GOAL 1: DATA ACQUISITION AND DATA WRANGLING(CLEANING THE DATA) USING PYTHON LIBRARIES (PANDAS AND NUMPY)

In [20]:
```
# importing the required libraries
import pandas as pd
import numpy as np
```

In [21]:
```
cd C:\Users\User\Desktop\PIP - CTS\ibm-hr-analytics-attrition-dataset
```

```
C:\Users\User\Desktop\PIP - CTS\ibm-hr-analytics-attrition-dataset
```

In [22]:
```
#To read the contents of the csv file
df = pd.read_csv('EmployeeAttrition_data.csv')
```

In [23]:
```
#To find the top 5 rows data in the dataset
df.tail()
```

Out[23]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...
995	43	No	Travel_Rarely	930	Research & Development	6	3	Medical	1	1402	...
996	27	No	NaN	205	Sales	10	3	Marketing	1	1403	...
997	27	Yes	NaN	135	Research & Development	17	4	Life Sciences	1	1405	...
998	26	No	Travel_Rarely	683	Research & Development	2	1	Medical	1	1407	...
999	42	No	Travel_Rarely	1147	Human Resources	10	3	Human Resources	1	1408	...

5 rows × 35 columns

In [24]:
```
#To check in the dataframe the total columns and non null entries in each
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1000 non-null   int64 
 1   Attrition                 1000 non-null   object
 2   BusinessTravel            970 non-null    object
 3   DailyRate                 1000 non-null   int64 
 4   Department                1000 non-null   object
 5   DistanceFromHome          1000 non-null   int64 
 6   Education                 1000 non-null   int64 
 7   EducationField            1000 non-null   object
 8   EmployeeCount             1000 non-null   int64 
 9   EmployeeNumber            1000 non-null   int64 
 10  EnvironmentSatisfaction   1000 non-null   int64 
 11  Gender                    1000 non-null   object
 12  HourlyRate                1000 non-null   int64 
 13  JobInvolvement            1000 non-null   int64 
 14  JobLevel                  1000 non-null   int64 
 15  JobRole                   1000 non-null   object
 16  JobSatisfaction           1000 non-null   int64 
 17  MaritalStatus             985 non-null    object
 18  MonthlyIncome             1000 non-null   int64 
 19  MonthlyRate               1000 non-null   int64 
 20  NumCompaniesWorked        1000 non-null   int64 
 21  Over18                    1000 non-null   object
 22  OverTime                  1000 non-null   object
 23  PercentSalaryHike         1000 non-null   int64 
 24  PerformanceRating         1000 non-null   int64 
 25  RelationshipSatisfaction  1000 non-null   int64 
 26  StandardHours             1000 non-null   int64 
 27  StockOptionLevel          1000 non-null   int64 
 28  TotalWorkingYears         1000 non-null   int64 
 29  TrainingTimesLastYear     1000 non-null   int64 
 30  WorkLifeBalance           1000 non-null   int64 
 31  JoiningYearinCompany      1000 non-null   int64 
 32  YearsInCurrentRole        1000 non-null   int64 
 33  YearsSinceLastPromotion   1000 non-null   int64 
 34  YearsWithCurrManager      1000 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 273.6+ KB
```

In [25]:
```
#To describe the dataset, Transpose of data is taken for better readability of data.
df.describe().T
```

Out[25]:

	count	mean	std	min	25%	50%	75%	max
Age	1000.0	36.992	9.417783	18.0	30.00	36.0	43.00	60.0
DailyRate	1000.0	808.437	405.508487	102.0	470.75	817.0	1157.25	1499.0
DistanceFromHome	1000.0	9.067	8.108900	1.0	2.00	7.0	14.00	29.0
Education	1000.0	2.868	1.030358	1.0	2.00	3.0	4.00	5.0
EmployeeCount	1000.0	1.000	0.000000	1.0	1.00	1.0	1.00	1.0
EmployeeNumber	1000.0	690.073	406.416188	1.0	341.75	678.0	1038.25	1408.0
EnvironmentSatisfaction	1000.0	2.731	1.083426	1.0	2.00	3.0	4.00	4.0
HourlyRate	1000.0	65.163	20.209227	30.0	48.00	65.0	83.00	100.0
JobInvolvement	1000.0	2.730	0.703986	1.0	2.00	3.0	3.00	4.0
JobLevel	1000.0	2.095	1.139857	1.0	1.00	2.0	3.00	5.0
JobSatisfaction	1000.0	2.769	1.098565	1.0	2.00	3.0	4.00	4.0
MonthlyIncome	1000.0	6627.086	4842.436233	1009.0	2868.00	4936.0	8723.00	19999.0
MonthlyRate	1000.0	14186.355	7051.393949	2094.0	8166.25	14019.0	20296.75	26999.0
NumCompaniesWorked	1000.0	2.689	2.533120	0.0	1.00	1.5	4.00	9.0
PercentSalaryHike	1000.0	15.192	3.657118	11.0	12.00	14.0	18.00	25.0
PerformanceRating	1000.0	3.154	0.361129	3.0	3.00	3.0	3.00	4.0
RelationshipSatisfaction	1000.0	2.741	1.087705	1.0	2.00	3.0	4.00	4.0
StandardHours	1000.0	80.000	0.000000	80.0	80.00	80.0	80.00	80.0
StockOptionLevel	1000.0	0.762	0.836694	0.0	0.00	1.0	1.00	3.0
TotalWorkingYears	1000.0	11.410	8.006748	0.0	6.00	10.0	16.00	40.0
TrainingTimesLastYear	1000.0	2.773	1.311942	0.0	2.00	3.0	3.00	6.0
WorkLifeBalance	1000.0	2.763	0.698082	1.0	2.00	3.0	3.00	4.0
JoiningYearinCompany	1000.0	2012.866	6.355032	1980.0	2011.00	2015.0	2017.00	2020.0
YearsInCurrentRole	1000.0	4.266	3.635720	0.0	2.00	3.0	7.00	18.0
YearsSinceLastPromotion	1000.0	2.235	3.302830	0.0	0.00	1.0	3.00	15.0
YearsWithCurrManager	1000.0	4.168	3.630283	0.0	2.00	3.0	7.00	17.0

In [26]:
```
#to find the shape(rows,columns) of the dataset
df.shape
```

Out[26]:
```
(1000, 35)
```

In [27]:
```
#To find the unique set of data in each column which is having categorical data
print(df['BusinessTravel'].unique())
print(df['Department'].unique())
print(df['EducationField'].unique())
print(df['Gender'].unique())
print(df['JobRole'].unique())
print(df['MaritalStatus'].unique())
print(df['Over18'].unique())
print(df['OverTime'].unique())
```

```
['Travel_Rarely' 'Travel_Frequently' nan 'Non-Travel']
['Sales' 'Research & Development' 'Human Resources']
['Life Sciences' 'Other' 'Medical' 'Mdcl' 'life sciences' 'Marketing'
 'Technical Degree' 'Human Resources']
['Female' 'Male']
['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
['Single' 'Married' nan 'Divorced']
['Y']
['Yes' 'No']
```

From the above cell, we can see we have some nan values and some repitions of data in 'BusinessTravel', 'EducationalField' and 'MaritalStatus' columns

Clean up the Educationfield column

In [28]:
```
#Check the count of unique values in EducationField
print(df['EducationField'].value_counts())
print("Total counts is:",(df['EducationField'].value_counts().sum()))
```

```
Life Sciences       403
Medical             303
Marketing           115
Technical Degree     87
Other                52
Human Resources      15
life sciences        14
Mdcl                 11
Name: EducationField, dtype: int64
Total counts is: 1000
```

In [29]:
```
#To clean up the data in EducationalField column by using replace for life sciences and Mdcl data
df['EducationField'] = df['EducationField'].replace('life sciences','Life Sciences')
df['EducationField'] = df['EducationField'].replace('Mdcl','Medical')
```

In [30]:
```
#Checking the unique values and value_counts again after cleaning the data
print(df['EducationField'].unique())
print(df['EducationField'].value_counts())
print("Total counts is:",(df['EducationField'].value_counts().sum()))
```

```
['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']
Life Sciences       417
Medical             314
Marketing           115
Technical Degree     87
Other                52
Human Resources      15
Name: EducationField, dtype: int64
Total counts is: 1000
```

Clean up the BusinessTravel column

In [31]:
```
#To find the count of each unique value in BusinessTravel column
print(df['BusinessTravel'].value_counts())
print("Total counts is:",(df['BusinessTravel'].value_counts().sum()))
```

```
Travel_Rarely        677
Travel_Frequently    192
Non-Travel           101
Name: BusinessTravel, dtype: int64
Total counts is: 970
```

In [32]:
```
#To fill the missing values with the most repeated data in that column or with the mode value
df['BusinessTravel'] = df['BusinessTravel'].fillna(df['BusinessTravel'].mode()[0])
```

In [33]:
```
#Checking again the counts to see if the missing values are treated or not
print(df['BusinessTravel'].value_counts())
print("Total counts is:",(df['BusinessTravel'].value_counts().sum()))
```

```
Travel_Rarely        707
Travel_Frequently    192
Non-Travel           101
Name: BusinessTravel, dtype: int64
Total counts is: 1000
```

Clean up the MaritalStatus column

In [34]:
```
#To find the count of each unique value in MaritalStatus column
print(df['MaritalStatus'].value_counts())
print("Total counts is:",(df['MaritalStatus'].value_counts().sum()))
```

```
Married     427
Single      329
Divorced    229
Name: MaritalStatus, dtype: int64
Total counts is: 985
```

In [35]:
```
#To fill the missing values with the most repeated data in that column or with the mode value
df['MaritalStatus'] = df['MaritalStatus'].fillna('Married',axis=0)
```

In [36]:
```
#Checking again the counts to see if the missing values are treated or not
print(df['MaritalStatus'].value_counts())
print("Total counts is:",(df['MaritalStatus'].value_counts().sum()))
```

```
Married     442
Single      329
Divorced    229
Name: MaritalStatus, dtype: int64
Total counts is: 1000
```

Creating new columns and dropping the unnecessary columns from the dataframe

In [38]:
```
#Creating a new column which has the total years employee worked in the company since joining year is unnecessary
df['YearsinCompany'] = 2020 - df['JoiningYearinCompany']
```

In [39]:
```
#Checking if the new column is added to the dataframe or not
df.head()
```

Out[39]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...
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 × 36 columns

In [40]:
```
#Dropping or removing the unnecessary columns
df = df.drop('EmployeeNumber', axis = 1)
df = df.drop('StandardHours', axis = 1)
df = df.drop('EmployeeCount', axis = 1)
df = df.drop('Over18', axis = 1)
df = df.drop('JoiningYearinCompany', axis = 1)
```

In [41]:
```
#Checking the shape of dataframe after removing the columns
df.shape
```

Out[41]: (1000, 31)

In [42]:
```
#Separating the output (Attrition column data) from the inputs(rest all features)
df['Age_Years'] = df['Age']
#Remove the first column called age
df = df.drop('Age', axis = 1)
df.tail()
```

Out[42]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRat
995	No	Travel_Rarely	930	Research & Development	6	3	Medical	1	Female	7
996	No	Travel_Rarely	205	Sales	10	3	Marketing	4	Female	9
997	Yes	Travel_Rarely	135	Research & Development	17	4	Life Sciences	4	Female	5
998	No	Travel_Rarely	683	Research & Development	2	1	Medical	1	Male	3
999	No	Travel_Rarely	1147	Human Resources	10	3	Human Resources	3	Female	3

5 rows × 31 columns

In [43]:
```
#Creating a new dataframe (new_df) with same data as dataframe df , so that new_df will be used in EDA
new_df = df.copy()
new_df.shape
```

Out[43]: (1000, 31)

In [44]:
```
df.head()
```

Out[44]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate
0	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	2	Female	94
1	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	3	Male	61
2	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	4	Male	92
3	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	4	Female	56
4	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	Male	40

