In [ ]:
```
pip install Pillow
```

In [3]:
```
from IPython.display import Image, display

# Specify the path to your image file
image_path = '"C:\Users\jangi\Downloads\HR_Employee_Attrition_iamge.png"

# Display the image
display(Image(filename=image_path))

```

HR Employee Attrition Data Analysis

Objective

- The aim of this dataset is to build a model that can predict the attrition of the employees based on employee factors.

Import the Modules

EDA and Preprocessing

Machine Learning Algorithms

- Logistic Regression
- Random Forest

- Decision Tree

In [71]:
```
import pandas as pd
import numpy as np


import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.metrics import confusion_matrix, classification_report, accur
from sklearn.model_selection import cross_val_score


import warnings
warnings.filterwarnings('ignore')

```

Read the CSV File

In [4]:
```
df = pd.read_csv(r"C:\Users\jangi\Downloads\HR_Employee_Attrition_Data.csv
```

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

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educ
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

Basic information

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

Out[4]:
```
(2940, 35)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2940 entries, 0 to 2939
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       2940 non-null   int64
 1   Attrition                 2940 non-null   object
 2   BusinessTravel            2940 non-null   object
 3   DailyRate                 2940 non-null   int64
 4   Department                2940 non-null   object
 5   DistanceFromHome          2940 non-null   int64
 6   Education                 2940 non-null   int64
 7   EducationField            2940 non-null   object
 8   EmployeeCount             2940 non-null   int64
 9   EmployeeNumber            2940 non-null   int64
 10  EnvironmentSatisfaction   2940 non-null   int64
 11  Gender                    2940 non-null   object
 12  HourlyRate                2940 non-null   int64
 13  JobInvolvement            2940 non-null   int64
 14  JobLevel                  2940 non-null   int64
 15  JobRole                   2940 non-null   object
 16  JobSatisfaction           2940 non-null   int64
 17  MaritalStatus             2940 non-null   object
 18  MonthlyIncome             2940 non-null   int64
 19  MonthlyRate               2940 non-null   int64
 20  NumCompaniesWorked        2940 non-null   int64
 21  Over18                    2940 non-null   object
 22  OverTime                  2940 non-null   object
 23  PercentSalaryHike         2940 non-null   int64
 24  PerformanceRating         2940 non-null   int64
 25  RelationshipSatisfaction  2940 non-null   int64
 26  StandardHours             2940 non-null   int64
 27  StockOptionLevel          2940 non-null   int64
 28  TotalWorkingYears         2940 non-null   int64
 29  TrainingTimesLastYear     2940 non-null   int64
 30  WorkLifeBalance           2940 non-null   int64
 31  YearsAtCompany            2940 non-null   int64
 32  YearsInCurrentRole        2940 non-null   int64
 33  YearsSinceLastPromotion   2940 non-null   int64
 34  YearsWithCurrManager      2940 non-null   int64
dtypes: int64(26), object(9)
memory usage: 804.0+ KB
```

In [6]:
```
1 df.describe()
```

Out[6]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	2940.000000	2940.000000	2940.000000	2940.000000	2940.0	2940.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1470.50
std	9.133819	403.440447	8.105485	1.023991	0.0	848.84
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00
25%	30.000000	465.000000	2.000000	2.000000	1.0	735.75
50%	36.000000	802.000000	7.000000	3.000000	1.0	1470.50
75%	43.000000	1157.000000	14.000000	4.000000	1.0	2205.25
max	60.000000	1499.000000	29.000000	5.000000	1.0	2940.00

8 rows × 26 columns

In [7]:
```
1 df.isna().sum()
```

Out[7]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [8]:
```
1 df.duplicated()
```

Out[8]:
```
0       False
1       False
2       False
3       False
4       False
        ...
2935    False
2936    False
2937    False
2938    False
2939    False
Length: 2940, dtype: bool
```

EDA & PreProcessing

In [263]:
```
1 df.describe().T.style.background_gradient(cmap = 'Reds')
```

Out[263]:

	count	mean	std	min	25%	
Age	2940.000000	36.923810	9.133819	18.000000	30.000000	
DailyRate	2940.000000	802.485714	403.440447	102.000000	465.000000	8
Department	2940.000000	1.260544	0.527703	0.000000	1.000000	
DistanceFromHome	2940.000000	9.192517	8.105485	1.000000	2.000000	
Education	2940.000000	2.912925	1.023991	1.000000	2.000000	
EducationField	2940.000000	2.247619	1.331143	0.000000	1.000000	
EmployeeCount	2940.000000	1.000000	0.000000	1.000000	1.000000	
EmployeeNumber	2940.000000	1470.500000	848.849221	1.000000	735.750000	14
EnvironmentSatisfaction	2940.000000	2.721769	1.092896	1.000000	2.000000	
Gender	2940.000000	0.600000	0.489981	0.000000	0.000000	
HourlyRate	2940.000000	65.891156	20.325969	30.000000	48.000000	
JobInvolvement	2940.000000	2.729932	0.711440	1.000000	2.000000	
JobLevel	2940.000000	2.063946	1.106752	1.000000	1.000000	
JobSatisfaction	2940.000000	2.728571	1.102658	1.000000	2.000000	
MaritalStatus	2940.000000	1.097279	0.729997	0.000000	1.000000	
MonthlyIncome	2940.000000	6502.931293	4707.155770	1009.000000	2911.000000	49
MonthlyRate	2940.000000	14313.103401	7116.575021	2094.000000	8045.000000	142
NumCompaniesWorked	2940.000000	2.693197	2.497584	0.000000	1.000000	
PercentSalaryHike	2940.000000	15.209524	3.659315	11.000000	12.000000	
PerformanceRating	2940.000000	3.153741	0.360762	3.000000	3.000000	
RelationshipSatisfaction	2940.000000	2.712245	1.081025	1.000000	2.000000	
StandardHours	2940.000000	80.000000	0.000000	80.000000	80.000000	
StockOptionLevel	2940.000000	0.793878	0.851932	0.000000	0.000000	
TotalWorkingYears	2940.000000	11.279592	7.779458	0.000000	6.000000	
TrainingTimesLastYear	2940.000000	2.799320	1.289051	0.000000	2.000000	
WorkLifeBalance	2940.000000	2.761224	0.706356	1.000000	2.000000	
YearsAtCompany	2940.000000	7.008163	6.125483	0.000000	3.000000	
YearsInCurrentRole	2940.000000	4.229252	3.622521	0.000000	2.000000	
YearsSinceLastPromotion	2940.000000	2.187755	3.221882	0.000000	0.000000	
YearsWithCurrManager	2940.000000	4.123129	3.567529	0.000000	2.000000	

In [10]:
```
pip install dataprep
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Requirement already satisfied: dataprep in c:\users\jangi\appdata\roaming
\python\python310\site-packages (0.4.5)
Requirement already satisfied: wordcloud<2.0,>=1.8 in c:\users\jangi\appda
ta\roaming\python\python310\site-packages (from dataprep) (1.9.2)
Requirement already satisfied: nltk<4.0.0,>=3.6.7 in c:\programdata\anacon
da3\lib\site-packages (from dataprep) (3.7)
Requirement already satisfied: rapidfuzz<3.0.0,>=2.1.2 in c:\users\jangi\a
ppdata\roaming\python\python310\site-packages (from dataprep) (2.15.2)
Requirement already satisfied: numpy<2.0,>=1.21 in c:\programdata\anaconda
3\lib\site-packages (from dataprep) (1.23.5)
Requirement already satisfied: dask[array,dataframe,delayed]>=2022.3.0 in
c:\programdata\anaconda3\lib\site-packages (from dataprep) (2022.7.0)
Requirement already satisfied: jinja2<3.1,>=3.0 in c:\users\jangi\appdata
\roaming\python\python310\site-packages (from dataprep) (3.0.3)
Requirement already satisfied: ipywidgets<8.0,>=7.5 in c:\programdata\anac
onda3\lib\site-packages (from dataprep) (7.6.5)
Requirement already satisfied: bokeh<3,>=2 in c:\programdata\anaconda3\lib
```

In [264]:
```
from dataprep.eda import create_report
report = create_report(df, title= 'Data Report')
report
```

```
  0%|          | 0/4747 [00:00<?, ?it/s]
```

Out[264]:

Data Report

Data Report Overview

Variables ≡

Age Attrition BusinessTravel DailyRate Department DistanceFromHome Education EducationField EmployeeCount EmployeeNumber EnvironmentSatisfaction Gender HourlyRate JobInvolvement JobLevel JobRole JobSatisfaction MaritalStatus MonthlyIncome MonthlyRate NumCompaniesWorked Over18 OverTime PercentSalaryHike PerformanceRating RelationshipSatisfaction StandardHours StockOptionLevel TotalWorkingYears TrainingTimesLastYear WorkLifeBalance YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager

Interactions Correlations Missing Values

Overview

Dataset Statistics

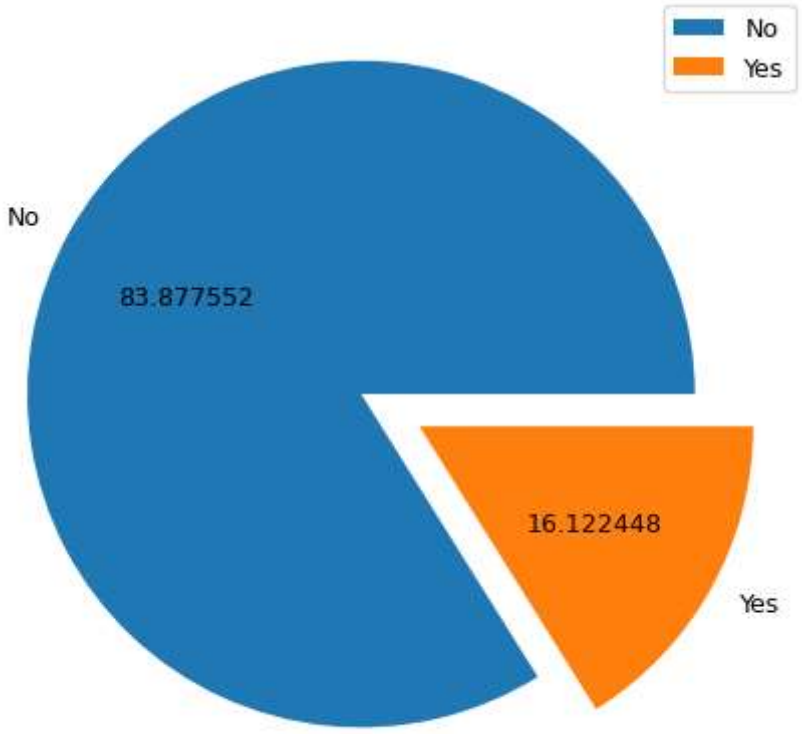
In [76]:
```
df.columns
```

Out[76]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

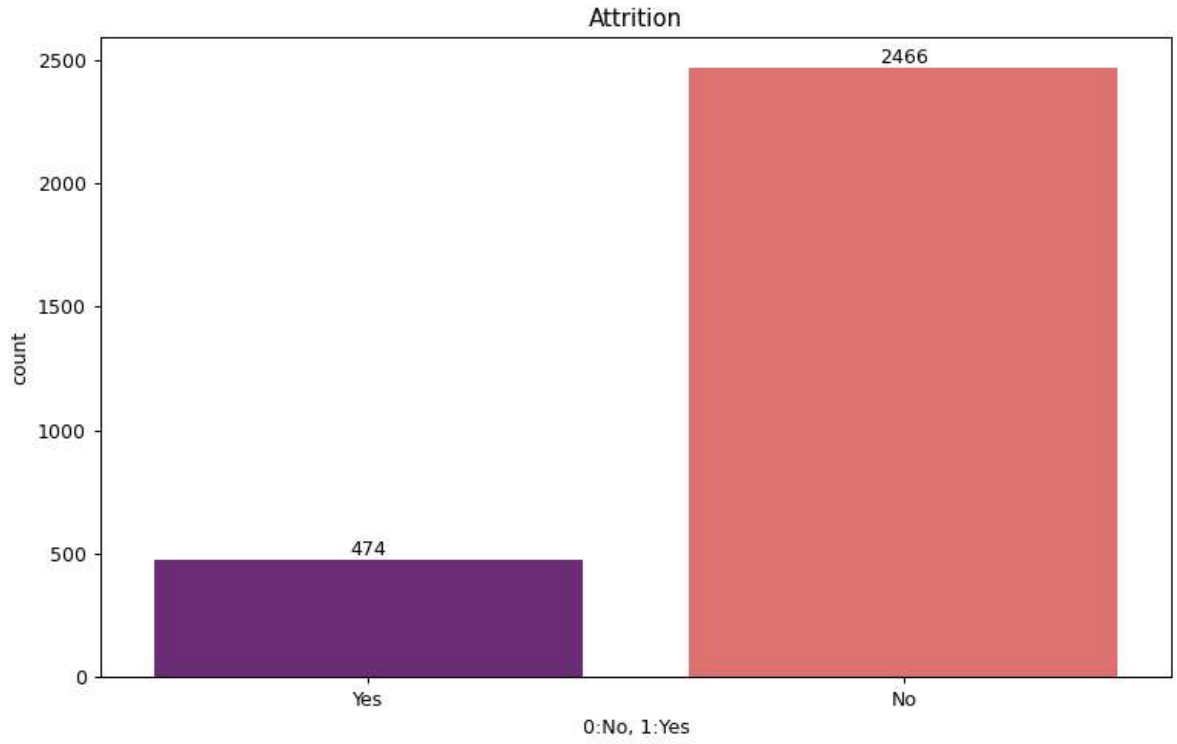
In [79]:
```
categorical_col = df.select_dtypes(include = ['object']).columns
numerical_col = df.select_dtypes(exclude = ['object']).columns
```

In [80]:
```
df['Gender'].replace(['F'],'Female', inplace = True)
df['MaritalStatus'].replace(['M'],'Married', inplace = True)
```

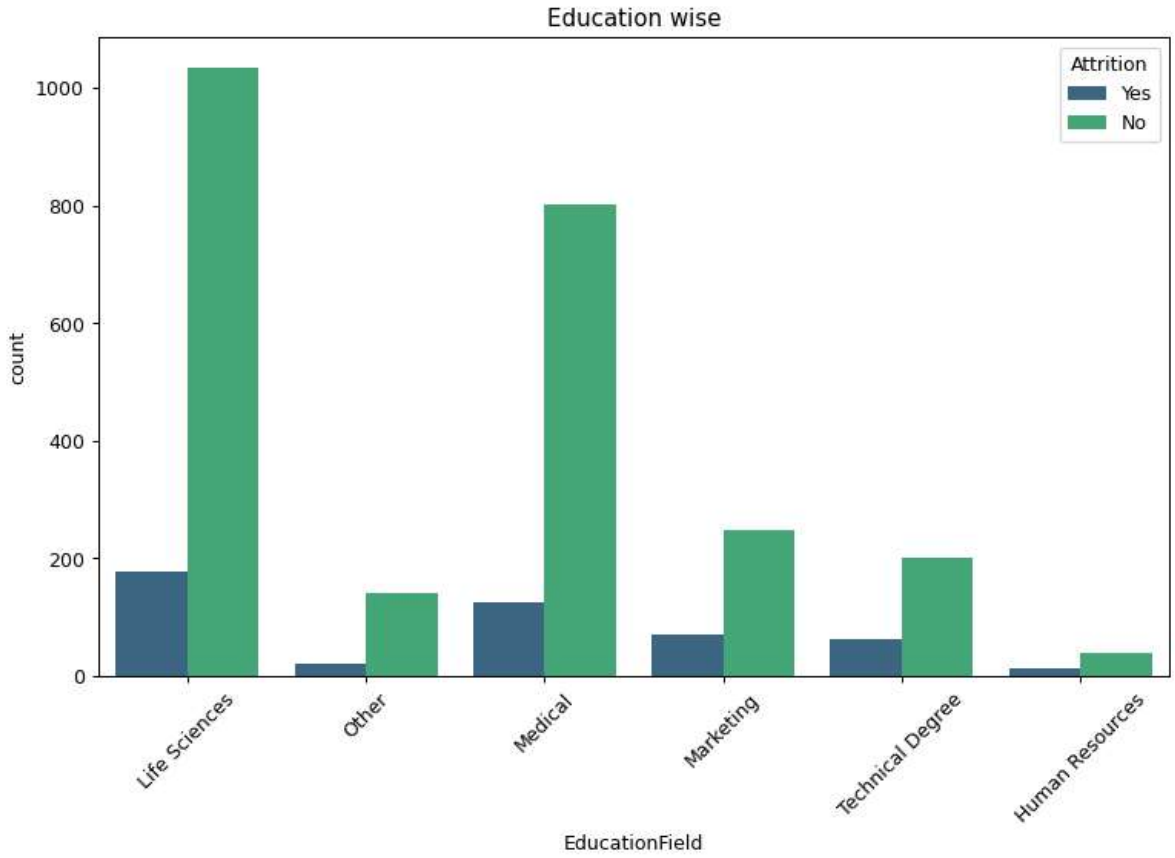
In [81]:
```
plt.figure(figsize = (10,6), dpi = 90)
plt.pie(df['Attrition'].value_counts(), labels = df['Attrition'].value_cou
        autopct = "%2f", explode = (0.1,0.1))
plt.legend()
plt.show()
```



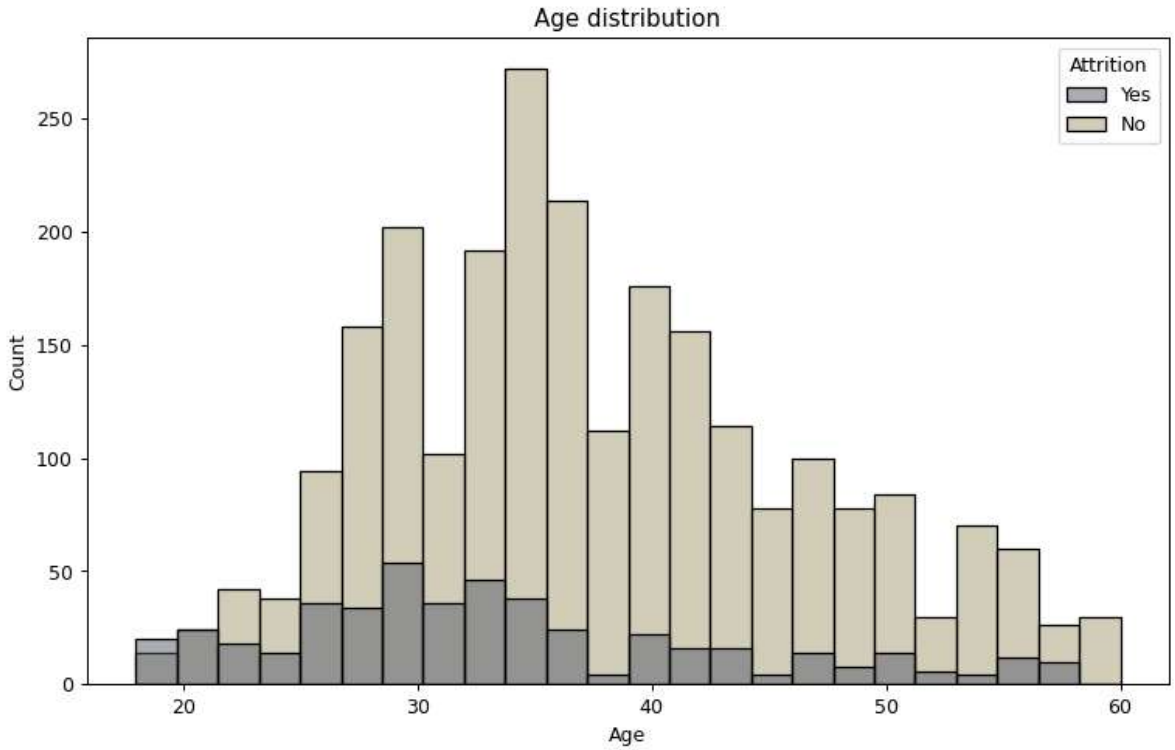
In [14]:
```
plt.figure(figsize = (10,6), dpi = 90)
ax = sns.countplot(x = "Attrition",data = df , palette='magma')
plt.title('Attrition')
plt.xlabel('0:No, 1:Yes')
for i in ax.containers:
    ax.bar_label(i)
```



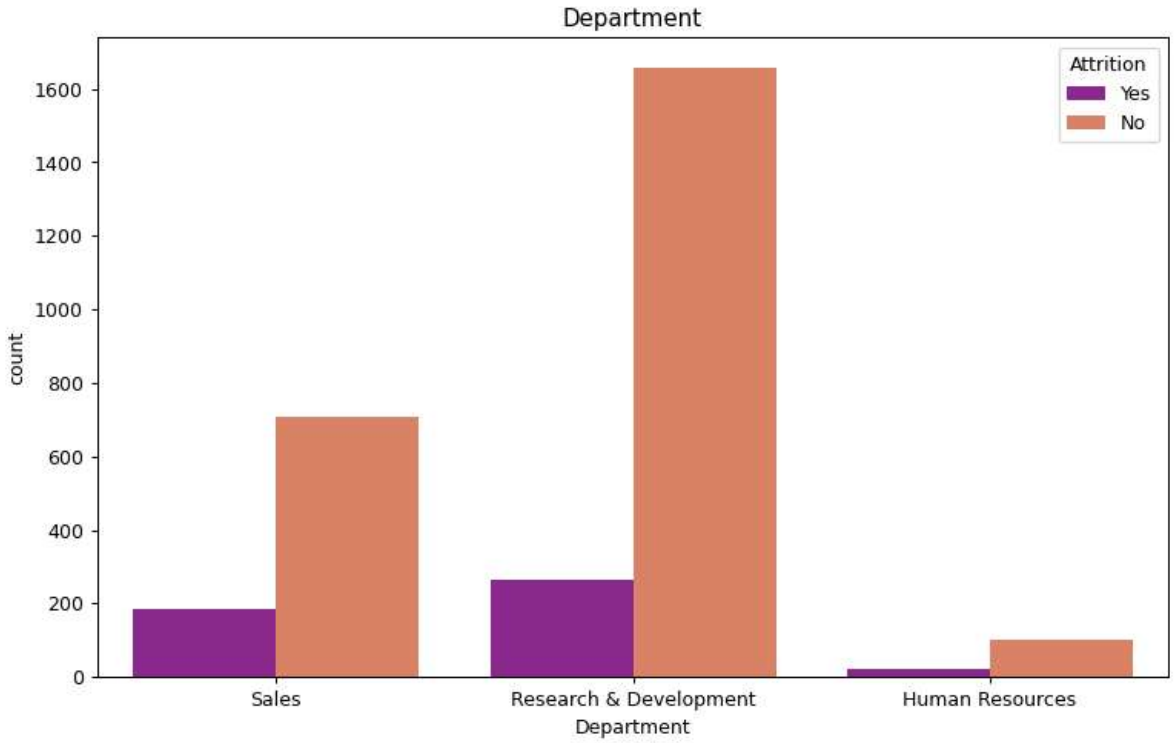
In [15]:
```
plt.figure(figsize = (10,6), dpi = 90)
sns.countplot(x = "EducationField",hue='Attrition', data = df, palette='vi
plt.title('Education wise')
plt.xticks(rotation=45)
plt.show()
```



In [16]:
```
plt.figure(figsize = (10,6), dpi = 90)
sns.histplot(x = "Age", hue='Attrition', data = df, palette='cividis')
plt.title('Age distribution')
plt.show()
```

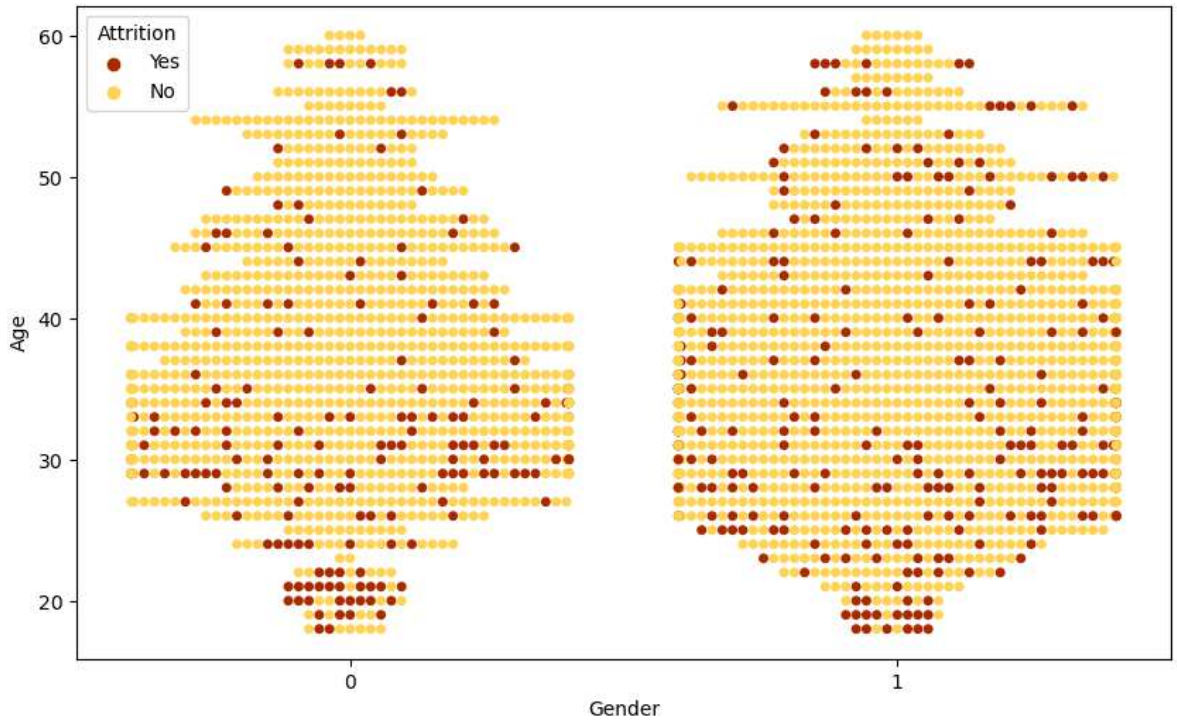


In [17]:
```
plt.figure(figsize = (10,6), dpi = 90)
ax = sns.countplot(x = "Department", hue='Attrition', data = df, palette=
plt.title('Department')
plt.show()

```



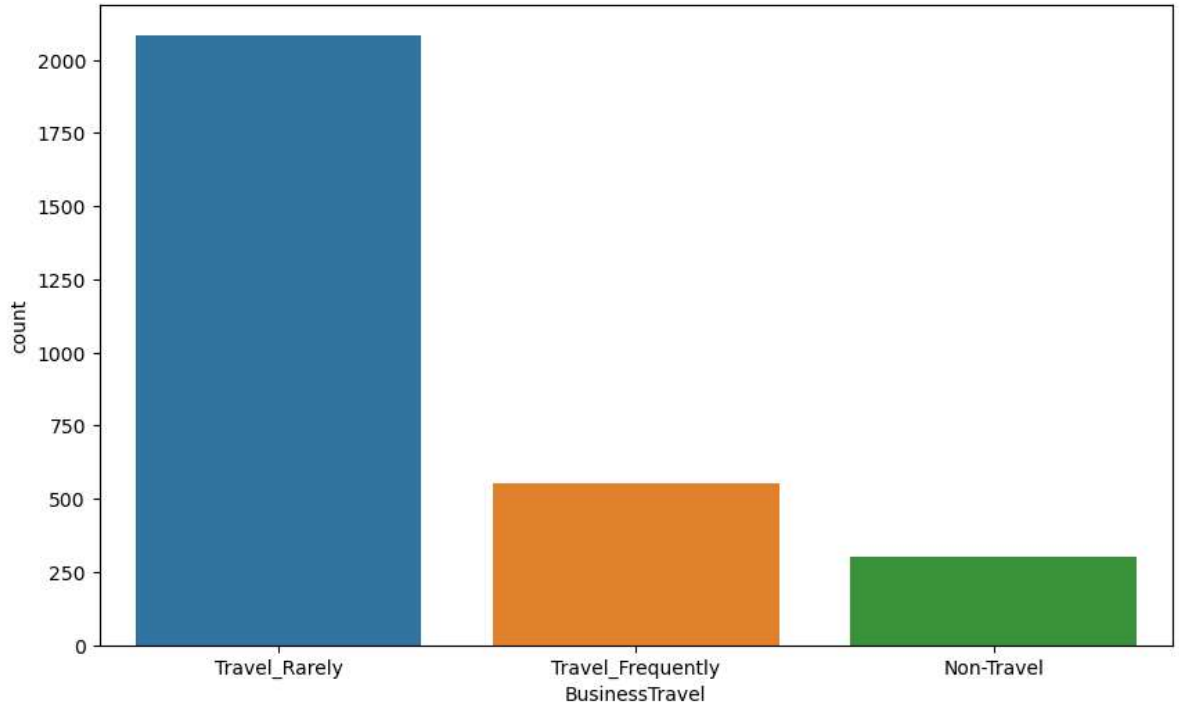
In [265]:
```
plt.figure(figsize = (10,6), dpi = 100)
ax = sns.swarmplot(x = "Gender", y = "Age", hue = "Attrition" ,data = df,
plt.xlabel("Gender")
plt.ylabel("Age")

plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as fl
oats or dates. If these strings should be plotted as numbers, cast to the app
ropriate data type before plotting.
Using categorical units to plot a list of strings that are all parsable as fl
oats or dates. If these strings should be plotted as numbers, cast to the app
ropriate data type before plotting.
```



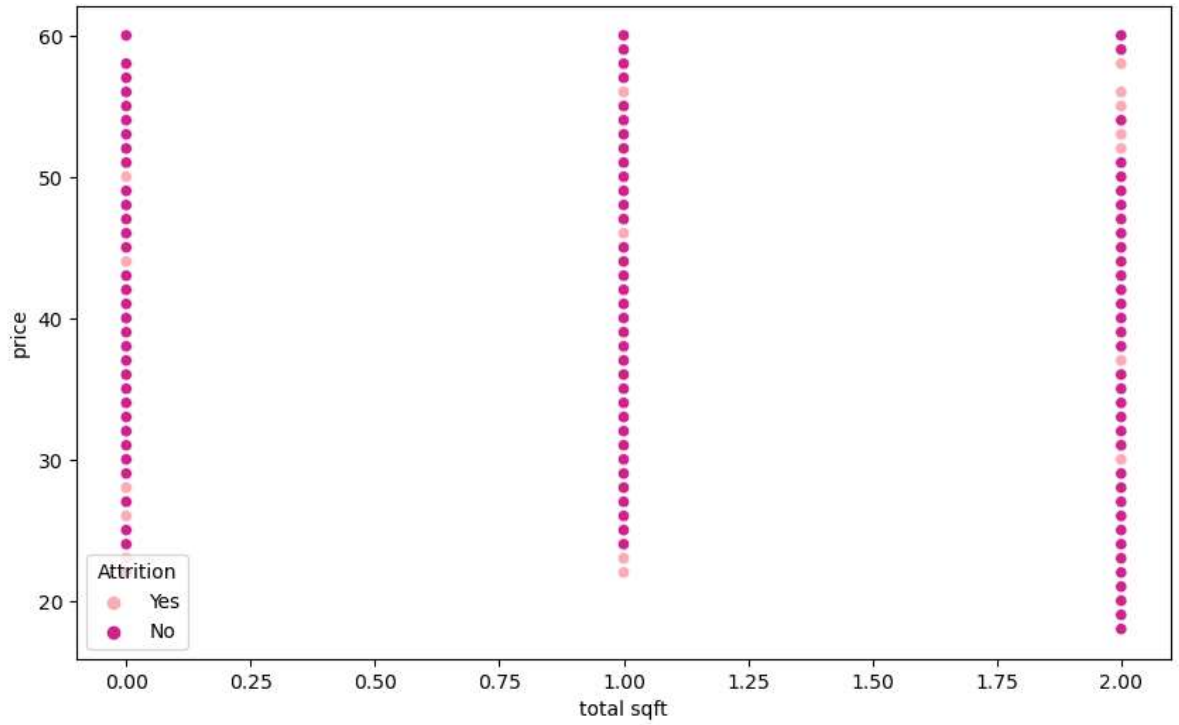
In [266]:
```
plt.figure(figsize = (10,6))
print(df['BusinessTravel'].value_counts())
sns.countplot(x = "BusinessTravel", data = df)
plt.show()
```

```
Travel_Rarely        2086
Travel_Frequently     554
Non-Travel            300
Name: BusinessTravel, dtype: int64
```



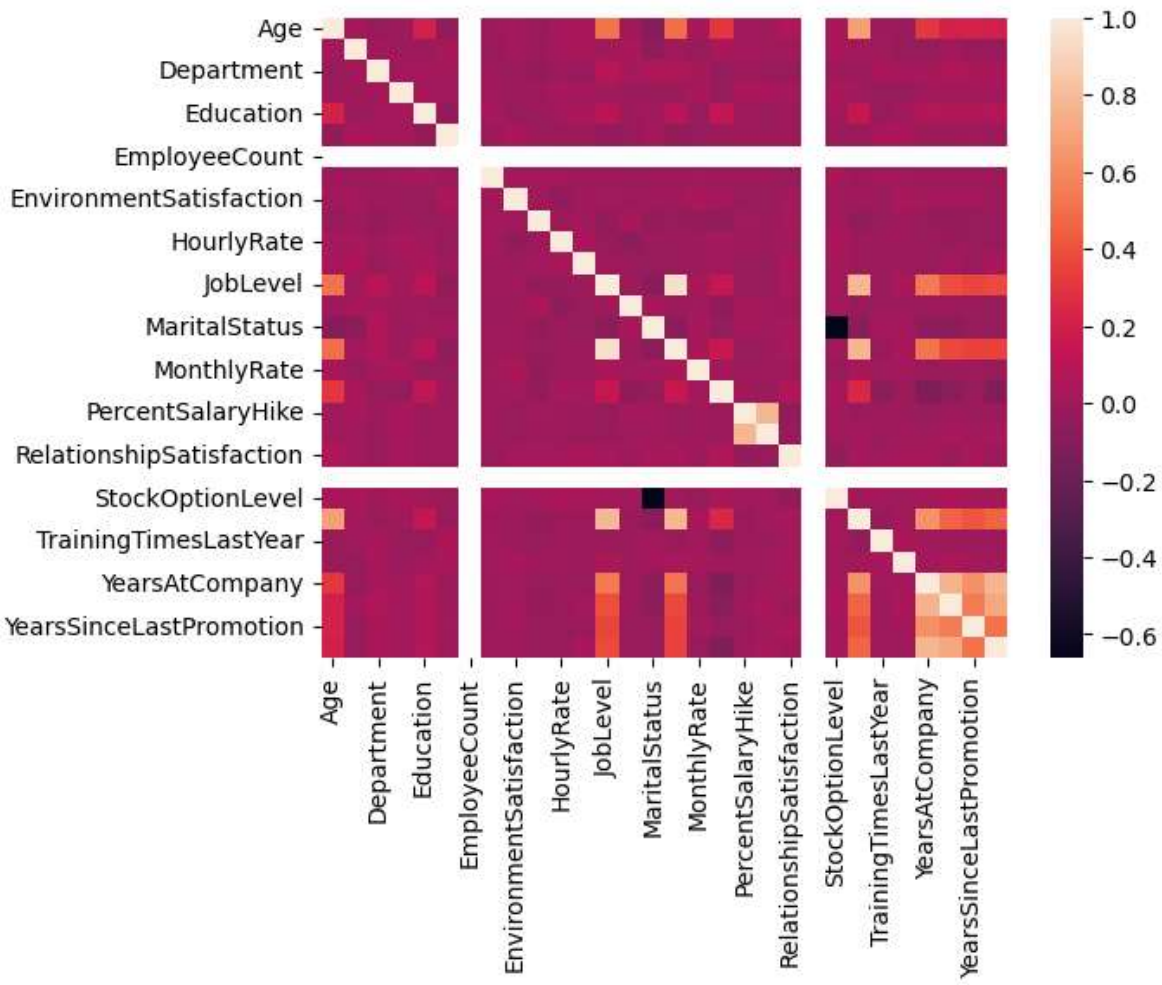
In [267]:

```
plt.figure(figsize = (10,6), dpi = 100)
ax = sns.scatterplot(x = "MaritalStatus", y = "Age", hue = "Attrition" ,da
plt.xlabel("total sqft")
plt.ylabel("price")

plt.show()
```



In [268]:
```
1 sns.heatmap(df.corr())
```

Out[268]: <Axes: >



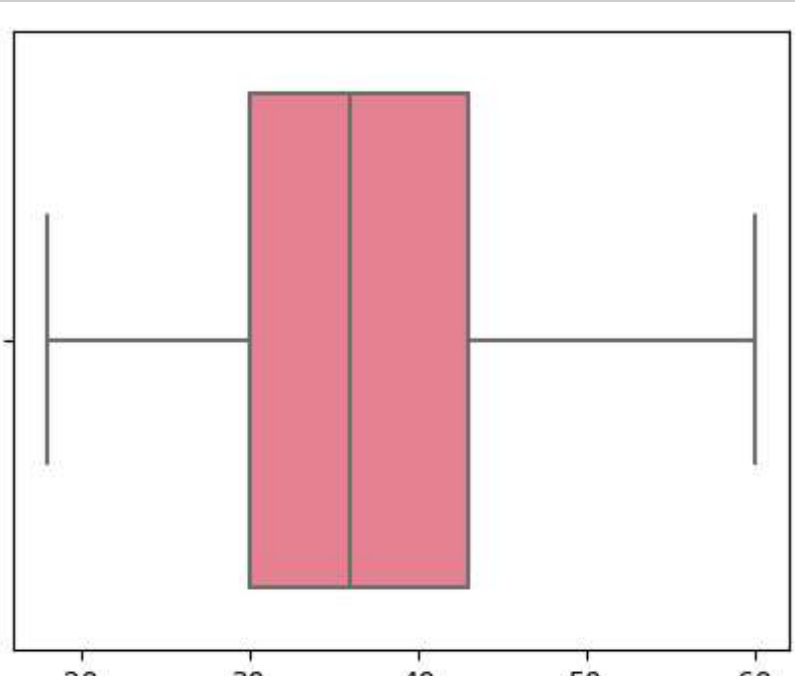
Check the Outliers

- A few data points that are significently different from the rest of the data points
- if any data points is far from the mean values can be treated as outliers
- outliers are only checked for numerical values
- to detect outliers we used boxplots

In [72]:
```
df.columns
```

Out[72]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [269]:
```
def boxplots(col):
    plt.figure(figsize =(5,4))
    sns.boxplot(df, x=col, palette = 'husl')
    plt.show()

for i in list(df.select_dtypes(exclude=['object']).columns)[0:]:
    boxplots(i)
```



In [270]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2940 entries, 0 to 2939
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       2940 non-null   int64 
 1   Attrition                 2940 non-null   object
 2   BusinessTravel            2940 non-null   object
 3   DailyRate                 2940 non-null   int64 
 4   Department                2940 non-null   int64 
 5   DistanceFromHome          2940 non-null   int64 
 6   Education                 2940 non-null   int64 
 7   EducationField            2940 non-null   int64 
 8   EmployeeCount             2940 non-null   int64 
 9   EmployeeNumber            2940 non-null   int64 
 10  EnvironmentSatisfaction   2940 non-null   int64 
 11  Gender                    2940 non-null   int64 
 12  HourlyRate                2940 non-null   int64 
 13  JobInvolvement            2940 non-null   int64 
 14  JobLevel                  2940 non-null   int64 
 15  JobRole                   2940 non-null   object
 16  JobSatisfaction           2940 non-null   int64 
 17  MaritalStatus             2940 non-null   int64 
 18  MonthlyIncome             2940 non-null   int64 
 19  MonthlyRate               2940 non-null   int64 
 20  NumCompaniesWorked        2940 non-null   int64 
 21  Over18                    2940 non-null   object
 22  OverTime                  2940 non-null   object
 23  PercentSalaryHike         2940 non-null   int64 
 24  PerformanceRating         2940 non-null   int64 
 25  RelationshipSatisfaction  2940 non-null   int64 
 26  StandardHours             2940 non-null   int64 
 27  StockOptionLevel          2940 non-null   int64 
 28  TotalWorkingYears         2940 non-null   int64 
 29  TrainingTimesLastYear     2940 non-null   int64 
 30  WorkLifeBalance           2940 non-null   int64 
 31  YearsAtCompany            2940 non-null   int64 
 32  YearsInCurrentRole        2940 non-null   int64 
 33  YearsSinceLastPromotion   2940 non-null   int64 
 34  YearsWithCurrManager      2940 non-null   int64 
dtypes: int64(30), object(5)
memory usage: 804.0+ KB
```

In [271]:
```
1 df.describe(include='object').T
```

Out[271]:

	count	unique	top	freq
Attrition	2940	2	No	2466
BusinessTravel	2940	3	Travel_Rarely	2086
JobRole	2940	9	Sales Executive	652
Over18	2940	1	Y	2940
OverTime	2940	2	No	2108

In [272]:
```
1 df.describe(include='int').T
```

Out[272]:

	count	mean	std	min	25%	50%	75%
Age	2940.0	36.923810	9.133819	18.0	30.00	36.0	43.00
DailyRate	2940.0	802.485714	403.440447	102.0	465.00	802.0	1157.00
Department	2940.0	1.260544	0.527703	0.0	1.00	1.0	2.00
DistanceFromHome	2940.0	9.192517	8.105485	1.0	2.00	7.0	14.00
Education	2940.0	2.912925	1.023991	1.0	2.00	3.0	4.00
EducationField	2940.0	2.247619	1.331143	0.0	1.00	2.0	3.00
EmployeeCount	2940.0	1.000000	0.000000	1.0	1.00	1.0	1.00
EmployeeNumber	2940.0	1470.500000	848.849221	1.0	735.75	1470.5	2205.25
EnvironmentSatisfaction	2940.0	2.721769	1.092896	1.0	2.00	3.0	4.00
Gender	2940.0	0.600000	0.489981	0.0	0.00	1.0	1.00
HourlyRate	2940.0	65.891156	20.325969	30.0	48.00	66.0	84.00
JobInvolvement	2940.0	2.729932	0.711440	1.0	2.00	3.0	3.00
JobLevel	2940.0	2.063946	1.106752	1.0	1.00	2.0	3.00
JobSatisfaction	2940.0	2.728571	1.102658	1.0	2.00	3.0	4.00
MaritalStatus	2940.0	1.097279	0.729997	0.0	1.00	1.0	2.00
MonthlyIncome	2940.0	6502.931293	4707.155770	1009.0	2911.00	4919.0	8380.00
MonthlyRate	2940.0	14313.103401	7116.575021	2094.0	8045.00	14235.5	20462.00
NumCompaniesWorked	2940.0	2.693197	2.497584	0.0	1.00	2.0	4.00
PercentSalaryHike	2940.0	15.209524	3.659315	11.0	12.00	14.0	18.00
PerformanceRating	2940.0	3.153741	0.360762	3.0	3.00	3.0	3.00
RelationshipSatisfaction	2940.0	2.712245	1.081025	1.0	2.00	3.0	4.00
StandardHours	2940.0	80.000000	0.000000	80.0	80.00	80.0	80.00
StockOptionLevel	2940.0	0.793878	0.851932	0.0	0.00	1.0	1.00
TotalWorkingYears	2940.0	11.279592	7.779458	0.0	6.00	10.0	15.00
TrainingTimesLastYear	2940.0	2.799320	1.289051	0.0	2.00	3.0	3.00
WorkLifeBalance	2940.0	2.761224	0.706356	1.0	2.00	3.0	3.00
YearsAtCompany	2940.0	7.008163	6.125483	0.0	3.00	5.0	9.00
YearsInCurrentRole	2940.0	4.229252	3.622521	0.0	2.00	3.0	7.00
YearsSinceLastPromotion	2940.0	2.187755	3.221882	0.0	0.00	1.0	3.00
YearsWithCurrManager	2940.0	4.123129	3.567529	0.0	2.00	3.0	7.00

Fix outlier or Remove outlier

IQR Method -

- we can cap the value between the upper bound and lower bound

In [273]:
```
def outlier(data):
    q1 = data.quantile(0.25)
    q3 = data.quantile(0.75)
    iqr = q3 - q1
    upper_bound = q3 + 1.5 * iqr
    lower_bound = q1 - 1.5 * iqr
    return data.clip(upper_bound,lower_bound)
```

In [274]:
```
df.head()
```

Out[274]:

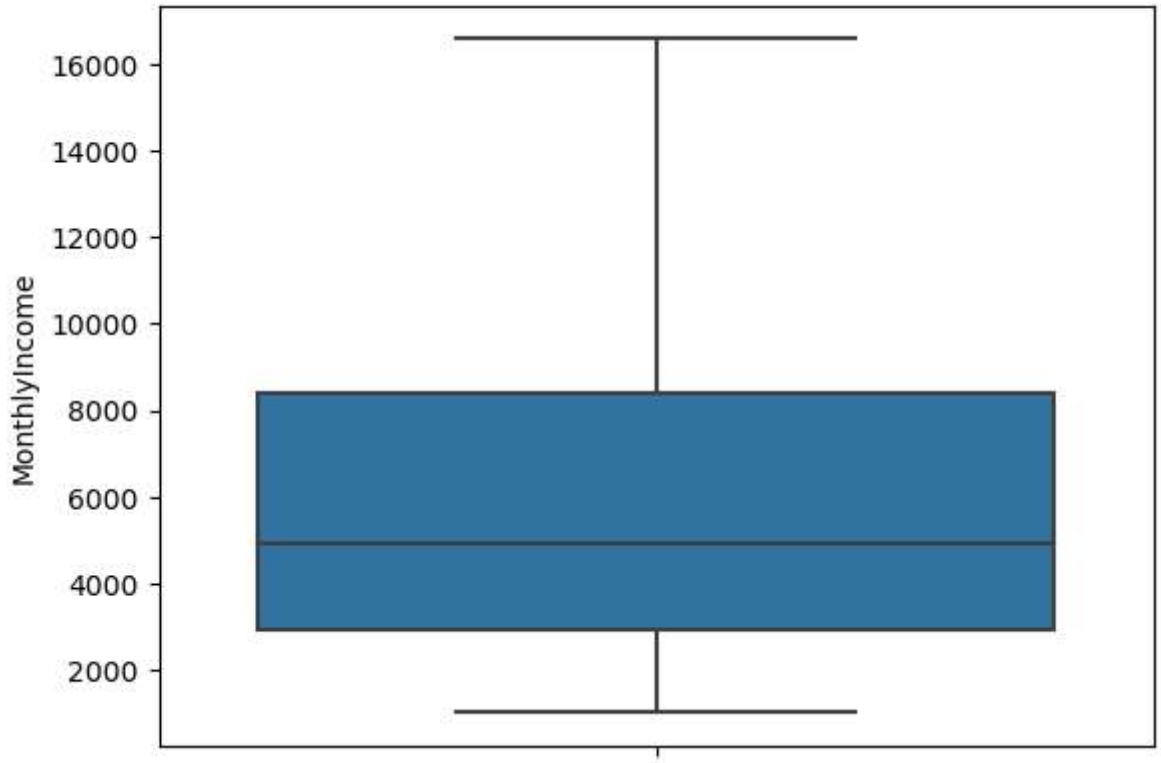
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
0	41	Yes	Travel_Rarely	1102	2	1	2	
1	49	No	Travel_Frequently	279	1	8	1	
2	37	Yes	Travel_Rarely	1373	1	2	2	
3	33	No	Travel_Frequently	1392	1	3	4	
4	27	No	Travel_Rarely	591	1	2	1	

5 rows × 35 columns

In [275]:
```
df["MonthlyIncome"] = outlier(df.MonthlyIncome)
```

In [276]:
```
1 sns.boxplot(y="MonthlyIncome" , data=df )
```

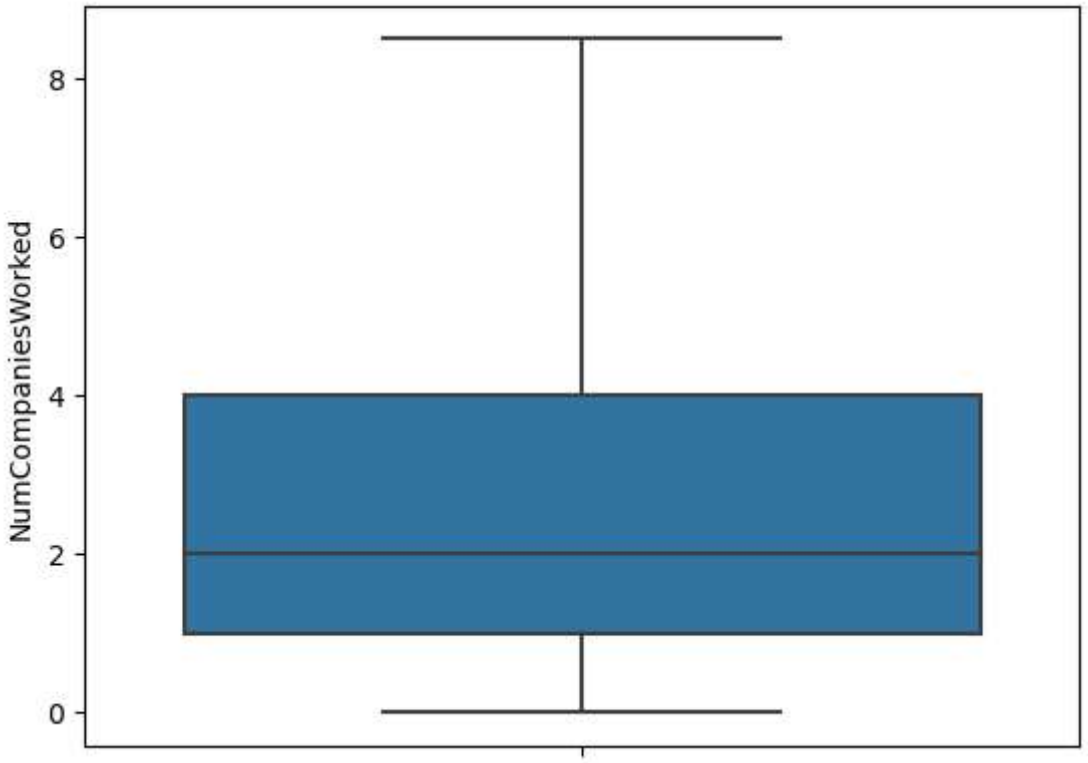
Out[276]: <Axes: ylabel='MonthlyIncome'>



In [277]:
```
1 df["NumCompaniesWorked"] = outlier(df.NumCompaniesWorked)
```

In [278]:
```
1 sns.boxplot(y="NumCompaniesWorked" , data=df )
```

Out[278]: <Axes: ylabel='NumCompaniesWorked'>



In [279]:
```
1 df["PerformanceRating"] = outlier(df.PerformanceRating)
```

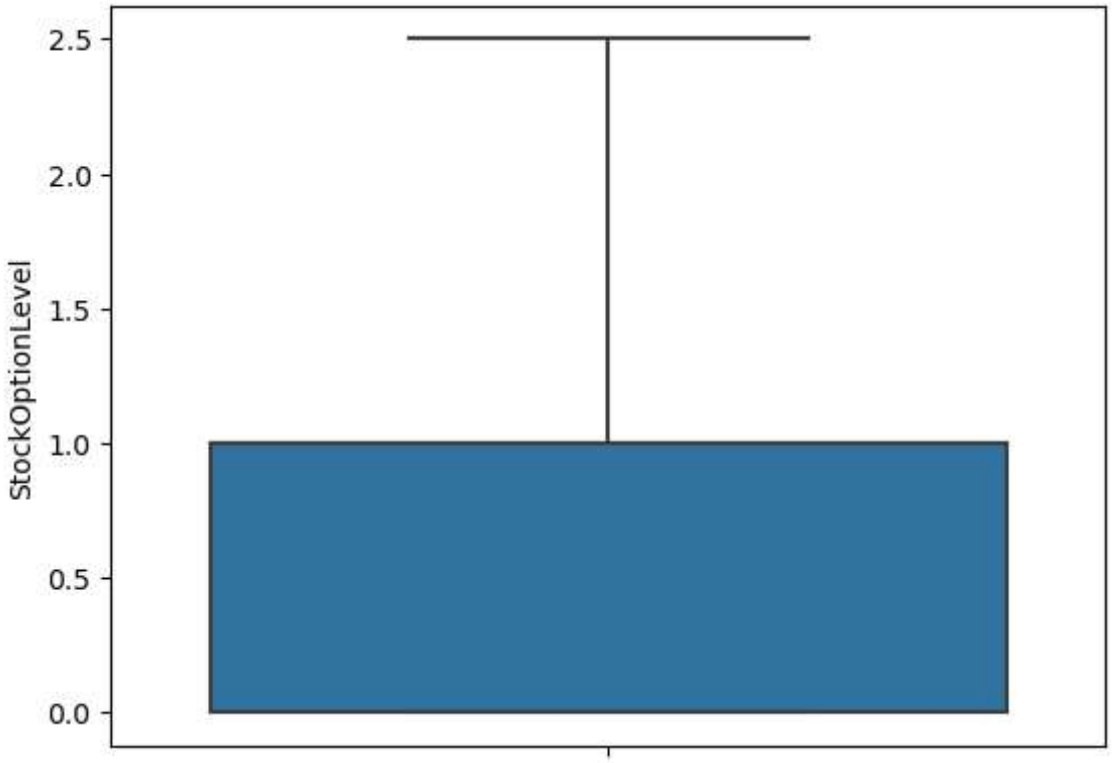
In [280]:
```
1 sns.boxplot(y="PerformanceRating" , data=df )
```

Out[280]: <Axes: ylabel='PerformanceRating'>

In [281]:
```
1 df["StockOptionLevel"] = outlier(df.StockOptionLevel)
```

In [282]:
```
1 sns.boxplot(y="StockOptionLevel" , data=df )
```

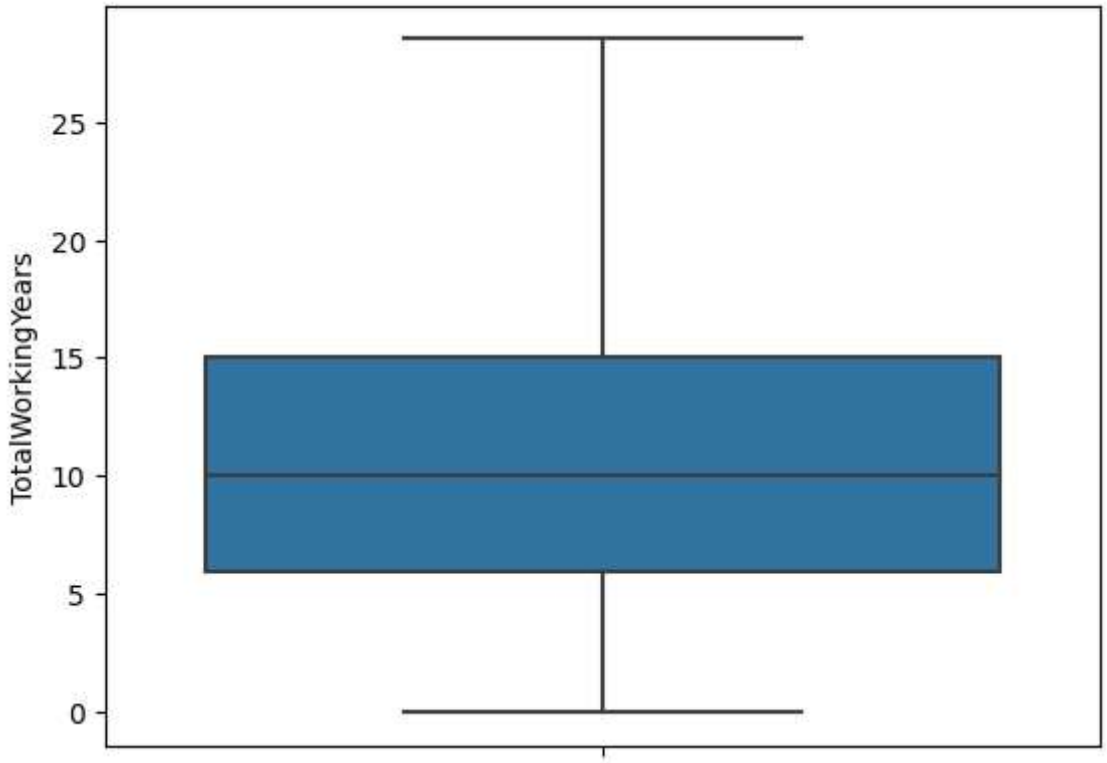
Out[282]: <Axes: ylabel='StockOptionLevel'>



In [283]:
```
1 df["TotalWorkingYears"] = outlier(df.TotalWorkingYears)
```

In [284]:
```
1 sns.boxplot(y="TotalWorkingYears" , data=df )
```

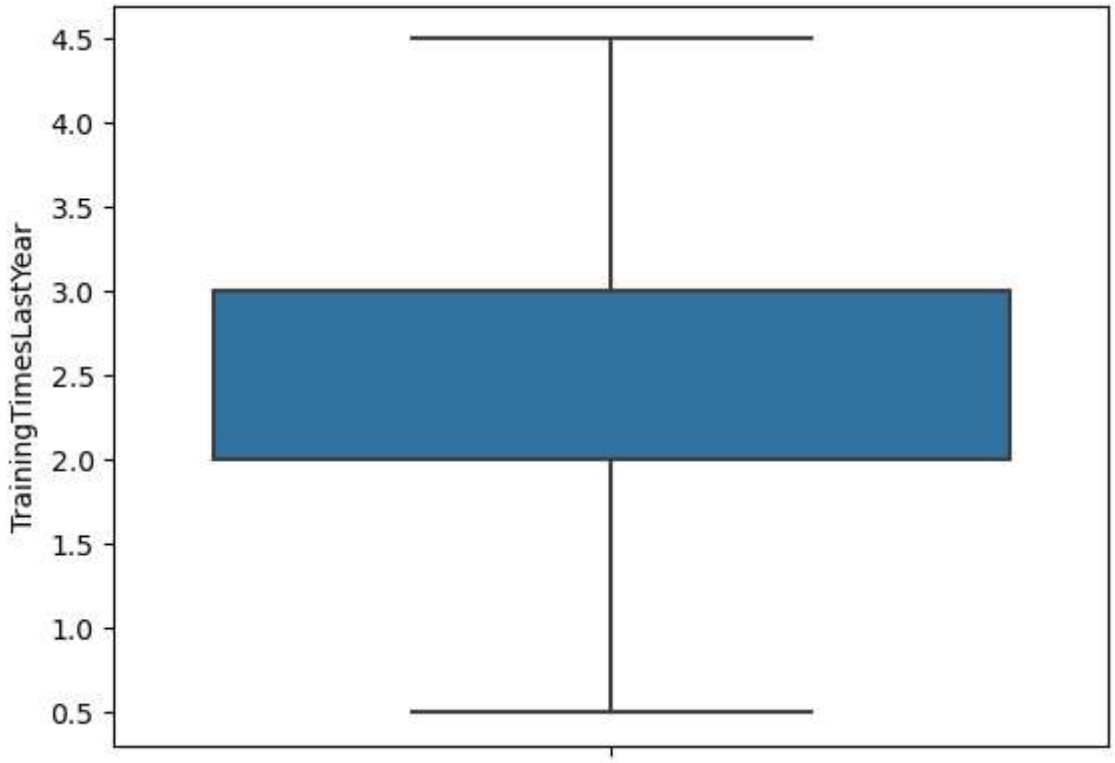
Out[284]: <Axes: ylabel='TotalWorkingYears'>



In [285]:
```
1 df["TrainingTimesLastYear"] = outlier(df.TrainingTimesLastYear)
```

In [286]:
```
1 sns.boxplot(y="TrainingTimesLastYear" , data=df )
```

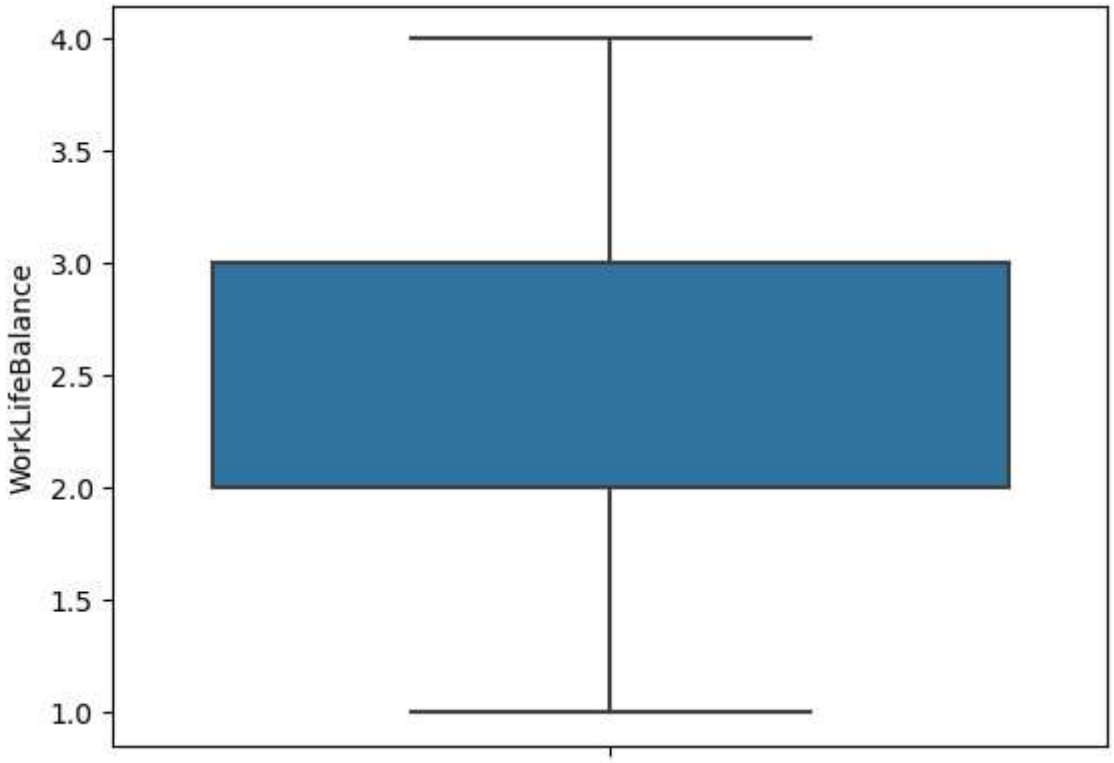
Out[286]: <Axes: ylabel='TrainingTimesLastYear'>



In [287]:
```
1 df["WorkLifeBalance"] = outlier(df.WorkLifeBalance)
```

In [288]:
```
1 sns.boxplot(y="WorkLifeBalance" , data=df )
```

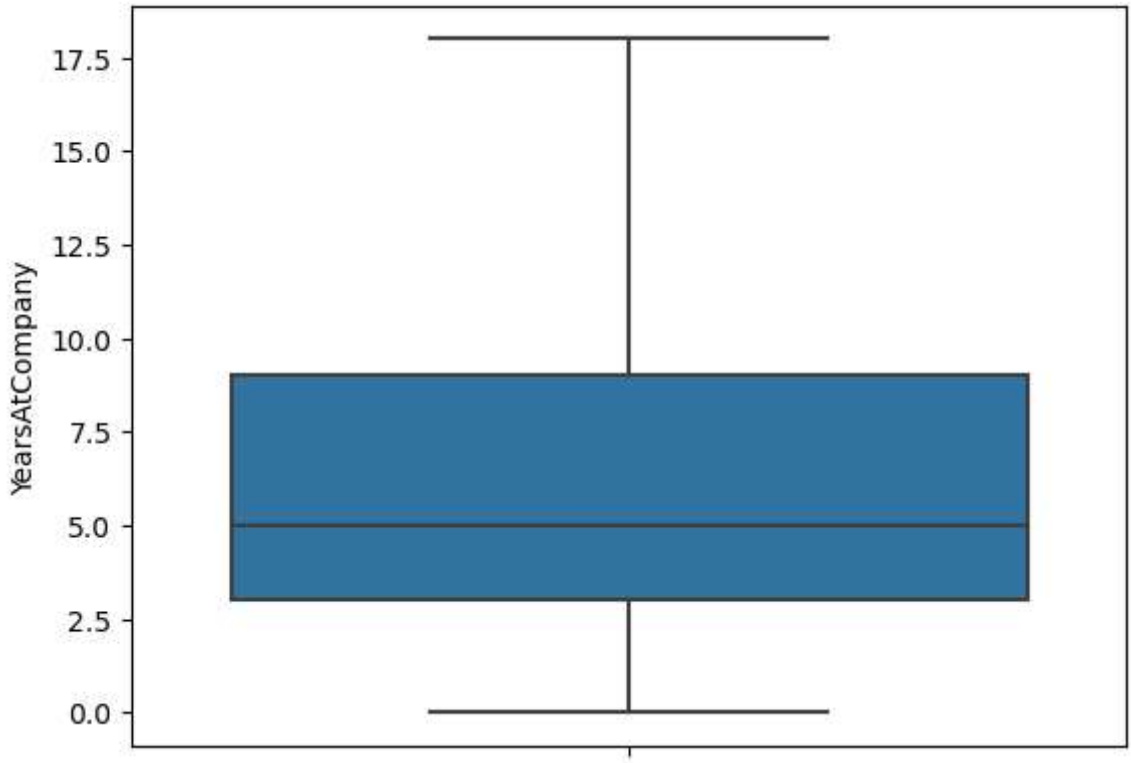
Out[288]: <Axes: ylabel='WorkLifeBalance'>



In [289]:
```
1 df["YearsAtCompany"] = outlier(df.YearsAtCompany)
```

In [290]:
```
1 sns.boxplot(y="YearsAtCompany" , data=df )
```

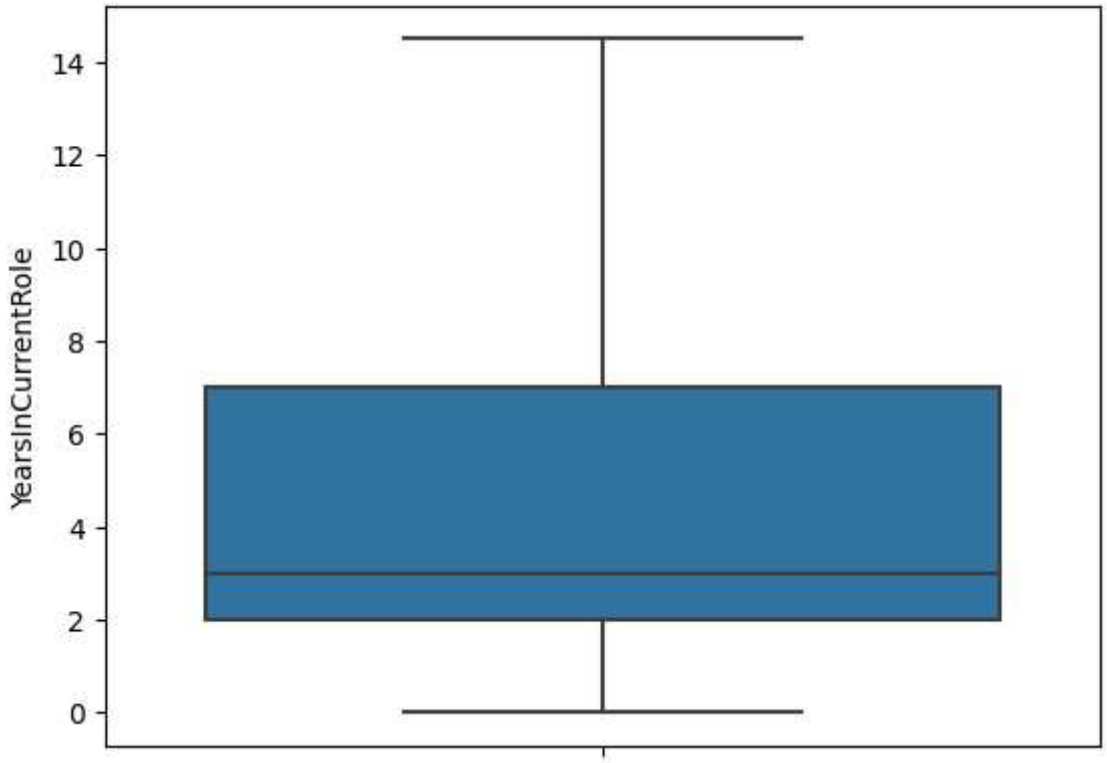
Out[290]: <Axes: ylabel='YearsAtCompany'>



In [291]:
```
1 df["YearsInCurrentRole"] = outlier(df.YearsInCurrentRole)
```

In [292]:
```
1 sns.boxplot(y="YearsInCurrentRole" , data=df )
```

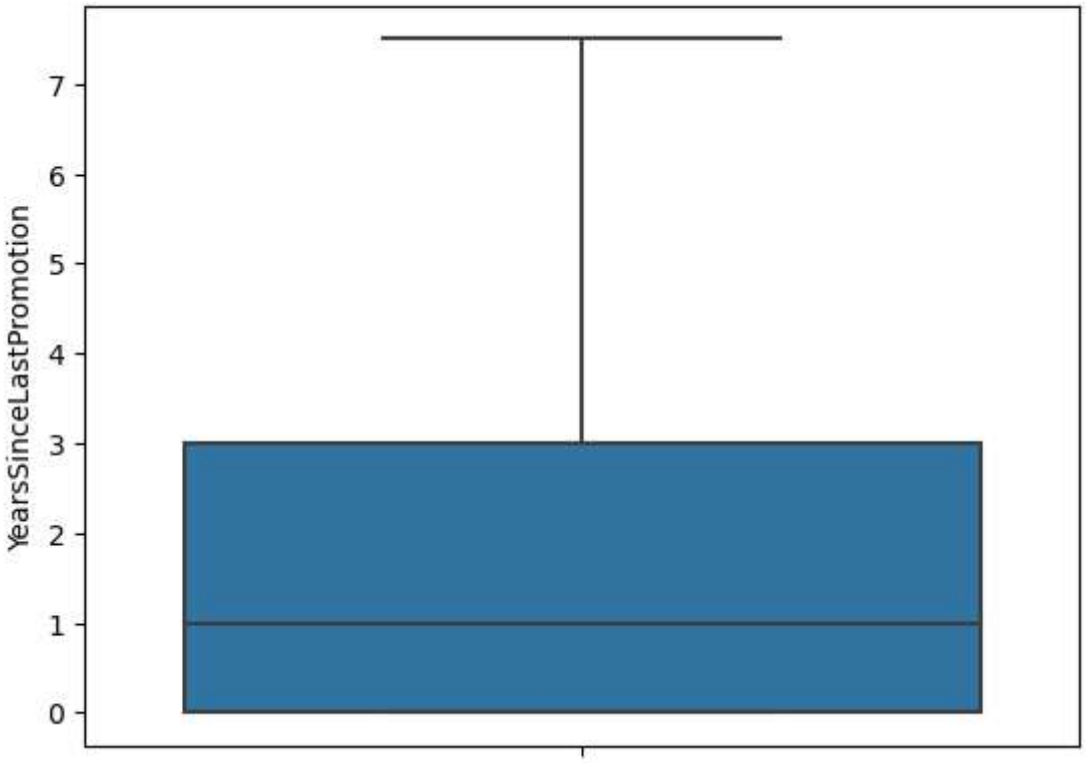
Out[292]: <Axes: ylabel='YearsInCurrentRole'>



In [293]:
```
1 df["YearsSinceLastPromotion"] = outlier(df.YearsSinceLastPromotion)
```

In [294]:
```
1 sns.boxplot(y="YearsSinceLastPromotion" , data=df )
```

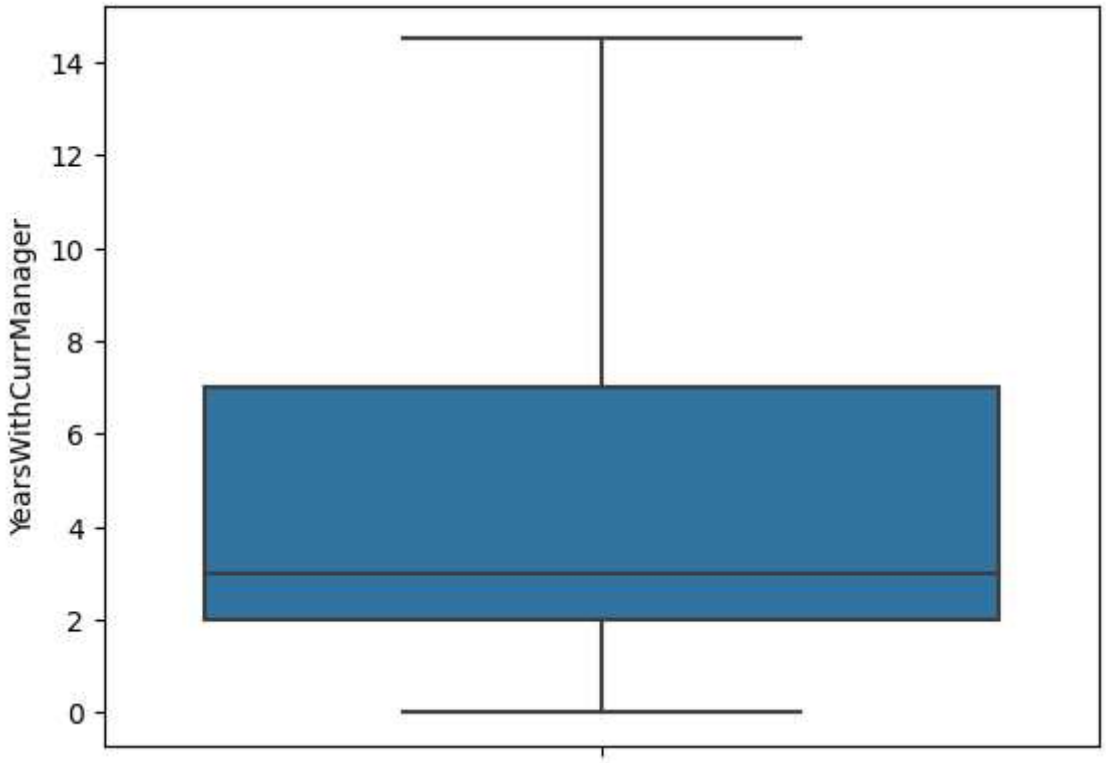
Out[294]: <Axes: ylabel='YearsSinceLastPromotion'>



In [295]:
```
1 df["YearsWithCurrManager"] = outlier(df.YearsWithCurrManager)
```

In [296]:
```
1 sns.boxplot(y="YearsWithCurrManager" , data=df )
```

Out[296]: <Axes: ylabel='YearsWithCurrManager'>



Models Buildings :

- logistic Regression
- Random Forest
- Decision Tree

In [11]:
```
1 correlations = df.corr()
2 correlations
```

Out[11]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount
Age	1.000000	0.010661	-0.001686	0.208034	NaN
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN
Education	0.208034	-0.016806	0.021042	1.000000	NaN
EmployeeCount	NaN	NaN	NaN	NaN	NaN
EmployeeNumber	-0.005175	-0.025742	0.016464	0.020950	NaN
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN
StandardHours	NaN	NaN	NaN	NaN	NaN
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN

26 rows × 26 columns

Model-1 : Logistic Regression Algorthim

In [12]:
```
df.head()
```

Out[12]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educ
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Lif
1	49	No	Travel_Frequently	279	Research & Development	8	1	Lif
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Lif
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [13]:
```
df['Gender'].replace(['F'],'Female', inplace = True)
df['MaritalStatus'].replace(['M'],'Married', inplace = True)
```

In [14]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df['Department'] = le.fit_transform(df['Department'])
df['EducationField'] = le.fit_transform(df['EducationField'])
df['Gender'] = le.fit_transform(df['Gender'])
df['MaritalStatus'] = le.fit_transform(df['MaritalStatus'])
```

Data(Train-Test) Split

In [15]:
```
x=df.drop(['Gender','Attrition','JobRole','Over18', 'OverTime','BusinessTr
y=df[['Gender']]
```

In [16]:
```
x.head(2)
```

Out[16]:

	Age	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount
0	41	1102	2	1	2	1	1
1	49	279	1	8	1	1	1

2 rows × 29 columns

In [17]:
```
1 x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2940 entries, 0 to 2939
Data columns (total 29 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       2940 non-null   int64
 1   DailyRate                 2940 non-null   int64
 2   Department                2940 non-null   int32
 3   DistanceFromHome          2940 non-null   int64
 4   Education                 2940 non-null   int64
 5   EducationField            2940 non-null   int32
 6   EmployeeCount             2940 non-null   int64
 7   EmployeeNumber            2940 non-null   int64
 8   EnvironmentSatisfaction   2940 non-null   int64
 9   HourlyRate                2940 non-null   int64
 10  JobInvolvement            2940 non-null   int64
 11  JobLevel                  2940 non-null   int64
 12  JobSatisfaction           2940 non-null   int64
 13  MaritalStatus             2940 non-null   int32
 14  MonthlyIncome             2940 non-null   int64
 15  MonthlyRate               2940 non-null   int64
 16  NumCompaniesWorked        2940 non-null   int64
 17  PercentSalaryHike         2940 non-null   int64
 18  PerformanceRating         2940 non-null   int64
 19  RelationshipSatisfaction  2940 non-null   int64
 20  StandardHours             2940 non-null   int64
 21  StockOptionLevel          2940 non-null   int64
 22  TotalWorkingYears         2940 non-null   int64
 23  TrainingTimesLastYear     2940 non-null   int64
 24  WorkLifeBalance           2940 non-null   int64
 25  YearsAtCompany            2940 non-null   int64
 26  YearsInCurrentRole        2940 non-null   int64
 27  YearsSinceLastPromotion   2940 non-null   int64
 28  YearsWithCurrManager      2940 non-null   int64
dtypes: int32(3), int64(26)
memory usage: 631.8 KB
```

In [18]:
```
1 y.head(2)
```

Out[18]:

	Gender
0	0
1	1

In [19]:
```
1 from sklearn.model_selection import train_test_split
2 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, r
```

In [20]:
```
1 x_train.shape , x_test.shape
```

Out[20]:
```
((2352, 29), (588, 29))
```

In [21]:
```
y_train.shape , y_test.shape
```

Out[21]:
```
((2352, 1), (588, 1))
```

Logistic Regression Method

In [22]:
```
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression(random_state= 100)
logit.fit(x_train, y_train)
```

Out[22]:
```
LogisticRegression(random_state=100)
```
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.

Prediction

In [23]:
```
y_pred_train_log = logit.predict(x_train)
y_pred_test_log = logit.predict(x_test)
```

Evaluate test data Accuracy

In [24]:
```
from sklearn.metrics import confusion_matrix,classification_report, accura
accuracy_log_test=accuracy_score(y_test,y_pred_test_log)
print('Logistic regression Test accuracy:', accuracy_score(y_test, y_pred_
```

```
Logistic regression Test accuracy: 0.6071428571428571
```

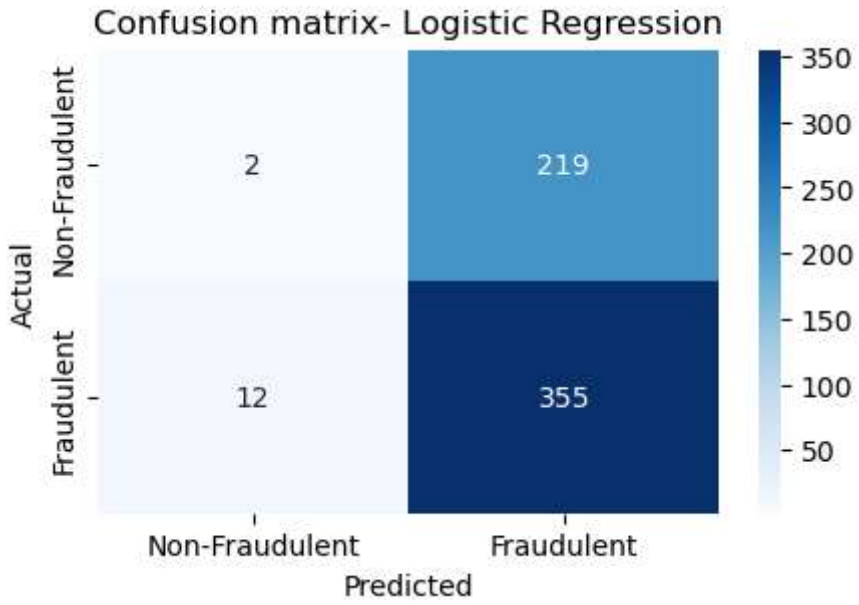
Evaluate train data Accuracy

In [25]:
```

from sklearn.metrics import confusion_matrix,classification_report, accura
accuracy_train = accuracy_score(y_train,y_pred_train_log)

print('Logistic regression Train accuracy:', accuracy_score(y_train, y_pre

```

```
Logistic regression Train accuracy: 0.594812925170068
```

Confusion Martrix - Logistic Regression

In [256]:
```
Labels = ['Non-Fraudulent', 'Fraudulent']
plt.figure(figsize=(5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_log),xticklabels=Labels,
            yticklabels=Labels,cmap='Blues',annot=True, fmt='g')
plt.title("Confusion matrix- Logistic Regression")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



AUC (Area under the curve) & ROC (Receiver operating characteristics)

- It is one of the most important evaluation metrics for checking classification model's performance.
- It is also written as AUROC (Area Under the Receiver Operating Characteristics)
- ROC is a probability curve and AUC represents the degree or measure of separability.
- It tells how much the model is capable of distinguishing between classes.

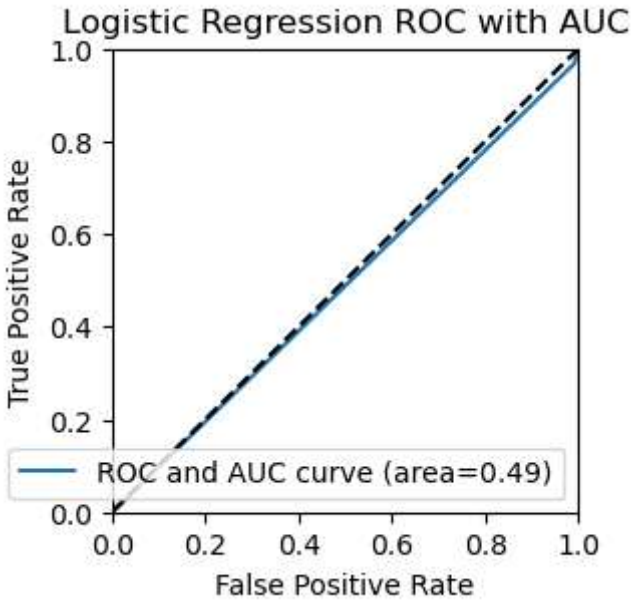
In [26]:
```
from sklearn.metrics import roc_auc_score
logit_roc_auc = roc_auc_score(y_test, y_pred_test_log)
print(logit_roc_auc)
```

```
0.48817611303586617
```

In [27]:
```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_log)
display(fpr[:10])
display(tpr[:10])
display(thresholds[:10])
```

```
array([0.        , 0.99095023, 1.        ])

array([0.        , 0.96730245, 1.        ])

array([2, 1, 0])
```

In [28]:
```
plt.figure(figsize=(3,3))
plt.plot(fpr, tpr, label="ROC and AUC curve (area=%0.2f)" % logit_roc_auc)
plt.plot([0,1],[0,1], 'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Logistic Regression ROC with AUC")
plt.legend(loc='lower right')
plt.show()
```



Model-2 : Random Forest Algorithm

Feature Scaling

In [29]:
```
1 x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2940 entries, 0 to 2939
Data columns (total 29 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       2940 non-null   int64
 1   DailyRate                 2940 non-null   int64
 2   Department                2940 non-null   int32
 3   DistanceFromHome          2940 non-null   int64
 4   Education                 2940 non-null   int64
 5   EducationField            2940 non-null   int32
 6   EmployeeCount             2940 non-null   int64
 7   EmployeeNumber            2940 non-null   int64
 8   EnvironmentSatisfaction   2940 non-null   int64
 9   HourlyRate                2940 non-null   int64
 10  JobInvolvement            2940 non-null   int64
 11  JobLevel                  2940 non-null   int64
 12  JobSatisfaction           2940 non-null   int64
 13  MaritalStatus             2940 non-null   int32
 14  MonthlyIncome             2940 non-null   int64
 15  MonthlyRate               2940 non-null   int64
 16  NumCompaniesWorked        2940 non-null   int64
 17  PercentSalaryHike         2940 non-null   int64
 18  PerformanceRating         2940 non-null   int64
 19  RelationshipSatisfaction  2940 non-null   int64
 20  StandardHours             2940 non-null   int64
 21  StockOptionLevel          2940 non-null   int64
 22  TotalWorkingYears         2940 non-null   int64
 23  TrainingTimesLastYear     2940 non-null   int64
 24  WorkLifeBalance           2940 non-null   int64
 25  YearsAtCompany            2940 non-null   int64
 26  YearsInCurrentRole        2940 non-null   int64
 27  YearsSinceLastPromotion   2940 non-null   int64
 28  YearsWithCurrManager      2940 non-null   int64
dtypes: int32(3), int64(26)
memory usage: 631.8 KB
```

In [30]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x1=sc.fit_transform(x)
pd.DataFrame(x1).head(2)
```

Out[30]:

	0	1	2	3	4	5	6	7	8	
0	0.446350	0.742527	1.401512	-1.010909	-0.891688	-0.937414	0.0	-1.731462	-0.660531	1.3
1	1.322365	-1.297775	-0.493817	-0.147150	-1.868426	-0.937414	0.0	-1.730284	0.254625	-0.2

2 rows × 29 columns

Check Balance Data

In [31]:
```
y.value_counts()
```

Out[31]:
```
Gender
1         1764
0         1176
dtype: int64
```

Conculsion - Data is Imbalaced

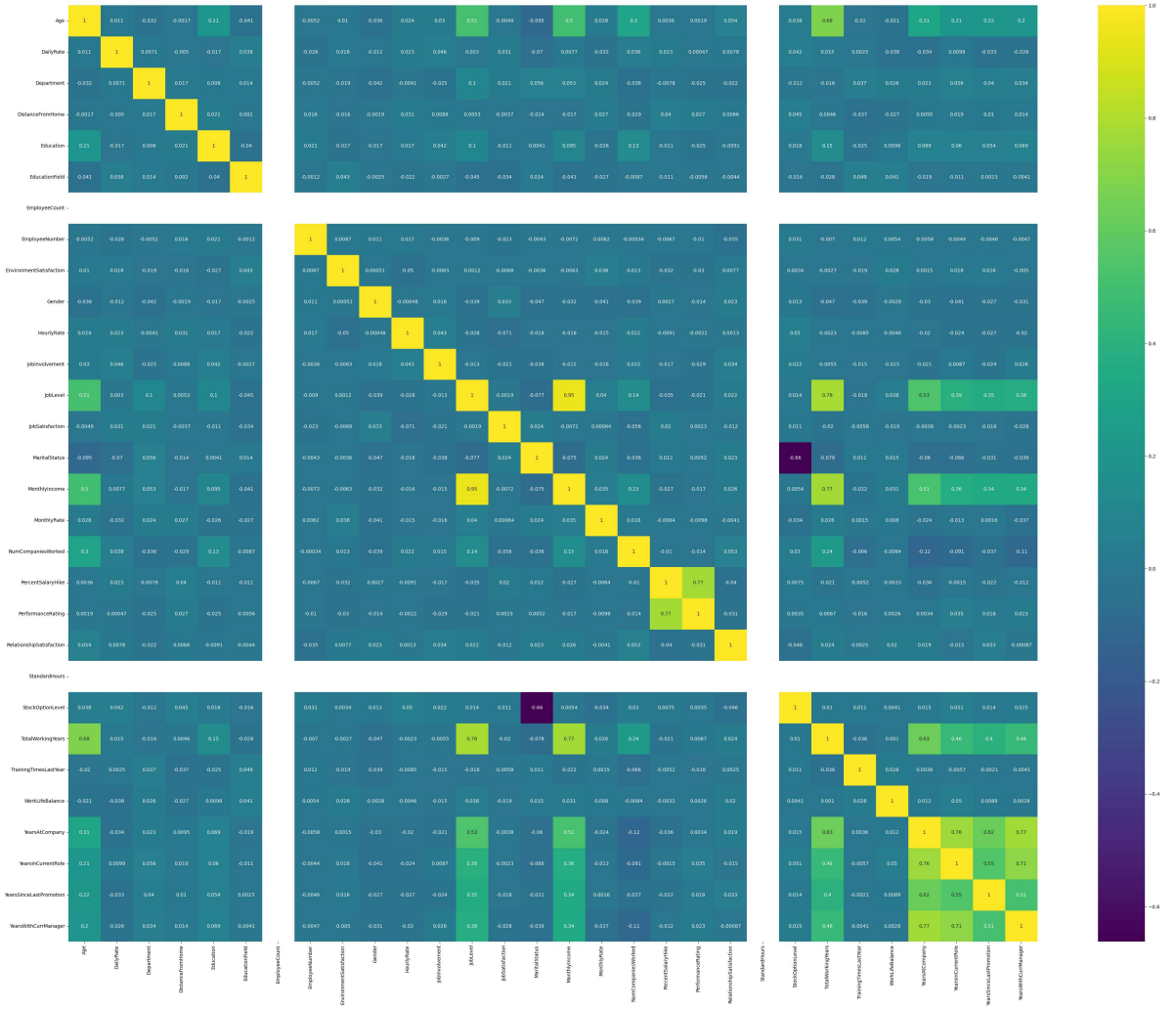
In [33]:
```
# Done Under Sampling to balaced the data
import imblearn
from imblearn.under_sampling import RandomUnderSampler
ros = RandomUnderSampler()
x_un,y_un = ros.fit_resample(x1,y)
print(x_un.shape,y_un.shape,y.shape)
```

```
(2352, 29) (2352, 1) (2940, 1)
```

In [34]:
```
y_un.value_counts()
```

Out[34]:
```
Gender
0         1176
1         1176
dtype: int64
```

In [41]:
```
plt.figure(figsize = (45, 35))
sns.heatmap(df.corr(), annot = True, cmap = 'viridis')
plt.show()
```



Model Building

In [43]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 200, oob_score = False)
rf.fit(x_train,y_train)
```

Out[43]:
```
RandomForestClassifier(n_estimators=200)
```

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.

Prediction

In [44]:
```
y_pred_train_rf = rf.predict(x_train)
y_pred_test_rf = rf.predict(x_test)
```

Evaluate

In [45]:
```
accuracy_rf_test = accuracy_score(y_test,y_pred_test_rf)
accuracy_rf_train = accuracy_score(y_train,y_pred_train_rf)
print('Random Forest - Train accuracy:', accuracy_score(y_train, y_pred_tr
print('-------'*10)
print('Random Forest - Test accuracy:', accuracy_score(y_test, y_pred_test
```

```
Random Forest - Train accuracy: 1.0
----------------------------------------------------------------------
Random Forest - Test accuracy: 0.9149659863945578
```

Confusion Matrix

In [46]:
```
print(confusion_matrix(y_test,y_pred_test_rf))
```

```
[[185  36]
 [ 14 353]]
```

In [47]:
```
print(confusion_matrix(y_train,y_pred_train_rf))
```

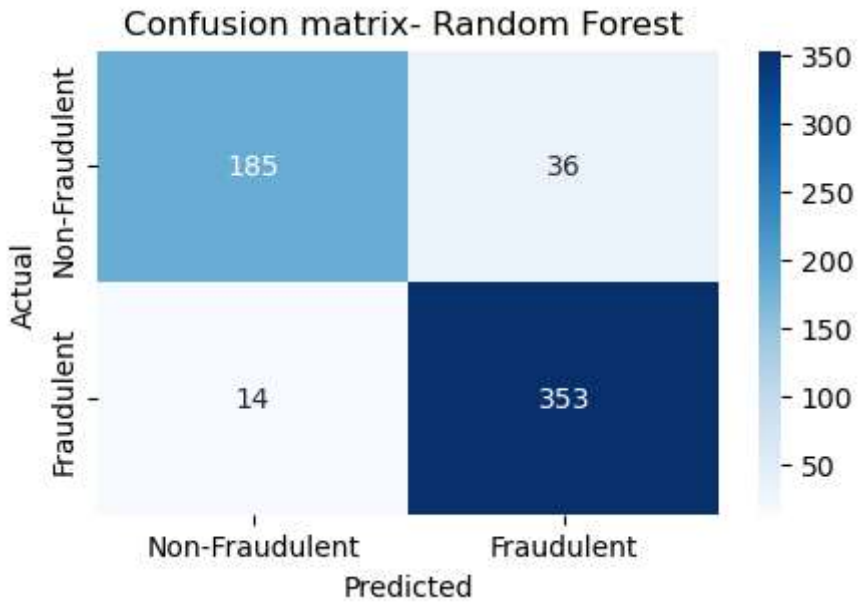
```
[[ 955    0]
 [   0 1397]]
```

In [53]:
```
print('Random Forest Train data accuracy')
acc = accuracy_score (y_train, y_pred_train_rf)
print('Accuracy score is', acc)

```

```
Random Forest Train data accuracy
Accuracy score is 1.0
```

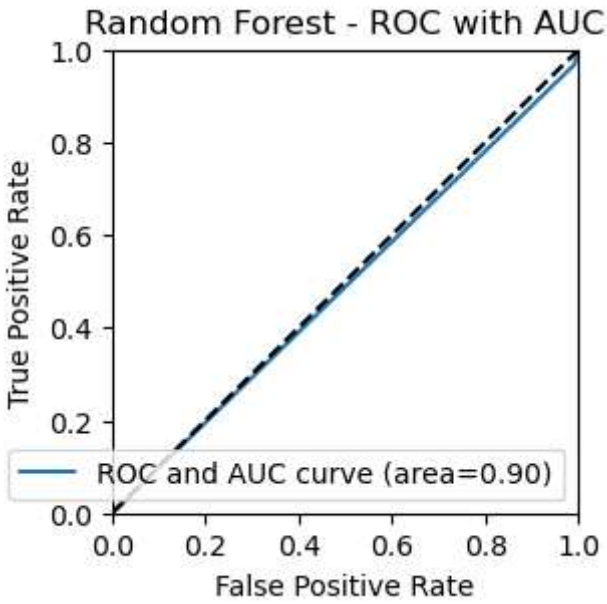
In [55]:

```
Labels = ['Non-Fraudulent', 'Fraudulent']
plt.figure(figsize = (5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_rf), xticklabels = Labels,
            yticklabels = Labels, cmap = 'Blues', annot = True, fmt = 'g')
plt.title("Confusion matrix- Random Forest ")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



In [56]:
```
rf_roc_auc = roc_auc_score(y_test, y_pred_test_rf)
print(rf_roc_auc)
plt.figure(figsize = (3,3))
plt.plot(fpr, tpr, label = "ROC and AUC curve (area=%0.2f)" % rf_roc_auc)
plt.plot([0,1],[0,1], 'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Random Forest - ROC with AUC")
plt.legend(loc = 'lower right')
plt.show()
```

```
0.8994784667168062
```



Cross validation because of underfitting issue

In [57]:
```
from sklearn.model_selection import cross_val_score
train_accuracy_rf = cross_val_score(rf, x_train, y_train, cv = 10)
crossval_train_rf = train_accuracy_rf.mean()
test_accuracy_rf = cross_val_score(rf, x_test, y_test, cv = 10)
crossval_test_rf = test_accuracy_rf.mean()

print('Random forest after Cross validation Train accuracy:', crossval_tra
print('-------'*10)
print('Random forest  after Cross validation Test accuracy:', crossval_tes
```

```
Random forest after Cross validation Train accuracy: 0.8877623512441399
----------------------------------------------------------------------
Random forest  after Cross validation Test accuracy: 0.6545587375803623
```

Model-3 : Decision Tree

- A decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

Model building

In [58]:
```
from sklearn.tree import DecisionTreeClassifier,plot_tree
dtree = DecisionTreeClassifier()
dtree.fit(x_train,y_train)
```

Out[58]:
```
DecisionTreeClassifier()
```
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.

Prediction

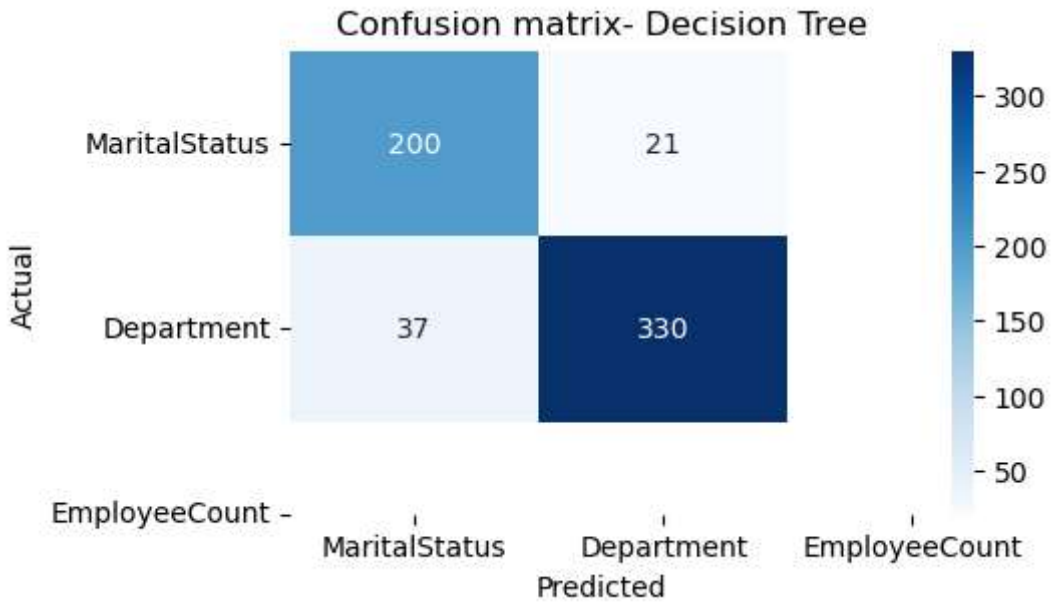
In [60]:
```
y_pred_train_dtree = dtree.predict(x_train)
y_pred_test_dtree = dtree.predict(x_test)
```

Evaluate

In [61]:
```
accuracy_dtree_test = accuracy_score(y_test,y_pred_test_dtree)
accuracy_dtree_train = accuracy_score(y_train,y_pred_train_dtree)

print('Decision Tree - Train accuracy:', accuracy_score(y_train, y_pred_tr
print('-------'*10)
print('Decision Tree - Test accuracy:', accuracy_score(y_test, y_pred_test
```

```
Decision Tree - Train accuracy: 1.0
----------------------------------------------------------------------
Decision Tree - Test accuracy: 0.9013605442176871
```

In [63]:
```
Labels = ['MaritalStatus','Department','EmployeeCount']
plt.figure(figsize = (5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_dtree),xticklabels = Label
            yticklabels = Labels, cmap = 'Blues', annot = True, fmt = 'g')
plt.title("Confusion matrix- Decision Tree")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



Using Post prunning method to handle overfitting problem

In [66]:
```
def dtree_model(model):
    model_preds = model.predict(x_test)
    print(classification_report(y_test,model_preds))
    print('\n')
    plt.figure(figsize = (15,12), dpi = 150)
    plot_tree(model, filled = True, feature_names = x.columns)
plt.show()
```

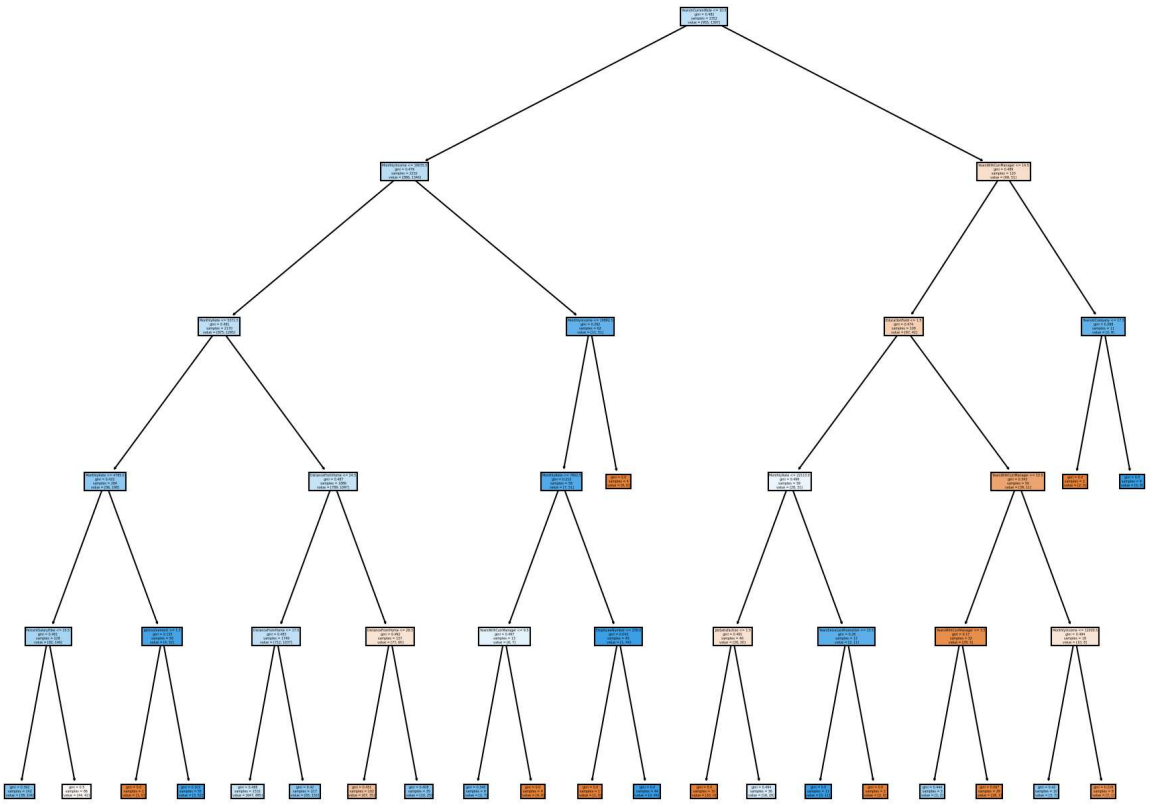
In [67]:
```
# max depth at 5
prunned_dtree = DecisionTreeClassifier(max_depth = 5)
prunned_dtree.fit(x_train,y_train)
```

Out[67]:
```
DecisionTreeClassifier(max_depth=5)
```

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 [68]:
```
1 dtree_model(prunned_dtree)
```

```
              precision    recall  f1-score   support

           0       0.55      0.16      0.25       221
           1       0.65      0.92      0.76       367

    accuracy                           0.63       588
   macro avg       0.60      0.54      0.50       588
weighted avg       0.61      0.63      0.57       588
```



Prediction

In [73]:
```
1 y_pred_prunned_train = prunned_dtree.predict(x_train)
2 y_pred_prunned_test = prunned_dtree.predict(x_test)
```

Evaluate

In [70]:
```
print('Decision Tree post prunning- Train accuracy:',accuracy_score(y_trai
print('-------'*10)
print('Decision Tree post prunning- Test accuracy:', accuracy_score(y_test
```

```
Decision Tree post prunning- Train accuracy: 0.6326530612244898
----------------------------------------------------------------------
Decision Tree post prunning- Test accuracy: 0.6343537414965986
```

In [ ]:
```

```