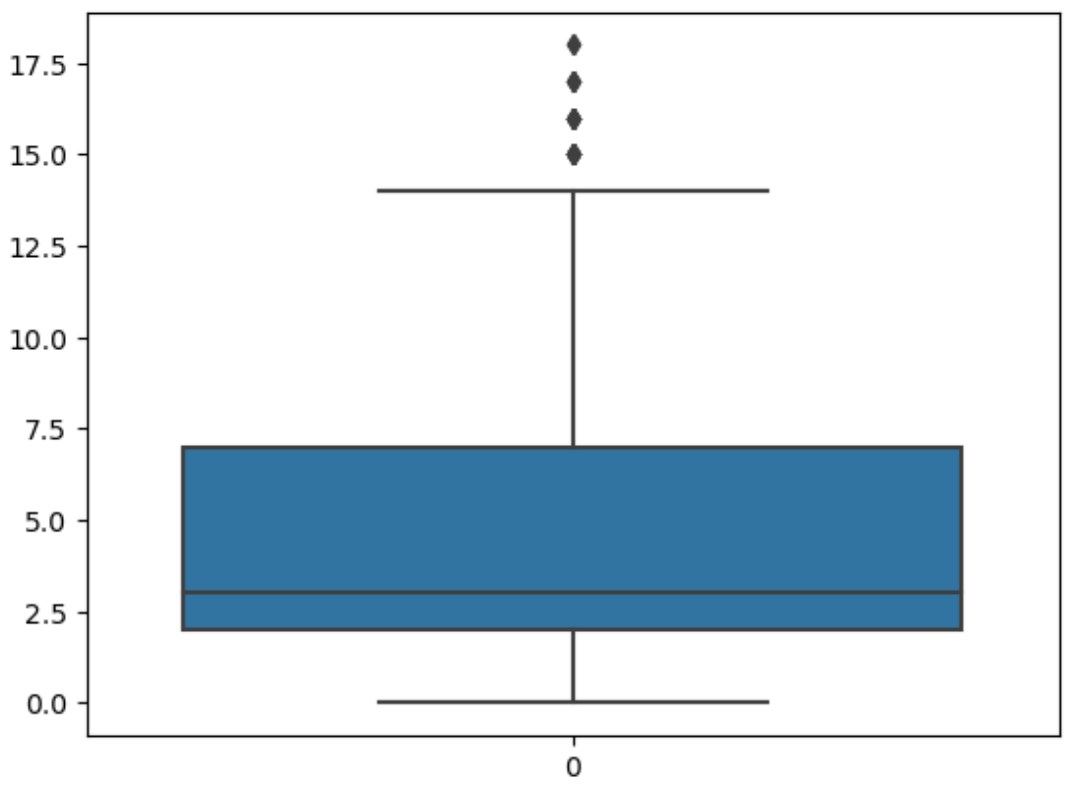
In [33]:
```
sns.boxplot(dataset.YearsAtCompany)
```