5 rows × 31 columns

In [26]:
```
#To export data from df dataframe to an excel
df.to_excel (r'C:\Users\User\Desktop\PIP - CTS\export_dataframe.xlsx', index = False, header=True)
```

Converting Categorical data to numerical data

In [45]:
```
#Importing the LabelEncoder to convert the categorical data to numeric since machine understands only numeric data
from sklearn.preprocessing import LabelEncoder
#Instantiate the LabelEncoder class
le = LabelEncoder()
```

In [46]:
```
#Applying the fit_transform method of LabelEncoder to the attrition column which is having categorical data
df['Attrition'] = le.fit_transform(df['Attrition'])
```

In [47]:
```
# See the Attrition column , it has been converted to 1 for Yes and 0 for No values using LabelEncoder
df.head()
```

Out[47]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate
0	1	Travel_Rarely	1102	Sales	1	2	Life Sciences	2	Female	94
1	0	Travel_Frequently	279	Research & Development	8	1	Life Sciences	3	Male	61
2	1	Travel_Rarely	1373	Research & Development	2	2	Other	4	Male	92
3	0	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	4	Female	56
4	0	Travel_Rarely	591	Research & Development	2	1	Medical	1	Male	40

5 rows × 31 columns

In [48]:
```
#Checking the datatype of a caetgorical column
if df['Department'].dtype == np.object:
    print('True')
else:
    print('False')
```

```
True
```

In [49]:
```
# Using for loop to apply the fit_transform method to remaining columns having categorical data
for column in df.columns:
        if df[column].dtype == np.object:
            df[column] = LabelEncoder().fit_transform(df[column])
        else:
            continue
```

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

Out[50]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRat
995	0	2	930	1	6	3	3	1	0	7
996	0	2	205	2	10	3	2	4	0	9
997	1	2	135	1	17	4	1	4	0	5
998	0	2	683	1	2	1	3	1	1	3
999	0	2	1147	0	10	3	0	3	0	3

5 rows × 31 columns

OutlierDetection

In [51]:
```
#Importing the visualization libraries(matplotlib and seaborn)
import matplotlib.pyplot as plt
%matplotlib inline


import seaborn as sns
```

In [52]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 31 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Attrition                 1000 non-null   int32
 1   BusinessTravel            1000 non-null   int32
 2   DailyRate                 1000 non-null   int64
 3   Department                1000 non-null   int32
 4   DistanceFromHome          1000 non-null   int64
 5   Education                 1000 non-null   int64
 6   EducationField            1000 non-null   int32
 7   EnvironmentSatisfaction   1000 non-null   int64
 8   Gender                    1000 non-null   int32
 9   HourlyRate                1000 non-null   int64
 10  JobInvolvement            1000 non-null   int64
 11  JobLevel                  1000 non-null   int64
 12  JobRole                   1000 non-null   int32
 13  JobSatisfaction           1000 non-null   int64
 14  MaritalStatus             1000 non-null   int32
 15  MonthlyIncome             1000 non-null   int64
 16  MonthlyRate               1000 non-null   int64
 17  NumCompaniesWorked        1000 non-null   int64
 18  OverTime                  1000 non-null   int32
 19  PercentSalaryHike         1000 non-null   int64
 20  PerformanceRating         1000 non-null   int64
 21  RelationshipSatisfaction  1000 non-null   int64
 22  StockOptionLevel          1000 non-null   int64
 23  TotalWorkingYears         1000 non-null   int64
 24  TrainingTimesLastYear     1000 non-null   int64
 25  WorkLifeBalance           1000 non-null   int64
 26  YearsInCurrentRole        1000 non-null   int64
 27  YearsSinceLastPromotion   1000 non-null   int64
 28  YearsWithCurrManager      1000 non-null   int64
 29  YearsinCompany            1000 non-null   int64
 30  Age_Years                 1000 non-null   int64
dtypes: int32(8), int64(23)
memory usage: 211.1 KB
```

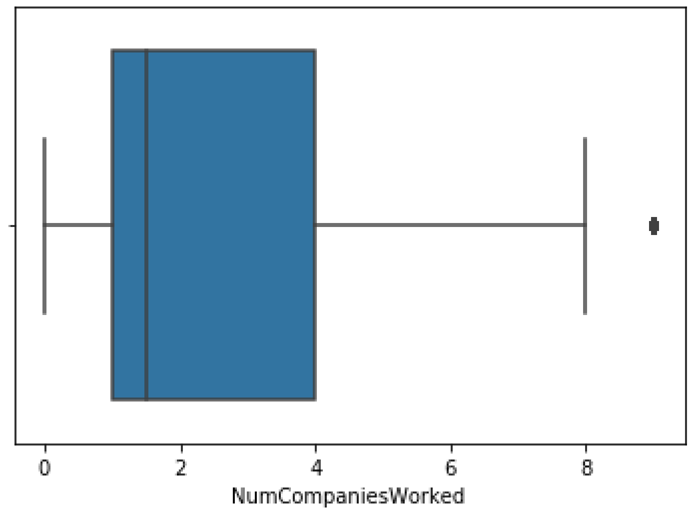
In statistics, an outlier is an observation point that is distant from other observations.

The above definition suggests that outlier is something which is separate/different from the crowd.

**In descriptive statistics, a box plot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. Outliers may be plotted as individual points.

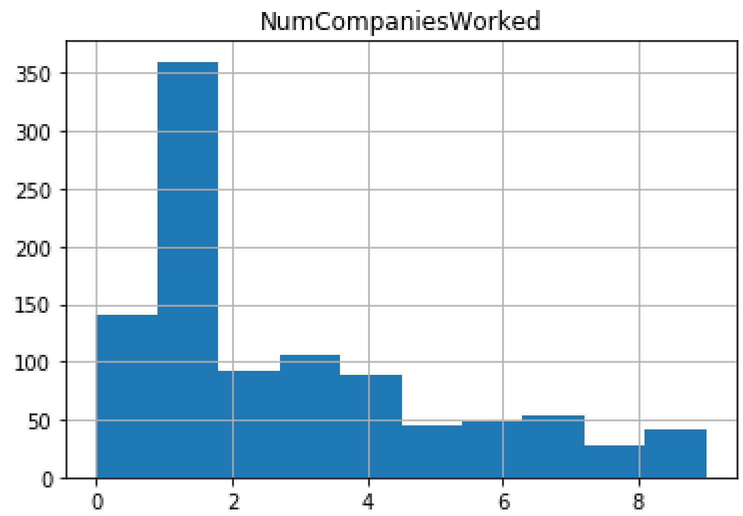
In [53]:
```
sns.boxplot(x=df['NumCompaniesWorked'])
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x27858ba588>



In [54]:
```
df.hist(['NumCompaniesWorked'])
```

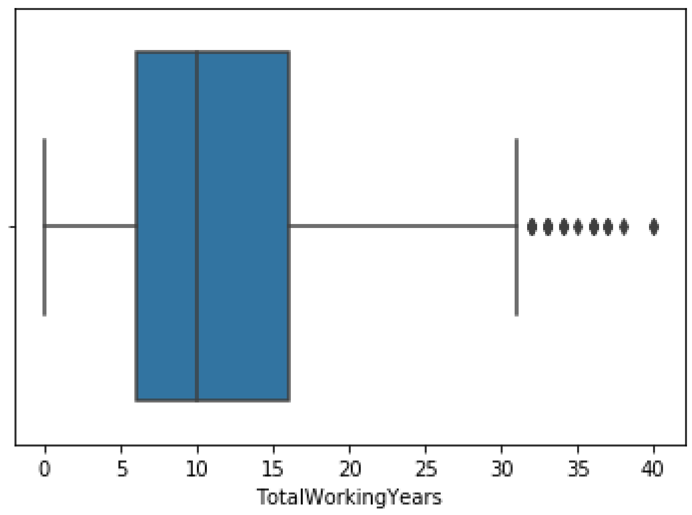
Out[54]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000000278600C588>]],
      dtype=object)
```



**** In the above figure we have plotted (boxplot and histogram) for 'NumCompaniesWorked' column and found that we have very less outliers which can be ignored.**

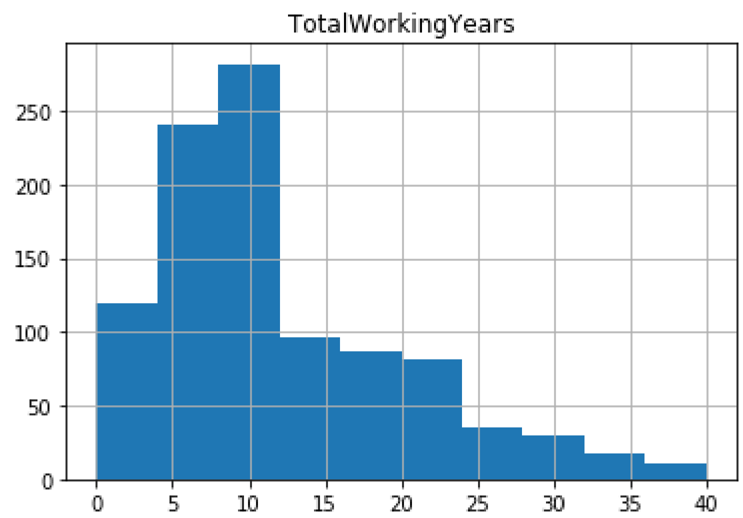
In [55]:
```
sns.boxplot(x=df['TotalWorkingYears'])
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x27860c5c48>



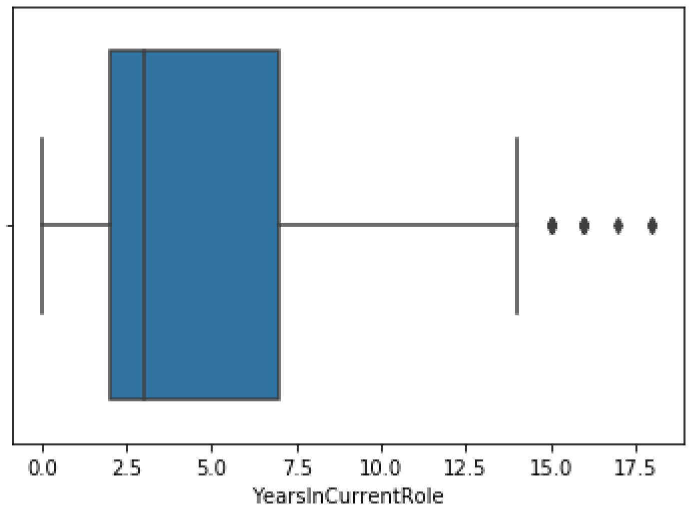
In [56]:
```
df.hist(['TotalWorkingYears'])
```

Out[56]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000002786146748>]],
      dtype=object)
```

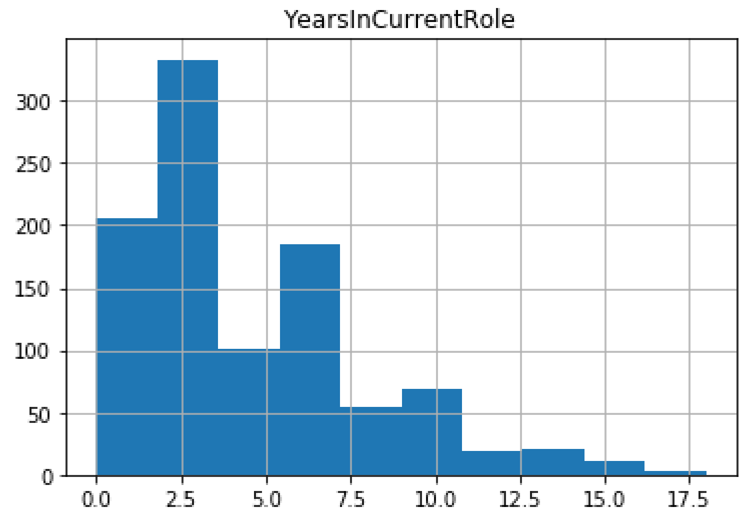


**** In the above figure we have plotted (boxplot and histogram) for 'TotalWorkingYears' column and found that we have some outliers that needs to be corrected.**

In [57]:
```
sns.boxplot(x=df['YearsInCurrentRole'])
```
Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27861ae648>
```

