HR Analytics Project- Understanding the Attrition in HR

In [1]:

```
#Importing the libreris like pandas, numpy for selecing the data and convering the data to
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
#Creating the variable df and loading the dataframe  to the variable df
df=pd.read_excel('ibm.hr.xlsx')
df
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Em
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	
...	...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	

1470 rows × 35 columns

In [3]:

```
#Checking the dataframe for null values, if null value present we need to remove the null
df.isnull().sum()
```

Out[3]:

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
                            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
```

No null value present in the data frame df

In [4]:

```
#cheking the datatype of dataframe columns
df.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
```

Exploratory Data Analysis

In [5]:

```
# to know the statistical data we use describe function
df.describe()
```

Out[5]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	Environme
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	

8 rows × 26 columns

In [6]:

```
#Mapping the attrition 1 - yes and 0 - no in the new column

df["left"] = np.where(df["Attrition"] == "Yes",1,0)
```

In [7]:

```
import matplotlib.pyplot as plt
import seaborn as sns
def NumericalVariables_targetPlots(df,segment_by,target_var = "Attrition"):
    """A function for plotting the distribution of numerical variables and its effect on a

    fig, ax = plt.subplots(ncols= 2, figsize = (14,6))

    #boxplot for comparison
    sns.boxplot(x = target_var, y = segment_by, data=df, ax=ax[0])
    ax[0].set_title("Comparision of " + segment_by + " vs " + target_var)

    #distribution plot
    ax[1].set_title("Distribution of "+segment_by)
    ax[1].set_ylabel("Frequency")
    sns.distplot(a = df[segment_by], ax=ax[1], kde=False)

    plt.show()
```

In [8]:

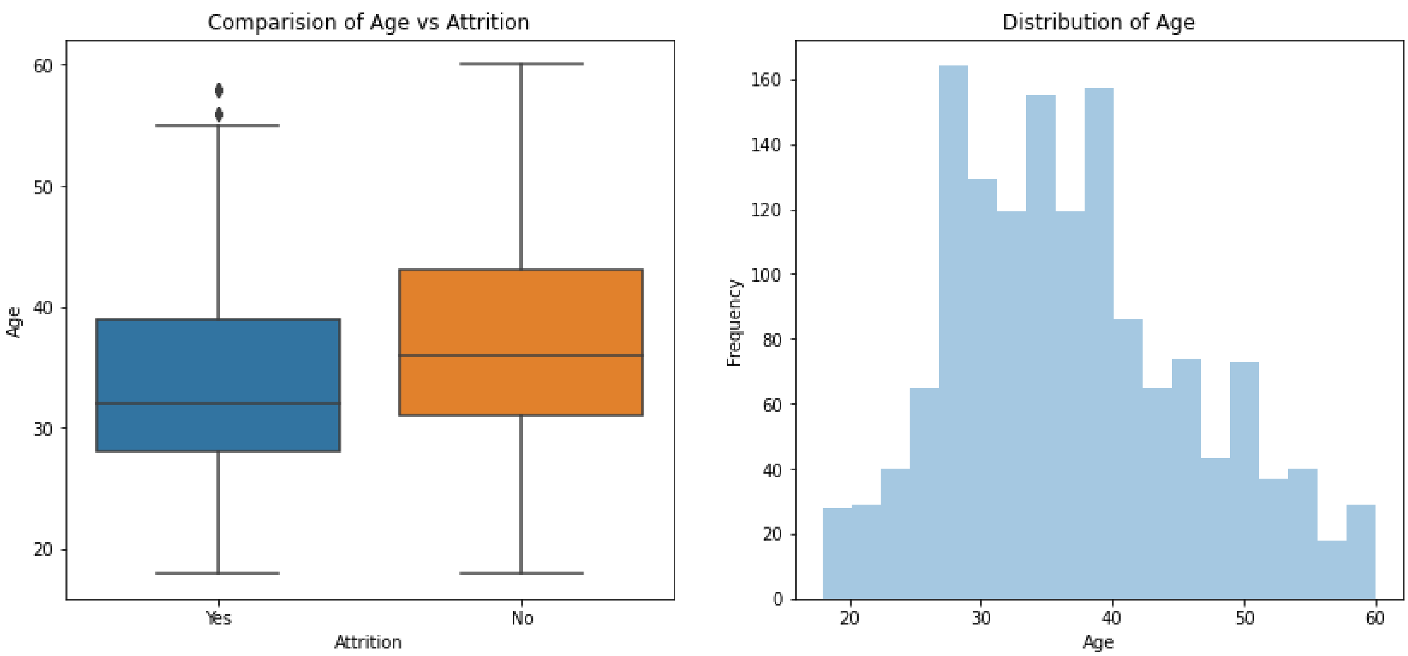
```
def CategoricalVariables_targetPlots(df, segment_by,invert_axis = False, target_var = "lef

    """A function for Plotting the effect of variables(categorical data) on attrition """

    fig, ax = plt.subplots(ncols= 2, figsize = (14,6))
```

Loading [MathJax]/extensions/Safe.js

```
    #countplot for distribution along with target variable
    #invert axis variable helps to inter change the axis so that names of categories doesn
    if invert_axis == False:
        sns.countplot(x = segment_by, data=df,hue="Attrition",ax=ax[0])
    else:
        sns.countplot(y = segment_by, data=df,hue="Attrition",ax=ax[0])

    ax[0].set_title("Comparision of " + segment_by + " vs " + "Attrition")

    #plot the effect of variable on attrition
    if invert_axis == False:
        sns.barplot(x = segment_by, y = target_var ,data=df,ci=None)
    else:
        sns.barplot(y = segment_by, x = target_var ,data=df,ci=None)

    ax[1].set_title("Attrition rate by {}".format(segment_by))
    ax[1].set_ylabel("Average(Attrition)")
    plt.tight_layout()

    plt.show()
```

In [9]:

```
NumericalVariables_targetPlots(df,segment_by="Age")
```

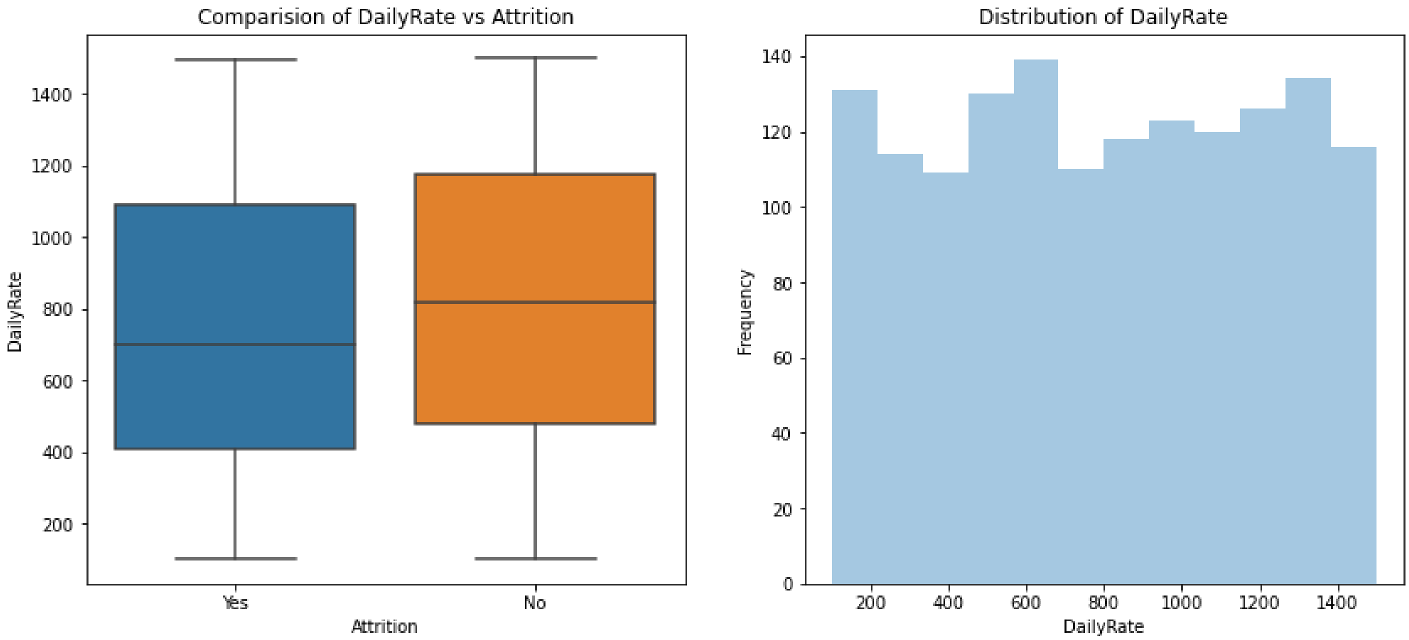


1) Minimum age is 18 Yrs and Maximum age is 60 Yrs.

2) majority of people who left the company are below 40 years and among the people who didn't left the company are of age 32 to 40 years

In [10]:

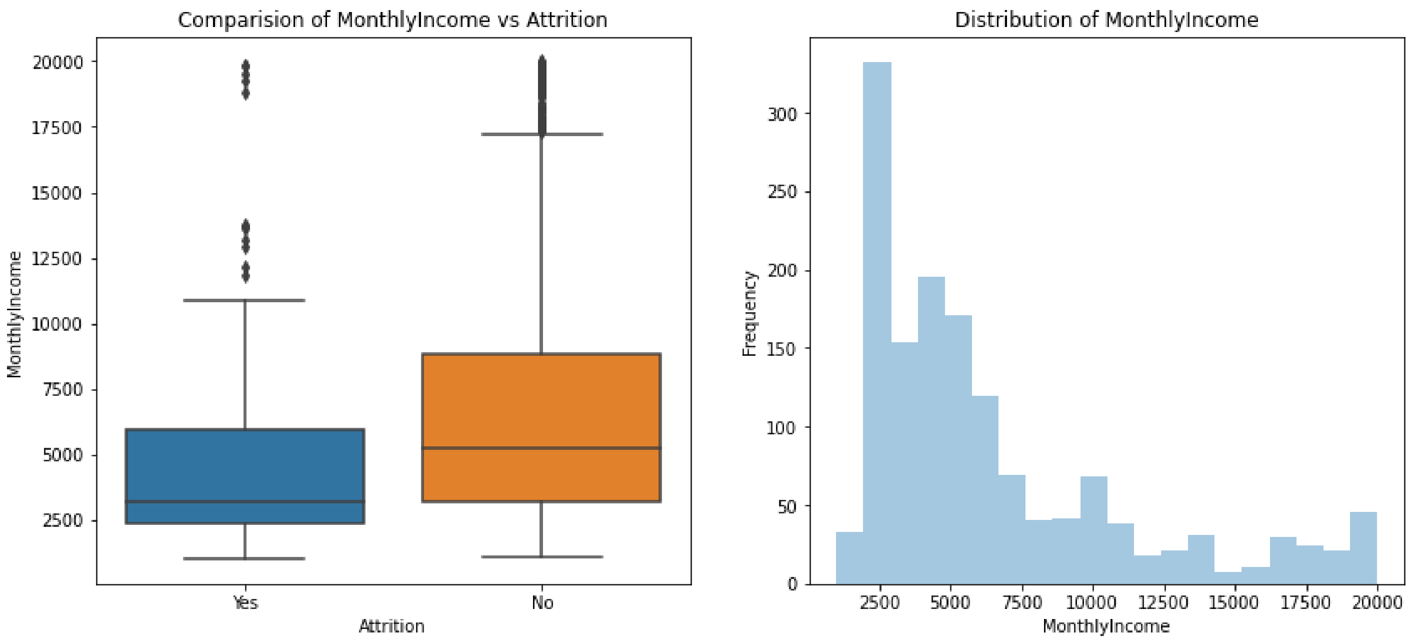
```
NumericalVariables_targetPlots(df,"DailyRate")
```



1) by graph we can say that the persons the persons who are working for less daily rate are leaving the company

In [11]:

```
NumericalVariables_targetPlots(df,"MonthlyIncome")
```



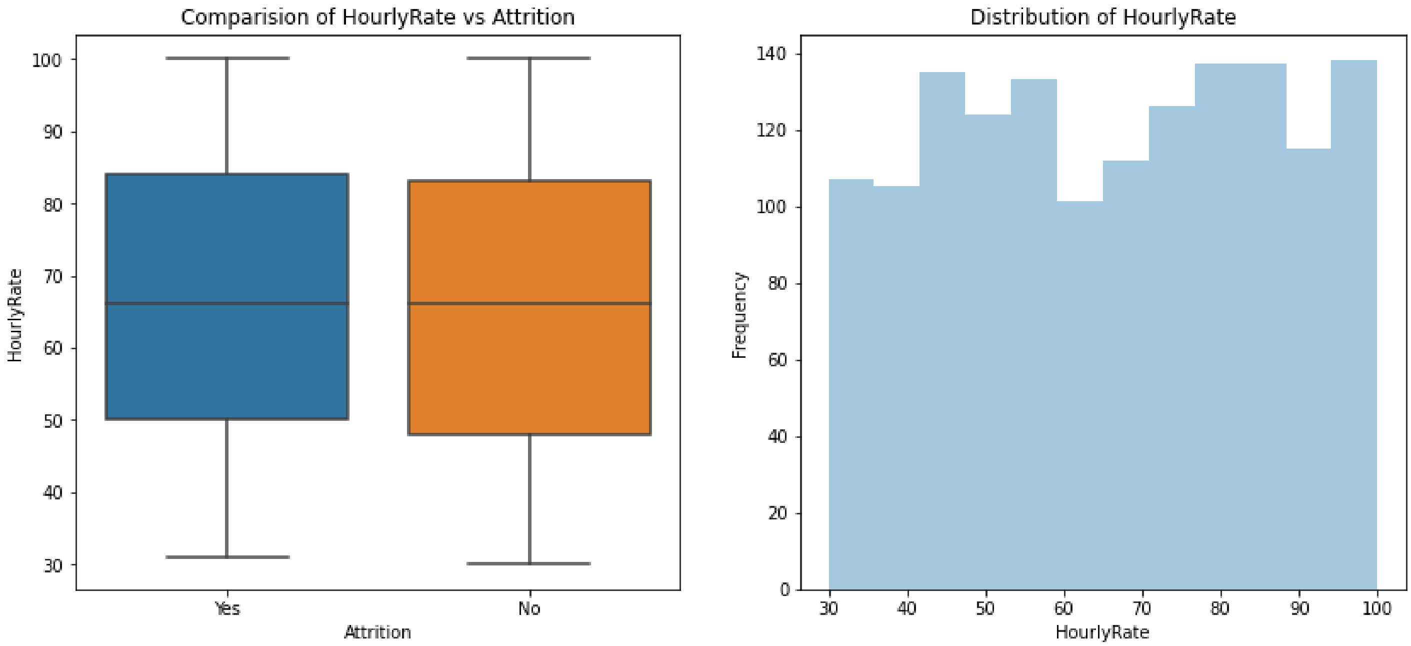
1) by graph we can say that the persons the persons who are working for less daily rate are leaving the company

2) some outliers are present in the monthly columns

In [12]:

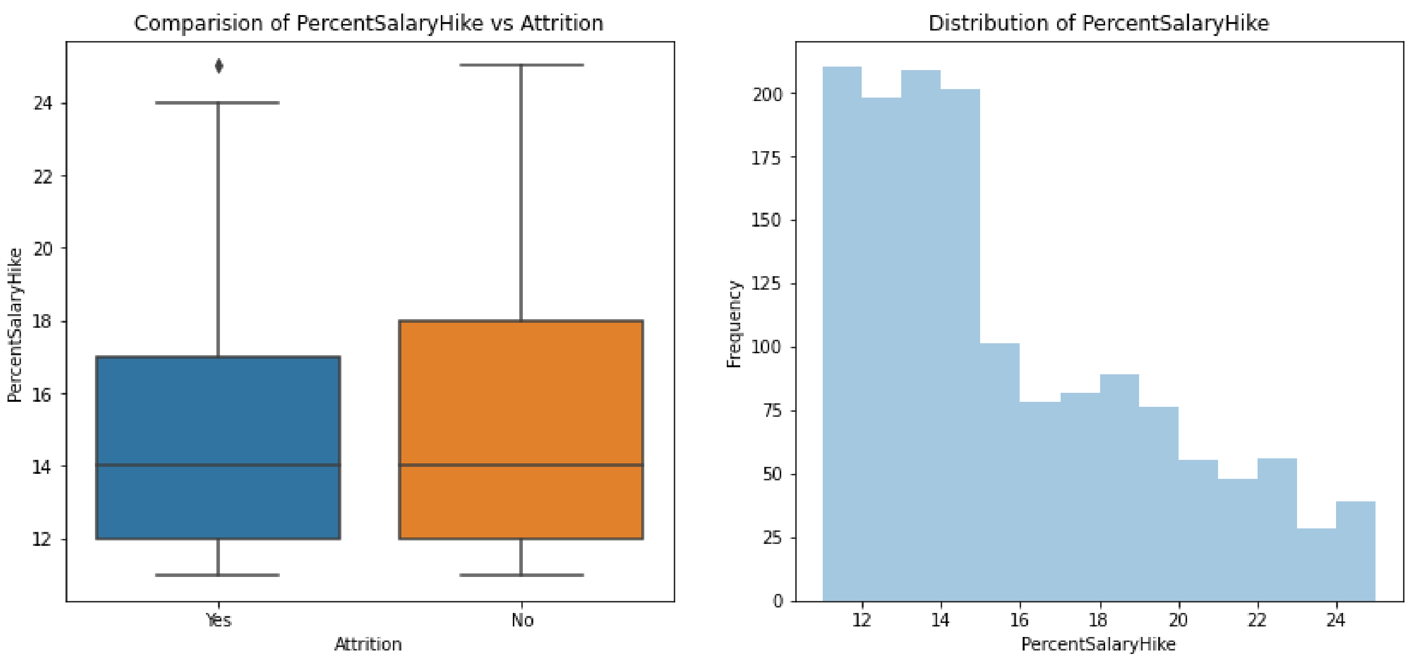
```
NumericalVariables_targetPlots(df,"HourlyRate")
```

Loading [MathJax]/extensions/Safe.js



1) by graph we can say that there is no significan change in the hourly rate and attrition , hourly rate has not much impact on attrition

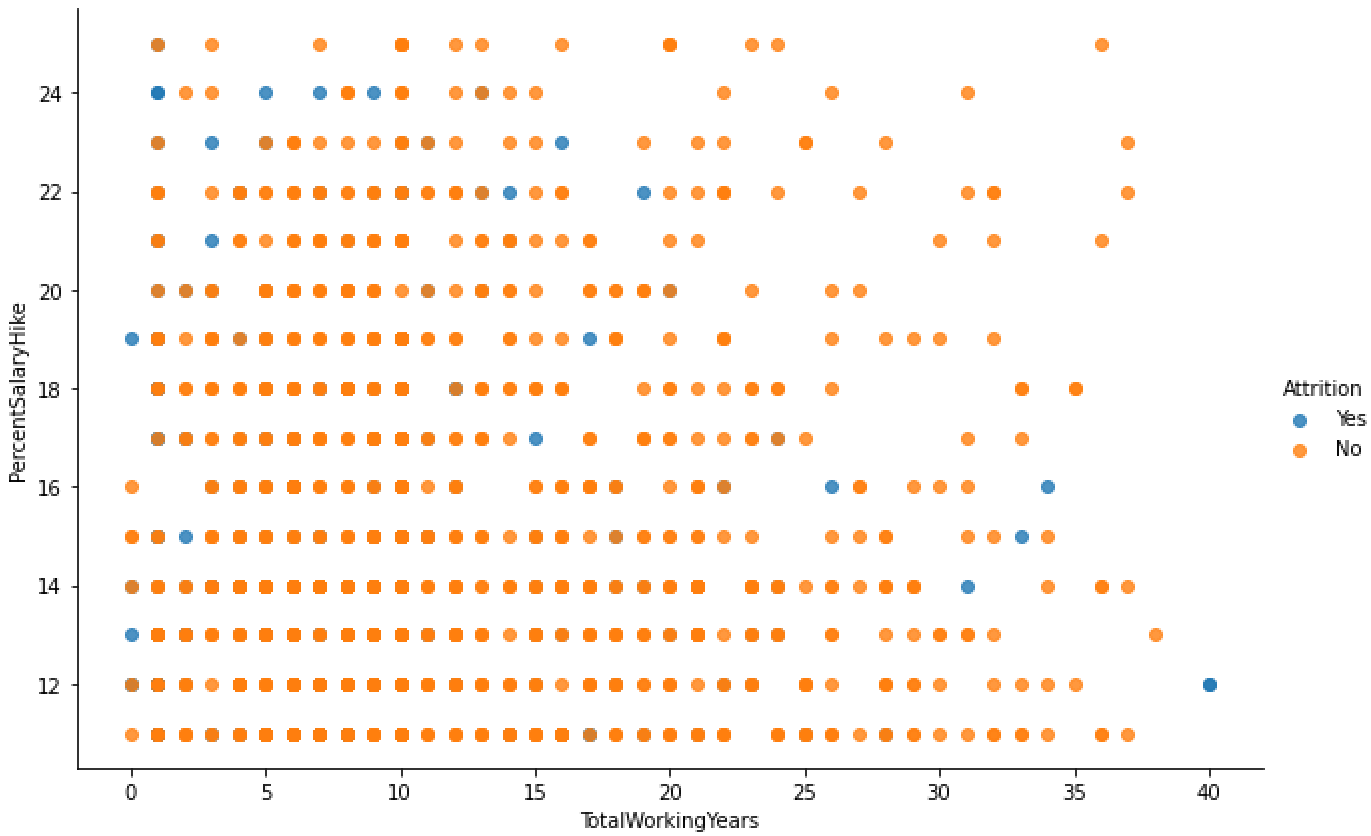
In [13]:
```
NumericalVariables_targetPlots(df,"PercentSalaryHike")
```



1) maximum (arround 50%) of the employee who got the salary hike of below 16 have left the company

In [14]:
```
sns.lmplot(x = "TotalWorkingYears", y = "PercentSalaryHike", data=df,fit_reg=False,hue="At
           aspect=1.5)

plt.show()
```

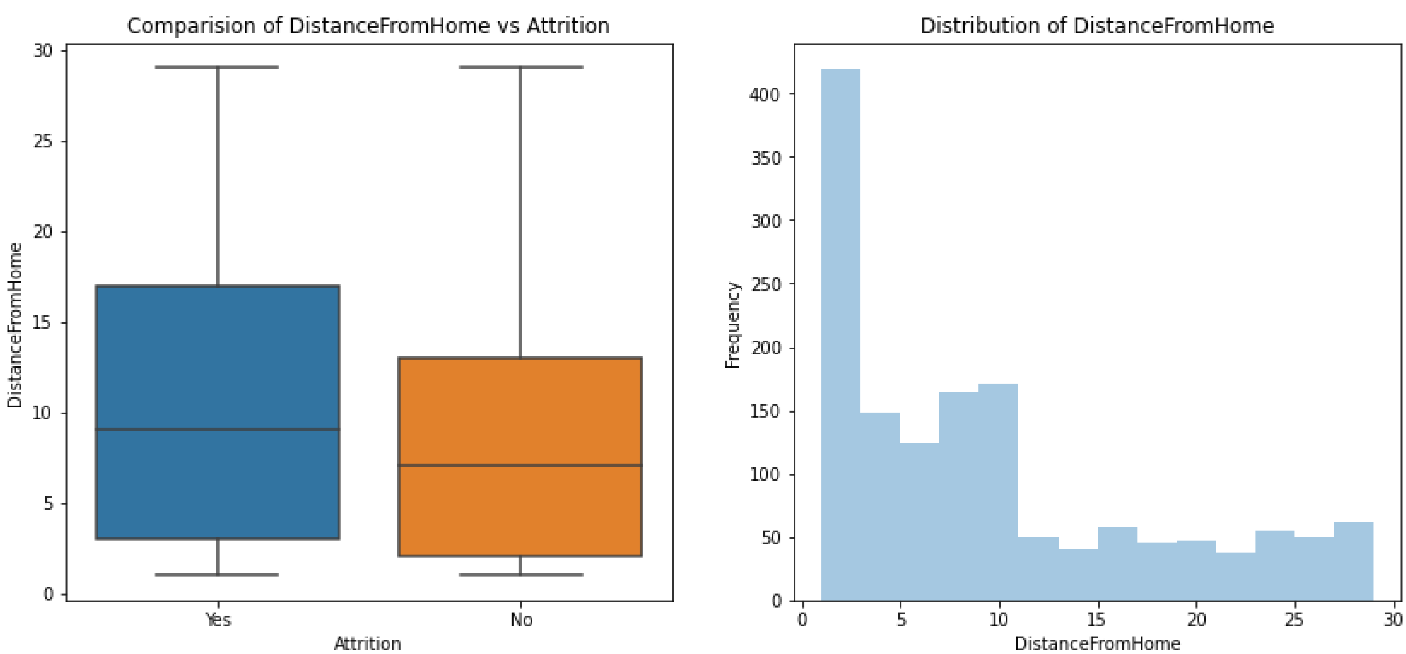
Loading [MathJax]/extensions/Safe.js



1) There is no linear relation between the total woking years, percentsalaryhike and attrition

2) More number of employees who are having the totalworkingyears below 20years have left the company

In [15]:
```
NumericalVariables_targetPlots(df,"DistanceFromHome")
```

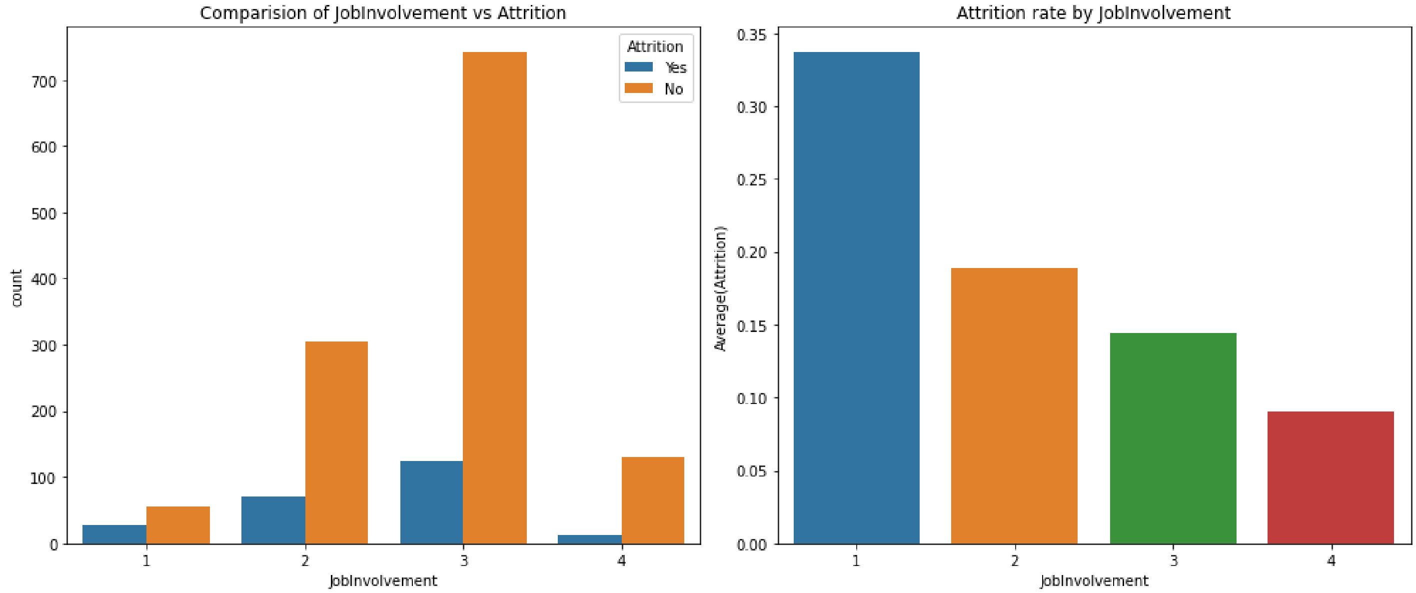


1) maximum of employee stay near by the company

2) by graph we can say that the employee coming from the far away have left the company (majarity)

In [16]:
```
CategoricalVariables_targetPlots(df,"JobInvolvement")
```

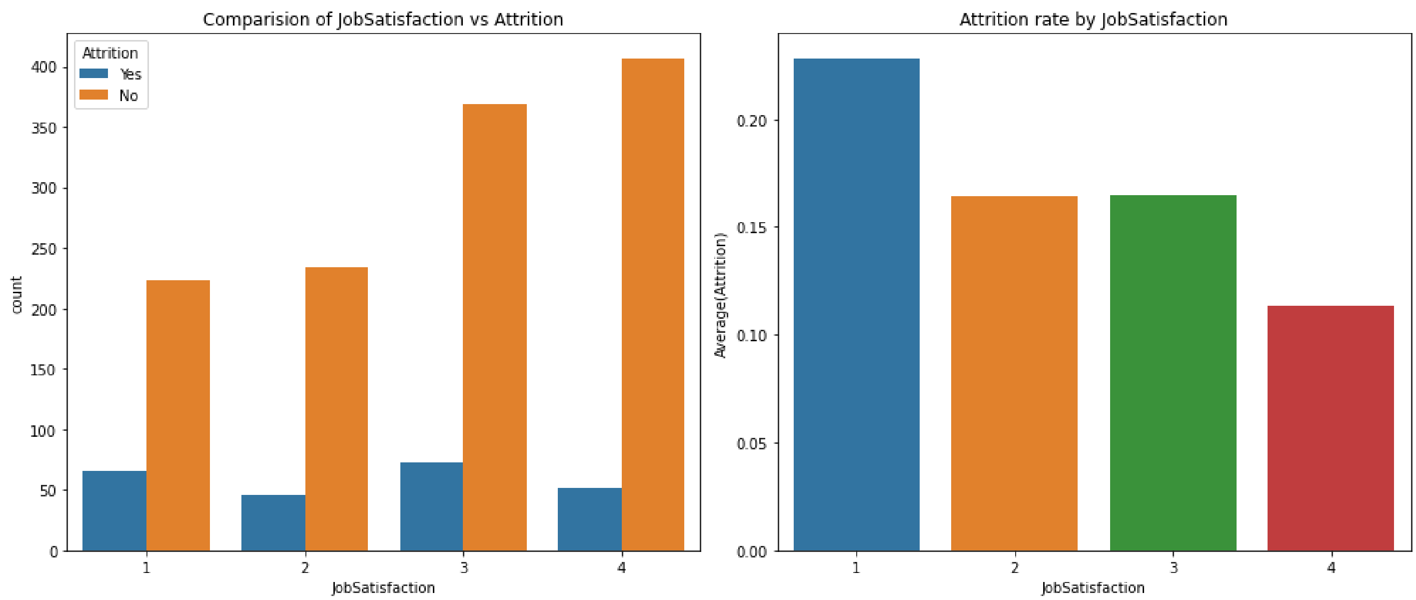
Loading [MathJax]/extensions/Safe.js



1) level 1 as maximum of employees

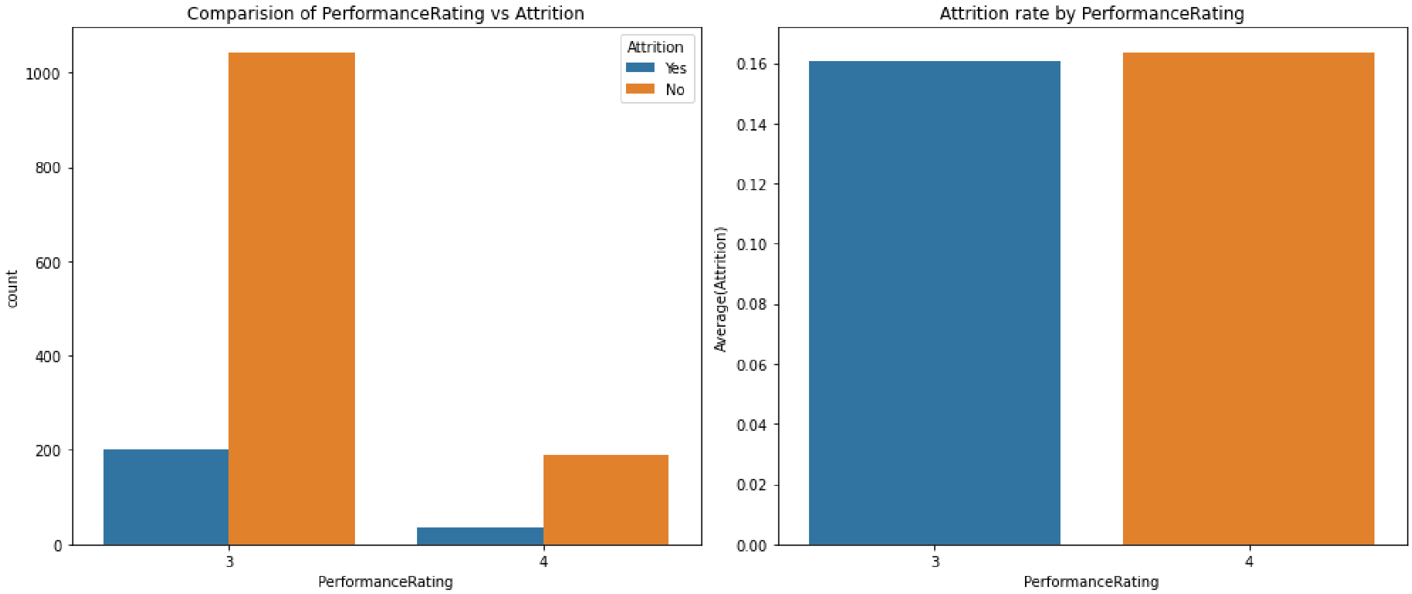
2) the maximum employees who left the company are in level 2 and 3

In [17]:
```
CategoricalVariables_targetPlots(df,"JobSatisfaction")
```



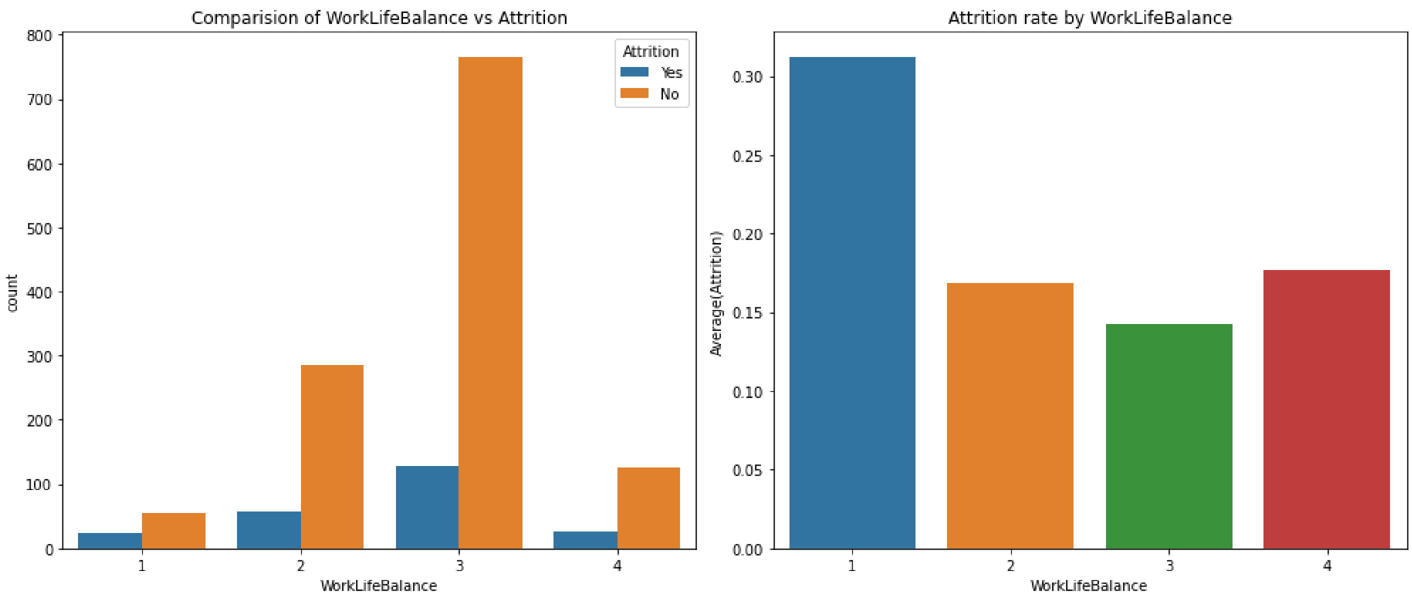
1) the employees having the job satisifaction level 1 and 2 have left the company more when compare to jobsatisifaction level 2 and 4

In [18]:
```
CategoricalVariables_targetPlots(df,"PerformanceRating")
```

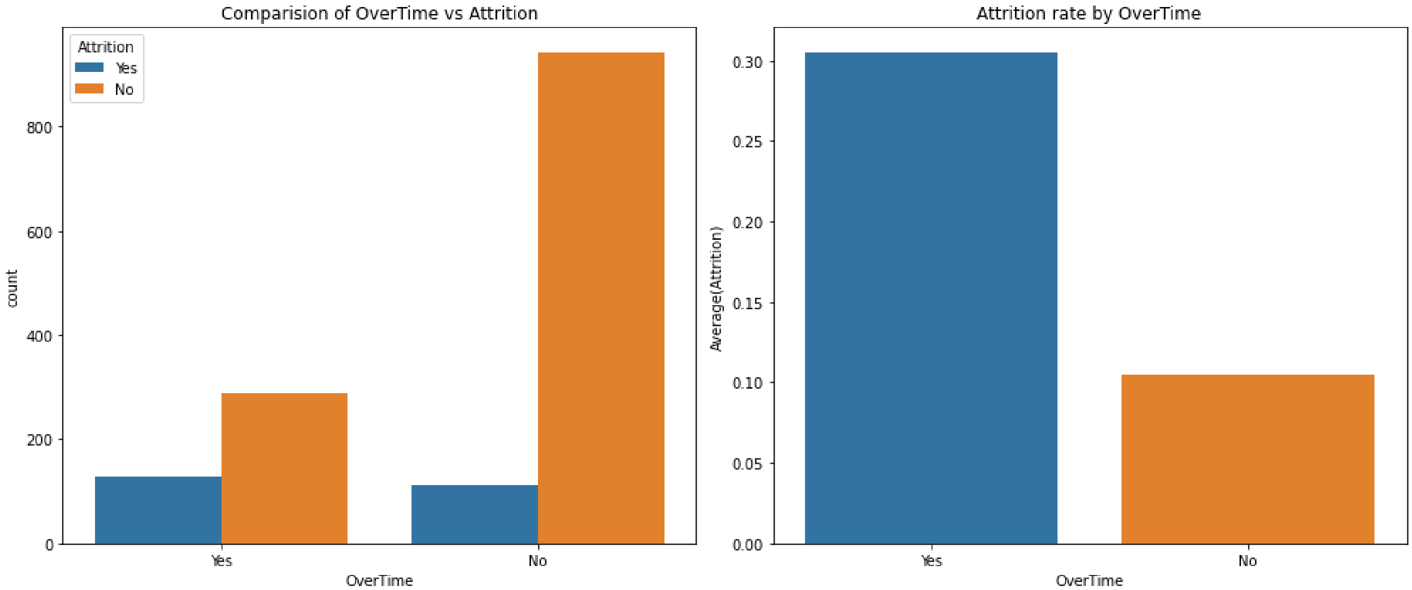


1) the employees who's performancerating 3 have left the company more when compare to performance rate 4

In [19]:
```
CategoricalVariables_targetPlots(df,"WorkLifeBalance")
```



In [20]:
```
CategoricalVariables_targetPlots(df,"OverTime")
```

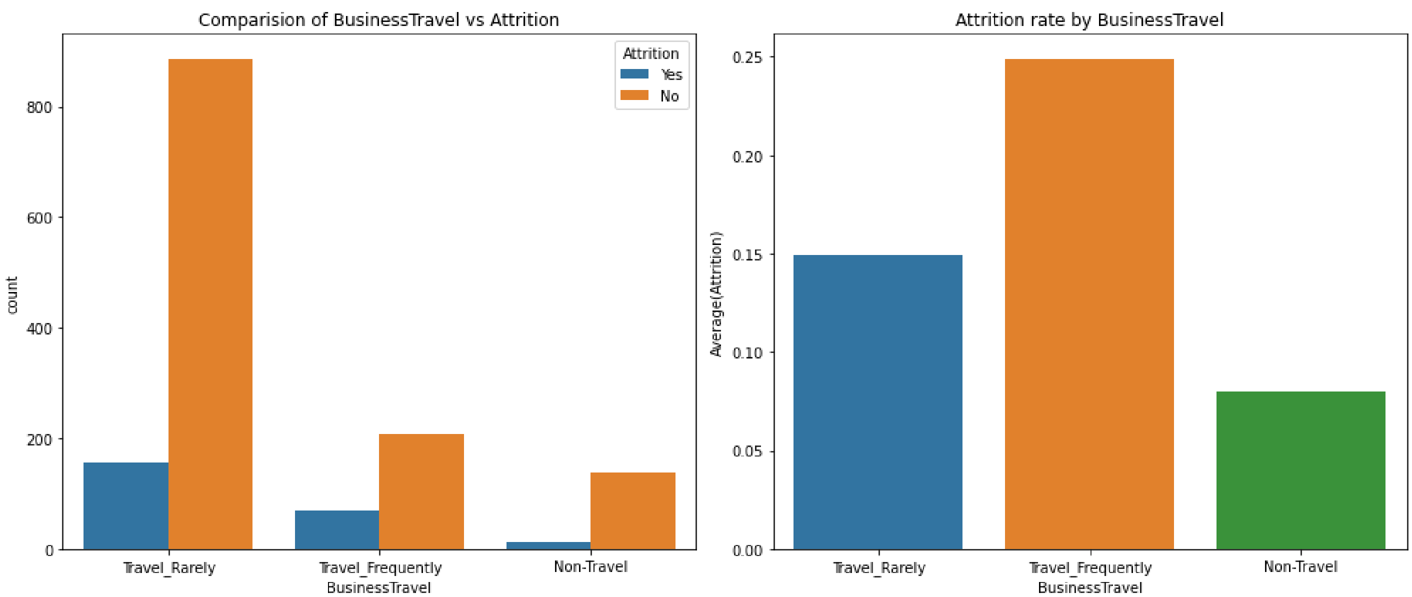


1) by graph we can say that the maximum number of employees work overtime

2) the employee who left the company, in that the employee who work overnight is more

In [21]:

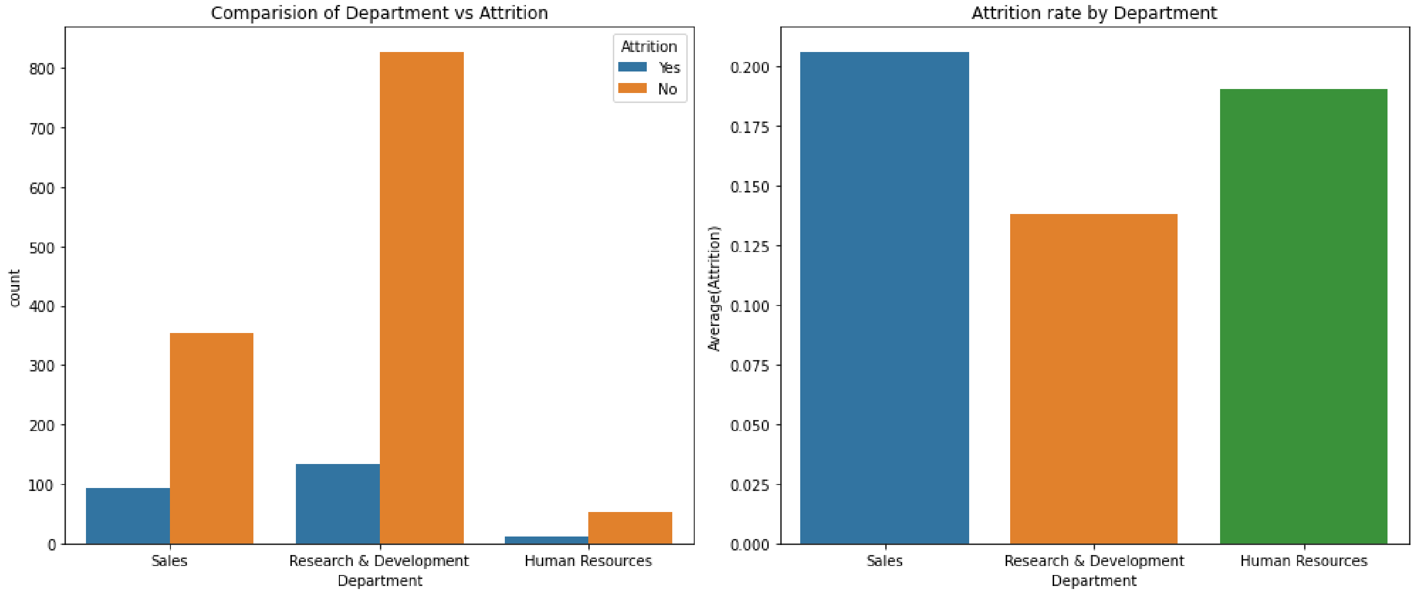
```
CategoricalVariables_targetPlots(df,segment_by="BusinessTravel")
```



In [22]:

```
CategoricalVariables_targetPlots(df,segment_by="Department")
```

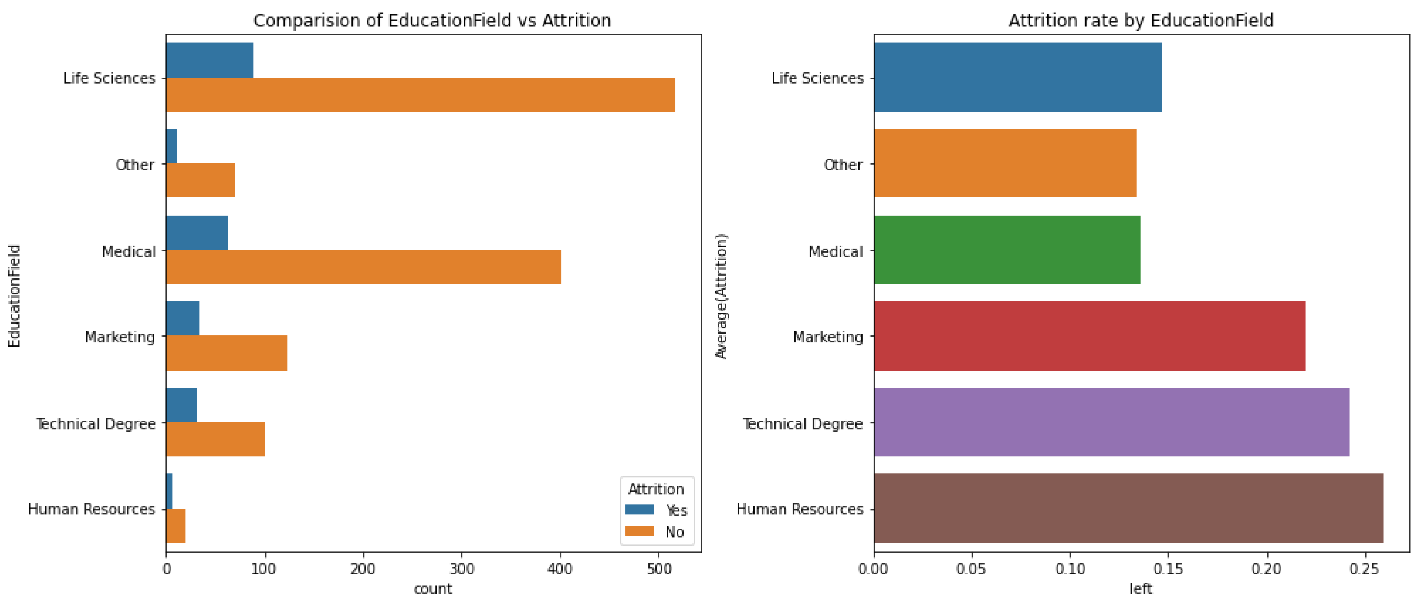
Loading [MathJax]/extensions/Safe.js



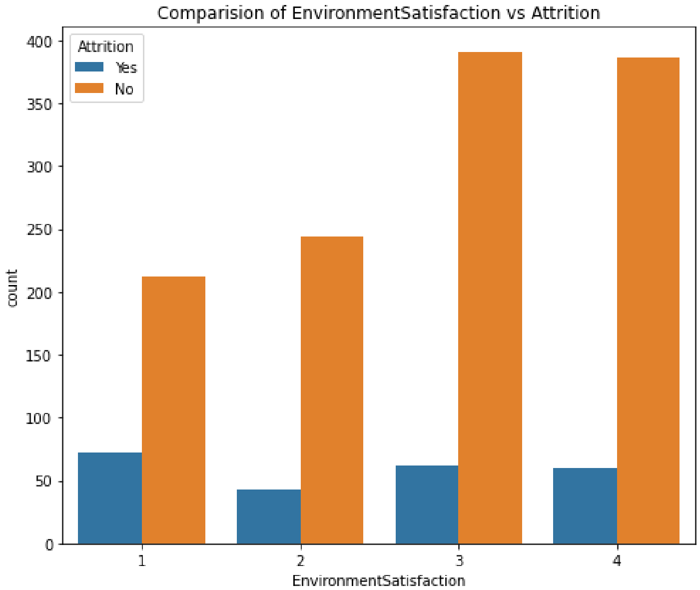
1) number of employee present in the sales is more and then folled by human resources and Research and development

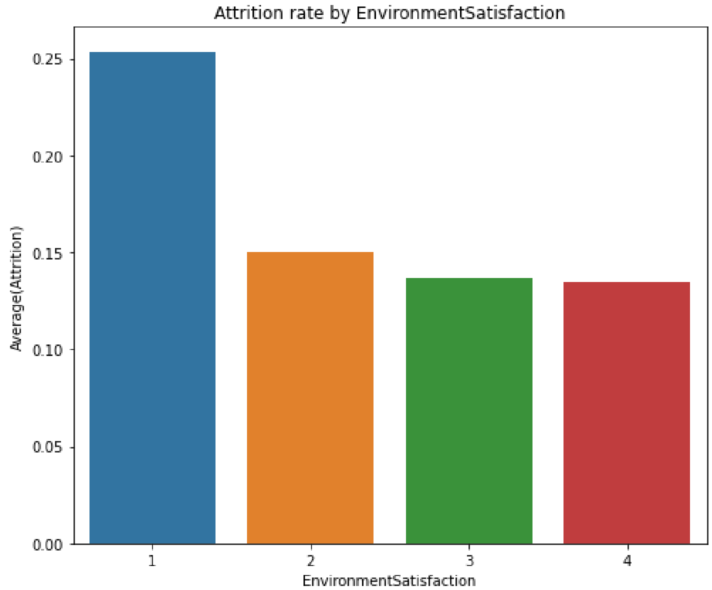
2) the more number of employee left the company in the research and development and then followed by sales and humnan resources

In [23]:
```
CategoricalVariables_targetPlots(df,"EducationField",invert_axis=True)
```



In [24]:
```
CategoricalVariables_targetPlots(df,"EnvironmentSatisfaction")
```

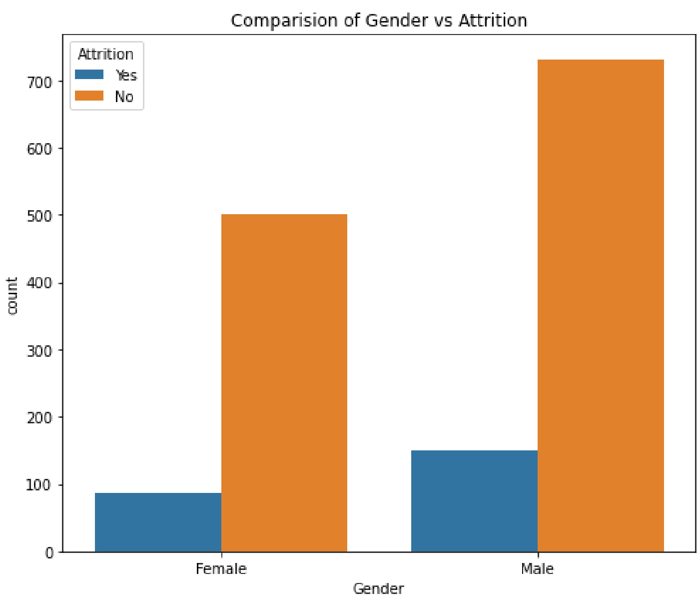


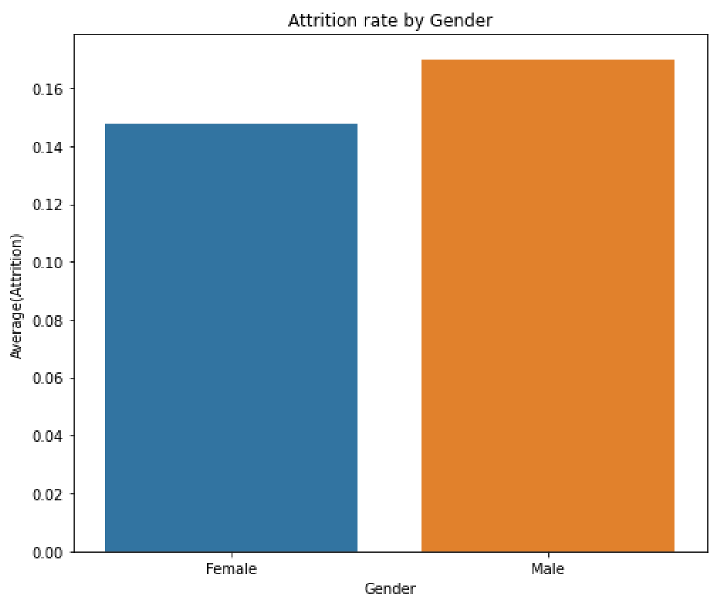


1) the employee who are having the environmentsatisfaction 1 left the company and followed by 3, 4 and 3 environmentsatisfaction

In [25]:

```
CategoricalVariables_targetPlots(df,"Gender")
```

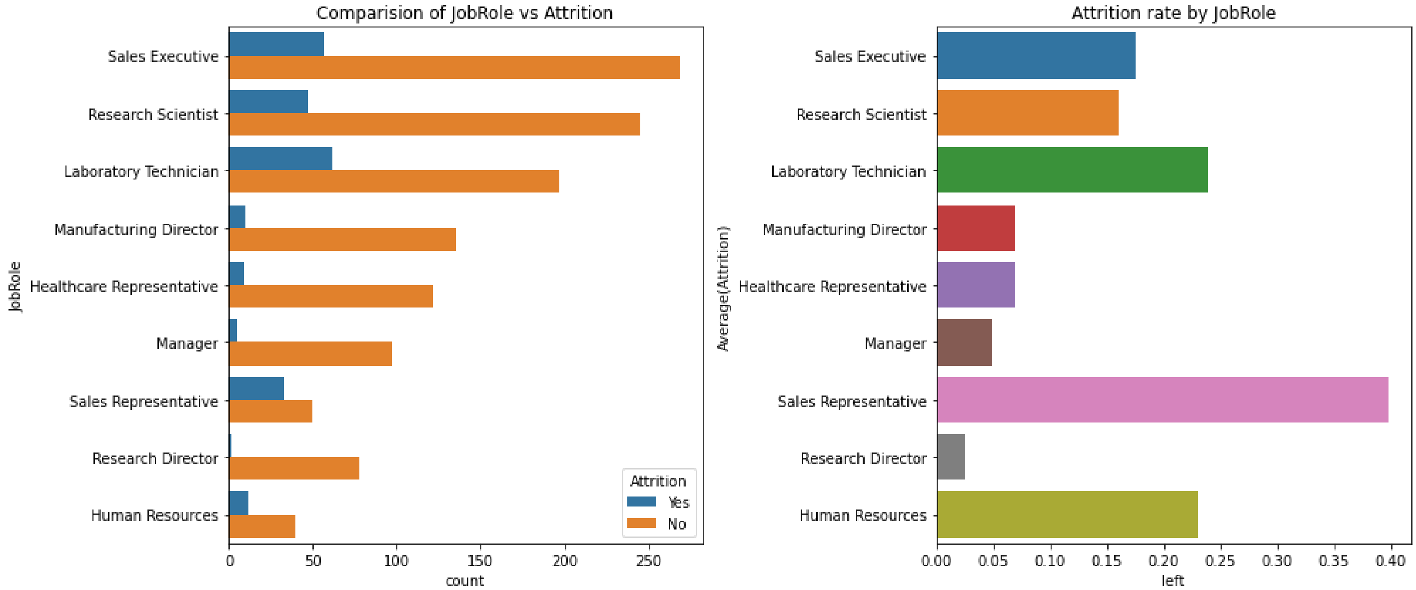




1) male workers are more in the company then that of female worker

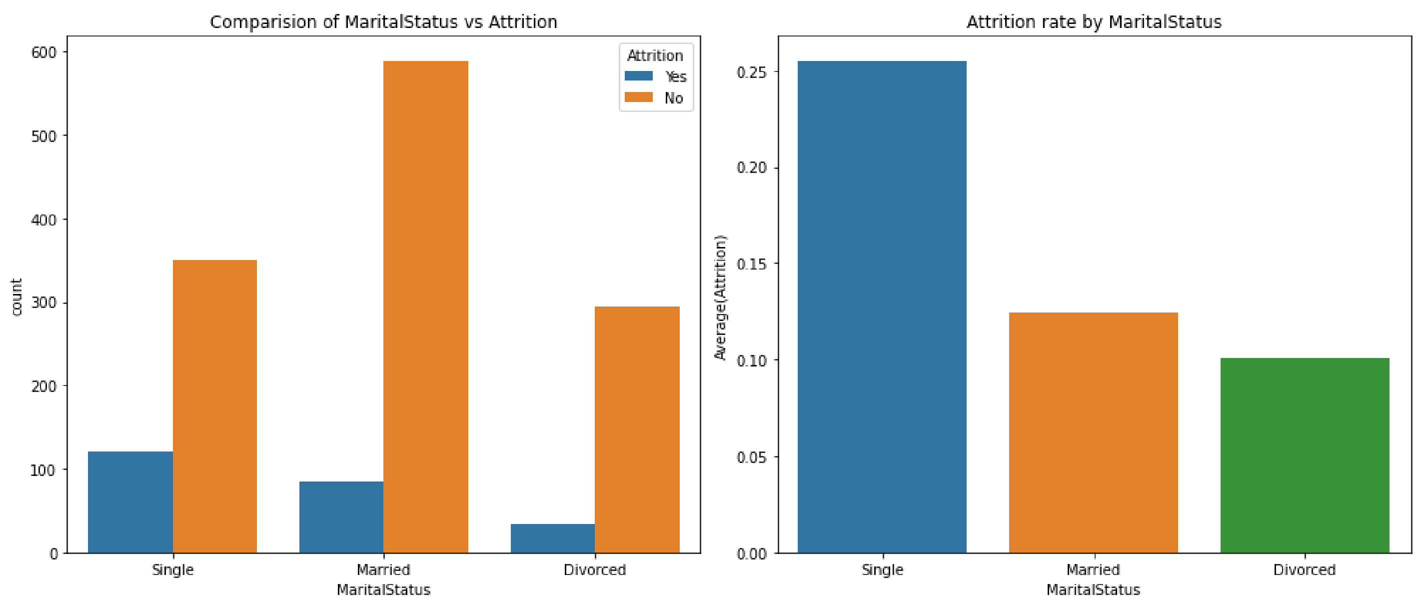
In [26]:

```
CategoricalVariables_targetPlots(df,"JobRole",invert_axis=True)
```



In [27]:

```
CategoricalVariables_targetPlots(df,"MaritalStatus")
```



1) maritalstatus single are more in the company

In [28]:

```
# changing the categorical value to numerical value by labelencoder tecnique
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
df['Attrition']=le.fit_transform(df['Attrition'])
df['BusinessTravel']=le.fit_transform(df['BusinessTravel'])
df['Department']=le.fit_transform(df['Department'])
df['EducationField']=le.fit_transform(df['EducationField'])
df['Gender']=le.fit_transform(df['Gender'])
df['JobRole']=le.fit_transform(df['JobRole'])
df['MaritalStatus']=le.fit_transform(df['MaritalStatus'])
df['Over18']=le.fit_transform(df['Over18'])
df['OverTime']=le.fit_transform(df['OverTime'])
df
```

Out[28]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Empl
0	41	1	2	1102	2	1	2	1	
1	49	0	1	279	1	8	1	1	

Loading [MathJax]/extensions/Safe.js

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Empl
2	37	1	2	1373	1	2	2	4	
3	33	0	1	1392	1	3	4	1	
4	27	0	2	591	1	2	1	3	
...	...	...	...	...	...	...	...	...	
1465	36	0	1	884	1	23	2	3	
1466	39	0	2	613	1	6	1	3	
1467	27	0	2	155	1	4	3	1	
1468	49	0	1	1023	2	2	3	3	
1469	34	0	2	628	1	8	3	3	

1470 rows × 36 columns

In [29]:

```
df.skew()
```

Out[29]:

```
Age                         0.413286
Attrition                   1.844366
BusinessTravel             -1.439006
DailyRate                  -0.003519
Department                  0.172231
DistanceFromHome            0.958118
Education                  -0.289681
EducationField              0.550371
EmployeeCount               0.000000
EmployeeNumber              0.016574
EnvironmentSatisfaction    -0.321654
Gender                     -0.408665
HourlyRate                 -0.032311
JobInvolvement             -0.498419
JobLevel                    1.025401
JobRole                    -0.357270
JobSatisfaction            -0.329672
MaritalStatus              -0.152175
MonthlyIncome               1.369817
MonthlyRate                 0.018578
NumCompaniesWorked          1.026471
Over18                      0.000000
OverTime                    0.964489
PercentSalaryHike           0.821128
PerformanceRating           1.921883
RelationshipSatisfaction   -0.302828
StandardHours               0.000000
StockOptionLevel            0.968980
TotalWorkingYears           1.117172
TrainingTimesLastYear       0.553124
WorkLifeBalance            -0.552480
YearsAtCompany              1.764529
YearsInCurrentRole          0.917363
YearsSinceLastPromotion     1.984290
YearsWithCurrManager        0.833451
left                        1.844366
dtype: float64
```

In [30]:

```
df.isnull().sum()
```

```
Age                         0
```

Loading [MathJax]/extensions/Safe.js

```
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
left                        0
dtype: int64
```

In [31]:

```
#The columns like EmployeeCount,Gender,HourlyRate,Over18,StandardHours,left does not have
df=df.drop(columns=['EmployeeCount','Gender','HourlyRate','Over18','StandardHours','left']
df
```

Out[31]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Empl
0	41	1	2	1102	2	1	2	1	
1	49	0	1	279	1	8	1	1	
2	37	1	2	1373	1	2	2	4	
3	33	0	1	1392	1	3	4	1	
4	27	0	2	591	1	2	1	3	
...	...	...	...	...	...	...	...	...	
1465	36	0	1	884	1	23	2	3	
1466	39	0	2	613	1	6	1	3	
1467	27	0	2	155	1	4	3	1	
1468	49	0	1	1023	2	2	3	3	
1469	34	0	2	628	1	8	3	3	

1470 rows × 30 columns

In [32]:

```python
# checking for co relation
df.corr()
```

Out[32]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
Age	1.000000	-0.159205	0.024751	0.010661	-0.031882	-0.001686	0.208
Attrition	-0.159205	1.000000	0.000074	-0.056652	0.063991	0.077924	-0.031
BusinessTravel	0.024751	0.000074	1.000000	-0.004086	-0.009044	-0.024469	0.000
DailyRate	0.010661	-0.056652	-0.004086	1.000000	0.007109	-0.004985	-0.016
Department	-0.031882	0.063991	-0.009044	0.007109	1.000000	0.017225	0.007
DistanceFromHome	-0.001686	0.077924	-0.024469	-0.004985	0.017225	1.000000	0.021
Education	0.208034	-0.031373	0.000757	-0.016806	0.007996	0.021042	1.000
EducationField	-0.040873	0.026846	0.023724	0.037709	0.013720	0.002013	-0.039
EmployeeNumber	-0.010145	-0.010577	-0.015578	-0.050990	-0.010895	0.032916	0.042
EnvironmentSatisfaction	0.010146	-0.103369	0.004174	0.018355	-0.019395	-0.016075	-0.027
JobInvolvement	0.029820	-0.130016	0.039062	0.046135	-0.024586	0.008783	0.042
JobLevel	0.509604	-0.169105	0.019311	0.002966	0.101963	0.005303	0.101
JobRole	-0.122427	0.067151	0.002724	-0.009472	0.662431	-0.001015	0.004
JobSatisfaction	-0.004892	-0.103481	-0.033962	0.030571	0.021001	-0.003669	-0.011
MaritalStatus	-0.095029	0.162070	0.024001	-0.069586	0.056073	-0.014437	0.004
MonthlyIncome	0.497855	-0.159840	0.034319	0.007707	0.053130	-0.017014	0.094
MonthlyRate	0.028051	0.015170	-0.014107	-0.032182	0.023642	0.027473	-0.026
NumCompaniesWorked	0.299635	0.043494	0.020875	0.038153	-0.035882	-0.029251	0.126
OverTime	0.028062	0.246118	0.016543	0.009135	0.007481	0.025514	-0.020
PercentSalaryHike	0.003634	-0.013478	-0.029377	0.022704	-0.007840	0.040235	-0.011
PerformanceRating	0.001904	0.002889	-0.026341	0.000473	-0.024604	0.027110	-0.024
RelationshipSatisfaction	0.053535	-0.045872	-0.035986	0.007846	-0.022414	0.006557	-0.009
StockOptionLevel	0.037510	-0.137145	-0.016727	0.042143	-0.012193	0.044872	0.018
TotalWorkingYears	0.680381	-0.171063	0.034226	0.014515	-0.015762	0.004628	0.148
TrainingTimesLastYear	-0.019621	-0.059478	0.015240	0.002453	0.036875	-0.036942	-0.025
WorkLifeBalance	-0.021490	-0.063939	-0.011256	-0.037848	0.026383	-0.026556	0.009
YearsAtCompany	0.311309	-0.134392	-0.014575	-0.034055	0.022920	0.009508	0.069
YearsInCurrentRole	0.212901	-0.160545	-0.011497	0.009932	0.056315	0.018845	0.060
YearsSinceLastPromotion	0.216513	-0.033019	-0.032591	-0.033229	0.040061	0.010029	0.054
YearsWithCurrManager	0.202089	-0.156199	-0.022636	-0.026363	0.034282	0.014406	0.069

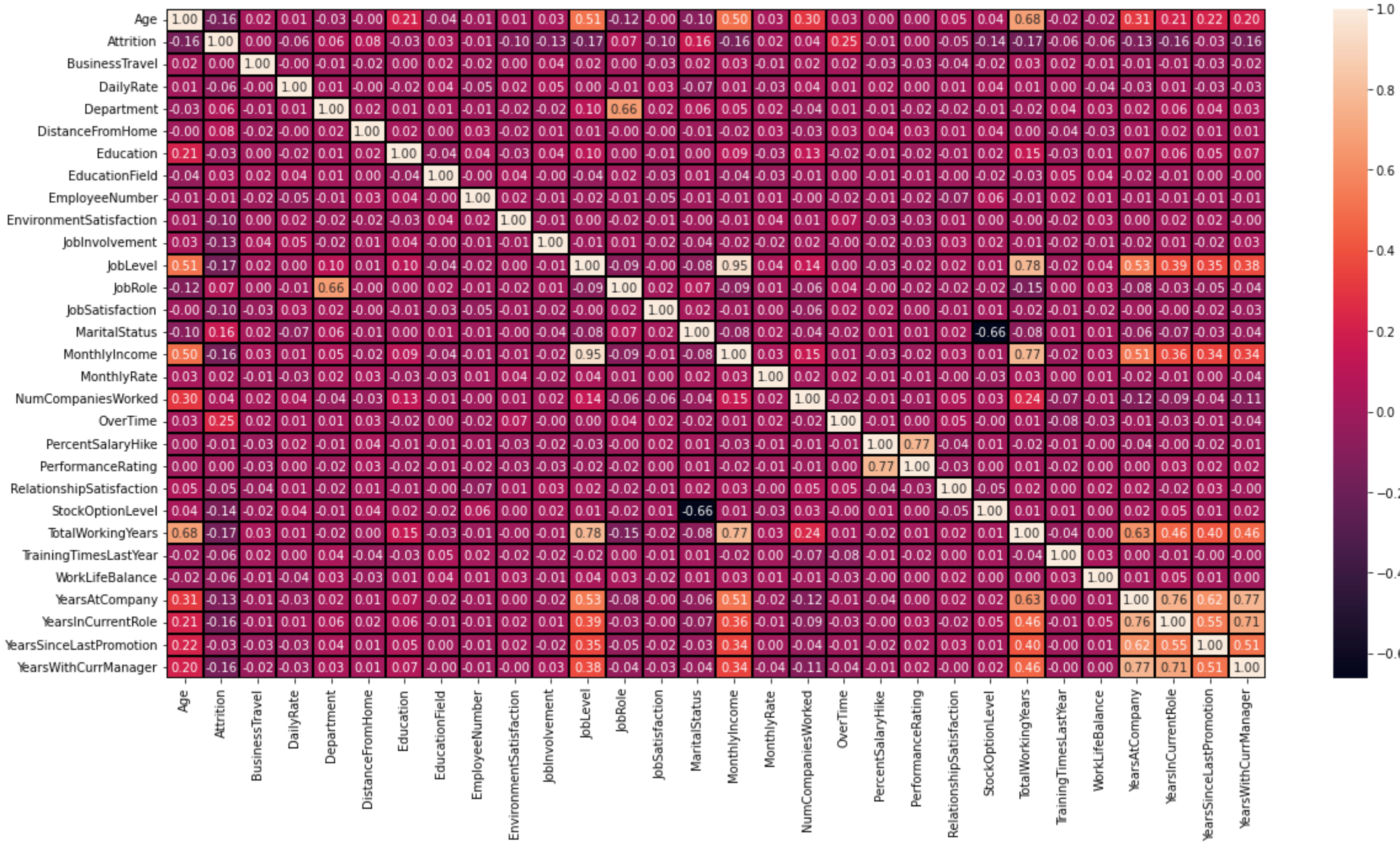
30 rows × 30 columns

In [33]:

```python
#Reprasantation of co relation values by using the heatmap
import matplotlib.pyplot as plt
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True,linewidth=0.1,linecolor='black',fmt='0.2f')
```

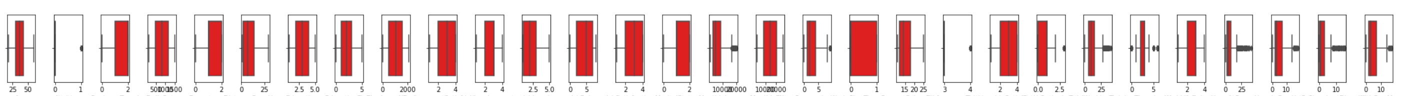
Out[33]: <AxesSubplot:>

Loading [MathJax]/extensions/Safe.js



In [34]:

```python
#check for outliers with the help of box plot
columns_list=df.columns.values
ncol=100
nrows=50
plt.figure(figsize=(ncol,ncol))
for i in range (0,len(columns_list)):
    plt.subplot(nrows,ncol,i+1)
    sns.boxplot(df[columns_list[i]],color='red',orient='h')
    plt.tight_layout()
```



by graph we come to know that outliers present in the data frame

In [35]:

```python
# checking how much number of outliers are present in the dataframe
from scipy.stats import zscore
z=np.abs(zscore(df))
print(df.shape)
print(z.shape)
threshold=3
print(np.where(z>3))
len(np.where(z>3)[0])
```

```
(1470, 30)
(1470, 30)
(array([  28,   45,   62,   62,   63,   64,   85,   98,   98,  110,  123,
        123,  123,  126,  126,  126,  153,  178,  187,  187,  190,  190,
        218,  231,  231,  237,  237,  270,  270,  281,  326,  386,  386,
        401,  411,  425,  425,  427,  445,  466,  473,  477,  535,  561,
        561,  584,  592,  595,  595,  595,  616,  624,  635,  653,  653,
        677,  686,  701,  716,  746,  749,  752,  799,  838,  861,  861,
        875,  875,  894,  914,  914,  918,  922,  926,  926,  937,  956,
        962,  976,  976, 1008, 1024, 1043, 1078, 1078, 1086, 1086, 1093,
        116, 1116, 1135, 1138, 1138, 1156, 1184, 1221, 1223, 1242,
```

```
        1295, 1301, 1301, 1303, 1327, 1331, 1348, 1351, 1401, 1414, 1430],
       dtype=int64), array([29, 28, 26, 28, 27, 28, 23, 23, 26, 28, 27, 28, 29, 23, 26, 28,
29,
        28, 23, 29, 26, 27, 28, 27, 29, 26, 28, 23, 26, 27, 28, 28, 29, 23,
        26, 26, 28, 28, 23, 27, 26, 26, 28, 26, 29, 28, 26, 23, 26, 28, 29,
        23, 29, 26, 28, 26, 29, 28, 27, 27, 26, 28, 28, 28, 26, 28, 28, 29,
        23, 26, 28, 26, 28, 28, 29, 28, 23, 26, 27, 28, 28, 27, 23, 28, 29,
        26, 28, 28, 26, 23, 26, 26, 26, 28, 28, 23, 28, 28, 28, 28, 23, 28,
        28, 27, 28, 29, 27, 23, 28, 27], dtype=int64))
```

Out[35]:

```
110
```

In [36]:

```python
# 110 outliers are present in the df data frame
df_new=df[(z<3).all(axis=1)]
print(df_new.shape)
```

```
(1387, 30)
```

after removing the outliers the shape of the df is reduced to 1380*30

In [37]:

```python
#df_new is the data frame after removing the outliers
df_new
```

Out[37]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Empl
0	41	1	2	1102	2	1	2	1	
1	49	0	1	279	1	8	1	1	
2	37	1	2	1373	1	2	2	4	
3	33	0	1	1392	1	3	4	1	
4	27	0	2	591	1	2	1	3	
...	...	...	...	...	...	...	...	...	
1465	36	0	1	884	1	23	2	3	
1466	39	0	2	613	1	6	1	3	
1467	27	0	2	155	1	4	3	1	
1468	49	0	1	1023	2	2	3	3	
1469	34	0	2	628	1	8	3	3	

1387 rows × 30 columns

In [38]:

```python
df_new1=df_new.drop(columns=['Attrition'])
df_new1
```

Out[38]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNuml
0	41	2	1102	2	1	2	1	
1	49	1	279	1	8	1	1	
2	37	2	1373	1	2	2	4	
3	33	1	1392	1	3	4	1	
4	27	2	591	1	2	1	3	
...	...	...	...	...	...	...	...	
1465	36	1	884	1	23	2	3	20

Loading [MathJax]/extensions/Safe.js

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNuml
1466	39	2	613	1	6	1	3	20
1467	27	2	155	1	4	3	1	20
1468	49	1	1023	2	2	3	3	20
1469	34	2	628	1	8	3	3	20

1387 rows × 29 columns

In [39]:

```
# assigning the x and y values
x=df_new1.iloc[:,:]
y=df_new.iloc[:,1:2]
print(x)
print(y)
```

```
      Age  BusinessTravel  DailyRate  Department  DistanceFromHome  Education  \
0      41               2       1102           2                 1          2
1      49               1        279           1                 8          1
2      37               2       1373           1                 2          2
3      33               1       1392           1                 3          4
4      27               2        591           1                 2          1
...   ...             ...        ...         ...               ...        ...
1465   36               1        884           1                23          2
1466   39               2        613           1                 6          1
1467   27               2        155           1                 4          3
1468   49               1       1023           2                 2          3
1469   34               2        628           1                 8          3

      EducationField  EmployeeNumber  EnvironmentSatisfaction  JobInvolvement  \
0                  1               1                        2               3
1                  1               2                        3               2
2                  4               4                        4               2
3                  1               5                        4               3
4                  3               7                        1               3
...              ...             ...                      ...             ...
1465               3            2061                        3               4
1466               3            2062                        4               2
1467               1            2064                        2               4
1468               3            2065                        4               2
1469               3            2068                        2               4

      ...  PerformanceRating  RelationshipSatisfaction  StockOptionLevel  \
0     ...                  3                         1                 0
1     ...                  4                         4                 1
2     ...                  3                         2                 0
3     ...                  3                         3                 0
4     ...                  3                         4                 1
...   ...                ...                       ...               ...
1465  ...                  3                         3                 1
1466  ...                  3                         1                 1
1467  ...                  4                         2                 1
1468  ...                  3                         4                 0
1469  ...                  3                         1                 0

      TotalWorkingYears  TrainingTimesLastYear  WorkLifeBalance  \
0                     8                      0                1
1                    10                      3                3
2                     7                      3                3
3                     8                      3                3
4                     6                      3                3
                    ...                    ...              ...
```

```
1465                17                      3                 3
1466                 9                      5                 3
1467                 6                      0                 3
1468                17                      3                 2
1469                 6                      3                 4

      YearsAtCompany  YearsInCurrentRole  YearsSinceLastPromotion  \
0                  6                   4                        0
1                 10                   7                        1
2                  0                   0                        0
3                  8                   7                        3
4                  2                   2                        2
...              ...                 ...                      ...
1465               5                   2                        0
1466               7                   7                        1
1467               6                   2                        0
1468               9                   6                        0
1469               4                   3                        1

      YearsWithCurrManager
0                        5
1                        7
2                        0
3                        0
4                        2
...                    ...
1465                     3
1466                     7
1467                     3
1468                     8
1469                     2

[1387 rows x 29 columns]
      Attrition
0             1
1             0
2             1
3             0
4             0
...         ...
1465          0
1466          0
1467          0
1468          0
1469          0

[1387 rows x 1 columns]
```

In [40]:

```
#removing the skewness by yeo johnson method
from sklearn.preprocessing import power_transform
x=power_transform(x,method='yeo-johnson')
x
```

Out[40]:

```
array([[ 0.61013332,  0.63872976,  0.75061538, ...,  0.29052433,
        -1.07353381,  0.58217664],
       [ 1.37182973, -1.38077628, -1.34337244, ...,  1.0065754 ,
         0.19316755,  1.01807316],
       [ 0.18248603,  0.63872976,  1.33708042, ..., -1.57181404,
        -1.07353381, -1.52842596],
       ...,
       [-1.0804891 ,  0.63872976, -1.75453754, ..., -0.39076907,
        -1.07353381,  0.01867962],
       [ 1.37182973, -1.38077628,  0.57328582, ...,  0.79376377,
        -1.07353381,  1.20578193],
```

```
       [-0.16377603,  0.63872976, -0.37222758, ..., -0.01873824,
         0.19316755, -0.3478709 ]])
```

In [41]:

```
# checking the skweness after removing the skewness by yeo-johnson method
pd.DataFrame(x).skew()
```

Out[41]:

```
0    -0.004079
1    -0.960583
2    -0.199742
3     0.015095
4    -0.008149
5    -0.103747
6    -0.008642
7    -0.287518
8    -0.205472
9    -0.018801
10    0.110769
11   -0.337641
12   -0.217730
13   -0.158253
14    0.027700
15   -0.176560
16    0.016175
17    0.954751
18    0.112128
19    0.000000
20   -0.191406
21    0.089929
22   -0.009666
23    0.057949
24   -0.011133
25   -0.025230
26   -0.069631
27    0.212301
28   -0.070570
dtype: float64
```

In [42]:

```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
x1=mms.fit_transform(x)
```

In [43]:

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=99,test_size=0.2)
dtc.fit(x_train,y_train)
pred_test=dtc.predict(x_test)
print('accurecy score = ',accuracy_score(y_test,pred_test)*100)
print(confusion_matrix(y_test,pred_test))
print(classification_report(y_test,pred_test))
from sklearn.model_selection import cross_val_score
cv_score=cross_val_score(dtc,x1,y,cv=5)
cv_mean=cv_score.mean()
print("cross_val_score=",cv_mean*100)
```

```
accurecy score =  84.89208633093526
[[219  22]
 [ 20  17]]
              precision    recall  f1-score   support
```

```
           0       0.92      0.91      0.91       241
           1       0.44      0.46      0.45        37

    accuracy                           0.85       278
   macro avg       0.68      0.68      0.68       278
weighted avg       0.85      0.85      0.85       278

cross_val_score= 77.14411864010596
```

In [49]:

```python
# spliting the x_train, x_test, y_train,y_test  for analysing the model performance by usi
# KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=99,test_size=0.2)
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
pred_knn=dtc.predict(x_test)
print(accuracy_score(y_test,pred_knn)*100)
print(confusion_matrix(y_test,pred_knn))
print(classification_report(y_test,pred_knn))
from sklearn.model_selection import cross_val_score
cv_score=cross_val_score(knn,x1,y,cv=5)
cv_mean=cv_score.mean()
print("cross_val_score=",cv_mean*100)
```

```
84.89208633093526
[[219  22]
 [ 20  17]]
              precision    recall  f1-score   support

           0       0.92      0.91      0.91       241
           1       0.44      0.46      0.45        37

    accuracy                           0.85       278
   macro avg       0.68      0.68      0.68       278
weighted avg       0.85      0.85      0.85       278

cross_val_score= 85.72526816092252
```

In [46]:

```python
# SVC
from sklearn.svm import SVC
svc=SVC()
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=99,test_size=0.2)
svc.fit(x_train,y_train)
pred_svc=dtc.predict(x_test)
print(accuracy_score(y_test,pred_svc)*100)
print(confusion_matrix(y_test,pred_svc))
print(classification_report(y_test,pred_svc))
from sklearn.model_selection import cross_val_score
cv_score=cross_val_score(svc,x1,y,cv=5)
cv_mean=cv_score.mean()
print("cross_val_score=",cv_mean*100)
```

```
84.89208633093526
[[219  22]
 [ 20  17]]
              precision    recall  f1-score   support

           0       0.92      0.91      0.91       241
           1       0.44      0.46      0.45        37

    accuracy                           0.85       278
   macro avg       0.68      0.68      0.68       278
```

```
weighted avg       0.85      0.85      0.85       278

cross_val_score= 87.02256966989586
```

In [47]:

```
# LogisticRegression
from sklearn.linear_model import LogisticRegression
```

In [48]:

```
lrr=LogisticRegression()
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=99,test_size=0.2)
lrr.fit(x_train,y_train)
pred_test=dtc.predict(x_test)
print('accurecy score = ',accuracy_score(y_test,pred_test)*100)
print(confusion_matrix(y_test,pred_test))
print(classification_report(y_test,pred_test))
from sklearn.model_selection import cross_val_score
cv_score=cross_val_score(lrr,x1,y,cv=5)
cv_mean=cv_score.mean()
print("cross_val_score=",cv_mean*100)
```

```
accurecy score =  84.89208633093526
[[219  22]
 [ 20  17]]
              precision    recall  f1-score   support

           0       0.92      0.91      0.91       241
           1       0.44      0.46      0.45        37

    accuracy                           0.85       278
   macro avg       0.68      0.68      0.68       278
weighted avg       0.85      0.85      0.85       278

cross_val_score= 87.52772511232891
```

The difference betwen the KNeighborsClassifier accuracy score and the cross validation accuracy score is less so i am considering the KNeighborsClassifier and hyper parameter tuning done to improve the model accuracy score

For improving the model performance we will go for hyper parameter tuning

In [57]:

```
# after knowing the better performance model algorithm , we are performing the hyperparame
from sklearn.model_selection import GridSearchCV,RepeatedStratifiedKFold
seed=42
```

In [62]:

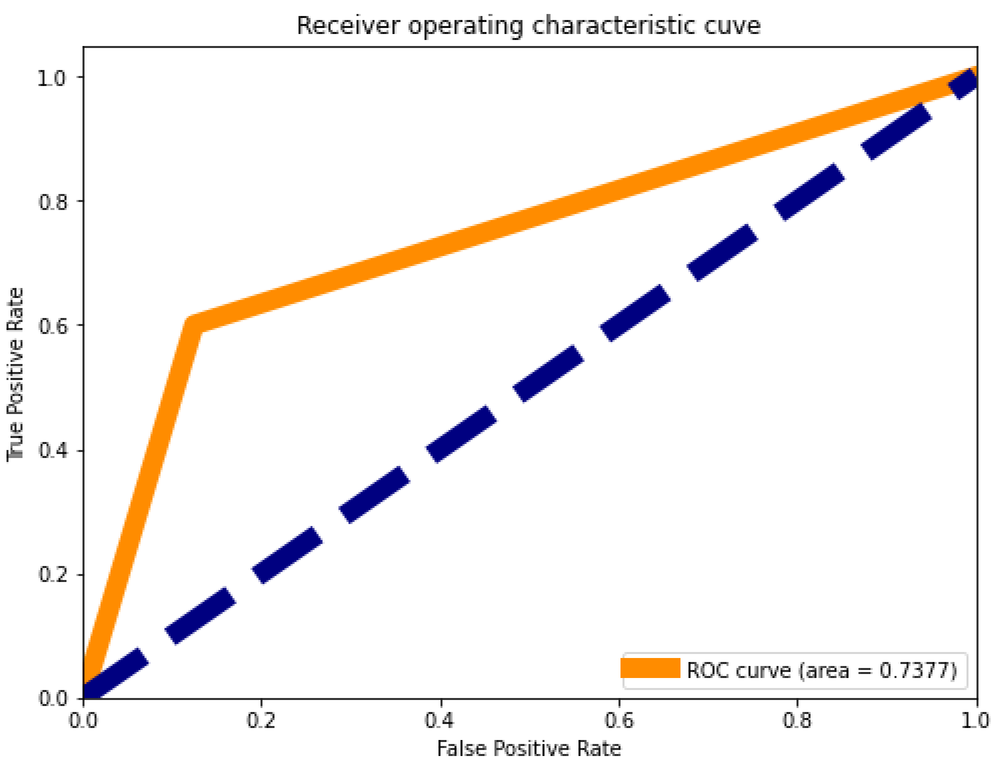
```
knn_prams={'n_neighbors':range(1,30,2),
          'weights':['uniform','distance'],
          'metric':['euclidean','manhattan','minkowski'],
          'leaf_size':range(1,50,5)}
knn=KNeighborsClassifier()
cv=RepeatedStratifiedKFold(n_splits=10,n_repeats=3,random_state=seed)
grid_search=GridSearchCV(estimator=knn,param_grid=knn_prams,n_jobs=1,cv=cv,scoring='accura
grid_results=grid_search.fit(x_train,y_train)
final_model=knn.set_params(**grid_results.best_params_)
final_model.fit(x_train,y_train)
y_pred=final_model.predict(x_test)
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
                    sults.best_params_)
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       241
           1       0.60      0.08      0.14        37

    accuracy                           0.87       278
   macro avg       0.74      0.54      0.54       278
weighted avg       0.84      0.87      0.83       278

[[239   2]
 [ 34   3]]
{'leaf_size': 1, 'metric': 'manhattan', 'n_neighbors': 9, 'weights': 'uniform'}
```

After hyper parametr tuning the ACCURACY score incresed from 84.89 to 87 so we can consider the KNeighborsClassifier so we are saving the KNeighborsClassifier results

In [68]:

```
#aoc-roc curve
from sklearn.metrics import roc_curve,auc
fpr, tpr, thresholds = roc_curve(y_pred,y_test)
roc_auc=auc(fpr,tpr)


plt.figure(figsize=(8, 6))
plt.plot( fpr, tpr,color='darkorange',lw=10,label='ROC curve (area = %0.4f)' % roc_auc)
plt.plot([0, 1], [0, 1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic cuve')
plt.legend(loc="lower right")
plt.show()
```



1) area under the curve is 73.77percent

2) last step of the projct,we know the better performance algorith which is KNeighborsClassifier hyperparameter tuning also done after that we need to save the model by usig pickel

In [ ]:

Loading [MathJax]/extensions/Safe.js