Out[33]: <Axes: >



In [34]:
```
sns.boxplot(dataset.YearsInCurrentRole)
```

Out[34]: <Axes: >



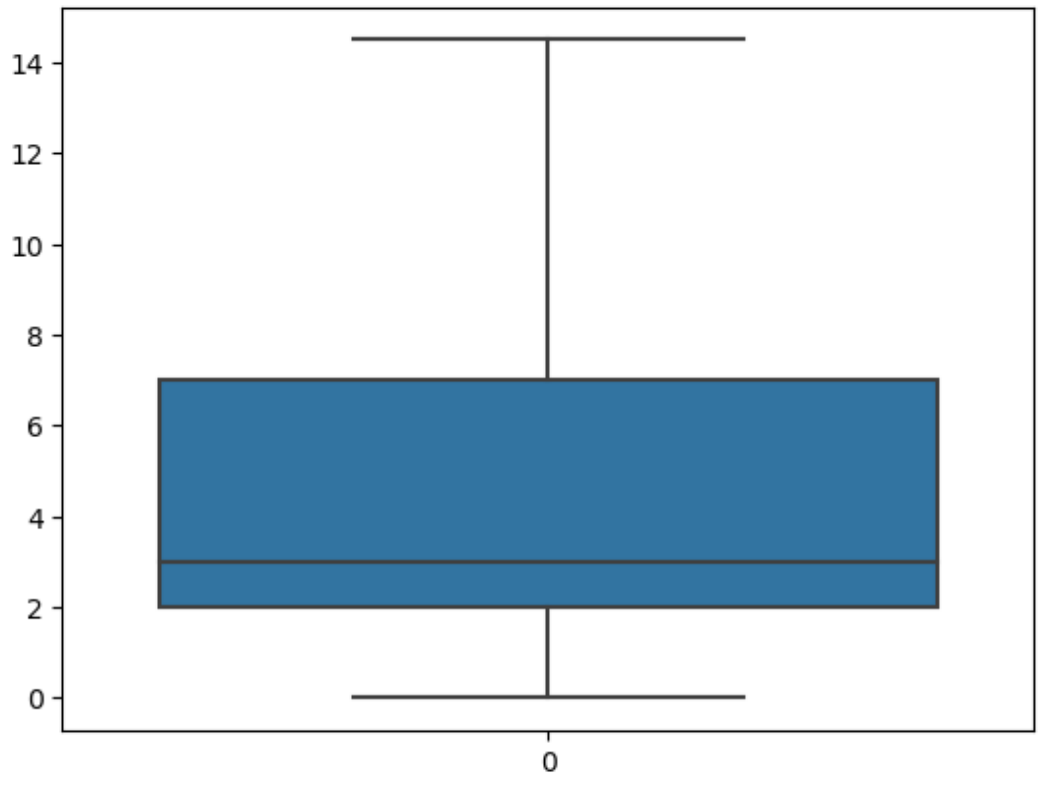
In [35]:
```
q1 = dataset.YearsInCurrentRole.quantile(0.25)
q3 = dataset.YearsInCurrentRole.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [36]:
```
dataset['YearsInCurrentRole'] = np.where(dataset['YearsInCurrentRole'] > upperlimit , upperlimit,dataset['YearsInCurrentR
dataset['YearsInCurrentRole'] = np.where(dataset['YearsInCurrentRole'] < lowerlimit, lowerlimit, dataset['YearsInCurrentR
```

In [37]:
```
sns.boxplot(dataset.YearsInCurrentRole)
```

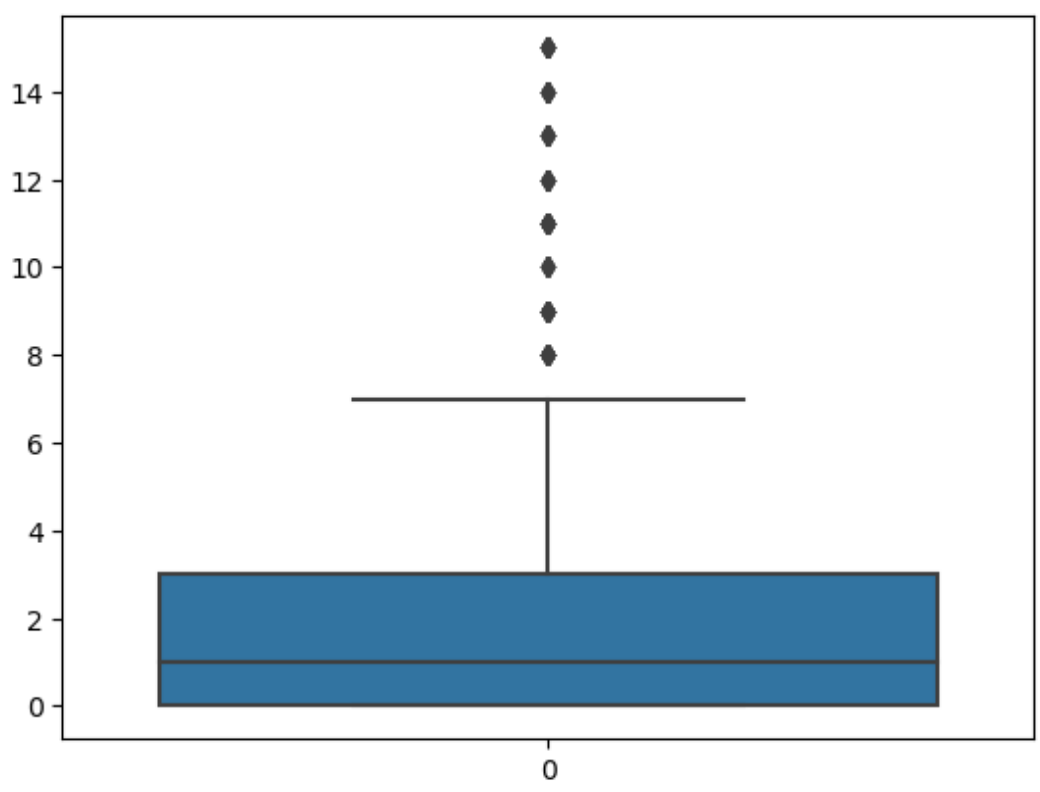