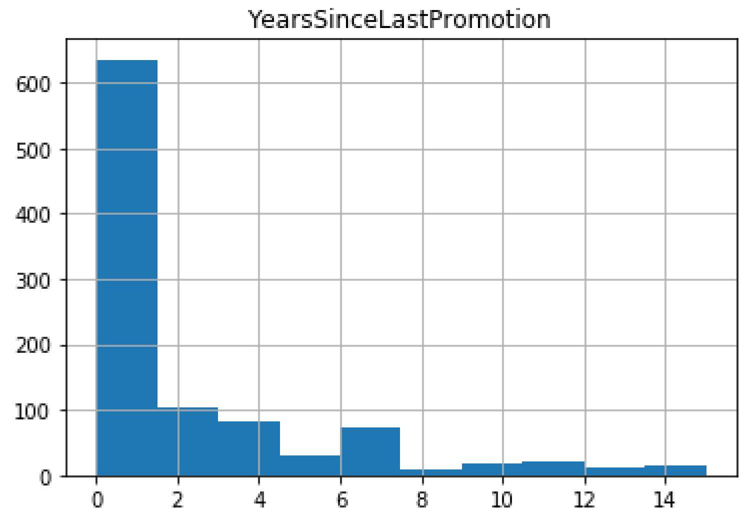


In [58]:
```
df.hist(['YearsInCurrentRole'])
```
Out[58]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000000278626F088>]],
      dtype=object)
```



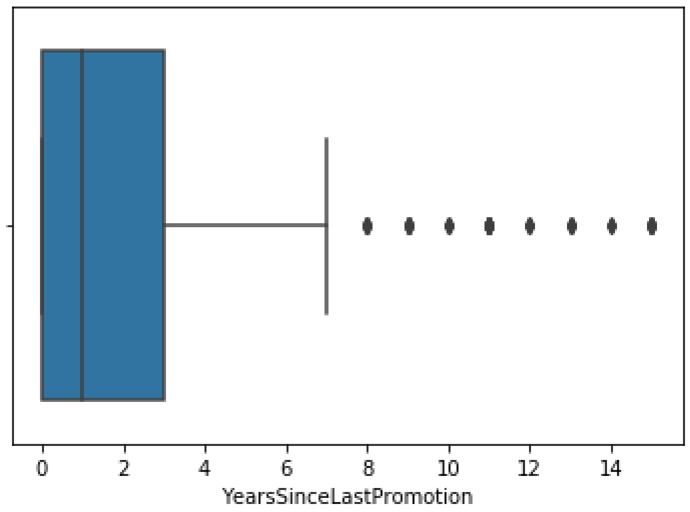
**** In the above figure we have plotted (boxplot and histogram) for 'YearsInCUrrentRole' column and found that we have some outliers that needs to be corrected.**

In [59]:
```
df.hist(['YearsSinceLastPromotion'])
```
Out[59]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000027862FAA48>]],
      dtype=object)
```



In [60]:
```
sns.boxplot(x=df['YearsSinceLastPromotion'])
```

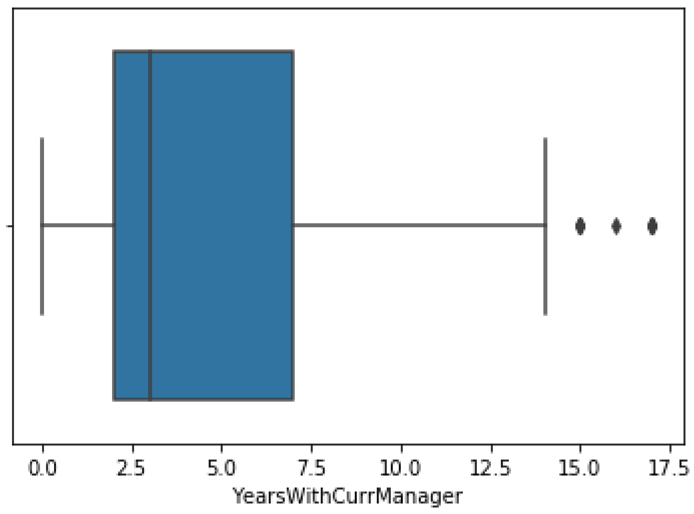
Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x27863829c8>



**** In the above figure we have plotted (boxplot and histogram) for 'YearsSinceLastPromotion' column and found that we have some outliers that needs to be corrected.**

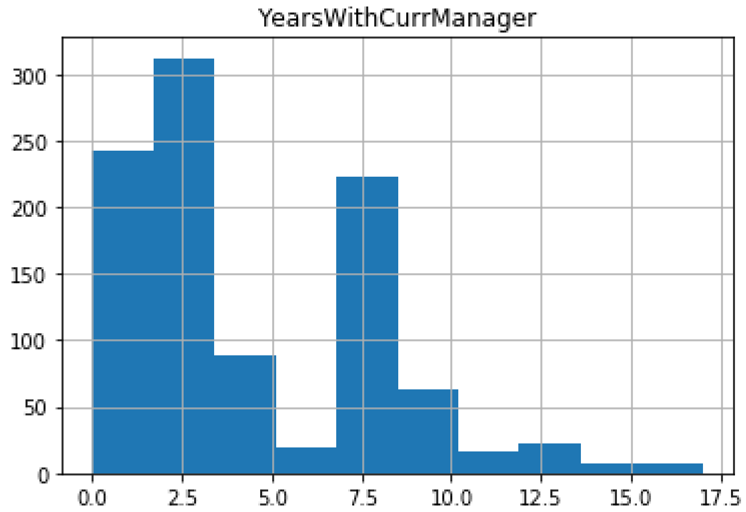
In [61]:
```
sns.boxplot(x=df['YearsWithCurrManager'])
```

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x27863f2a08>



In [62]:
```
df.hist(['YearsWithCurrManager'])
```

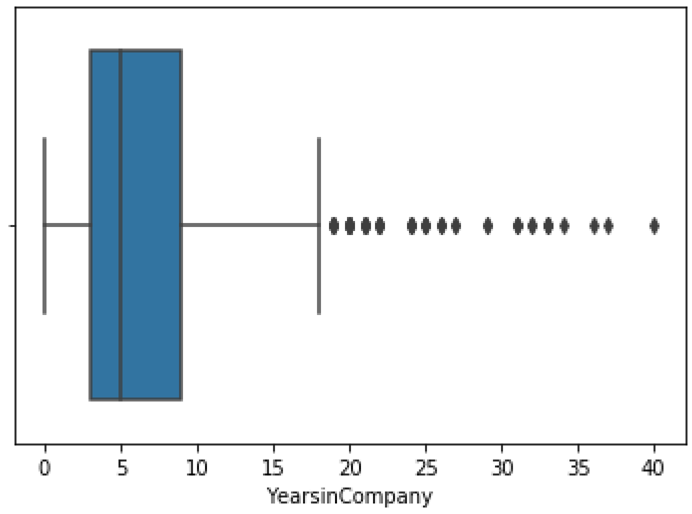
Out[62]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000000278645DF48>]],
      dtype=object)
```



**** In the above figure we have plotted (boxplot and histogram) for 'YearsWithCurrManager' column and found that we have some outliers that needs to be corrected.**

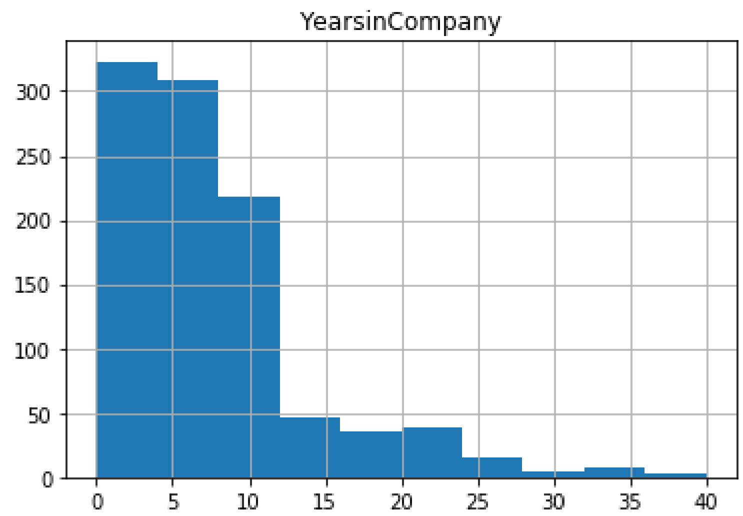
In [63]:
```
sns.boxplot(x=df['YearsinCompany'])
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x27864ed2c8>



In [64]:
```
df.hist(['YearsinCompany'])
```

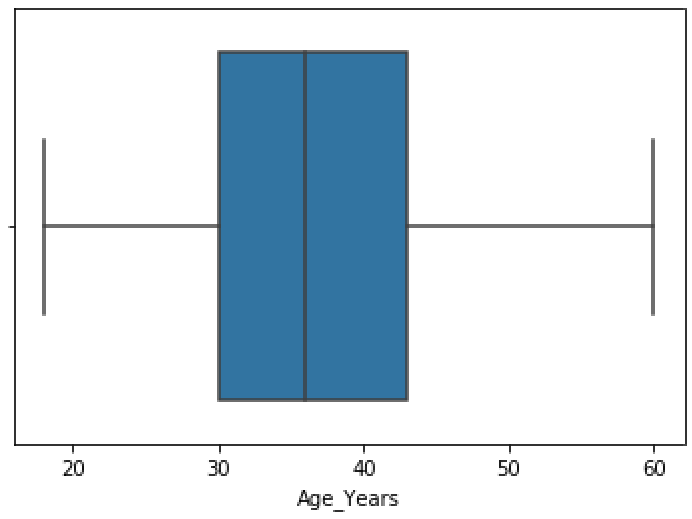
Out[64]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000002786532F88>]],
      dtype=object)
```



**** In the above figure we have plotted (boxplot and histogram) for 'YearsinCompany' column and found that we have some outliers that needs to be corrected.**

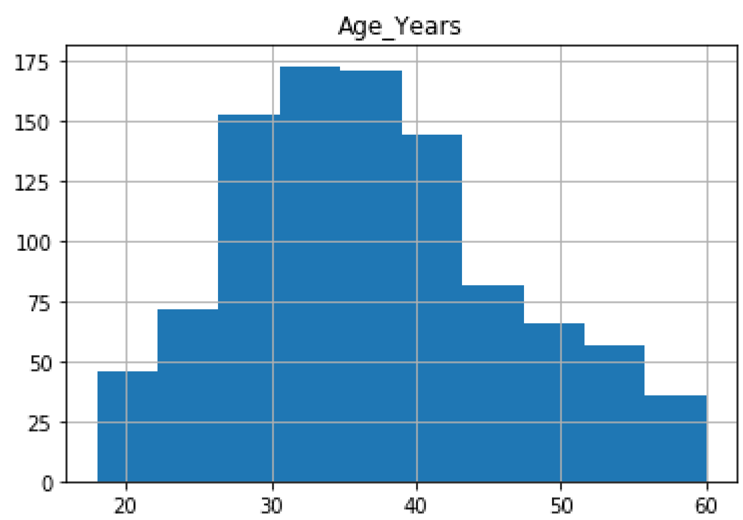
In [65]:
```
sns.boxplot(x=df['Age_Years'])
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x27865f1548>



In [66]:
```
df.hist(['Age_Years'])
```

Out[66]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000027865A72C8>]],
      dtype=object)
```



**** In the above figure we have plotted (boxplot and histogram) for 'Age_Years' column and found no outliers for this.**

Outliers Correction

*****Skewness refers to distortion or asymmetry in a symmetrical bell curve, or normal distribution, in a set of data. If the curve is shifted to the left or to the right, it is said to be skewed.**

*****Ideally, the skewness value should be between -1 and +1, and any major deviation from this range indicates the presence of extreme values.**

***** Check for skewness: [https://en.wikipedia.org/wiki/Skewness (https://en.wikipedia.org/wiki/Skewness)](https://en.wikipedia.org/wiki/Skewness)**

In [67]:
```
##Here I am printing out the skewness of those columns where we had seen some outliers.
print(df['YearsinCompany'].skew())
print(df['YearsWithCurrManager'].skew())
print(df['YearsSinceLastPromotion'].skew())
print(df['YearsInCurrentRole'].skew())
print(df['TotalWorkingYears'].skew())
```

```
1.7608766176932167
0.8253205002646364
1.9608612190990318
0.8925762914864066
1.0734748040426976
```

From the above skewness results , we can make the outlier corrections for 'YearsinCompany' and 'YearsSinceLastPromotion' since those values goes beyond 1 , rest all looks good.

In [68]:
```
#Creating a new dataframe and storing the filtered results(that lie between 25percent to 75percent) for 'YearsinCompan
y'column
Q1 = df['YearsinCompany'].quantile(0.25)
Q3 = df['YearsinCompany'].quantile(0.75)
IQR = Q3 - Q1
filtered_df = df.query('(@Q1 - 1.5 * @IQR) <= YearsinCompany <= (@Q3 + 1.5 * @IQR)')
```

In [69]:
```
#The skewness got reduced to 0.80 from 1.76 after outliers removal
print(filtered_df['YearsinCompany'].skew())
```

```
0.8011134281101522
```

In [70]:
```
#Finding the shape of original dataframe and the filtered one to see the no of records filtered out
df.shape, filtered_df.shape
```

Out[70]:
```
((1000, 31), (923, 31))
```

In [71]:
```
#Creating a new DF and storing the filtered results(that lie between 25percent to 75percent) for 'YearsSinceLastPromot
ion'column
Q1 = filtered_df['YearsSinceLastPromotion'].quantile(0.25)
Q3 = filtered_df['YearsSinceLastPromotion'].quantile(0.75)
IQR = Q3 - Q1
filtered_df1 = filtered_df.query('(@Q1 - 1.5 * @IQR) <= YearsSinceLastPromotion <= (@Q3 + 1.5 * @IQR)')
```

In [72]:
```
#The skewness got reduced to 0.99 from 1.96 after outliers removal
print(filtered_df1['YearsinCompany'].skew())
```

```
0.9993965848397647
```

In [73]:
```
#Finding the shape of original dataframe and the filtered one to see the no of records filtered out
df.shape, filtered_df.shape, filtered_df1.shape
```

Out[73]:
```
((1000, 31), (923, 31), (814, 31))
```

***** So here we concluded that after outliers correction and removal we have now 814 rows and 31 columns and this will be our final dataset on which we will apply our Machine Learning algorithms and models.**

GOAL 2: EXPLORATORY DATA ANALYSIS (EDA) USING PYTHON LIBRARIES (MATPLOTLIB,SEABORN)

In [74]:
```
#Importing the visualization libraries(matplotlib and seaborn)
import matplotlib.pyplot as plt
%matplotlib inline


import seaborn as sns
```

In [54]:
```
new_df.head()
```

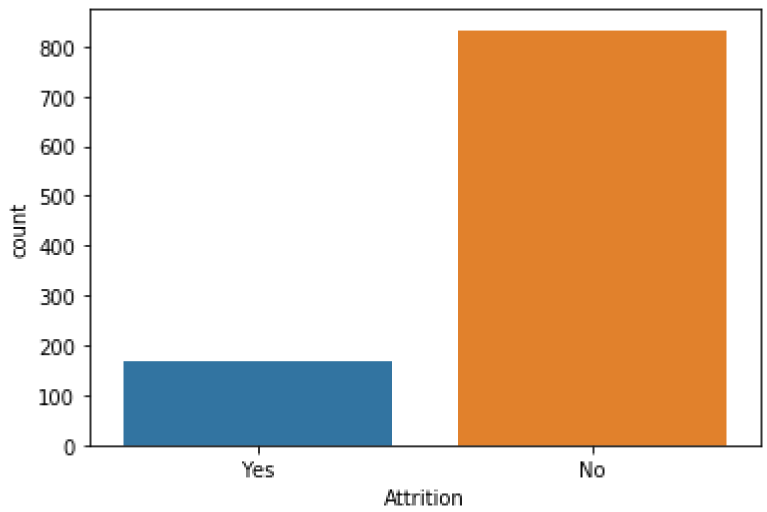
Out[54]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate
0	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	2	Female	94
1	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	3	Male	61
2	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	4	Male	92
3	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	4	Female	56
4	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	Male	40

5 rows × 31 columns

In [55]:
```
#To check the count of Attrition
sns.countplot(new_df['Attrition'])
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x71f93eda08>



Here we can see the count of attrition is less (like somewhere within 200 for a total 1000 records).

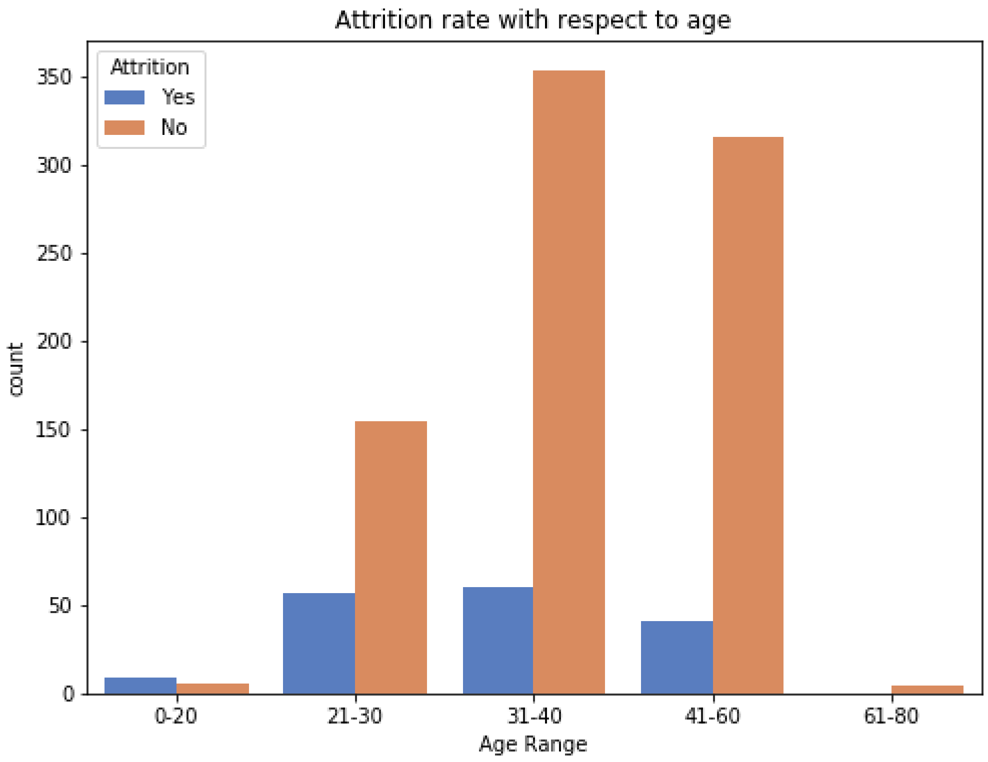
In [56]:
```
# Creating bins for Age_Years column so that it will be easy to visualize
bins = [0, 20, 30, 40, 60, 80]
Labels = ['0-20', '21-30','31-40','41-60','61-80']
new_df['Age Range'] = pd.cut(new_df['Age_Years'],bins=bins, labels=Labels, right=False)
new_df.tail()
```

Out[56]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRat
995	No	Travel_Rarely	930	Research & Development	6	3	Medical	1	Female	7
996	No	Travel_Rarely	205	Sales	10	3	Marketing	4	Female	9
997	Yes	Travel_Rarely	135	Research & Development	17	4	Life Sciences	4	Female	5
998	No	Travel_Rarely	683	Research & Development	2	1	Medical	1	Male	3
999	No	Travel_Rarely	1147	Human Resources	10	3	Human Resources	3	Female	3

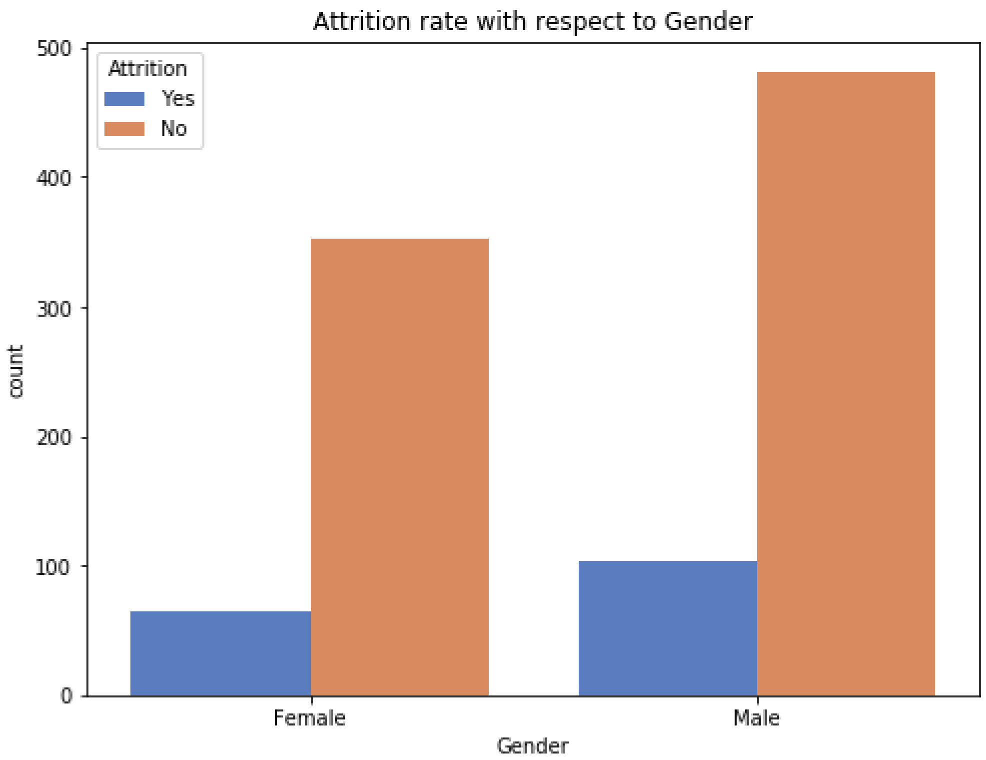
5 rows × 32 columns

In [57]:
```
#Appyling the Age Range data to view
plt.figure(figsize=(8,6))
plt.title('Attrition rate with respect to age')
sns.countplot(x='Age Range', hue='Attrition', data = new_df, palette="muted");
```