In [37]:
```
sns.boxplot(dataset.YearsInCurrentRole)
```

Out[37]: <Axes: >



In [38]:
```
sns.boxplot(dataset.YearsSinceLastPromotion)
```

Out[38]: <Axes: >



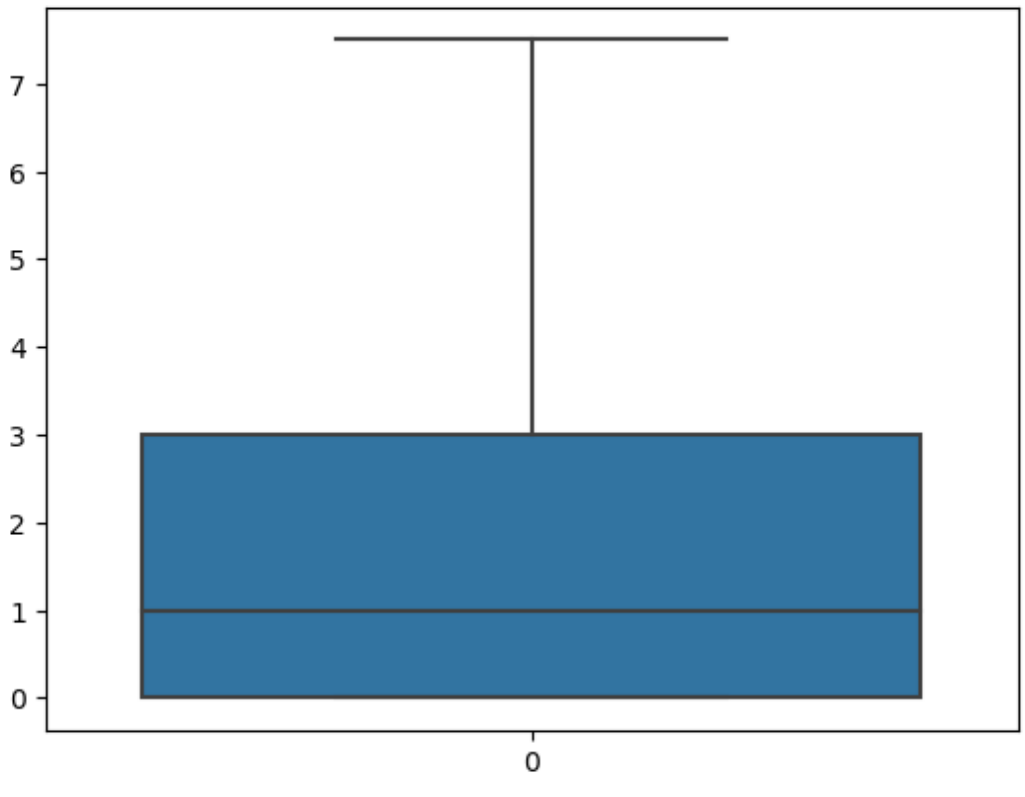
In [39]:
```
q1 = dataset.YearsSinceLastPromotion.quantile(0.25)
q3 = dataset.YearsSinceLastPromotion.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [40]:
```
'YearsSinceLastPromotion'] = np.where(dataset['YearsSinceLastPromotion'] > upperlimit , upperlimit,dataset['YearsSinceLas
'YearsSinceLastPromotion'] = np.where(dataset['YearsSinceLastPromotion'] < lowerlimit, lowerlimit, dataset['YearsSinceLas
```