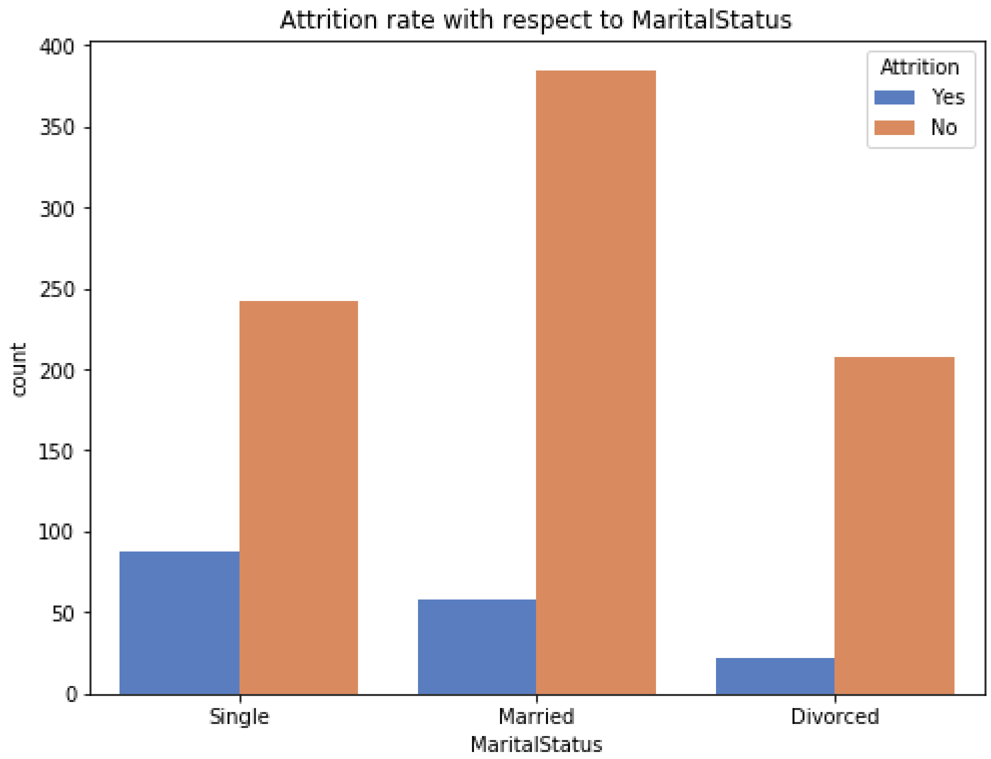
From this graph we can see the attrition count is high somewhere in 21-30 and 31-40 years of age.

In [58]:
```
#This is to check which has the highest attriction rate -Male/Female
plt.figure(figsize=(8,6))
plt.title('Attrition rate with respect to Gender')
sns.countplot(x='Gender', hue='Attrition', data = new_df, palette="muted");
```



In this plot we can clearly see that the attrition rate is higher for Males than females.

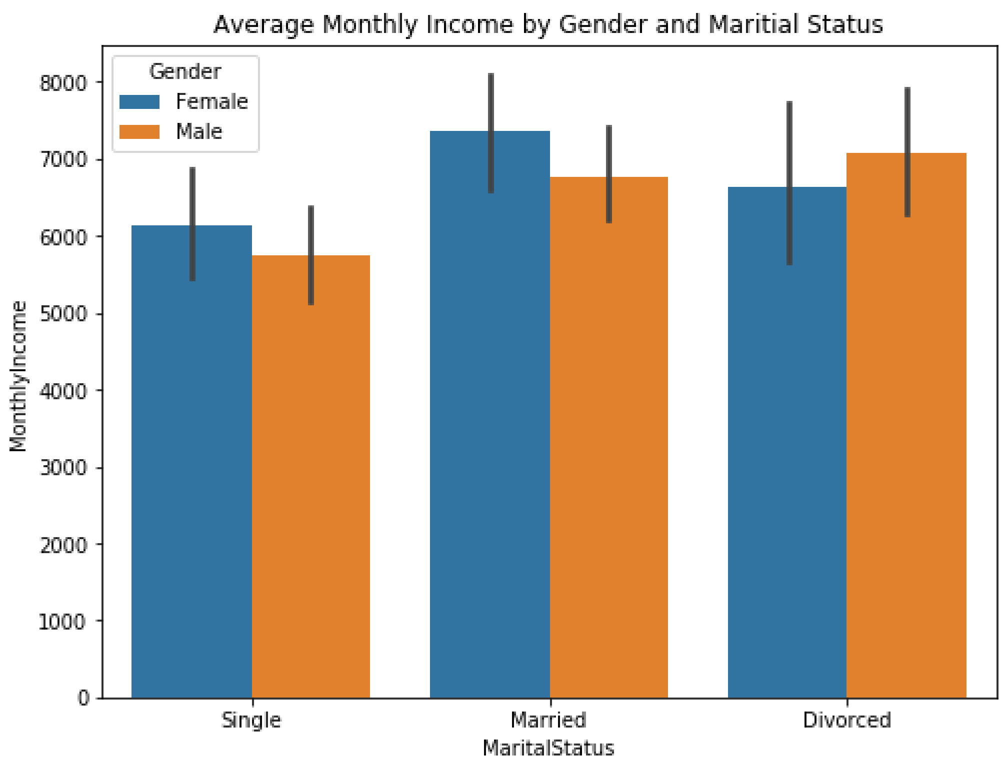
In [59]:
```
#This is to check which MaritalStatus has the highest attriction rate
plt.figure(figsize=(8,6))
plt.title('Attrition rate with respect to MaritalStatus')
sns.countplot(x='MaritalStatus', hue='Attrition', data = new_df, palette="muted");
```



The people who are Single tend to leave the companies more than married or divorced people.

In [60]:
```
##To figure out the average monthly income with respect to Gender and MaritalStatus
plt.figure(figsize=(8,6))
plt.title('Average Monthly Income by Gender and Maritial Status')
sns.barplot(x="MaritalStatus", y="MonthlyIncome", hue="Gender", data=new_df)
```

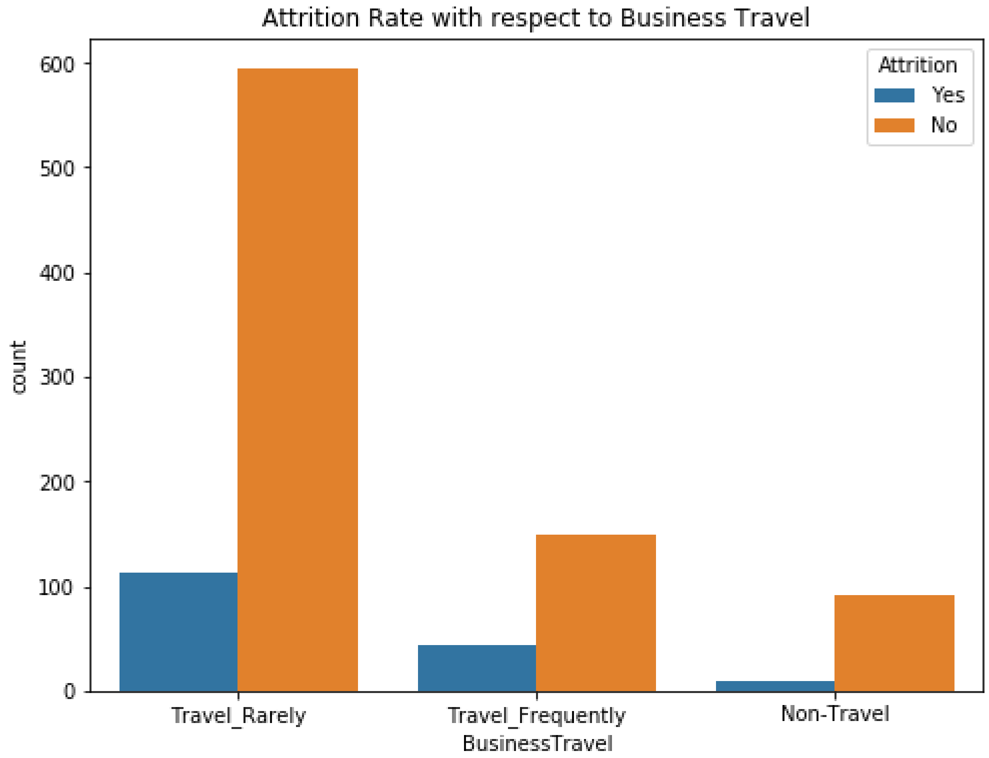
Out[60]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x71fa5e5d08>
```



From this graph we are seeing that Married and Divorced people get a better salary than Single people, hence that may be the reason single people tend to leave the companies more frequent.

In [61]:
```
#To figure out the attrition rate with respect to BusinessTravel categories
plt.figure(figsize=(8,6))
plt.title('Attrition Rate with respect to Business Travel')
sns.countplot(x="BusinessTravel", hue="Attrition", data=new_df)
```

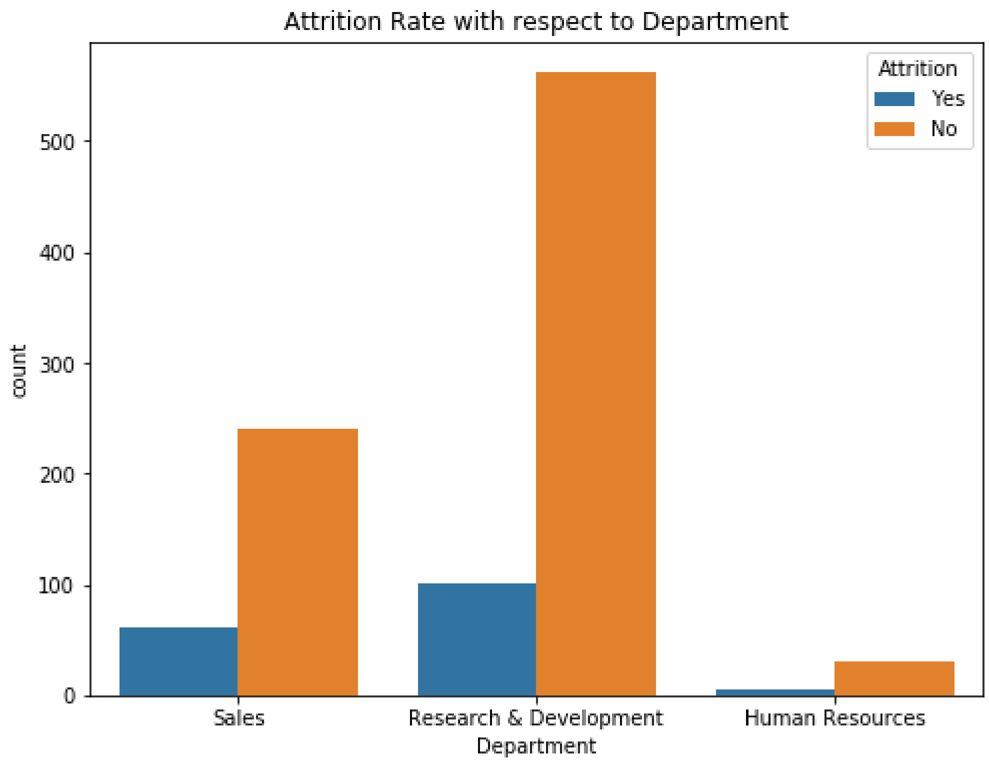
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x71fa6772c8>



People who travel rarely or who are rarely given a chance to move onsite tend to leave the companies as seen in this graph.

In [62]:
```
#To figure out the attrition rate with respect to different Departments
plt.figure(figsize=(8,6))
plt.title('Attrition Rate with respect to Department')
sns.countplot(x="Department", hue="Attrition", data=new_df)
```

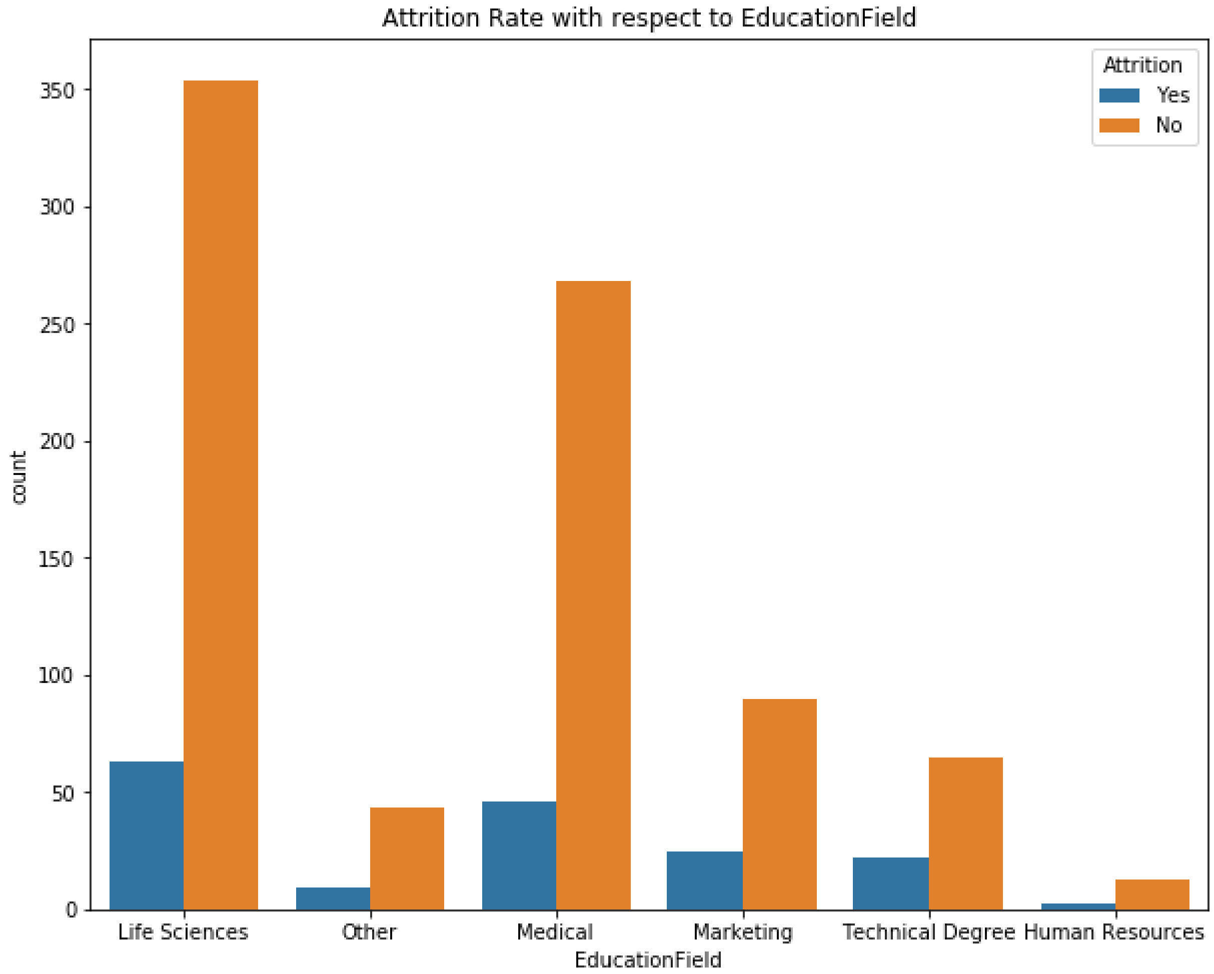
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x71f956bc48>



This above plot shows that attrition rate is minimal for Human Resources department and highest for R&D department.

In [63]:
```
#To figure out the attrition rate with respect to different Education Fields
plt.figure(figsize=(10,8))
plt.title('Attrition Rate with respect to EducationField')
sns.countplot(x="EducationField", hue="Attrition", data=new_df)
```

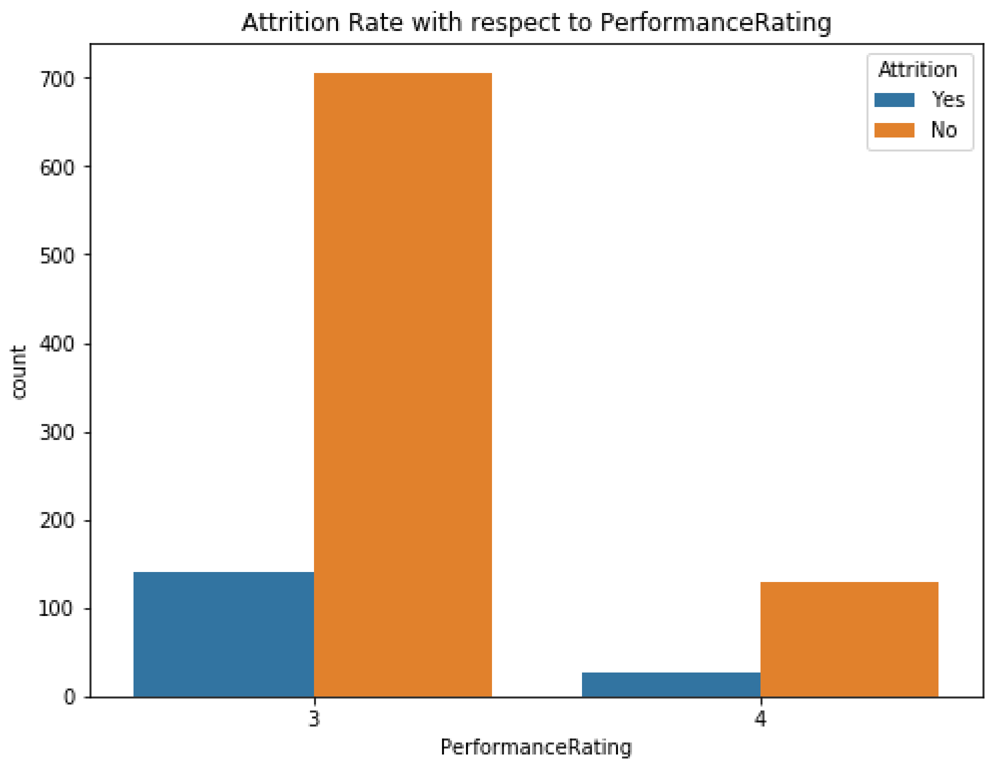
Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x71f94cb508>



Similarly the above graph shows the attrition rate as per the Education field and as per that its higher for LifeSciences category.

In [64]:
```
#To figure out the attrition rate with respect to PerformanceRating
plt.figure(figsize=(8,6))
plt.title('Attrition Rate with respect to PerformanceRating')
sns.countplot(x="PerformanceRating", hue="Attrition", data=new_df)
```

Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x71f95d6308>



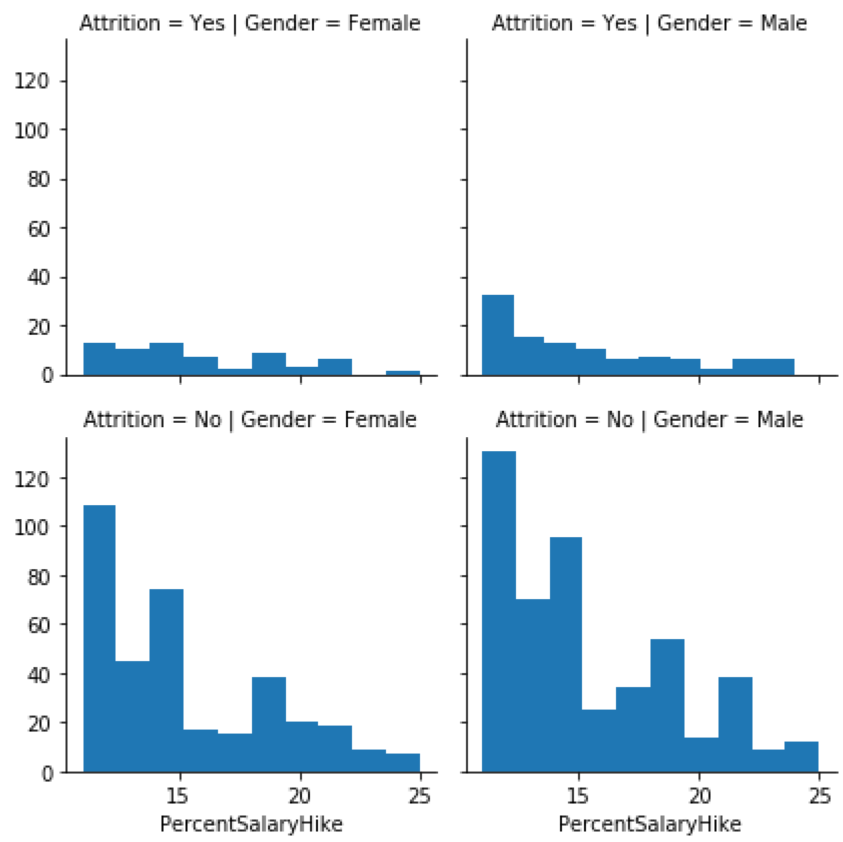
As per this dataset people in the organization are provided with ratings 3(Excellent) and 4(Outstanding), so clearly the attrition rate would be much lower for people with 4th rating as shown in above graph.

In [65]:
```
#Which range of age is involved in Overtime
plt.figure(figsize=(12,8))
plt.title('Age Distribution of Employees who have worked Over Time')
#sns.distplot(hr.YearsAtCompany, bins = np.linspace(0,40,40))
sns.distplot(new_df.Age_Years[new_df.OverTime == 'Yes'] , color = 'Blue',kde = True,bins = np.linspace(0,70,35))
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x71fa6eb5c8>

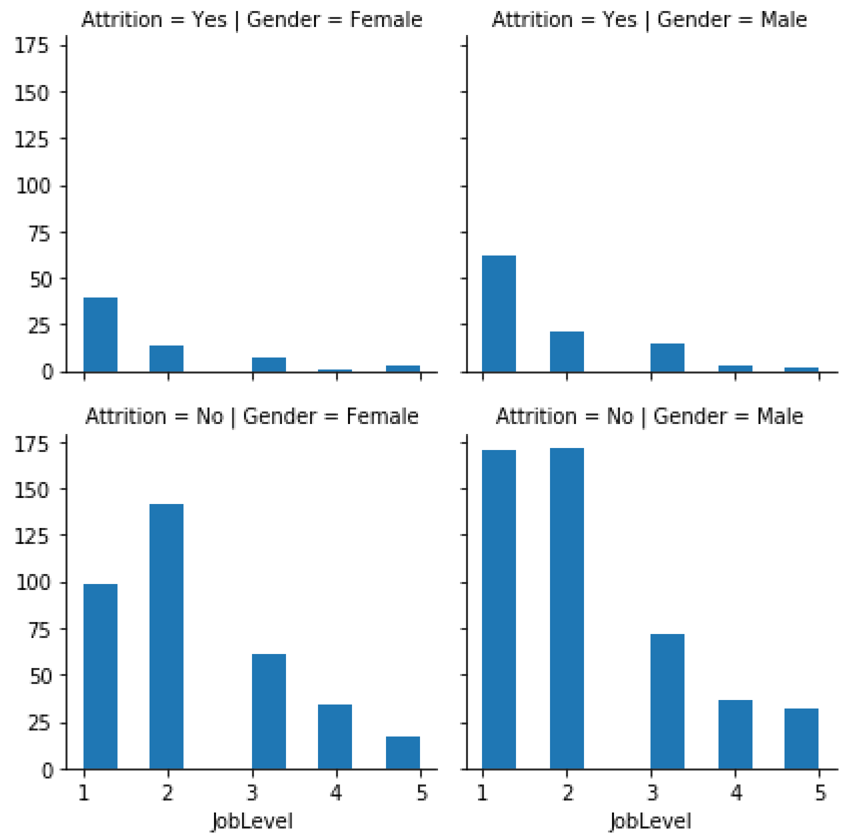
This above graph is having a uniformly ditributed data which shows that people in all age categories are doing overtime and mostly between 30-50 years of age.