In [41]:
```
sns.boxplot(dataset.YearsSinceLastPromotion)
```

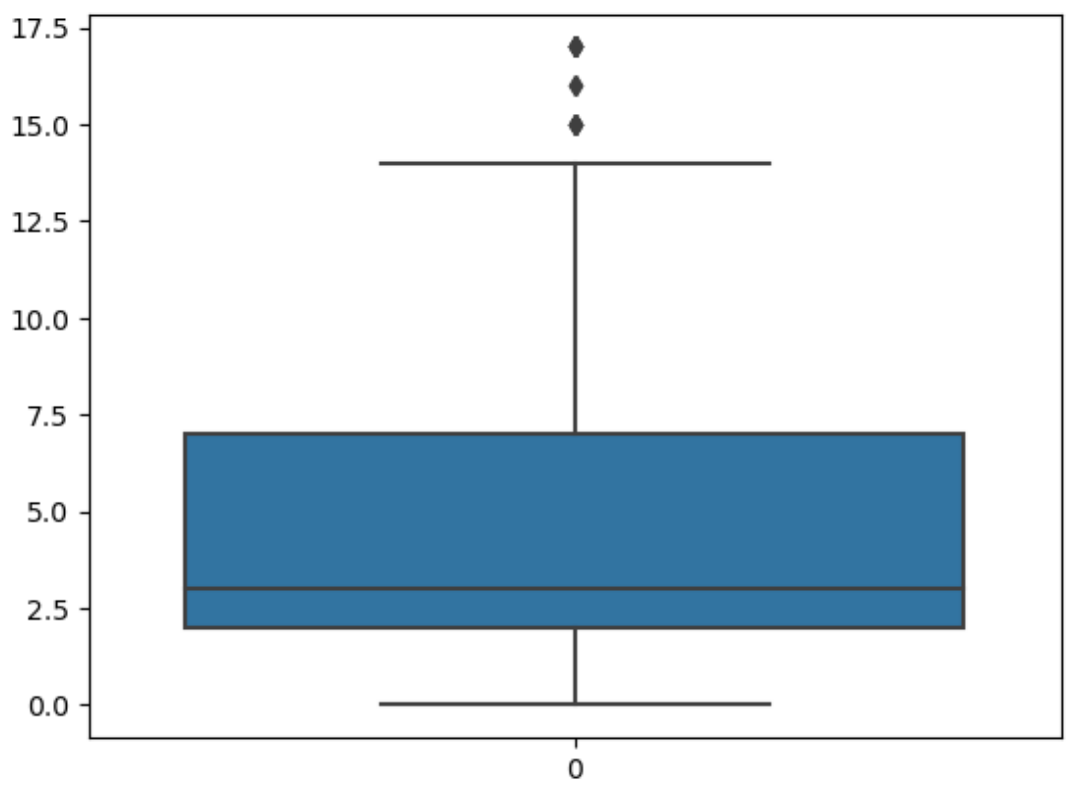
In [41]:
```
sns.boxplot(dataset.YearsSinceLastPromotion)
```