In [66]:
```
##With respect to gender and attrition , what is the Percentage in salary hike?
g =sns.FacetGrid(data = new_df,col = 'Gender', row = 'Attrition')
g = g.map(plt.hist, 'PercentSalaryHike')
```



This graph shows the percent hikes for males and females (with and without attrition), and we can clearly see that people with lower salary hikes tend to leave the companies more (both males and females)

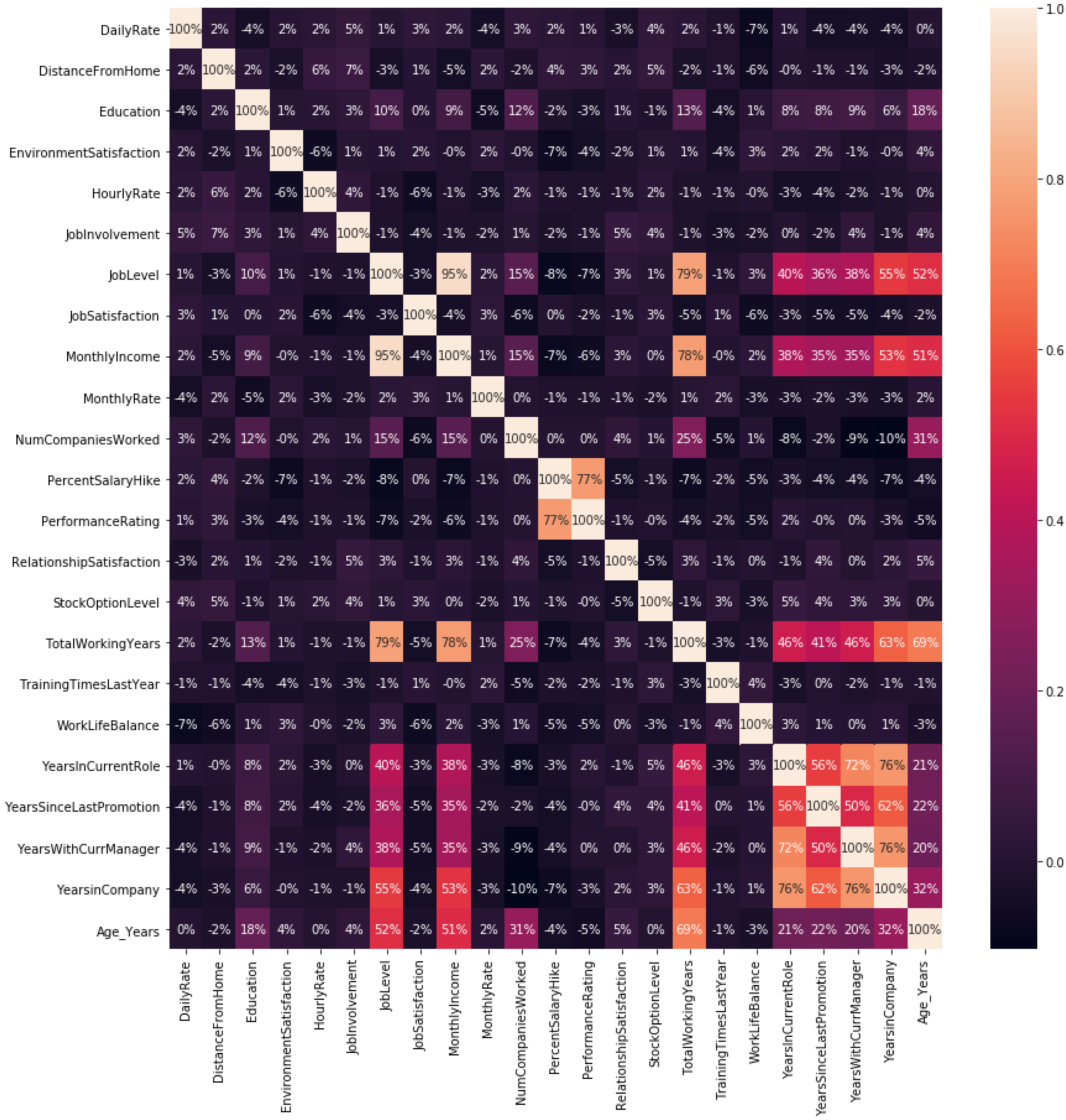
In [67]:
```
#With respect to gender and attrition , what is the job satisfaction
g =sns.FacetGrid(data = new_df,col = 'Gender', row = 'Attrition')
g = g.map(plt.hist, 'JobLevel')
```



This graph shows the JobLevel for both males and females and with respect to that what is the attrition.As Job level increase people become more stable and attrition count is low for them.

In [68]:
```
#Plotting the heatmap for the dataframe to show the correlations among the different features
plt.figure(figsize=(14,14))  #14in by 14in
sns.heatmap(new_df.corr(), annot=True, fmt='.0%')
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x71f7479f08>
```



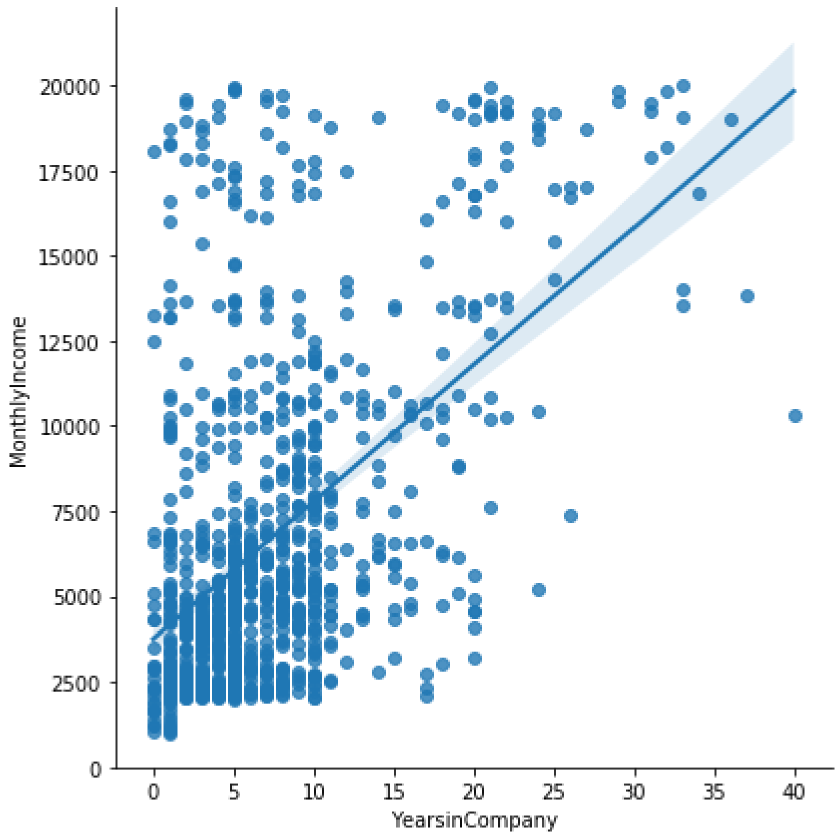
This is a heatmap and this graph shows the correlation among each other , Every feature is highly correlated with itself (100%)and JobLevel-MonthlyIncome , TotalWorkingyears-JobLevel, TotalWorkingYears-MonthlyIncome are also some having high correlations with each other.

In [69]:
```
#Line plot to display the relationship between yearsincompany and the Monthly income
plt.figure(figsize=(10,8))
sns.lmplot("YearsinCompany", "MonthlyIncome", data=new_df, size=6)
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\regression.py:574: UserWarning: The `size` parameter has been renam
ed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

Out[69]: <seaborn.axisgrid.FacetGrid at 0x71fb38e208>

<Figure size 720x576 with 0 Axes>



This is a line plot which shows that as the years in comapny increases the monthly income also increases.

GOAL 3: Apply Machine Learning algorithms to ensure the best fit model from EDA and calculate accuracy.

In [75]:
```
# We will be taking the filtered_df1 dataframe which is the final one after cleaning and outliers removal
filtered_df1.shape
```

Out[75]: (814, 31)

In [76]:
```
filtered_df1.head().T
```

Out[76]:

	0	1	2	3	4
Attrition	1	0	1	0	0
BusinessTravel	2	1	2	1	2
DailyRate	1102	279	1373	1392	591
Department	2	1	1	1	1
DistanceFromHome	1	8	2	3	2
Education	2	1	2	4	1
EducationField	1	1	4	1	3
EnvironmentSatisfaction	2	3	4	4	1
Gender	0	1	1	0	1
HourlyRate	94	61	92	56	40
JobInvolvement	3	2	2	3	3
JobLevel	2	2	1	1	1
JobRole	7	6	2	6	2
JobSatisfaction	4	2	3	3	2
MaritalStatus	2	1	2	1	1
MonthlyIncome	5993	5130	2090	2909	3468
MonthlyRate	19479	24907	2396	23159	16632
NumCompaniesWorked	8	1	6	1	9
OverTime	1	0	1	1	0
PercentSalaryHike	11	23	15	11	12
PerformanceRating	3	4	3	3	3
RelationshipSatisfaction	1	4	2	3	4
StockOptionLevel	0	1	0	0	1
TotalWorkingYears	8	10	7	8	6
TrainingTimesLastYear	0	3	3	3	3
WorkLifeBalance	1	3	3	3	3
YearsInCurrentRole	4	7	0	7	2
YearsSinceLastPromotion	0	1	0	3	2
YearsWithCurrManager	5	7	0	0	2
YearsinCompany	6	10	0	8	2
Age_Years	41	49	37	33	27

In [77]:
```
#Separate the data into X and y that will be fed to the Machine Learning algorithms
X = filtered_df1.iloc[:,1:]
y = filtered_df1.iloc[:,:1]
```

In [78]:
```
#Check the shape of X and y
X.shape,y.shape
```

Out[78]: ((814, 30), (814, 1))

In [79]:
```
#Normalizing the data so that all the data will be scaled to same level before fed to ML
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_scaled = sc.fit_transform(X)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)
X_scaled.head()
```

Out[79]:

	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	JobInvol
0	0.590907	0.715526	1.457402	-1.008419	-0.807298	-0.927035	-0.681092	-1.20185	1.425654	0.
1	-0.931240	-1.339815	-0.509970	-0.130099	-1.777964	-0.927035	0.235286	0.83205	-0.216820	-1.
2	0.590907	1.392315	-0.509970	-0.882944	-0.807298	1.336784	1.151664	0.83205	1.326110	-1.
3	-0.931240	1.439765	-0.509970	-0.757470	1.134034	-0.927035	1.151664	-1.20185	-0.465679	0.
4	0.590907	-0.560633	-0.509970	-0.882944	-1.777964	0.582178	-1.597470	0.83205	-1.262031	0.

5 rows × 30 columns

Apply SmoteTomek Upsampling since the target data is baised

In [80]:
```
#Checking the target column 'Attrition' where the count of records for both 0 and 1 class shows the baisness towards c
lass 0
y['Attrition'].value_counts()
```

Out[80]:
```
0    675
1    139
Name: Attrition, dtype: int64
```

In [81]:
```
#Splitting the data into train and test set in 75:25 ratio
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled ,y, test_size = 0.25, random_state = 1)
```

In [82]:
```
Y_train['Attrition'].value_counts()
```

Out[82]:
```
0    508
1    102
Name: Attrition, dtype: int64
```

In [83]:
```
#Applying the SMOTETomek upsampling so that the baised class will now be equal to other class
from imblearn.combine import SMOTETomek
sm = SMOTETomek(random_state=1)
X_train_resample, y_train_resample = sm.fit_resample(X_train, Y_train)
```

In [84]:
```
#After upsampling now both classes have same records
y_train_resample['Attrition'].value_counts()
```

Out[84]:
```
1    508
0    508
Name: Attrition, dtype: int64
```

1. Apply Logistic regression

In [85]:
```
#Instantiating and fitting the model to the training data only (X_train and Y_train)
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression(penalty='l1',solver='liblinear',random_state=1 )
logreg.fit(X_train_resample, y_train_resample)
#Checking the Training model accuracy
print(logreg.score(X_train_resample, y_train_resample))
print(logreg.score(X_train, Y_train))
print(logreg.score(X_test,Y_test))
```

```
0.8346456692913385
0.8032786885245902
0.8333333333333334

C:\Users\User\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [86]:
```
y_pred =logreg.predict(X_test)
```

In [87]:
```
from sklearn import metrics
print (metrics.accuracy_score(Y_test, y_pred))
```

```
0.8333333333333334
```

In [88]:
```
print (metrics.confusion_matrix(Y_test, y_pred))
print (metrics.classification_report(Y_test, y_pred))
```

```
[[139  28]
 [  6  31]]
              precision    recall  f1-score   support

           0       0.96      0.83      0.89       167
           1       0.53      0.84      0.65        37

    accuracy                           0.83       204
   macro avg       0.74      0.84      0.77       204
weighted avg       0.88      0.83      0.85       204
```

Logistic Regression Model has done a good job in predicting the no of people leaving the company i.e. it has correctly predicted 31 records correctly out of total 37.