Out[41]: <Axes: >



In [42]:
```
sns.boxplot(dataset.YearsWithCurrManager)
```

Out[42]: <Axes: >



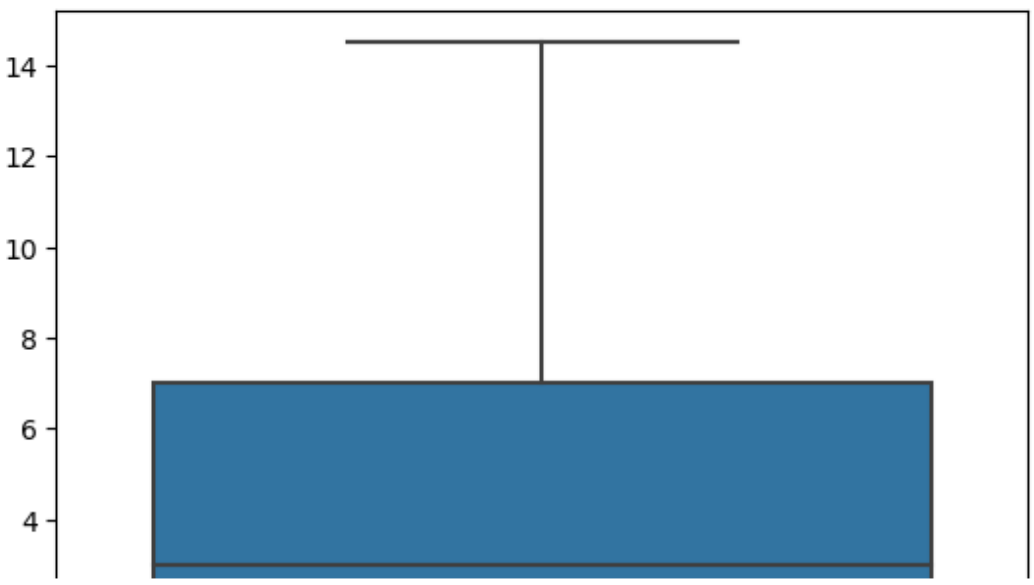
In [43]:
```
q1 = dataset.YearsWithCurrManager.quantile(0.25)
q3 = dataset.YearsWithCurrManager.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [44]:
```
taset['YearsWithCurrManager'] = np.where(dataset['YearsWithCurrManager'] > upperlimit ,upperlimit, dataset['YearsWithCurr
taset['YearsWithCurrManager'] = np.where(dataset['YearsWithCurrManager'] < lowerlimit, lowerlimit, dataset['YearsWithCurr
```

In [45]:
```
sns.boxplot(dataset.YearsWithCurrManager)
```

In [45]:
```
sns.boxplot(dataset.YearsWithCurrManager)
```

Out[45]: <Axes: >



In [46]:
```
y = dataset['Attrition']
y.head()
```

Out[46]:
```
0    Yes
1     No
2    Yes
3     No
4     No
Name: Attrition, dtype: object
```

In [47]:
```
x = dataset
```

In [48]:
```
x = x.drop(columns=['Attrition'])
```

In [49]:
```
x.head()
```

Out[49]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatis
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	

5 rows × 34 columns

In [50]:
```
le = LabelEncoder() #creating a object for label encoder
```

In [51]:
```
x['BusinessTravel'] = le.fit_transform(x['BusinessTravel'])
```

In [52]:
```
x['Department'] = le.fit_transform(x['Department'])
```

In [53]:
```
x['EducationField'] = le.fit_transform(x['EducationField'])
```

In [54]:
```
x['Gender'] = le.fit_transform(x['Gender'])
```

In [55]:
```
x['JobRole'] = le.fit_transform(x['JobRole'])
```

In [56]:
```
x['MaritalStatus'] = le.fit_transform(x['MaritalStatus'])
```

In [57]:
```
x['Over18'] = le.fit_transform(x['Over18'])
```

In [58]:
```
x['OverTime'] = le.fit_transform(x['OverTime'])
```

In [59]:
```
y = le.fit_transform(y)
```

In [60]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

In [61]:
```
sc = StandardScaler()
```

In [61]:
```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
x_train
```

Out[61]:
```
array([[-0.75016842,  0.58888875, -0.58244694, ...,  1.35715165,
        -0.37275049, -0.59310587],
       [-0.41863372,  0.58888875, -1.12354988, ..., -1.1694503 ,
        -0.77459466, -1.16535711],
       [ 0.90750511,  0.58888875, -0.04626313, ..., -0.32724965,
        -0.77459466, -0.02085464],
       ...,
       [ 0.68648197,  0.58888875,  0.91542436, ...,  1.35715165,
         2.23923658,  1.12364784],
       [ 0.13392413,  0.58888875, -1.3252337 , ..., -1.1694503 ,
        -0.77459466, -0.87923149],
       [ 0.35494726,  0.58888875, -0.36600577, ..., -1.1694503 ,
        -0.77459466, -1.16535711]])
```

LOGISTIC REGRESSION

In [62]:
```
from sklearn.linear_model import LogisticRegression
```

In [63]:
```
model=LogisticRegression()
```

In [64]:
```
model.fit(x_train,y_train)
```

Out[64]:
```
LogisticRegression
LogisticRegression()
```

In [65]:
```
pred=model.predict(x_test)
```

In [66]:
```
pred
```

Out[66]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0])
```

In [67]:
```
y_test
```

Out[67]:
```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0])
```

In [68]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [69]:
```
accuracy_score(y_test,pred)
```

In [69]:
```
accuracy_score(y_test,pred)
```

Out[69]:
```
0.8888888888888888
```

In [70]:
```
confusion_matrix(y_test,pred)
```

Out[70]:
```
array([[367,   4],
       [ 45,  25]], dtype=int64)
```

In [71]:
```
probability = model.predict_proba(x_test)[:,1]
```

In [71]:
```
probability = model.predict_proba(x_test)[:,1]
probability
```

Out[71]:
```
array([0.19579831, 0.15552362, 0.38995702, 0.08682671, 0.6922058 ,
       0.0624921 , 0.62952533, 0.07396905, 0.00573277, 0.43260736
```

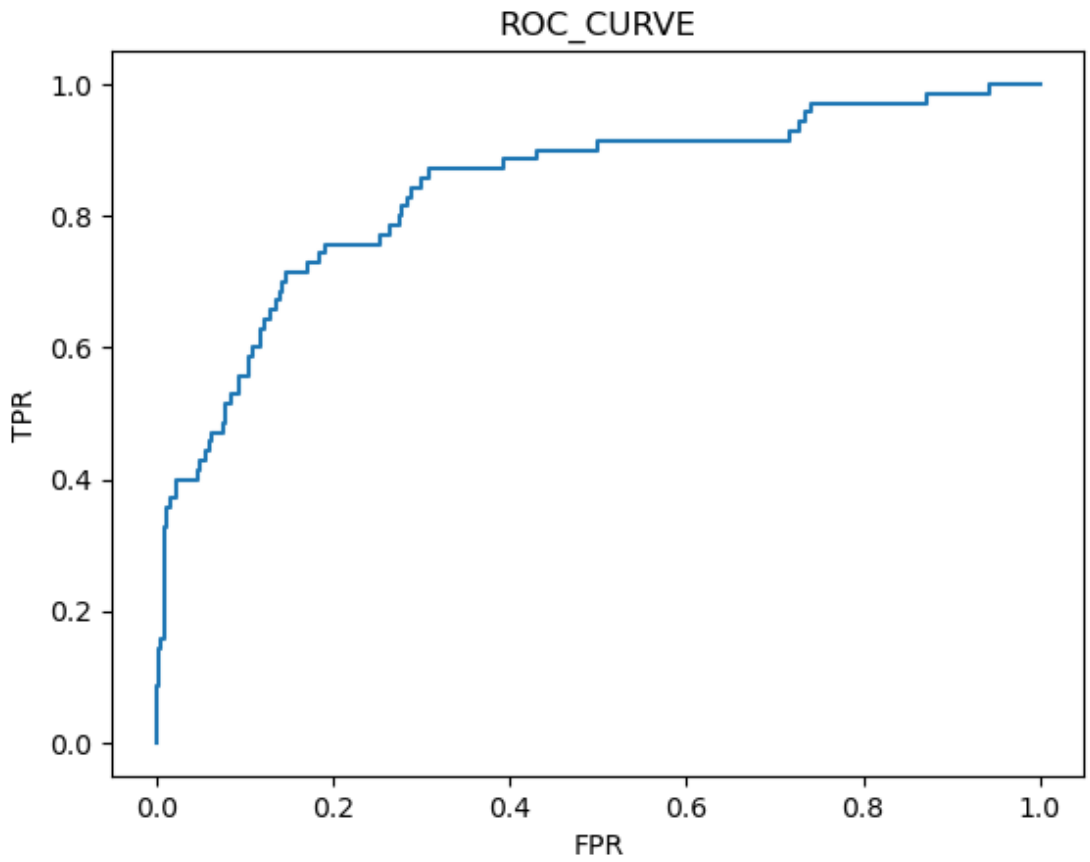
Out[71]:
```
array([0.19579831, 0.15552362, 0.38995702, 0.08682671, 0.6922058 ,
       0.0624921 , 0.62952533, 0.07396905, 0.00573277, 0.43260736,
       0.04490463, 0.27352994, 0.01159387, 0.68417371, 0.12930571,
       0.03383768, 0.09221393, 0.14720723, 0.03944072, 0.29868575,
       0.20294639, 0.01186285, 0.05437088, 0.05106273, 0.6995925 ,
       0.43699338, 0.06113834, 0.05009991, 0.71272923, 0.05319752,
       0.00921142, 0.02232563, 0.05813214, 0.15944243, 0.06812333,
       0.02652088, 0.09072814, 0.05606023, 0.02953579, 0.03892011,
       0.02803701, 0.01956232, 0.00896522, 0.01398542, 0.02507106,
       0.51386154, 0.42116382, 0.00167935, 0.74326518, 0.60569995,
       0.06691342, 0.56245756, 0.12524887, 0.36712016, 0.66045072,
       0.2686972 , 0.0142516 , 0.30452917, 0.01892498, 0.21461241,
       0.01166206, 0.2736003 , 0.13107796, 0.05263695, 0.33766339,
       0.01855701, 0.35959885, 0.15427192, 0.09709966, 0.06467332,
       0.04843011, 0.25751633, 0.08204146, 0.05983077, 0.1490895 ,
       0.05669258, 0.04768474, 0.0662595 , 0.21092731, 0.0302912 ,
       0.00512108, 0.01894607, 0.12352564, 0.01896209, 0.02261006,
       0.06210019, 0.00235425, 0.02447942, 0.03488668, 0.09450654,
       0.2903063 , 0.203084  , 0.29037019, 0.21700654, 0.01294784,
       0.19081892, 0.31690863, 0.2295535 , 0.0507254 , 0.0457513 ,
       0.36730484, 0.77546176, 0.17129448, 0.01572669, 0.1132287 ,
       0.02527927, 0.05881631, 0.17768066, 0.06721389, 0.09340386,
       0.06632914, 0.02492894, 0.02041484, 0.21205651, 0.05106419,
       0.02856211, 0.045039  , 0.10162959, 0.00558798, 0.01131369,
       0.15983685, 0.03119699, 0.09094822, 0.86577378, 0.02329163,
       0.01303435, 0.00435058, 0.14032879, 0.18743659, 0.02768259,
       0.00759633, 0.33839497, 0.57901869, 0.3819565 , 0.04446413,
       0.42320152, 0.61415077, 0.23639291, 0.05612529, 0.25489927,
       0.06749645, 0.05777534, 0.05543002, 0.19093678, 0.29213723,
       0.02503402, 0.11222578, 0.00179009, 0.12979696, 0.20240728,
       0.04238634, 0.11169611, 0.04162917, 0.09341743, 0.03621538,
       0.01662032, 0.01830264, 0.09338259, 0.00884409, 0.01453791,
       0.27305791, 0.0098961 , 0.18380707, 0.8744936 , 0.11300511,
       0.22439753, 0.20818325, 0.13336251, 0.03497385, 0.00264867,
       0.03178672, 0.14899891, 0.10386491, 0.12633262, 0.0055426 ,
       0.09421436, 0.10346566, 0.07542156, 0.04793617, 0.07507159,
       0.02296713, 0.11144322, 0.00248745, 0.80091957, 0.05056744,
       0.02296964, 0.41531014, 0.0502381 , 0.76502424, 0.09428294,
       0.35112774, 0.43239374, 0.47322341, 0.04270531, 0.05728281,
       0.21947094, 0.04084905, 0.00704577, 0.30508954, 0.07484356,
       0.21573131, 0.19327207, 0.75030378, 0.04946435, 0.34416735,
       0.03689173, 0.49075683, 0.00338736, 0.20792491, 0.02278638,
       0.11254303, 0.22950906, 0.07155397, 0.08053402, 0.25405405,
       0.01592012, 0.01458023, 0.07134449, 0.0166122 , 0.15624144,
       0.11284808, 0.23716724, 0.7813573 , 0.07129064, 0.42346859,
       0.01085563, 0.11290392, 0.2128908 , 0.39785767, 0.03617695,
       0.02421154, 0.31375247, 0.03873665, 0.01960503, 0.1646768 ,
       0.36172753, 0.30211788, 0.00643235, 0.05095244, 0.00630729,
       0.1326142 , 0.30225639, 0.01093046, 0.14838264, 0.049809  ,
       0.02732427, 0.41167585, 0.33034077, 0.05338394, 0.11905567,
       0.4225208 , 0.30177191, 0.85579031, 0.03221388, 0.19903778,
       0.06616074, 0.00246061, 0.70137888, 0.39165098, 0.36639266,
       0.3586306 , 0.03209488, 0.21616405, 0.052809  , 0.05233305,
       0.12244982, 0.00390377, 0.26596829, 0.46731341, 0.05307816,
       0.08669414, 0.01107614, 0.19711223, 0.07259327, 0.01511039,
       0.01599685, 0.06017988, 0.31892086, 0.26656807, 0.19115016,
       0.23785227, 0.01095053, 0.12064797, 0.08864395, 0.01303785,
       0.16784956, 0.00535149, 0.29129199, 0.00180718, 0.0193986 ,
       0.18169888, 0.86182474, 0.07205707, 0.21491777, 0.04950216,
       0.0243806 , 0.34735529, 0.13148419, 0.8181816 , 0.00213566,
       0.2121886 , 0.06164699, 0.12428891, 0.01499585, 0.07089641,
       0.07293826, 0.18405985, 0.16660644, 0.0022344 , 0.01406096,
       0.13854347, 0.01792022, 0.1268    , 0.15751838, 0.20038518,
       0.27241236, 0.19269631, 0.05796374, 0.07670279, 0.01737317,
       0.2054389 , 0.015812  , 0.44603607, 0.29741988, 0.10996515,
       0.05539215, 0.3798851 , 0.23183818, 0.00387448, 0.06416398,
       0.01977093, 0.31012706, 0.04049341, 0.55941864, 0.24809263,
       0.01422493, 0.12981161, 0.0017436 , 0.22861076, 0.0511786 ,
       0.09989979, 0.56964403, 0.05963581, 0.52953235, 0.07306062,
       0.0158111 , 0.16438662, 0.03614249, 0.01860067, 0.18851046,
       0.04379756, 0.4130974 , 0.07992365, 0.05111954, 0.13756952,
       0.00886905, 0.16617833, 0.11370354, 0.01145929, 0.17112443,
       0.00131049, 0.01343261, 0.04285953, 0.03108958, 0.14002884,
       0.08434439, 0.00404145, 0.33328994, 0.03307055, 0.1180385 ,
       0.00250866, 0.25224001, 0.02591001, 0.08862483, 0.33441248,
       0.02005985, 0.01449006, 0.0631204 , 0.62770516, 0.028856  ,
       0.28214059, 0.02326298, 0.08368591, 0.10210585, 0.00209815,
       0.32395596, 0.00198426, 0.38150659, 0.01392343, 0.14896799,
       0.0297138 , 0.23711494, 0.04540401, 0.27016254, 0.58242865,
       0.01747113, 0.05919349, 0.34813624, 0.05537212, 0.03734039,
       0.51573007, 0.07597715, 0.03490469, 0.03563915, 0.05515528,
       0.00873888, 0.32709479, 0.06985222, 0.01952601, 0.01365841,
       0.27075683, 0.02625795, 0.12410062, 0.415521  , 0.42297131,
       0.13352931, 0.15515134, 0.23398285, 0.20292476, 0.02199727,
       0.21497348, 0.08455902, 0.02739409, 0.01255949, 0.02110876,
       0.05149367, 0.08996571, 0.11016683, 0.20626417, 0.00987669,
       0.2933518 , 0.10665923, 0.01693612, 0.05090806, 0.05935808,
       0.07854326, 0.07044885, 0.19761001, 0.14180831, 0.0427378 ,
       0.00868604])
```

In [72]:
```
from sklearn.metrics import roc_curve
```

```
0.60868064])
```

In [72]:
```
from sklearn.metrics import roc_curve
fpr , tpr, thresholds = roc_curve(y_test, probability)
```

In [73]:
```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC_CURVE')
plt.show()
```



DECISION TREE

In [74]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
```

In [75]:
```
dtc.fit(x_train,y_train)
```

Out[75]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [76]:
```
pred = dtc.predict(x_test)
pred
```

Out[76]:
```
array([0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0])
```

In [77]:
```
y_test
```

In [77]:
```
y_test
```

Out[77]:
```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0])
```

In [78]:
```
accuracy_score(y_test,pred)
```

Out[78]:
```
0.7755102040816326
```

In [79]:
```
confusion_matrix(y_test,pred)
```

Out[79]:
```
array([[316,  55],
       [ 44,  26]], dtype=int64)
```

In [80]:
```
probability = dtc.predict_proba(x_test)[:,1]
probability
```

Out[80]:
```
array([0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0.,
       0., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
       0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
       1., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1.,
       0., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1.,
       0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 1., 0., 0., 0., 0., 0.,
       1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.])
```

In [81]:
```
from sklearn.metrics import roc_curve
fpr , tpr, thresholds = roc_curve(y_test, probability)
```

In [82]:
```
plt.plot(fpr,tpr)
```

In [82]:
```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC_CURVE')
plt.show()
```