2. Apply SVC Algorithm

In [89]:
```
from sklearn.svm import SVC
svc = SVC(C= 0.1, kernel='rbf', gamma= 0.1)
svc.fit(X_train_resample, y_train_resample)
#svc.fit(X_train, Y_train)
print(svc.score(X_train_resample, y_train_resample))
print(svc.score(X_test,Y_test))
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

0.8405511811023622
0.8235294117647058
```

In [90]:
```
y_pred_new = svc.predict(X_test)
from sklearn import metrics
print (metrics.accuracy_score(Y_test, y_pred_new))
```

```
0.8235294117647058
```

In [91]:
```
print (metrics.confusion_matrix(Y_test, y_pred_new))
print (metrics.classification_report(Y_test, y_pred_new))
```

```
[[163   4]
 [ 32   5]]
              precision    recall  f1-score   support

           0       0.84      0.98      0.90       167
           1       0.56      0.14      0.22        37

    accuracy                           0.82       204
   macro avg       0.70      0.56      0.56       204
weighted avg       0.79      0.82      0.78       204
```

3. Apply Decision Tree Classifier

In [92]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion ='entropy',max_depth=5,random_state=1)
dtc.fit(X_train_resample, y_train_resample)
print(dtc.score(X_train_resample, y_train_resample))
print(dtc.score(X_test,Y_test))
```

```
0.9094488188976378
0.8431372549019608
```

In [93]:
```
y_pred_dtc = svc.predict(X_test)
from sklearn import metrics
print (metrics.accuracy_score(Y_test, y_pred_dtc))
```

```
0.8235294117647058
```

In [94]:
```
print (metrics.confusion_matrix(Y_test, y_pred_dtc))
print (metrics.classification_report(Y_test, y_pred_dtc))
```

```
[[163   4]
 [ 32   5]]
              precision    recall  f1-score   support

           0       0.84      0.98      0.90       167
           1       0.56      0.14      0.22        37

    accuracy                           0.82       204
   macro avg       0.70      0.56      0.56       204
weighted avg       0.79      0.82      0.78       204
```

4. Apply RandomForestClassifier

In [95]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=25,
                                max_depth=5,random_state=1)
rfc.fit(X_train_resample, y_train_resample)
print(rfc.score(X_train_resample, y_train_resample))
print(rfc.score(X_test,Y_test))
```

```
C:\Users\User\anaconda3\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  after removing the cwd from sys.path.

0.9498031496062992
0.8529411764705882
```

In [96]:
```
y_pred_rfc = svc.predict(X_test)
from sklearn import metrics
print (metrics.accuracy_score(Y_test, y_pred_rfc))
```

```
0.8235294117647058
```

In [97]:
```
print (metrics.confusion_matrix(Y_test, y_pred_rfc))
print (metrics.classification_report(Y_test, y_pred_rfc))
```

```
[[163   4]
 [ 32   5]]
              precision    recall  f1-score   support

           0       0.84      0.98      0.90       167
           1       0.56      0.14      0.22        37

    accuracy                           0.82       204
   macro avg       0.70      0.56      0.56       204
weighted avg       0.79      0.82      0.78       204
```

APPLY BAGGING

In [98]:
```
from sklearn.ensemble import BaggingClassifier
bgcl = BaggingClassifier(n_estimators=60, max_samples=.5 , oob_score=True, random_state=42)

bgcl = bgcl.fit(X_train_resample, y_train_resample)
print(bgcl.oob_score_)
print(bgcl.score(X_train_resample, y_train_resample))
print(bgcl.score(X_test,Y_test))
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\ensemble\_bagging.py:645: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

0.9094488188976378
0.9881889763779528
0.8382352941176471
```

APPLY ADABOOST CLASSIFIER

In [99]:
```
from sklearn.ensemble import AdaBoostClassifier
abc = AdaBoostClassifier(random_state=42)

abc = abc.fit(X_train_resample, y_train_resample)
print(abc.score(X_train_resample, y_train_resample))
print(abc.score(X_test,Y_test))
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

0.9261811023622047
0.8725490196078431
```

In [100]:
```
y_prediction = abc.predict(X_test)
print (metrics.accuracy_score(Y_test, y_prediction))
```

```
0.8725490196078431
```

In [101]:
```
print (metrics.confusion_matrix(Y_test, y_prediction))
print (metrics.classification_report(Y_test, y_prediction))
```

```
[[152  15]
 [ 11  26]]
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       167
           1       0.63      0.70      0.67        37

    accuracy                           0.87       204
   macro avg       0.78      0.81      0.79       204
weighted avg       0.88      0.87      0.88       204
```

AdaBoost Classifier has predicted 178 records correctly out of 204 records achieving an overall accuracy of 87 percent. But for Attrition as Yes (1), it could predict ony 26 correctly out of 37 records achieving an recall score of 70 percent which is less than the LogisticRegression Model.

Taking a new test data with few records, cleaning the data as we did for our main data as above and to check the no of people which will be leaving the company or not.

In [102]:
```
df_newtestData = pd.read_excel('Test_2.xlsx')
df_newtestData.head()
```

Out[102]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	Environment!
0	34	Non-Travel	999	Research & Development	26	1	Technical Degree	1	1374	
1	40	Travel_Rarely	1202	Research & Development	2	1	Medical	1	1375	
2	34	Travel_Rarely	285	Research & Development	29	3	Medical	1	1377	
3	38	Travel_Frequently	693	Research & Development	7	3	Life Sciences	1	1382	
4	34	Travel_Rarely	404	Research & Development	2	4	Technical Degree	1	1383	

5 rows × 34 columns

In [103]:
```
print(df_newtestData['EducationField'].value_counts())
print("Total counts is:",(df_newtestData['EducationField'].value_counts().sum()))
```

```
Life Sciences       44
Medical             31
Technical Degree    12
Marketing           12
Other                4
Human Resources      1
Name: EducationField, dtype: int64
Total counts is: 104
```

In [104]:
```
print(df_newtestData['BusinessTravel'].value_counts())
print("Total counts is:",(df_newtestData['BusinessTravel'].value_counts().sum()))
```

```
Travel_Rarely        68
Travel_Frequently    24
Non-Travel           12
Name: BusinessTravel, dtype: int64
Total counts is: 104
```

In [105]:
```
#To find the count of each unique value in MaritalStatus column
print(df_newtestData['MaritalStatus'].value_counts())
print("Total counts is:",(df_newtestData['MaritalStatus'].value_counts().sum()))
```

```
Married     48
Single      31
Divorced    25
Name: MaritalStatus, dtype: int64
Total counts is: 104
```

In [106]:
```
df_newtestData['YearsinCompany'] = 2020 - df_newtestData['JoiningYearinCompany']
df_newtestData.head()
```

Out[106]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	Environment!
0	34	Non-Travel	999	Research & Development	26	1	Technical Degree	1	1374	
1	40	Travel_Rarely	1202	Research & Development	2	1	Medical	1	1375	
2	34	Travel_Rarely	285	Research & Development	29	3	Medical	1	1377	
3	38	Travel_Frequently	693	Research & Development	7	3	Life Sciences	1	1382	
4	34	Travel_Rarely	404	Research & Development	2	4	Technical Degree	1	1383	

5 rows × 35 columns

In [107]:
```
#Dropping or removing the unnecessary columns
df_newtestData = df_newtestData.drop('EmployeeNumber', axis = 1)
df_newtestData = df_newtestData.drop('StandardHours', axis = 1)
df_newtestData = df_newtestData.drop('EmployeeCount', axis = 1)

df_newtestData = df_newtestData.drop('Over18', axis = 1)
df_newtestData = df_newtestData.drop('JoiningYearinCompany', axis = 1)
```

In [108]:
```
df_newtestData.shape
```

Out[108]:
```
(104, 30)
```

In [109]:
```
#Importing the LabelEncoder to convert the categorical data to numeric since machine understands only numeric data
from sklearn.preprocessing import LabelEncoder
#Instantiate the LabelEncoder class
le1 = LabelEncoder()
```

In [110]:
```
# Using for loop to apply the fit_transform method to remaining columns having categorical data
for column in df_newtestData.columns:
        if df_newtestData[column].dtype == np.object:
            df_newtestData[column] = LabelEncoder().fit_transform(df_newtestData[column])
        else:
            continue
```

In [111]:
```
df_newtestData.isnull().sum()
```

Out[111]:
```
Age                         0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
YearsinCompany              0
dtype: int64
```

In [112]:
```
df_newtestData_scaled = sc.fit_transform(df_newtestData)
df_newtestData_scaled = pd.DataFrame(df_newtestData_scaled, columns=df_newtestData.columns)
df_newtestData_scaled.head()
```

Out[112]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRa
0	-0.205494	-2.222222	0.493748	-0.584317	1.815376	-1.591621	1.987691	-1.667968	-1.317893	1.45424
1	0.571646	0.666667	0.975828	-0.584317	-0.889805	-1.591621	0.526773	-0.745262	-1.317893	1.29827
2	-0.205494	0.666667	-1.201842	-0.584317	2.153524	0.169321	0.526773	-0.745262	0.758787	1.14230
3	0.312600	-0.777778	-0.232934	-0.584317	-0.326226	0.169321	-0.934144	1.100149	0.758787	-0.36543
4	-0.205494	0.666667	-0.919244	-0.584317	-0.889805	1.049792	1.987691	0.177443	-1.317893	1.76619

5 rows × 30 columns

APPLYING THE LOGISTIC REGRESSION MODEL TO ACHIEVE THE ATTRITION PREDICTION OF THIS NEW DATASET SINCE THIS MODEL HAS DONE A GOOD JOB IN PREDICTING THE PEOPLE LEAVING THE COMPANY.

In [119]:
```
y_predictionData = logreg.predict(df_newtestData_scaled)
y_predictionData
```

Out[119]:
```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0])
```

In [120]:
```
y_predictionData = pd.DataFrame(y_predictionData,columns=['AttritionPrediction'])
y_predictionData
```

Out[120]:

	AttritionPrediction
0	0
1	0
2	0
3	0
4	0
...	...
99	0
100	0
101	0
102	0
103	0

104 rows × 1 columns

In [121]:
```
y_predictionData['AttritionPrediction'].value_counts()
```

Out[121]:
```
0    73
1    31
Name: AttritionPrediction, dtype: int64
```

SO AS PER OUR PREDICTION (LOGISTICREGRESSIONMODEL), 73 PEOPLE WILL RETAIN IN THE COMPANY AND 31 PEOPLE WILL BE LEAVING OUT OF A TOTAL OF 104 PEOPLE

In [ ]: