CAPSTONE PROJECT - Swarn Priya

Prepare a model for the HR department to predict the Attrition and give the insights from the data about the important factors associated with the attrition so that HR can take the corrective or previntive measures to stop or control the attrition.

In [2]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from scipy.stats import norm
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,recall_score
from sklearn.metrics import roc_curve, auc
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [3]:

```
data = pd.read_csv("C:\\Users\\Swarn\\Downloads\\HR_Employee_Attrition_Data.csv")
data.head(10)
```

Out[3]:

	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	3	..
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	4	..
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	5	..
5	32	No	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	6	..
6	59	No	Travel_Rarely	1324	Research & Development	3	3	Medical	1	7	..
7	30	No	Travel_Rarely	1358	Research & Development	24	1	Life Sciences	1	8	..
8	38	No	Travel_Frequently	216	Research & Development	23	3	Life Sciences	1	9	..
9	36	No	Travel_Rarely	1299	Research & Development	27	3	Medical	1	10	..

10 rows × 35 columns

Exploratory Data Analysis (EDA)

In [4]:

```
data.shape
```

Out[4]:

```
(2940, 35)
```

In [5]:

```
data.columns.values
```

Out[5]:

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

In [6]:

```
data.dtypes
```

Out[6]:

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

In [7]:

```
data.describe()
```

Out[7]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	Jol
count	2940.000000	2940.000000	2940.000000	2940.000000	2940.0	2940.000000	2940.000000	2940.000000	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1470.500000	2.721769	65.891156	
std	9.133819	403.440447	8.105485	1.023991	0.0	848.849221	1.092896	20.325969	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	735.750000	2.000000	48.000000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1470.500000	3.000000	66.000000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	2205.250000	4.000000	84.000000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2940.000000	4.000000	100.000000	

8 rows × 26 columns

Only 26 columns are described instead of 35, because only numerical value will be described here.

If we see this describe values in detail, we wil see mostly the mean is statistically speaking, same as median, which means, not much outliers are there. Only in few columns, like YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager, we can see a little difference between the mdeian and the mode. There may be few utliers in these columns.

In [8]:

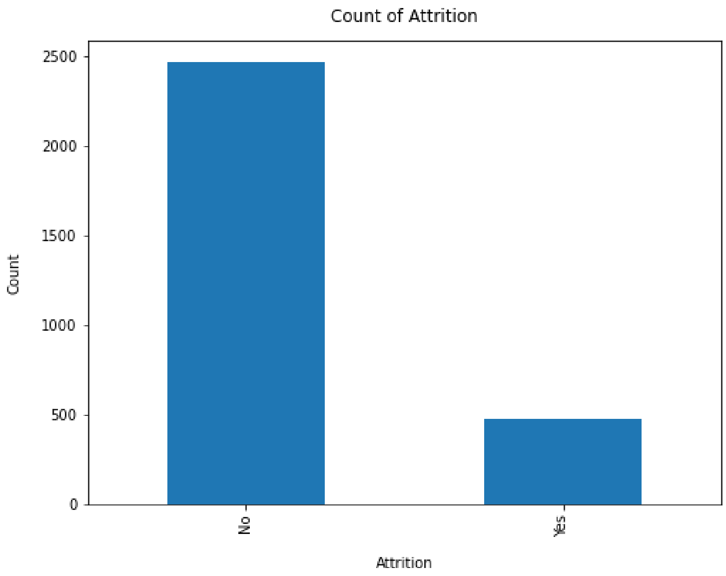
```
#To see the short Summary of the dataframe, we use this info function and we see no null values. (Verbose = True as there are
# many columns and wanted to check all the columns in one go)

data.info(verbose=True)
```

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

Data Exploration

In [9]:

```
data['Attrition'].value_counts().plot(kind='bar',figsize=(8,6))
plt.ylabel("Count",labelpad=14)
plt.xlabel("Attrition", labelpad=14)
plt.title("Count of Attrition",y=1.02)
plt.show()
```



In [10]:

```
print('Count')
print(data['Attrition'].value_counts())
print('\nPercentage')
print((data['Attrition'].value_counts()/len(data['Attrition']))*100)
```

```
Count
No     2466
Yes     474
Name: Attrition, dtype: int64

Percentage
No     83.877551
Yes    16.122449
Name: Attrition, dtype: float64
```

Insight 1::

This is just the prilimary graph and related data which shows the number of people who have left the organization. As seen from the graph and data, attrition level is 16% and this is very high value. So, we further analyse the data.

In [11]:

```
#Copying the data
data_copy = data.copy()
data_copy.shape
```

Out[11]:

```
(2940, 35)
```

In [12]:

```
data_copy.drop(['EmployeeNumber','Over18','StandardHours','EmployeeCount'],axis=1,inplace=True)
data_copy.shape
```

Out[12]:

```
(2940, 31)
```

The above columns are not going to contribute in our analysis in any which way.

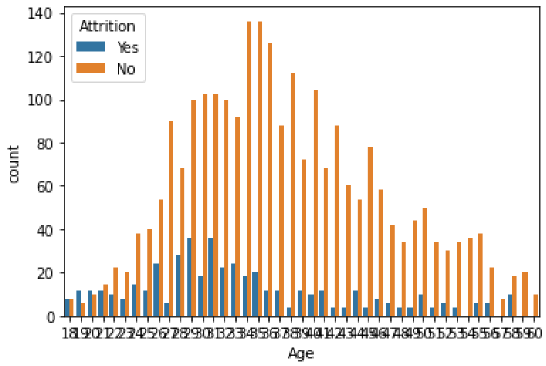
1. EmployeeNumber is the employee ID. Which is not required.
2. Over18 is an obvious column that anyone working in the organization is going to be over 18 plus all the values are Yes. No change at all.
3. StandardHours is also same for all the employees which is 80. Logically also, this is a constant thing, so attrition will not be dependent on this variable as well.

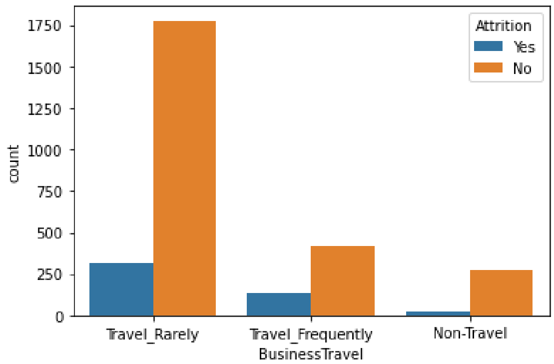
Categorical Data Analysis.

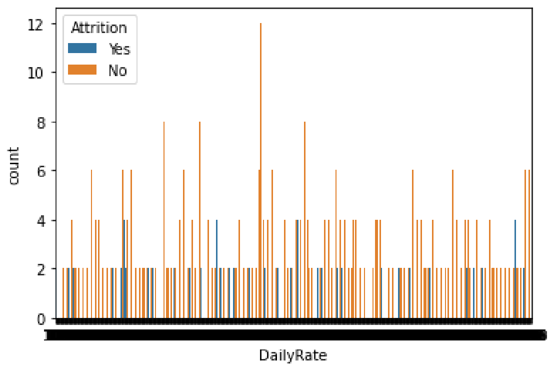
First of all, we do categorical data analysis.

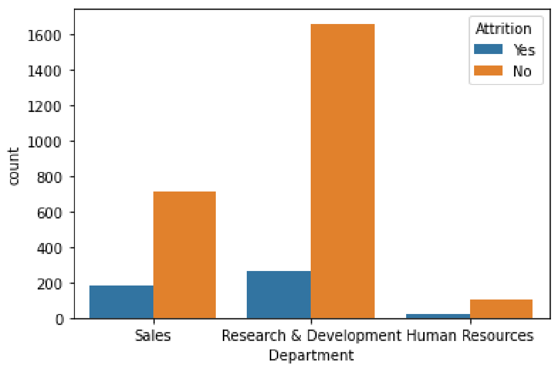
In [13]:

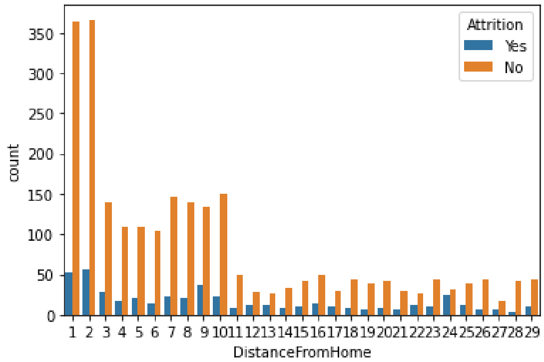
```
for i, predictor in enumerate(data_copy.drop(columns=['Attrition'])):
    plt.figure(i)
    sns.countplot(data=data_copy,x=predictor, hue='Attrition')

plt.show()

#Plotting all the column against the attrition rate, to see how the graph moves in each case.
```

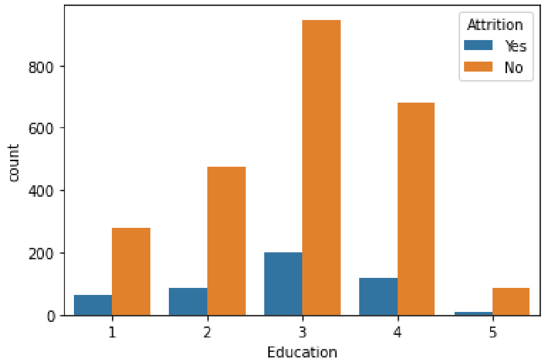


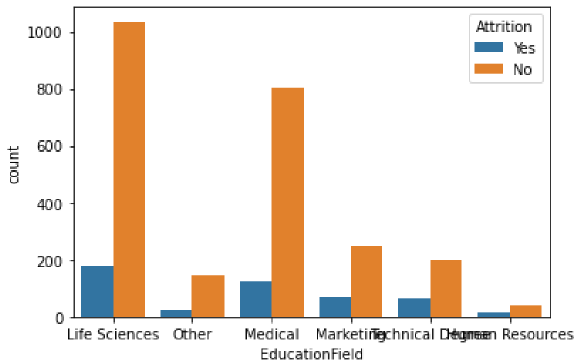


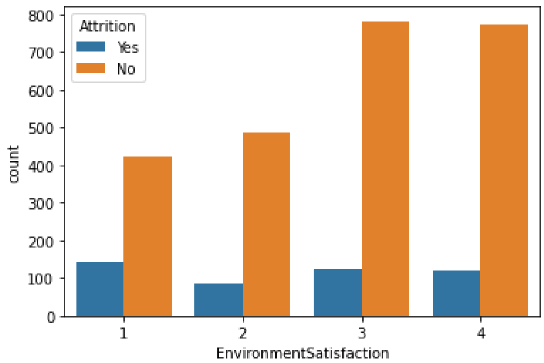


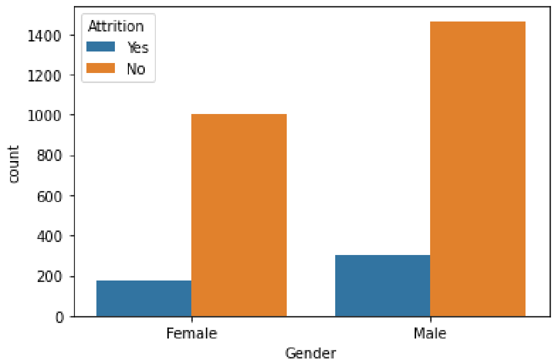


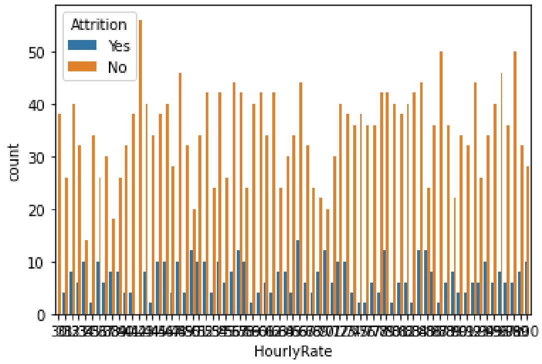

Attrition
Yes
No
count
DistanceFromHome

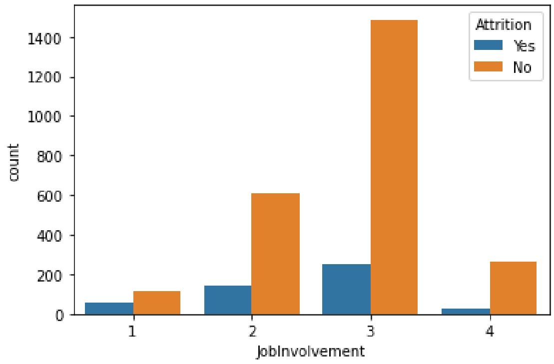

Attrition
Yes
No
count
1
2
3
4
5
Education

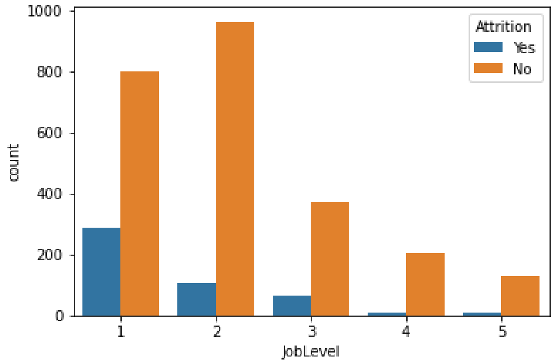

Attrition
Yes
No
count
Life Sciences
Other
Medical
EducationField

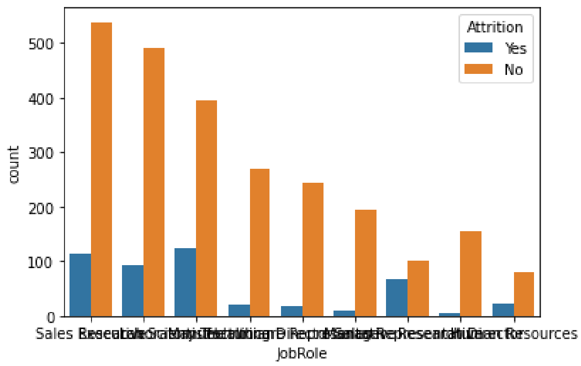

Attrition
Yes
No
count
1
2
3
4
EnvironmentSatisfaction

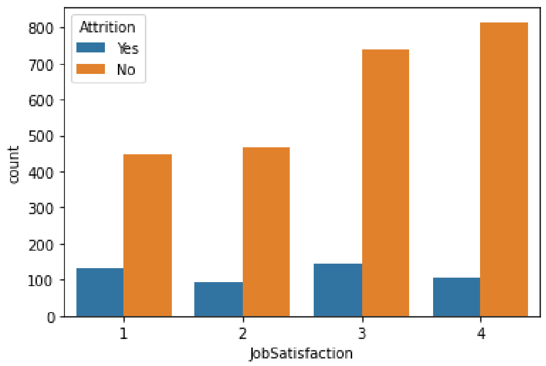

Attrition
Yes
No
count
Female
Male
Gender

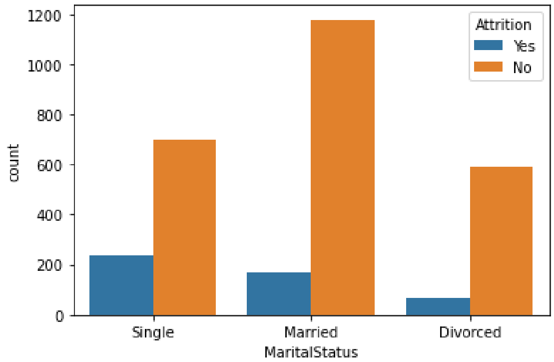

Attrition
Yes
No
count
0
10
20
30
40
50
HourlyRate

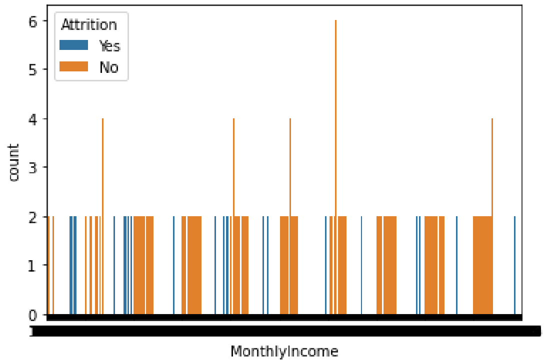

Attrition
Yes
No
count
0
200
400
600
800
1000
1200
1400
1
2
3
4
JobInvolvement

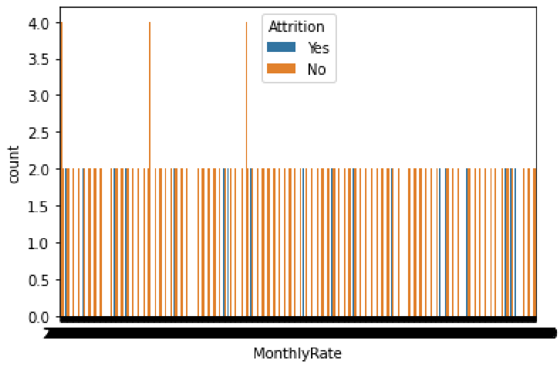

Attrition
Yes
No
count
0
200
400
600
800
1000
1
2
3
4
5
JobLevel

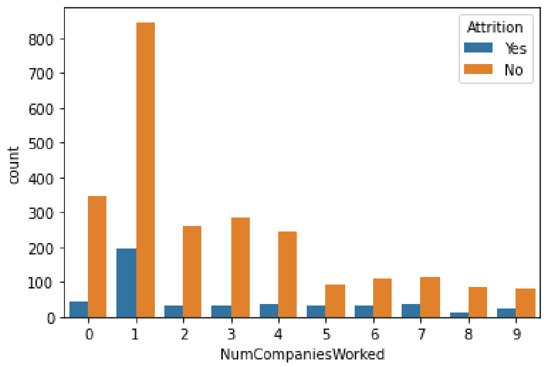

Attrition
Yes
No
count
0
100
200
300
400
500
JobRole

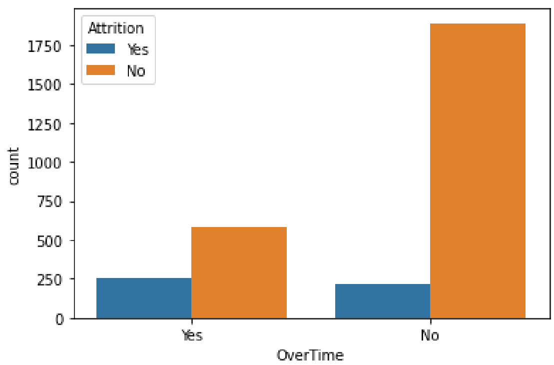

Attrition
Yes
No
count
0
100
200
300
400
500
600
700
800
1
2
3
4
JobSatisfaction

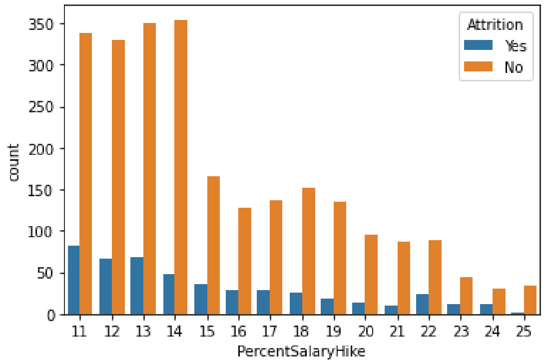

Attrition
Yes
No
count
0
200
400
600
800
1000
1200
Single
Married
Divorced
MaritalStatus

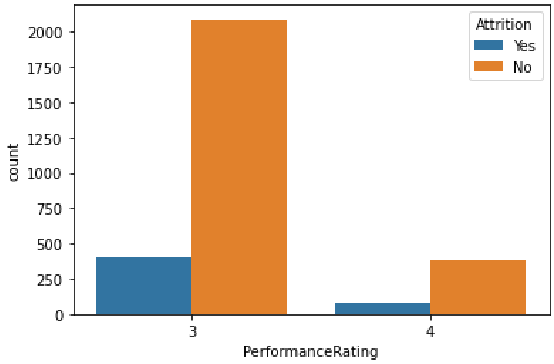

Attrition
Yes
No
count
0
1
2
3
4
5
6
MonthlyIncome

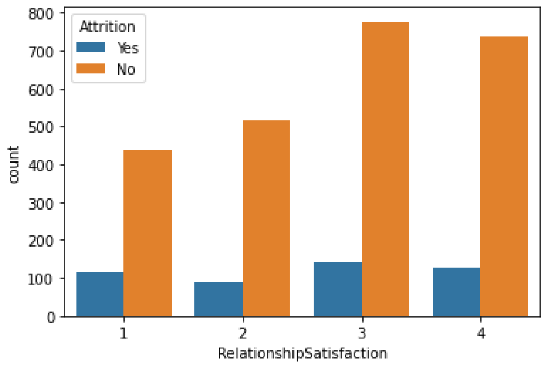

Attrition
Yes
No
count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
MonthlyRate

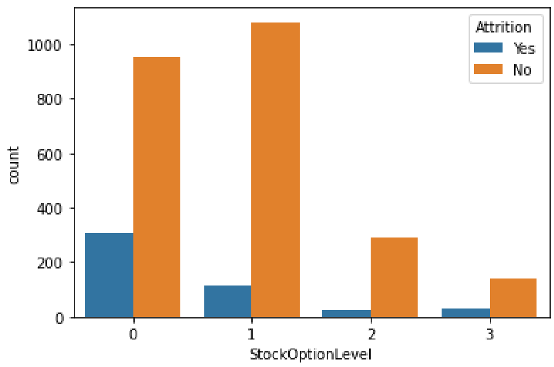

Attrition
Yes
No
count
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5
6
7
8
9
NumCompaniesWorked

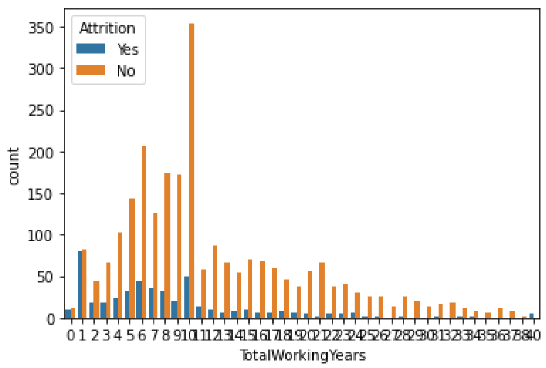

Attrition
Yes
No
count
0
250
500
750
1000
1250
1500
1750
Yes
No
OverTime

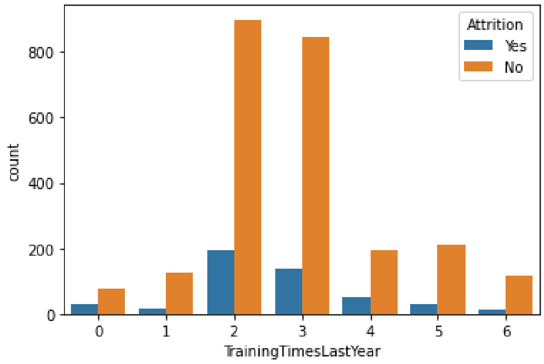

Attrition
Yes
No
count
0
50
100
150
200
250
300
350
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
PercentSalaryHike

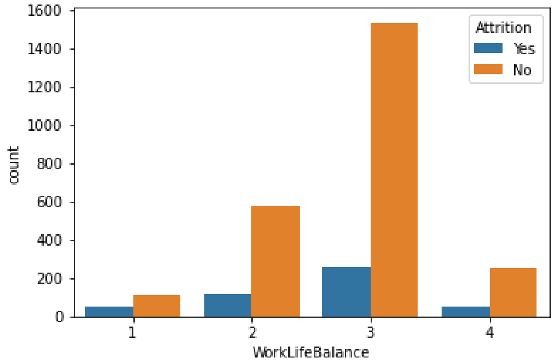

Attrition
Yes
No
count
0
250
500
750
1000
1250
1500
1750
2000
3
4
PerformanceRating

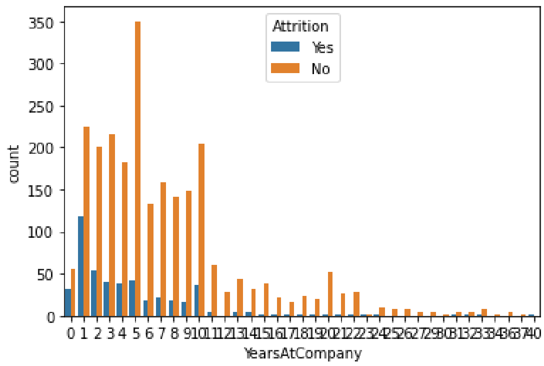

Attrition
Yes
No
count
0
100
200
300
400
500
600
700
800
1
2
3
4
RelationshipSatisfaction

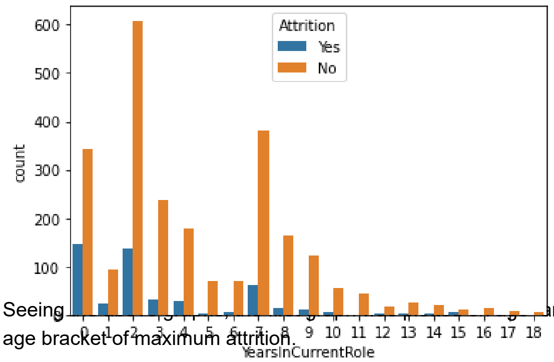

Attrition
Yes
No
count
0
200
400
600
800
1000
0
1
2
3
StockOptionLevel


Attrition
Yes
No
count
0
50
100
150
200
250
300
350
TotalWorkingYears

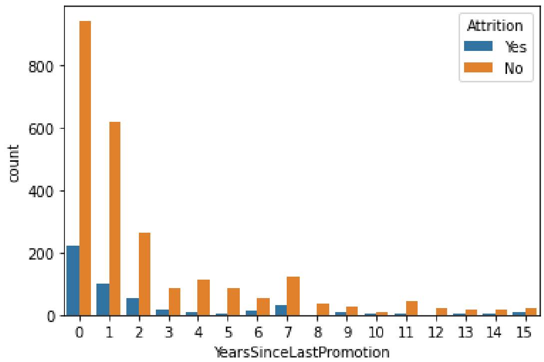






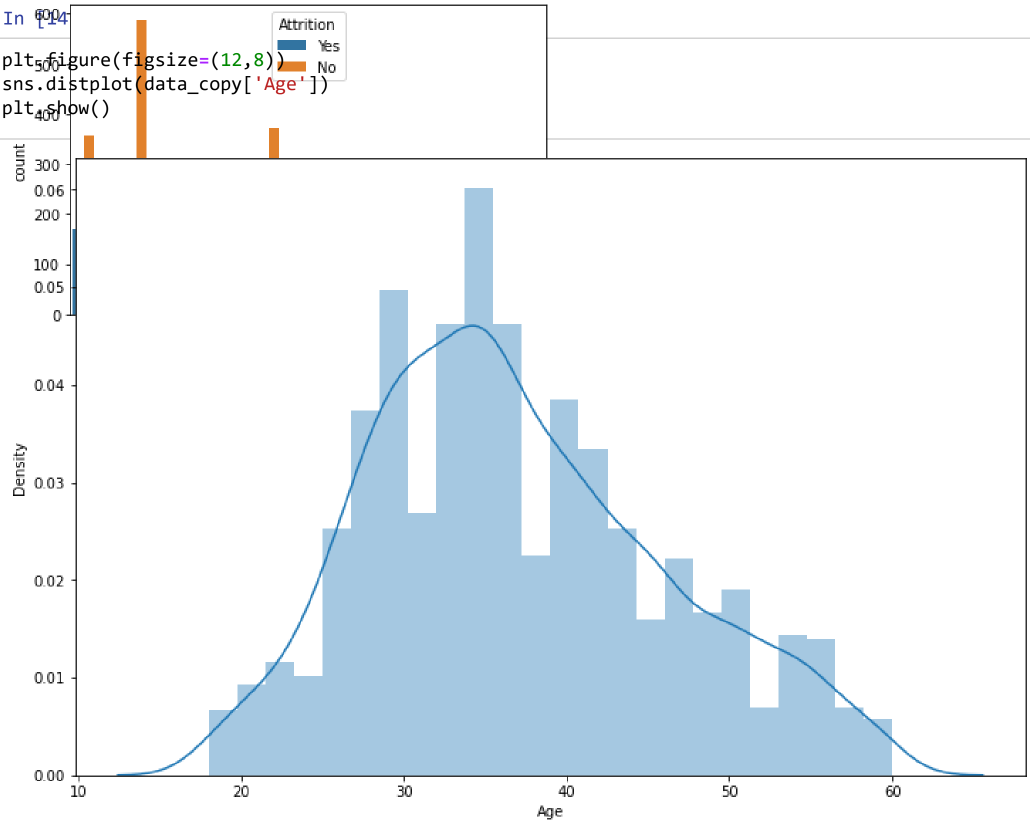


Seeing analysising the age group which is migrating the maximum. Seeing the same in distplot to get the age bracket of maximum attrition.



In [14]:

```
plt.figure(figsize=(12,8))
sns.distplot(data_copy['Age'])
plt.show()
```



In [15]:

```
plt.figure(figsize=(16,6))
sns.swarmplot(y='Age',x='Attrition',data=data_copy,hue = 'Department')
plt.show()
```

2nd Insight ::

Age Wise :: Attrition in the age bracket 25-35 is maximum.

But another observation is that there are maximum number of employees also same in that age bracket(approximately). So, let's look into the data further more.

In [ ]:

In [14]:

```
# Department-wise Atttrition %

depart_percent = data_copy.groupby(['Department'])
print(depart_percent.groups.keys())
print(data_copy['BusinessTravel'].count())
print("% of Attrition in HR-Department is :",depart_percent.get_group("Human Resources")['BusinessTravel'].value_counts().sum()/d
print("% of Attrition in R&D-Department is :",depart_percent.get_group("Research & Development")['BusinessTravel'].value_counts()
print("% of Attrition in Sales-Department is :",depart_percent.get_group("Sales")['BusinessTravel'].value_counts().sum()/data_cop
```

```
dict_keys(['Human Resources', 'Research & Development', 'Sales'])
2940
% of Attrition in HR-Department is : 4.285714285714286
% of Attrition in R&D-Department is : 65.37414965986395
% of Attrition in Sales-Department is : 30.34013605442177
```

In [19]:

```
#Getting the count of Business travel based on Department - HR and Attrition

departmentwisedata = data_copy.groupby(["Department","Attrition"])
print(departmentwisedata.groups.keys())

df1 = departmentwisedata.get_group(('Human Resources','No'))['BusinessTravel'].value_counts()
df1 = pd.DataFrame(df1)
df1=df1.reset_index()
df1['Department']="HR"
df1['Attrition'] = 'No'

df1
```

```
dict_keys([('Human Resources', 'No'), ('Human Resources', 'Yes'), ('Research & Development', 'No'), ('Research & De
velopment', 'Yes'), ('Sales', 'No'), ('Sales', 'Yes')])
```

Out[19]:

	index	BusinessTravel	Department	Attrition
0	Travel_Rarely	76	HR	No
1	Travel_Frequently	14	HR	No
2	Non-Travel	12	HR	No

In [20]:

```
df2 = departmentwisedata.get_group(('Human Resources','Yes'))['BusinessTravel'].value_counts()
df2 = df2.to_frame().reset_index()
df2['Department']="HR"
df2["Attrition"]="Yes"
df3=df1.append(df2,ignore_index=True)

#for getting the department Attrition %
depart_percent = data_copy.groupby(['Department'])

count_yes = departmentwisedata.get_group(('Human Resources','Yes'))['BusinessTravel'].value_counts().sum()
count_all = depart_percent.get_group(('Human Resources'))['BusinessTravel'].value_counts().sum()
attrition_HR = count_yes/count_all*100
# print(attrition_HR)
df3
```

Out[20]:

	index	BusinessTravel	Department	Attrition
0	Travel_Rarely	76	HR	No
1	Travel_Frequently	14	HR	No
2	Non-Travel	12	HR	No
3	Travel_Rarely	16	HR	Yes
4	Travel_Frequently	8	HR	Yes

In [21]:

```
#Getting the count of Business travel based on Department - R&D

df4 = departmentwisedata.get_group(("Research & Development",'No'))['BusinessTravel'].value_counts()
df4 = df4.to_frame().reset_index()
df4['Department'] = 'R&D'
df4['Attrition']='No'
df5 = df3.append(df4,ignore_index = True)
df5
```

Out[21]:

	index	BusinessTravel	Department	Attrition
0	Travel_Rarely	76	HR	No
1	Travel_Frequently	14	HR	No
2	Non-Travel	12	HR	No
3	Travel_Rarely	16	HR	Yes
4	Travel_Frequently	8	HR	Yes
5	Travel_Rarely	1188	R&D	No
6	Travel_Frequently	290	R&D	No
7	Non-Travel	178	R&D	No

In [22]:

```
#Getting the count of Business travel based on Department - R&D

df6 = departmentwisedata.get_group(("Research & Development",'Yes'))['BusinessTravel'].value_counts()
df6 = df6.to_frame().reset_index()
df6['Department'] = 'R&D'
df6['Attrition']='Yes'
df7 = df5.append(df6,ignore_index = True)

#Attrition % of R&D
count_yes = departmentwisedata.get_group(('Research & Development','Yes'))['BusinessTravel'].value_counts().sum()
count_all = depart_percent.get_group(('Research & Development'))['BusinessTravel'].value_counts().sum()
attrition_RD = count_yes/count_all*100
df7
```

Out[22]:

	index	BusinessTravel	Department	Attrition
0	Travel_Rarely	76	HR	No
1	Travel_Frequently	14	HR	No
2	Non-Travel	12	HR	No
3	Travel_Rarely	16	HR	Yes
4	Travel_Frequently	8	HR	Yes
5	Travel_Rarely	1188	R&D	No
6	Travel_Frequently	290	R&D	No
7	Non-Travel	178	R&D	No
8	Travel_Rarely	176	R&D	Yes
9	Travel_Frequently	74	R&D	Yes
10	Non-Travel	16	R&D	Yes

In [23]:

```
#Getting the count of Business travel based on Department - Sales


df8 = departmentwisedata.get_group(("Sales",'No'))['BusinessTravel'].value_counts()
df8 = df8.to_frame().reset_index()
df8['Department']='Sales'
df8['Attrition'] = 'No'
df9 = df7.append(df8,ignore_index=True)
df9
```

Out[23]:

	index	BusinessTravel	Department	Attrition
0	Travel_Rarely	76	HR	No
1	Travel_Frequently	14	HR	No
2	Non-Travel	12	HR	No
3	Travel_Rarely	16	HR	Yes
4	Travel_Frequently	8	HR	Yes
5	Travel_Rarely	1188	R&D	No
6	Travel_Frequently	290	R&D	No
7	Non-Travel	178	R&D	No
8	Travel_Rarely	176	R&D	Yes
9	Travel_Frequently	74	R&D	Yes
10	Non-Travel	16	R&D	Yes
11	Travel_Rarely	510	Sales	No
12	Travel_Frequently	112	Sales	No
13	Non-Travel	86	Sales	No

In [24]:

```
#Getting the count of Business travel based on Department - Sales


df10 = departmentwisedata.get_group(("Sales",'Yes'))['BusinessTravel'].value_counts()
df10 = df10.to_frame().reset_index()
df10['Department']='Sales'
df10['Attrition'] = 'Yes'
df10['Attrition-%'] = 0
df11 = df9.append(df10,ignore_index=True)

#Calculating the Attrition department-wise

df12 = data_copy.groupby(by = 'Department')
# print(df12.groups.keys())
count_HR = df12.get_group('Human Resources')['BusinessTravel'].value_counts().sum()
# print(count_HR)
count_RD = df12.get_group('Research & Development')['BusinessTravel'].value_counts().sum()
count_Sales = df12.get_group("Sales")['BusinessTravel'].value_counts().sum()

# adding the coloumn of attrition % within department and business travel
for i in range(len(df11)):
    if df11["Department"][i] == "HR":
        if df11['Attrition'][i] =='Yes':
            df11['Attrition-%'][i] = df11['BusinessTravel'][i]/count_HR *100
#              print(i,df11['Attrition-%'][i],df11['BusinessTravel'][i],df11['BusinessTravel'][i]/count_HR.sum())
    elif df11["Department"][i] == "R&D":
        if df11['Attrition'][i] =='Yes':
            df11['Attrition-%'][i] = df11['BusinessTravel'][i]/count_RD *100
    elif df11["Department"][i] == "Sales":
        if df11['Attrition'][i] =='Yes':
            df11['Attrition-%'][i] = df11['BusinessTravel'][i]/count_Sales *100


#Attrition % of Sales Department

count_yes = departmentwisedata.get_group(('Sales','Yes'))['BusinessTravel'].value_counts().sum()
count_all = depart_percent.get_group(('Sales'))['BusinessTravel'].value_counts().sum()
attrition_Sales = count_yes/count_all*100


#Attrition % of all the department
print("Percentage of attrition in HR is - ",attrition_HR)
print("Percentage of attrition in R&D is - ",attrition_RD)
print("Percentage of attrition in Sales is - ",attrition_Sales)


df11
```

```
Percentage of attrition in HR is -  19.047619047619047
Percentage of attrition in R&D is -  13.839750260145681
Percentage of attrition in Sales is -  20.62780269058296
```

Out[24]:

	index	BusinessTravel	Department	Attrition	Attrition-%
0	Travel_Rarely	76	HR	No	NaN
1	Travel_Frequently	14	HR	No	NaN
2	Non-Travel	12	HR	No	NaN
3	Travel_Rarely	16	HR	Yes	12.698413
4	Travel_Frequently	8	HR	Yes	6.349206
5	Travel_Rarely	1188	R&D	No	NaN
6	Travel_Frequently	290	R&D	No	NaN
7	Non-Travel	178	R&D	No	NaN

Insight 3

Highest attrition seen is in Sales Department = 20.63% And within sales department, employees who are 'Travel_Rarely' are leaving the organization the most which is about 13.45%.

HR department is very close Sales, when attrition percent is seen. It is = 19.04 Employees who 'Travel_Rarely' are the ones who are leaving the organiztion the most.

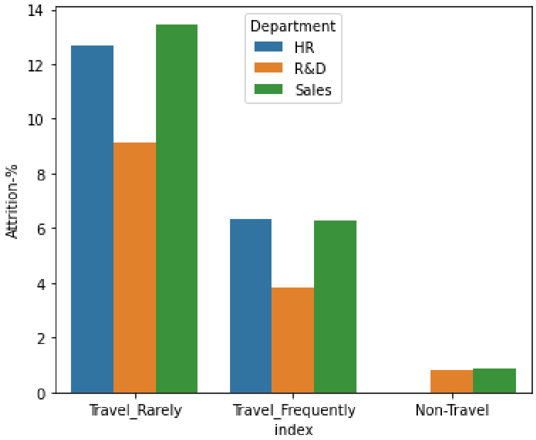
Least attrition is in R&D department. But interestingly, even in R&D:: Employees who 'Travel_Rarely' are the ones who are leaving the organiztion the most.

While, employee with 'Non_Travel' are leaving the organization in very few numbers.

In HR, there are no employees who has ' Non_Travel' and has left and in other departments also, attrition is very negligible when 'Non_Travels' are seen.

In [25]:

```
fig = plt.figure(figsize = (6,5))
sns.barplot(x='index',y='Attrition-%', data=df11, hue='Department')

plt.show()
```



As the 3rd insight suggest, the graph also says the same thing. Attrition is maximum in the group of employees who are in the business_Travel group of 'Travel_Rarely' and lease in group 'Non_Travel'.

Maximum attrition is in Sales followed by HR and then lastly in R&D.

In [28]:

```
#Education Level and Attrition level.

edulevelwisedata = data_copy.groupby(["EducationField","Attrition"])
print(edulevelwisedata.groups.keys())
```

```
dict_keys([('Human Resources', 'No'), ('Human Resources', 'Yes'), ('Life Sciences', 'No'), ('Life Sciences', 'Ye
s'), ('Marketing', 'No'), ('Marketing', 'Yes'), ('Medical', 'No'), ('Medical', 'Yes'), ('Other', 'No'), ('Other',
'Yes'), ('Technical Degree', 'No'), ('Technical Degree', 'Yes')])
```

In [29]:

```
# Making the dataframe for the Education Field = Human Resources.
eld = edulevelwisedata.get_group(('Human Resources', 'Yes'))['EducationField'].value_counts()
eld = pd.DataFrame(eld)
eld = eld.reset_index()
eld['Education Stream'] = 'HR'
eld['Attrition'] = 'Yes'
# print(eld)

eld1 = edulevelwisedata.get_group(('Human Resources', 'No'))['EducationField'].value_counts()
eld1 = eld1.to_frame().reset_index()
eld1['Education Stream'] = 'HR'
eld1['Attrition'] = 'No'
eld2 = eld1.append(eld,ignore_index=True)
eld2

# Making the dataframe for the Education Field = Life Sciences.
eld3 = edulevelwisedata.get_group(('Life Sciences', 'No'))['EducationField'].value_counts()
eld3 = eld3.to_frame().reset_index()
eld3['Education Stream'] = 'Life Scinces'
eld3['Attrition'] = 'No'
eld4 = eld2.append(eld3,ignore_index=True)
eld4

eld5 = edulevelwisedata.get_group(('Life Sciences', 'Yes'))['EducationField'].value_counts()
eld5 = eld5.to_frame().reset_index()
eld5['Education Stream'] = 'Life Scinces'
eld5['Attrition'] = 'Yes'
eld6 = eld4.append(eld5,ignore_index=True)
eld6

# Making the dataframe for the Education Field = Marketing.
eld7 = edulevelwisedata.get_group(('Marketing', 'No'))['EducationField'].value_counts()
eld7 = eld7.to_frame().reset_index()
eld7['Education Stream'] = 'Marketing'
eld7['Attrition'] = 'No'
eld8 = eld6.append(eld7,ignore_index=True)
eld8

eld9 = edulevelwisedata.get_group(('Marketing', 'Yes'))['EducationField'].value_counts()
eld9 = eld9.to_frame().reset_index()
eld9['Education Stream'] = 'Marketing'
eld9['Attrition'] = 'Yes'
eld10 = eld8.append(eld9,ignore_index=True)
eld10


# Making the dataframe for the Education Field = 'Medical'.
eld11 = edulevelwisedata.get_group(('Medical', 'No'))['EducationField'].value_counts()
eld11 = eld11.to_frame().reset_index()
eld11['Education Stream'] = 'Medical'
eld11['Attrition'] = 'No'
eld12 = eld10.append(eld11,ignore_index=True)
eld12

eld13 = edulevelwisedata.get_group(('Medical', 'Yes'))['EducationField'].value_counts()
eld13 = eld13.to_frame().reset_index()
eld13['Education Stream'] = 'Medical'
eld13['Attrition'] = 'Yes'
eld14 = eld12.append(eld13,ignore_index=True)
eld14


# Making the dataframe for the Education Field = 'Other'.
eld15 = edulevelwisedata.get_group(('Other','No'))['EducationField'].value_counts()
eld15 = eld15.to_frame().reset_index()
eld15['Education Stream'] = 'Other'
eld15['Attrition'] = 'No'
eld16 = eld14.append(eld15,ignore_index=True)
eld16

eld17 = edulevelwisedata.get_group(('Other','Yes'))['EducationField'].value_counts()
eld17 = eld17.to_frame().reset_index()
eld17['Education Stream'] = 'Other'
eld17['Attrition'] = 'Yes'
eld18 = eld16.append(eld17,ignore_index = True)
eld18


# Making the dataframe for the Education Field = 'Technical Degree'
eld19 = edulevelwisedata.get_group(('Technical Degree', 'No'))['EducationField'].value_counts()
eld19 = eld19.to_frame().reset_index()
eld19['Education Stream'] = 'Technical Degree'
eld19['Attrition'] = 'No'
eld20 = eld18.append(eld19,ignore_index=True)
```

```
eld20

eld21 = edulevelwisedata.get_group(('Technical Degree','Yes'))['EducationField'].value_counts()
eld21 = eld21.to_frame().reset_index()
eld21['Education Stream'] = 'Technical Degree'
eld21['Attrition'] = 'Yes'
eld22 = eld20.append(eld21,ignore_index = True)
eld22
```

Out[29]:

	index	EducationField	Education Stream	Attrition
0	Human Resources	40	HR	No
1	Human Resources	14	HR	Yes
2	Life Sciences	1034	Life Scinces	No
3	Life Sciences	178	Life Scinces	Yes
4	Marketing	248	Marketing	No
5	Marketing	70	Marketing	Yes
6	Medical	802	Medical	No
7	Medical	126	Medical	Yes
8	Other	142	Other	No
9	Other	22	Other	Yes
10	Technical Degree	200	Technical Degree	No
11	Technical Degree	64	Technical Degree	Yes

In [30]:

```
#Calculating the % of attrition when the educationfeild = Human resources
# eld_HR = eld1['EducationField'].sum()/(eld['EducationField'].sum()+eld1['EducationField'].sum())*100
#Calculating the Attrition department-wise

eld23 = data_copy.groupby(by = 'EducationField')
print(eld23.groups.keys())
count_HR = eld23.get_group('Human Resources')['EducationField'].value_counts().sum()
# print(count_HR)
count_LS = eld23.get_group('Life Sciences')['EducationField'].value_counts().sum()
count_Marketing = eld23.get_group("Marketing")['EducationField'].value_counts().sum()
count_medical = eld23.get_group('Medical')['EducationField'].value_counts().sum()
count_other = eld23.get_group('Other')['EducationField'].value_counts().sum()
count_TD = eld23.get_group("Technical Degree")['EducationField'].value_counts().sum()


#Calculating the attrition %
eld22['Attrition-%'] = 0

for i in range(len(eld22)):
    if eld22["Education Stream"][i] == "HR":
        if eld22['Attrition'][i] =='Yes':
            eld22['Attrition-%'][i] = eld22['EducationField'][i]/count_HR *100
#             print(i,df11['Attrition-%'][i],df11['BusinessTravel'][i],df11['BusinessTravel'][i]/count_HR.sum())
    elif eld22["Education Stream"][i] == "Life Scinces":
        if eld22['Attrition'][i] =='Yes':
            eld22['Attrition-%'][i] = eld22['EducationField'][i]/count_LS *100
    elif eld22["Education Stream"][i] == "Marketing":
        if eld22['Attrition'][i] =='Yes':
            eld22['Attrition-%'][i] = eld22['EducationField'][i]/count_Marketing *100
    elif eld22["Education Stream"][i] == "Medical":
        if eld22['Attrition'][i] =='Yes':
            eld22['Attrition-%'][i] = eld22['EducationField'][i]/count_medical *100
    elif eld22["Education Stream"][i] == "Other":
        if eld22['Attrition'][i] =='Yes':
            eld22['Attrition-%'][i] = eld22['EducationField'][i]/count_other *100
    elif eld22["Education Stream"][i] == "Technical Degree":
        if eld22['Attrition'][i] =='Yes':
            eld22['Attrition-%'][i] = eld22['EducationField'][i]/count_TD *100

eld22
```

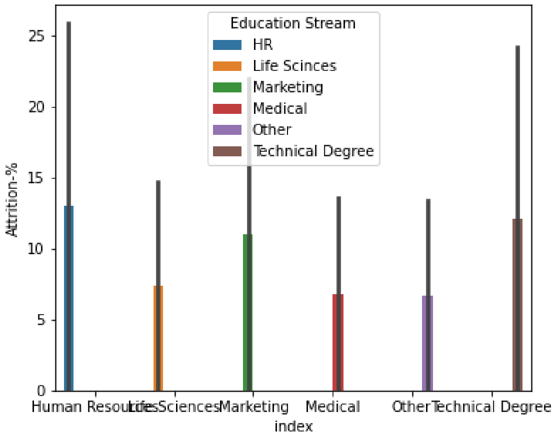
```
dict_keys(['Human Resources', 'Life Sciences', 'Marketing', 'Medical', 'Other', 'Technical Degree'])
```

Out[30]:

	index	EducationField	Education Stream	Attrition	Attrition-%
0	Human Resources	40	HR	No	0.000000
1	Human Resources	14	HR	Yes	25.925926
2	Life Sciences	1034	Life Scinces	No	0.000000
3	Life Sciences	178	Life Scinces	Yes	14.686469
4	Marketing	248	Marketing	No	0.000000
5	Marketing	70	Marketing	Yes	22.012579
6	Medical	802	Medical	No	0.000000
7	Medical	126	Medical	Yes	13.577586
8	Other	142	Other	No	0.000000
9	Other	22	Other	Yes	13.414634
10	Technical Degree	200	Technical Degree	No	0.000000
11	Technical Degree	64	Technical Degree	Yes	24.242424

In [31]:

```
fig = plt.figure(figsize = (6,5))
sns.barplot(x='index',y='Attrition-%', data=eld22, hue='Education Stream')

plt.show()
```



Insight 4

Attrtion in employees whose education stream is = Human Resources, Technical Degree and Marketing is very high. Within that department, (Statistically speaking) approx 25% employees leave the organization.

Remaining education stream employees - attrition is within 13-14% range.

In [32]:

```
#Environmental Satisfaction and Attrition level.

envwisedata = data_copy.groupby(["EnvironmentSatisfaction","Attrition"])
print(envwisedata.groups.keys())
```

```
dict_keys([(1, 'No'), (1, 'Yes'), (2, 'No'), (2, 'Yes'), (3, 'No'), (3, 'Yes'), (4, 'No'), (4, 'Yes')])
```

In [33]:

```
# Making the dataframe for the Environmental Satisfaction Level = 1.

esd = envwisedata.get_group((1, 'Yes'))['EnvironmentSatisfaction'].value_counts()
esd = pd.DataFrame(esd)
esd = esd.reset_index()
esd['Environment Satisfaction Level'] = 1
esd['Attrition'] = 'Yes'
# print(esd)

esd1 = envwisedata.get_group((1, 'No'))['EnvironmentSatisfaction'].value_counts()
esd1 = esd1.to_frame().reset_index()
esd1['Environment Satisfaction Level'] = '1'
esd1['Attrition'] = 'No'
esd2 = esd1.append(esd,ignore_index=True)
esd2

# Making the dataframe for the Environmental Satisfaction Level = 2
esd3 = envwisedata.get_group((2, 'No'))['EnvironmentSatisfaction'].value_counts()
esd3 = esd3.to_frame().reset_index()
esd3['Environment Satisfaction Level'] = 2
esd3['Attrition'] = 'No'
esd4 = esd2.append(esd3,ignore_index=True)
esd4

esd5 = envwisedata.get_group((2, 'Yes'))['EnvironmentSatisfaction'].value_counts()
esd5 = esd5.to_frame().reset_index()
esd5['Environment Satisfaction Level'] = '2'
esd5['Attrition'] = 'Yes'
esd6 = esd4.append(esd5,ignore_index=True)
esd6

# Making the dataframe for the Environmental Satisfaction Level = 3
esd7 = envwisedata.get_group((3, 'No'))['EnvironmentSatisfaction'].value_counts()
esd7 = esd7.to_frame().reset_index()
esd7['Environment Satisfaction Level'] = '3'
esd7['Attrition'] = 'No'
esd8 = esd6.append(esd7,ignore_index=True)
esd8

esd9 = envwisedata.get_group((3, 'Yes'))['EnvironmentSatisfaction'].value_counts()
esd9 = esd9.to_frame().reset_index()
esd9['Environment Satisfaction Level'] = '3'
esd9['Attrition'] = 'Yes'
esd10 = esd8.append(esd9,ignore_index=True)
esd10


# Making the dataframe for the Environmental Satisfaction Level = 4
esd11 = envwisedata.get_group((4, 'No'))['EnvironmentSatisfaction'].value_counts()
esd11 = esd11.to_frame().reset_index()
esd11['Environment Satisfaction Level'] = '4'
esd11['Attrition'] = 'No'
esd12 = esd10.append(esd11,ignore_index=True)
esd12

esd13 = envwisedata.get_group((4, 'Yes'))['EnvironmentSatisfaction'].value_counts()
esd13 = esd13.to_frame().reset_index()
esd13['Environment Satisfaction Level'] = '4'
esd13['Attrition'] = 'Yes'
esd14 = esd12.append(esd13,ignore_index=True)
esd14
```

Out[33]:

	index	EnvironmentSatisfaction	Environment Satisfaction Level	Attrition
0	1	424	1	No
1	1	144	1	Yes
2	2	488	2	No
3	2	86	2	Yes
4	3	782	3	No
5	3	124	3	Yes
6	4	772	4	No
7	4	120	4	Yes

In [42]:

```
#Calculating the % of attrition for differen Enviornment Satisfaction Level

esd15 = data_copy.groupby(by = 'EnvironmentSatisfaction')
print(esd15.groups.keys())
count_1 = esd15.get_group(1)['EnvironmentSatisfaction'].value_counts().sum()
count_2 = esd15.get_group(2)['EnvironmentSatisfaction'].value_counts().sum()
count_3 = esd15.get_group(3)['EnvironmentSatisfaction'].value_counts().sum()
count_4 = esd15.get_group(4)['EnvironmentSatisfaction'].value_counts().sum()
print(count_1,count_2,count_3,count_4)

#Calculating the attrition %
esd14['Attrition-%'] = 0

for i in range(len(esd14)):
    if esd14["Environment Satisfaction Level"][i] == 1:
        if esd14['Attrition'][i] =='Yes':
            esd14['Attrition-%'][i] = esd14['EnvironmentSatisfaction'][i]/count_1 *100
    elif esd14["Environment Satisfaction Level"][i] == '2':
        if esd14['Attrition'][i] =='Yes':
            esd14['Attrition-%'][i] = esd14['EnvironmentSatisfaction'][i]/count_2 *100
    elif esd14["Environment Satisfaction Level"][i] == '3':
        if esd14['Attrition'][i] =='Yes':
            esd14['Attrition-%'][i] = esd14['EnvironmentSatisfaction'][i]/count_3 *100
    elif esd14["Environment Satisfaction Level"][i] == '4':
        if esd14['Attrition'][i] =='Yes':
            esd14['Attrition-%'][i] = esd14['EnvironmentSatisfaction'][i]/count_4 *100


esd14
```

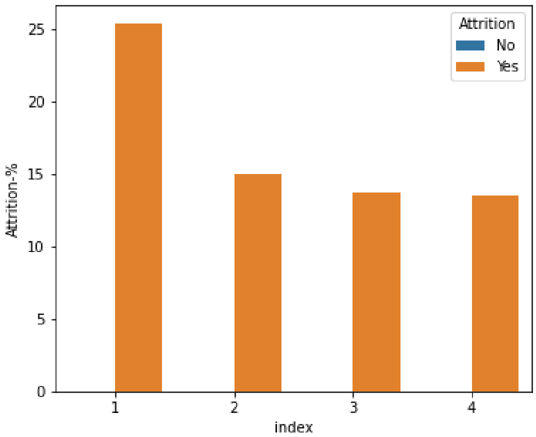
```
dict_keys([1, 2, 3, 4])
568 574 906 892
```

Out[42]:

	index	EnvironmentSatisfaction	Environment Satisfaction Level	Attrition	Attrition-%
0	1	424	1	No	0.000000
1	1	144	1	Yes	25.352113
2	2	488	2	No	0.000000
3	2	86	2	Yes	14.982578
4	3	782	3	No	0.000000
5	3	124	3	Yes	13.686534
6	4	772	4	No	0.000000
7	4	120	4	Yes	13.452915

In [50]:

```
fig = plt.figure(figsize = (6,5))
sns.barplot(x='index',y='Attrition-%', data=esd14, hue='Attrition')

plt.show()
```



Insight 5

Employees with Environment Satisfation Level 1 leave the organization the most. Attrition level of 25% is seen there, while in other cases, it is (statistically speaking) approximately same.

In [51]:

```
#Gender and Attrition level.

genderwisedata = data_copy.groupby(["Gender","Attrition"])
print(genderwisedata.groups.keys())
```

```
dict_keys([('Female', 'No'), ('Female', 'Yes'), ('Male', 'No'), ('Male', 'Yes')])
```

In [60]:

```
# Making the dataframe for the Gender and attrition level

gen = genderwisedata.get_group(('Female', 'No'))['Gender'].value_counts()
gen = pd.DataFrame(gen)
gen = gen.reset_index()
gen['Attrition'] = 'No'
gen

gen1 = genderwisedata.get_group(('Female', 'Yes'))['Gender'].value_counts()
gen1 = gen1.to_frame().reset_index()
gen1['Attrition'] = 'Yes'
gen2 = gen1.append(gen,ignore_index=True)
gen2

gen3 = genderwisedata.get_group(('Male', 'No'))['Gender'].value_counts()
gen3 = gen3.to_frame().reset_index()
gen3['Attrition'] = 'No'
gen4 = gen3.append(gen2,ignore_index=True)
gen4

gen5 = genderwisedata.get_group(('Male', 'Yes'))['Gender'].value_counts()
gen5 = gen5.to_frame().reset_index()
gen5['Attrition'] = 'Yes'
gen6 = gen5.append(gen4,ignore_index=True)
gen6
```

Out[60]:

	index	Gender	Attrition
0	Male	300	Yes
1	Male	1464	No
2	Female	174	Yes
3	Female	1002	No

In [79]:

```
#Calculating the % of attrition for differen gender

gen7 = data_copy.groupby(by = 'Gender')
print(gen7.groups.keys())
count_male = gen7.get_group("Male")['Gender'].value_counts().sum()
count_female = gen7.get_group('Female')['Gender'].value_counts().sum()
print(count_male,count_female)

#Calculating the attrition %
gen6['Attrition-%'] = 0

for i in range(len(gen6)):
    if gen6["index"][i] == "Male":
        if gen6['Attrition'][i] =='Yes':
            gen6['Attrition-%'][i] = gen6['Gender'][i]/count_male*100
    elif gen6["index"][i] == "Female":
        if gen6['Attrition'][i] =='Yes':
            gen6['Attrition-%'][i] = gen6['Gender'][i]/count_male*100


count_male_dep = gen7.get_group(("Male"))['Department'].value_counts().sum()
print("count_male_dep",count_male_dep)

gen6
```

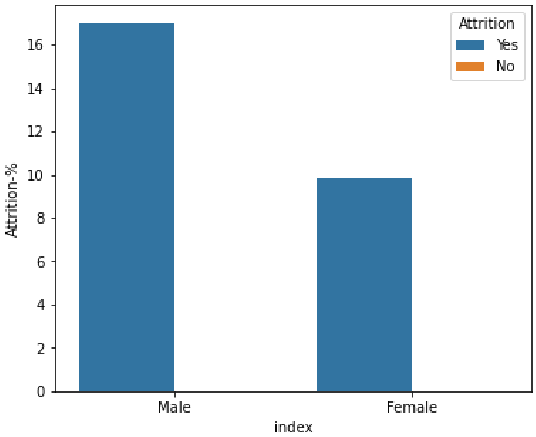
```
dict_keys(['Female', 'Male'])
1764 1176
count_male_dep 1764
```

Out[79]:

	index	Gender	Attrition	Attrition-%
0	Male	300	Yes	17.006803
1	Male	1464	No	0.000000
2	Female	174	Yes	9.863946
3	Female	1002	No	0.000000

In [81]:

```
fig = plt.figure(figsize = (6,5))
sns.barplot(x='index',y='Attrition-%', data=gen6, hue='Attrition')

plt.show()
```



Insight 6

Male gender's attrition is higher than that of female gender.

In [82]:

```
#Genderwise, and department wise attrition level


gen_dep_wisedata = data_copy.groupby(["Gender","Department","Attrition"])
print(gen_dep_wisedata.groups.keys())
```

```
dict_keys([('Female', 'Human Resources', 'No'), ('Female', 'Human Resources', 'Yes'), ('Female', 'Research & Develo
pment', 'No'), ('Female', 'Research & Development', 'Yes'), ('Female', 'Sales', 'No'), ('Female', 'Sales', 'Yes'),
('Male', 'Human Resources', 'No'), ('Male', 'Human Resources', 'Yes'), ('Male', 'Research & Development', 'No'),
('Male', 'Research & Development', 'Yes'), ('Male', 'Sales', 'No'), ('Male', 'Sales', 'Yes')])
```

In [104]:

```
# Making the dataframe for the Gender and department wise attrition level

gen_dep = gen_dep_wisedata.get_group(('Female', 'Human Resources', 'No'))['Gender'].value_counts()
gen_dep = pd.DataFrame(gen_dep)
gen_dep = gen_dep.reset_index()
gen_dep['Attrition'] = 'No'
gen_dep['Department'] = 'HR'
gen_dep

gen_dep1 = gen_dep_wisedata.get_group(('Female', 'Human Resources', 'Yes'))['Gender'].value_counts()
gen_dep1 = gen_dep1.to_frame().reset_index()
gen_dep1['Attrition'] = 'Yes'
gen_dep1['Department'] = 'HR'
gen_dep2 = gen_dep.append(gen_dep1,ignore_index=True)
gen_dep2

gen_dep3 = gen_dep_wisedata.get_group(('Female', 'Research & Development', 'No'))['Gender'].value_counts()
gen_dep3 = gen_dep3.to_frame().reset_index()
gen_dep3['Attrition'] = 'No'
gen_dep3['Department'] = 'R&D'
gen_dep4 = gen_dep2.append(gen_dep3,ignore_index=True)
gen_dep4

gen_dep5 = gen_dep_wisedata.get_group(('Female', 'Research & Development', 'Yes'))['Gender'].value_counts()
gen_dep5 = gen_dep5.to_frame().reset_index()
gen_dep5['Attrition'] = 'Yes'
gen_dep5['Department'] = 'R&D'
gen_dep6 = gen_dep4.append(gen_dep5,ignore_index=True)
gen_dep6

gen_dep7 = gen_dep_wisedata.get_group(('Female', 'Sales', 'No'))['Gender'].value_counts()
gen_dep7 = gen_dep7.to_frame().reset_index()
gen_dep7['Attrition'] = 'No'
gen_dep7['Department'] = 'Sales'
gen_dep8 = gen_dep6.append(gen_dep7,ignore_index=True)
gen_dep8

gen_dep9 = gen_dep_wisedata.get_group(('Female', 'Sales', 'Yes'))['Gender'].value_counts()
gen_dep9 = gen_dep9.to_frame().reset_index()
gen_dep9['Attrition'] = 'Yes'
gen_dep9['Department'] = 'Sales'
gen_dep10 = gen_dep8.append(gen_dep9,ignore_index=True)
gen_dep10

gen_dep11 = gen_dep_wisedata.get_group(('Male', 'Human Resources', 'No'))['Gender'].value_counts()
gen_dep11 = gen_dep11.to_frame().reset_index()
gen_dep11['Attrition'] = 'No'
gen_dep11['Department'] = 'HR'
gen_dep12 = gen_dep10.append(gen_dep11,ignore_index=True)
gen_dep12

gen_dep13 = gen_dep_wisedata.get_group(('Male', 'Human Resources', 'Yes'))['Gender'].value_counts()
gen_dep13 = gen_dep13.to_frame().reset_index()
gen_dep13['Attrition'] = 'Yes'
gen_dep13['Department'] = 'HR'
gen_dep14 = gen_dep12.append(gen_dep13,ignore_index=True)
gen_dep14


gen_dep15 = gen_dep_wisedata.get_group(('Male', 'Research & Development', 'No'))['Gender'].value_counts()
gen_dep15 = gen_dep15.to_frame().reset_index()
gen_dep15['Attrition'] = 'No'
gen_dep15['Department'] = 'R&D'
gen_dep16 = gen_dep14.append(gen_dep15,ignore_index=True)
gen_dep16

gen_dep17 = gen_dep_wisedata.get_group(('Male', 'Research & Development', 'Yes'))['Gender'].value_counts()
gen_dep17 = gen_dep17.to_frame().reset_index()
gen_dep17['Attrition'] = 'Yes'
gen_dep17['Department'] = 'R&D'
gen_dep18 = gen_dep16.append(gen_dep17,ignore_index=True)
gen_dep18


gen_dep19 = gen_dep_wisedata.get_group(('Male', 'Sales', 'No'))['Gender'].value_counts()
gen_dep19 = gen_dep19.to_frame().reset_index()
gen_dep19['Attrition'] = 'No'
gen_dep19['Department'] = 'Sales'
gen_dep20 = gen_dep18.append(gen_dep19,ignore_index=True)
gen_dep20

gen_dep21 = gen_dep_wisedata.get_group(('Male', 'Sales', 'Yes'))['Gender'].value_counts()
gen_dep21 = gen_dep21.to_frame().reset_index()
gen_dep21['Attrition'] = 'Yes'
```

```
gen_dep21['Department'] = 'Sales'
gen_dep22 = gen_dep20.append(gen_dep21,ignore_index=True)
gen_dep22
```

Out[104]:

	index	Gender	Attrition	Department
0	Female	28	No	HR
1	Female	12	Yes	HR
2	Female	672	No	R&D
3	Female	86	Yes	R&D
4	Female	302	No	Sales
5	Female	76	Yes	Sales
6	Male	74	No	HR
7	Male	12	Yes	HR
8	Male	984	No	R&D
9	Male	180	Yes	R&D
10	Male	406	No	Sales
11	Male	108	Yes	Sales

In [117]:

```
#Calculating the % of attrition for differen gender in different departments

gen_dep23 = data_copy.groupby(['Gender','Department'])
print(gen_dep23.groups.keys())

count_m_hr = gen_dep23.get_group(("Male",'Human Resources'))['Gender'].value_counts().sum()
count_f_hr = gen_dep23.get_group(("Female",'Human Resources'))['Gender'].value_counts().sum()
count_m_rd = gen_dep23.get_group(("Male",'Research & Development'))['Gender'].value_counts().sum()
count_f_rd = gen_dep23.get_group(("Female",'Research & Development'))['Gender'].value_counts().sum()
count_m_s = gen_dep23.get_group(("Male",'Sales'))['Gender'].value_counts().sum()
count_f_s = gen_dep23.get_group(("Female",'Sales'))['Gender'].value_counts().sum()
print(count_m_hr,count_f_hr,count_m_rd,count_f_rd,count_m_s,count_f_s)

#Calculating the attrition %

gen_dep22['Attrition-%'] = 0

for i in range(len(gen_dep22)):
    if gen_dep22["index"][i] == "Male":
        if gen_dep22["Department"][i] == 'HR':
            if gen_dep22['Attrition'][i] =='Yes':
                gen_dep22['Attrition-%'][i] = gen_dep22['Gender'][i]/count_m_hr*100
        elif gen_dep22["Department"][i] == 'R&D':
            if gen_dep22['Attrition'][i] =='Yes':
                gen_dep22['Attrition-%'][i] = gen_dep22['Gender'][i]/count_m_rd*100
        elif gen_dep22["Department"][i] =='Sales':
            if gen_dep22['Attrition'][i] =='Yes':
                gen_dep22['Attrition-%'][i] = gen_dep22['Gender'][i]/count_m_s*100

    elif gen_dep22["index"][i] == "Female":
        if gen_dep22["Department"][i] == 'HR':
            if gen_dep22['Attrition'][i] =='Yes':
                gen_dep22['Attrition-%'][i] = gen_dep22['Gender'][i]/count_f_hr*100
        elif gen_dep22["Department"][i] == 'R&D':
            if gen_dep22['Attrition'][i] =='Yes':
                gen_dep22['Attrition-%'][i] = gen_dep22['Gender'][i]/count_f_rd*100
        elif gen_dep22["Department"][i] == 'Sales':
            if gen_dep22['Attrition'][i] =='Yes':
                gen_dep22['Attrition-%'][i] = gen_dep22['Gender'][i]/count_f_s*100



gen_dep22
```

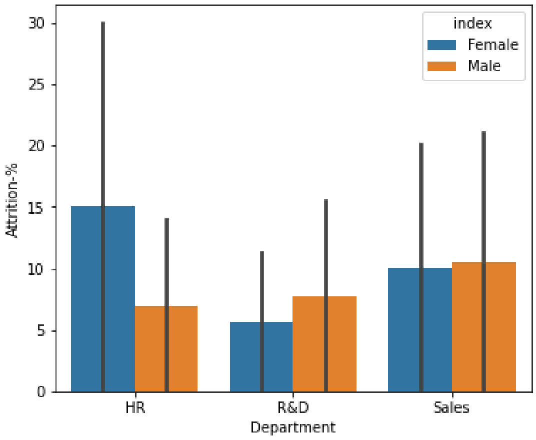
```
dict_keys([('Female', 'Human Resources'), ('Female', 'Research & Development'), ('Female', 'Sales'), ('Male', 'Huma
n Resources'), ('Male', 'Research & Development'), ('Male', 'Sales')])
86 40 1164 758 514 378
```

Out[117]:

	index	Gender	Attrition	Department	Attrition-%
0	Female	28	No	HR	0.000000
1	Female	12	Yes	HR	30.000000
2	Female	672	No	R&D	0.000000
3	Female	86	Yes	R&D	11.345646
4	Female	302	No	Sales	0.000000
5	Female	76	Yes	Sales	20.105820
6	Male	74	No	HR	0.000000
7	Male	12	Yes	HR	13.953488
8	Male	984	No	R&D	0.000000
9	Male	180	Yes	R&D	15.463918
10	Male	406	No	Sales	0.000000
11	Male	108	Yes	Sales	21.011673

In [118]:

```
fig = plt.figure(figsize = (6,5))
sns.barplot(x='Department',y='Attrition-%', data=gen_dep22, hue='index')

plt.show()
```



Insight 7

Female employees of HR department are the one who are moving out the most whic is 30%

While Sales department attrition rate shows an interesting data, both male and female employees are equally leaving the department and moving out of the organization.

In [121]:

```
#JobRole wise attrition level

jobrole_wisedata = data_copy.groupby(["JobRole","Attrition"])
print(jobrole_wisedata.groups.keys())
```

```
dict_keys([('Healthcare Representative', 'No'), ('Healthcare Representative', 'Yes'), ('Human Resources', 'No'),
('Human Resources', 'Yes'), ('Laboratory Technician', 'No'), ('Laboratory Technician', 'Yes'), ('Manager', 'No'),
('Manager', 'Yes'), ('Manufacturing Director', 'No'), ('Manufacturing Director', 'Yes'), ('Research Director', 'N
o'), ('Research Director', 'Yes'), ('Research Scientist', 'No'), ('Research Scientist', 'Yes'), ('Sales Executive',
'No'), ('Sales Executive', 'Yes'), ('Sales Representative', 'No'), ('Sales Representative', 'Yes')])
```

In [140]:

```
# Making the dataframe for the JobRole wise attrition level

jobrole = jobrole_wisedata.get_group(('Healthcare Representative','No'))['JobRole'].value_counts()
jobrole = pd.DataFrame(jobrole)
jobrole = jobrole.reset_index()
jobrole['Attrition'] = 'No'
jobrole

jobrole1 = jobrole_wisedata.get_group(('Healthcare Representative','Yes'))['JobRole'].value_counts()
jobrole1 = jobrole1.to_frame().reset_index()
jobrole1['Attrition'] = 'Yes'
jobrole2 = jobrole.append(jobrole1,ignore_index=True)
jobrole2

jobrole3 = jobrole_wisedata.get_group(('Human Resources','No'))['JobRole'].value_counts()
jobrole3 = jobrole3.to_frame().reset_index()
jobrole3['Attrition'] = 'No'
jobrole4 = jobrole2.append(jobrole3,ignore_index=True)
jobrole4

jobrole5 = jobrole_wisedata.get_group(('Human Resources','Yes'))['JobRole'].value_counts()
jobrole5 = jobrole5.to_frame().reset_index()
jobrole5['Attrition'] = 'Yes'
jobrole6 = jobrole4.append(jobrole5,ignore_index=True)
jobrole6

jobrole7 = jobrole_wisedata.get_group(('Laboratory Technician','No'))['JobRole'].value_counts()
jobrole7 = jobrole7.to_frame().reset_index()
jobrole7['Attrition'] = 'No'
jobrole8 = jobrole6.append(jobrole7,ignore_index=True)
jobrole8

jobrole9 = jobrole_wisedata.get_group(('Laboratory Technician','Yes'))['JobRole'].value_counts()
jobrole9 = jobrole9.to_frame().reset_index()
jobrole9['Attrition'] = 'Yes'
jobrole10 = jobrole8.append(jobrole9,ignore_index=True)
jobrole10

jobrole11 = jobrole_wisedata.get_group(('Manager','No'))['JobRole'].value_counts()
jobrole11 = jobrole11.to_frame().reset_index()
jobrole11['Attrition'] = 'No'
jobrole12 = jobrole10.append(jobrole11,ignore_index=True)
jobrole12

jobrole13 = jobrole_wisedata.get_group(('Manager','Yes'))['JobRole'].value_counts()
jobrole13 = jobrole13.to_frame().reset_index()
jobrole13['Attrition'] = 'Yes'
jobrole14 = jobrole12.append(jobrole13,ignore_index=True)
jobrole14

jobrole15 = jobrole_wisedata.get_group(('Manufacturing Director','No'))['JobRole'].value_counts()
jobrole15 = jobrole15.to_frame().reset_index()
jobrole15['Attrition'] = 'No'
jobrole16 = jobrole14.append(jobrole15,ignore_index=True)
jobrole16

jobrole17 = jobrole_wisedata.get_group(('Manufacturing Director','Yes'))['JobRole'].value_counts()
jobrole17= jobrole17.to_frame().reset_index()
jobrole17['Attrition'] = 'Yes'
jobrole18 = jobrole16.append(jobrole17,ignore_index=True)
jobrole18

jobrole19 = jobrole_wisedata.get_group(('Research Director','No'))['JobRole'].value_counts()
jobrole19= jobrole19.to_frame().reset_index()
jobrole19['Attrition'] = 'No'
jobrole20 = jobrole18.append(jobrole19,ignore_index=True)
jobrole20

jobrole21 = jobrole_wisedata.get_group(('Research Director','Yes'))['JobRole'].value_counts()
jobrole21= jobrole21.to_frame().reset_index()
jobrole21['Attrition'] = 'Yes'
jobrole22 = jobrole20.append(jobrole21,ignore_index=True)
jobrole22

jobrole23 = jobrole_wisedata.get_group(('Research Scientist','No'))['JobRole'].value_counts()
jobrole23= jobrole23.to_frame().reset_index()
jobrole23['Attrition'] = 'No'
jobrole24 = jobrole22.append(jobrole23,ignore_index=True)
jobrole24

jobrole25 = jobrole_wisedata.get_group(('Research Scientist','Yes'))['JobRole'].value_counts()
jobrole25= jobrole25.to_frame().reset_index()
jobrole25['Attrition'] = 'Yes'
jobrole26 = jobrole24.append(jobrole25,ignore_index=True)
```

```
jobrole26


jobrole27 = jobrole_wisedata.get_group(('Sales Executive','No'))['JobRole'].value_counts()
jobrole27= jobrole27.to_frame().reset_index()
jobrole27['Attrition'] = 'No'
jobrole28 = jobrole26.append(jobrole27,ignore_index=True)
jobrole28


jobrole29 = jobrole_wisedata.get_group(('Sales Executive','Yes'))['JobRole'].value_counts()
jobrole29= jobrole29.to_frame().reset_index()
jobrole29['Attrition'] = 'Yes'
jobrole30 = jobrole28.append(jobrole29,ignore_index=True)
jobrole30

jobrole31 = jobrole_wisedata.get_group(('Sales Representative','No'))['JobRole'].value_counts()
jobrole31= jobrole31.to_frame().reset_index()
jobrole31['Attrition'] = 'No'
jobrole32 = jobrole30.append(jobrole31,ignore_index=True)
jobrole32

jobrole33 = jobrole_wisedata.get_group(('Sales Representative','Yes'))['JobRole'].value_counts()
jobrole33= jobrole33.to_frame().reset_index()
jobrole33['Attrition'] = 'Yes'
jobrole34 = jobrole32.append(jobrole33,ignore_index=True)
jobrole34



262
```

Out[140]:

	index	JobRole	Attrition
0	Healthcare Representative	244	No
1	Healthcare Representative	18	Yes
2	Human Resources	80	No
3	Human Resources	24	Yes
4	Laboratory Technician	394	No
5	Laboratory Technician	124	Yes
6	Manager	194	No
7	Manager	10	Yes
8	Manufacturing Director	270	No
9	Manufacturing Director	20	Yes
10	Research Director	156	No
11	Research Director	4	Yes
12	Research Scientist	490	No
13	Research Scientist	94	Yes
14	Sales Executive	538	No
15	Sales Executive	114	Yes
16	Sales Representative	100	No
17	Sales Representative	66	Yes

In [144]:

```
#Calculating the % of attrition for differen job roles

jobrole35 = data_copy.groupby(by = 'JobRole')
print(jobrole35.groups.keys())

count_HReps=jobrole35.get_group("Healthcare Representative")['JobRole'].value_counts().sum()
count_HR=jobrole35.get_group("Human Resources")['JobRole'].value_counts().sum()
count_lab=jobrole35.get_group("Laboratory Technician")['JobRole'].value_counts().sum()
count_mgr=jobrole35.get_group("Manager")['JobRole'].value_counts().sum()
count_MD=jobrole35.get_group("Manufacturing Director")['JobRole'].value_counts().sum()
count_RD=jobrole35.get_group("Research Director")['JobRole'].value_counts().sum()
count_RS=jobrole35.get_group("Research Scientist")['JobRole'].value_counts().sum()
count_SE=jobrole35.get_group("Sales Executive")['JobRole'].value_counts().sum()
count_SR=jobrole35.get_group("Sales Representative")['JobRole'].value_counts().sum()
# print(count_HReps,count_HR)

#Calculating the attrition %
jobrole34['Attrition-%'] = 0

for i in range(len(jobrole34)):
    if jobrole34["index"][i] == "Healthcare Representative":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] = jobrole34['JobRole'][i]/count_HReps*100
    elif jobrole34["index"][i] == "Human Resources":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_HR*100
    elif jobrole34["index"][i] == "Laboratory Technician":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_lab*100
    elif jobrole34["index"][i] == "Manager":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_mgr*100
    elif jobrole34["index"][i] == "Manufacturing Director":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_MD*100
    elif jobrole34["index"][i] == "Research Director":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_RD*100
    elif jobrole34["index"][i] == "Research Scientist":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_RS*100
    elif jobrole34["index"][i] == "Sales Executive":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_SE*100
    elif jobrole34["index"][i] == "Sales Representative":
        if jobrole34['Attrition'][i] =='Yes':
            jobrole34['Attrition-%'][i] =jobrole34['JobRole'][i]/count_SR*100


jobrole34
```

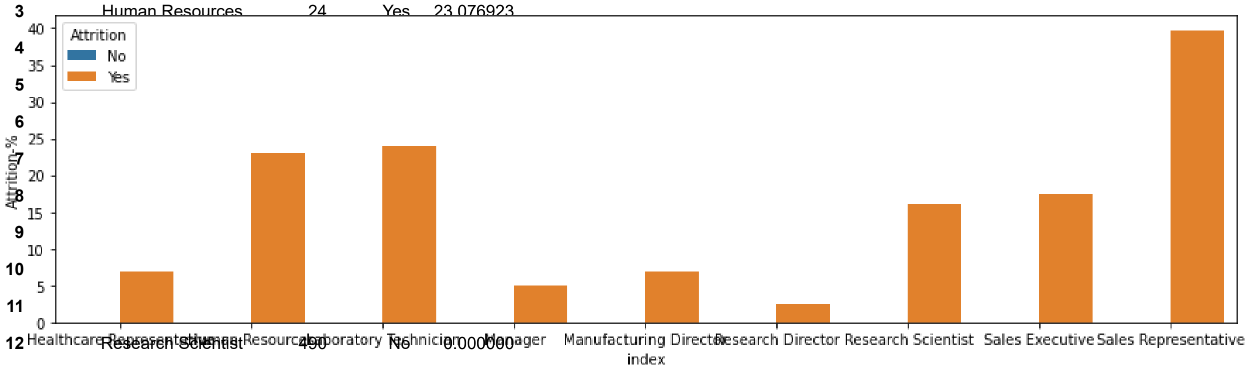
```
dict_keys(['Healthcare Representative', 'Human Resources', 'Laboratory Technician', 'Manager', 'Manufacturing Direc
tor', 'Research Director', 'Research Scientist', 'Sales Executive', 'Sales Representative'])
```

Out[144]:

	index	JobRole	Attrition	Attrition-%
0	Healthcare Representative	244	No	0.000000
1	Healthcare Representative	18	Yes	6.870229
2	Human Resources	80	No	0.000000
3	Human Resources	24	Yes	23.076923
4				
5				
6				
7				
8				
9				
10				
11				
12	Research Scientist	490	No	0.000000
13	Research Scientist	94	Yes	16.095890
14	Sales Executive	538	No	0.000000
15	Sales Executive	114	Yes	17.484663
16	Sales Representative	100	No	0.000000
17	Sales Representative	66	Yes	39.759036

In [149]:

```
fig = plt.figure(figsize = (15,4))
sns.barplot(x='index',y='Attrition-%', data=jobrole34, hue='Attrition')

plt.show()
```



Insight 8

Sales Representatives jobroles sees maximum attrition. We have already seen that the Sales department sees maximum attrition and this insight tells us, which job role within the department has maximum attrition.

Attrition is high for Human resources and lab technicians also.

In [150]:

```
#JobSatisfaction wise attrition level

jobsat_wisedata = data_copy.groupby(["JobSatisfaction","Attrition"])
print(jobsat_wisedata.groups.keys())
```

```
dict_keys([(1, 'No'), (1, 'Yes'), (2, 'No'), (2, 'Yes'), (3, 'No'), (3, 'Yes'), (4, 'No'), (4, 'Yes')])
```

In [166]:

```
# Making the dataframe for the JobSatisfaction wise attrition level

jobsat = jobsat_wisedata.get_group((1,'No'))['JobSatisfaction'].value_counts()
jobsat = pd.DataFrame(jobsat)
jobsat = jobsat.reset_index()
jobsat['Attrition'] = 'No'
jobsat

jobsat1 = jobsat_wisedata.get_group((1,'Yes'))['JobSatisfaction'].value_counts()
jobsat1 = jobsat1.to_frame().reset_index()
jobsat1['Attrition'] = 'Yes'
jobsat2 = jobsat.append(jobsat1,ignore_index=True)
jobsat2


jobsat3 = jobsat_wisedata.get_group((2,'No'))['JobSatisfaction'].value_counts()
jobsat3= jobsat3.to_frame().reset_index()
jobsat3['Attrition'] = 'No'
jobsat4 = jobsat2.append(jobsat3,ignore_index=True)
jobsat4

jobsat5 = jobsat_wisedata.get_group((2,'Yes'))['JobSatisfaction'].value_counts()
jobsat5 = jobsat5.to_frame().reset_index()
jobsat5['Attrition'] = 'Yes'
jobsat6 = jobsat4.append(jobsat5,ignore_index=True)
jobsat6

jobsat7 = jobsat_wisedata.get_group((3,'No'))['JobSatisfaction'].value_counts()
jobsat7 = jobsat7.to_frame().reset_index()
jobsat7['Attrition'] = 'No'
jobsat8 = jobsat6.append(jobsat7,ignore_index=True)
jobsat8

jobsat9 = jobsat_wisedata.get_group((3,'Yes'))['JobSatisfaction'].value_counts()
jobsat9 = jobsat9.to_frame().reset_index()
jobsat9['Attrition'] = 'Yes'
jobsat10 = jobsat8.append(jobsat9,ignore_index=True)
jobsat10

jobsat11 = jobsat_wisedata.get_group((4,'No'))['JobSatisfaction'].value_counts()
jobsat11= jobsat11.to_frame().reset_index()
jobsat11['Attrition'] = 'No'
jobsat12 = jobsat10.append(jobsat11,ignore_index=True)
jobsat12

jobsat13 = jobsat_wisedata.get_group((4,'Yes'))['JobSatisfaction'].value_counts()
jobsat13= jobsat13.to_frame().reset_index()
jobsat13['Attrition'] = 'Yes'
jobsat14 = jobsat12.append(jobsat13,ignore_index=True)
jobsat14
```

Out[166]:

	index	JobSatisfaction	Attrition
0	1	446	No
1	1	132	Yes
2	2	468	No
3	2	92	Yes
4	3	738	No
5	3	146	Yes
6	4	814	No
7	4	104	Yes

In [168]:

```
#Calculating the % of attrition for differen job roles

jobsat15 = data_copy.groupby(by = 'JobSatisfaction')
print(jobsat15.groups.keys())

count_1=jobsat15.get_group(1)['JobSatisfaction'].value_counts().sum()
count_2=jobsat15.get_group(2)['JobSatisfaction'].value_counts().sum()
count_3=jobsat15.get_group(3)['JobSatisfaction'].value_counts().sum()
count_4=jobsat15.get_group(4)['JobSatisfaction'].value_counts().sum()
print(count_1,count_2,count_3,count_4)

# Calculating the attrition %

jobsat14['Attrition-%'] = 0

for i in range(len(jobsat14)):
    if jobsat14["index"][i] == 1:
        if jobsat14['Attrition'][i] =='Yes':
            jobsat14['Attrition-%'][i] = jobsat14['JobSatisfaction'][i]/count_1*100
    elif jobsat14["index"][i] == 2:
        if jobsat14['Attrition'][i] =='Yes':
            jobsat14['Attrition-%'][i] = jobsat14['JobSatisfaction'][i]/count_2*100
    elif jobsat14["index"][i] == 3:
        if jobsat14['Attrition'][i] =='Yes':
            jobsat14['Attrition-%'][i] = jobsat14['JobSatisfaction'][i]/count_3*100
    elif jobsat14["index"][i] == 4:
        if jobsat14['Attrition'][i] =='Yes':
            jobsat14['Attrition-%'][i] = jobsat14['JobSatisfaction'][i]/count_4*100


jobsat14
```

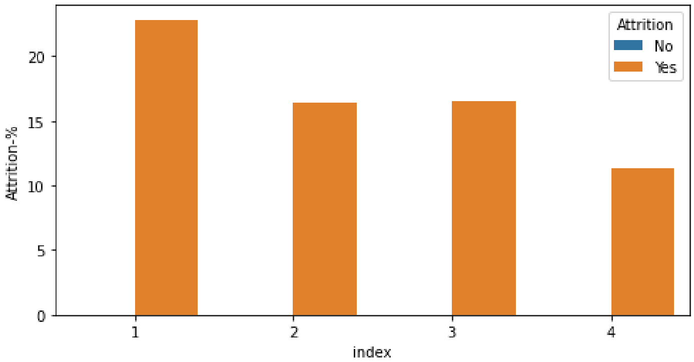
```
dict_keys([1, 2, 3, 4])
578 560 884 918
```

Out[168]:

	index	JobSatisfaction	Attrition	Attrition-%
0	1	446	No	0.000000
1	1	132	Yes	22.837370
2	2	468	No	0.000000
3	2	92	Yes	16.428571
4	3	738	No	0.000000
5	3	146	Yes	16.515837
6	4	814	No	0.000000
7	4	104	Yes	11.328976

In [169]:

```
fig=plt.figure(figsize = (8,4))
sns.barplot(x='index',y='Attrition-%',data = jobsat14,hue="Attrition")
plt.show()
```



Insight 9

As expected, employees with low satisfaction have highest attrition rate while the emplpyees with highest job satisfactio of 4 have the lowest attrition rate.

In [170]:

```
#Relation between Martial Status and attrition level

# mud - married, unmarried(single), divorced.

mud_wisedata = data_copy.groupby(["MaritalStatus","Attrition"])
print(mud_wisedata.groups.keys())
```

```
dict_keys([('Divorced', 'No'), ('Divorced', 'Yes'), ('Married', 'No'), ('Married', 'Yes'), ('Single', 'No'), ('Sing
le', 'Yes')])
```

In [179]:

```
# Making the dataframe for the MaritalStatus and attrition level

mud = mud_wisedata.get_group(('Divorced','Yes'))['MaritalStatus'].value_counts()
mud = pd.DataFrame(mud)
mud = mud.reset_index()
mud['Attrition'] = 'Yes'
mud

mud1 = mud_wisedata.get_group(('Divorced','No'))['MaritalStatus'].value_counts()
mud1 = mud1.to_frame().reset_index()
mud1['Attrition'] = 'No'
mud2 = mud1.append(mud,ignore_index=True)
mud2

mud3 = mud_wisedata.get_group(('Married','No'))['MaritalStatus'].value_counts()
mud3 = mud3.to_frame().reset_index()
mud3['Attrition'] = 'No'
mud4 = mud2.append(mud3,ignore_index=True)
mud4

mud5 = mud_wisedata.get_group(('Married','Yes'))['MaritalStatus'].value_counts()
mud5 = mud5.to_frame().reset_index()
mud5['Attrition'] = 'Yes'
mud6 = mud4.append(mud5,ignore_index=True)
mud6

mud7 = mud_wisedata.get_group(('Single','No'))['MaritalStatus'].value_counts()
mud7 = mud7.to_frame().reset_index()
mud7['Attrition'] = 'No'
mud8 = mud6.append(mud7,ignore_index=True)
mud8

mud9 = mud_wisedata.get_group(('Single','Yes'))['MaritalStatus'].value_counts()
mud9 = mud9.to_frame().reset_index()
mud9['Attrition'] = 'Yes'
mud10 = mud8.append(mud9,ignore_index=True)
mud10
```

Out[179]:

	index	MaritalStatus	Attrition
0	Divorced	588	No
1	Divorced	66	Yes
2	Married	1178	No
3	Married	168	Yes
4	Single	700	No
5	Single	240	Yes

In [187]:

```
#Calculating the % of attrition for differen marital status

mud11 = data_copy.groupby(by = 'MaritalStatus')
print(mud11.groups.keys())

count_m=mud11.get_group("Married")['MaritalStatus'].value_counts().sum()
count_d=mud11.get_group("Divorced")['MaritalStatus'].value_counts().sum()
count_u=mud11.get_group("Single")['MaritalStatus'].value_counts().sum()
print(count_m,count_d,count_u)

# Calculating the attrition %

mud10['Attrition-%'] = 0

for i in range(len(mud10)):
    if mud10["index"][i] == "Married":
        if mud10['Attrition'][i] =='Yes':
            mud10['Attrition-%'][i] = mud10['MaritalStatus'][i]/count_m*100
    elif mud10["index"][i] == "Divorced":
        if mud10['Attrition'][i] =='Yes':
            mud10['Attrition-%'][i] = mud10['MaritalStatus'][i]/count_d*100
    elif mud10["index"][i] == "Single":
        if mud10['Attrition'][i] =='Yes':
            mud10['Attrition-%'][i] = mud10['MaritalStatus'][i]/count_u*100



mud10
```

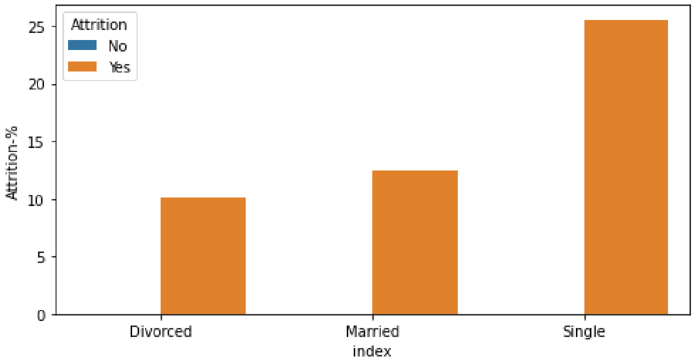
```
dict_keys(['Divorced', 'Married', 'Single'])
1346 654 940
```

Out[187]:

	index	MaritalStatus	Attrition	Attrition-%
0	Divorced	588	No	0.000000
1	Divorced	66	Yes	10.091743
2	Married	1178	No	0.000000
3	Married	168	Yes	12.481426
4	Single	700	No	0.000000
5	Single	240	Yes	25.531915

In [188]:

```
fig = plt.figure(figsize = (8,4))
sns.barplot(x='index',y='Attrition-%',data=mud10,hue ='Attrition')
plt.show()
```



Insight 10

Single/Unmarried emplopyees show a highest attrition rate of 25.5%. While the divorced and married employees shows approximately similar attrition rate.

In [212]:

```
#PLotting Attrition level and Monthly Income relation with the Marital Status.

fig = plt.figure(figsize=(10,4))
sns.stripplot(x='Attrition',y='MonthlyIncome',data=data_copy,jitter = True,hue='MaritalStatus',dodge = True,palette='afmhot')
plt.show()
```

Insight 11

Employees who are single and whose monthly income is around 2500-4500 leave the organiztion the most. The blue plot looks heavily populated.

In [183]:

```
#Relation between WorkLifeBalance and attrition level

wlb_wisedata = data_copy.groupby(["WorkLifeBalance","Attrition"])
print(wlb_wisedata.groups.keys())
```

```
dict_keys([(1, 'No'), (1, 'Yes'), (2, 'No'), (2, 'Yes'), (3, 'No'), (3, 'Yes'), (4, 'No'), (4, 'Yes')])
```

In [186]:

```
# Making the dataframe for the WorkLifeBalance and attrition level

wlb = wlb_wisedata.get_group((1,'Yes'))['WorkLifeBalance'].value_counts()
wlb = pd.DataFrame(wlb)
wlb = wlb.reset_index()
wlb['Attrition'] = 'Yes'
wlb

wlb1 = wlb_wisedata.get_group((1,'No'))['WorkLifeBalance'].value_counts()
wlb1 = wlb1.to_frame().reset_index()
wlb1['Attrition'] = 'No'
wlb2 = wlb.append(wlb1,ignore_index=True)
wlb2

wlb3 = wlb_wisedata.get_group((2,'Yes'))['WorkLifeBalance'].value_counts()
wlb3 = wlb3.to_frame().reset_index()
wlb3['Attrition'] = 'Yes'
wlb4 = wlb2.append(wlb3,ignore_index=True)
wlb4

wlb5 = wlb_wisedata.get_group((2,'No'))['WorkLifeBalance'].value_counts()
wlb5 = wlb5.to_frame().reset_index()
wlb5['Attrition'] = 'No'
wlb6 = wlb4.append(wlb5,ignore_index=True)
wlb6

wlb7 = wlb_wisedata.get_group((3,'Yes'))['WorkLifeBalance'].value_counts()
wlb7 = wlb7.to_frame().reset_index()
wlb7['Attrition'] = 'Yes'
wlb8 = wlb6.append(wlb7,ignore_index=True)
wlb8

wlb9 = wlb_wisedata.get_group((3,'No'))['WorkLifeBalance'].value_counts()
wlb9 = wlb9.to_frame().reset_index()
wlb9['Attrition'] = 'No'
wlb10 = wlb8.append(wlb9,ignore_index=True)
wlb10

wlb11 = wlb_wisedata.get_group((4,'Yes'))['WorkLifeBalance'].value_counts()
wlb11 = wlb11.to_frame().reset_index()
wlb11['Attrition'] = 'Yes'
wlb12= wlb10.append(wlb11,ignore_index=True)
wlb12

wlb13 = wlb_wisedata.get_group((4,'No'))['WorkLifeBalance'].value_counts()
wlb13= wlb13.to_frame().reset_index()
wlb13['Attrition'] = 'No'
wlb14= wlb12.append(wlb13,ignore_index=True)
wlb14
```

Out[186]:

	index	WorkLifeBalance	Attrition
0	1	50	Yes
1	1	110	No
2	2	116	Yes
3	2	572	No
4	3	254	Yes
5	3	1532	No
6	4	54	Yes
7	4	252	No

In [190]:

```
#Calculating the % of attrition for differen Work life balance levels.

wlb15 = data_copy.groupby(by = 'WorkLifeBalance')
print(wlb15.groups.keys())

count_1=wlb15.get_group(1)['WorkLifeBalance'].value_counts().sum()
count_2=wlb15.get_group(2)['WorkLifeBalance'].value_counts().sum()
count_3=wlb15.get_group(3)['WorkLifeBalance'].value_counts().sum()
count_4=wlb15.get_group(4)['WorkLifeBalance'].value_counts().sum()
print(count_1,count_2,count_3,count_4)

# Calculating the attrition %

wlb14['Attrition-%'] = 0

for i in range(len(wlb14)):
    if wlb14["index"][i] == 1:
        if wlb14['Attrition'][i] =='Yes':
            wlb14['Attrition-%'][i] = wlb14['WorkLifeBalance'][i]/count_1*100
    elif wlb14["index"][i] == 2:
        if wlb14['Attrition'][i] =='Yes':
            wlb14['Attrition-%'][i] = wlb14['WorkLifeBalance'][i]/count_2*100
    elif wlb14["index"][i] == 3:
        if wlb14['Attrition'][i] =='Yes':
            wlb14['Attrition-%'][i] = wlb14['WorkLifeBalance'][i]/count_3*100
    elif wlb14["index"][i] == 4:
        if wlb14['Attrition'][i] =='Yes':
            wlb14['Attrition-%'][i] = wlb14['WorkLifeBalance'][i]/count_4*100


wlb14
```

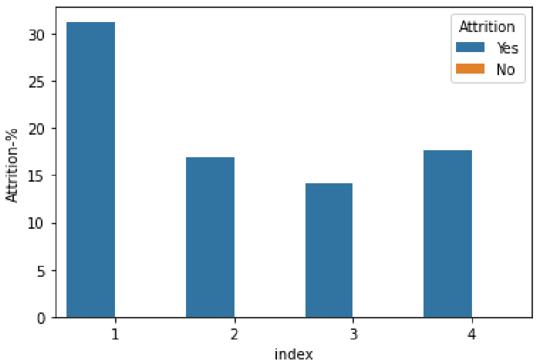
```
dict_keys([1, 2, 3, 4])
160 688 1786 306
```

Out[190]:

	index	WorkLifeBalance	Attrition	Attrition-%
0	1	50	Yes	31.250000
1	1	110	No	0.000000
2	2	116	Yes	16.860465
3	2	572	No	0.000000
4	3	254	Yes	14.221725
5	3	1532	No	0.000000
6	4	54	Yes	17.647059
7	4	252	No	0.000000

In [191]:

```
fig = plt.figure(figsize=(6,4))
sns.barplot(x='index',y='Attrition-%',data=wlb14,hue='Attrition')
plt.show()
```



Insight 12

Employees whose Work-LIfe Balance is 1, which is least, they are the ones who leave the organization the most. The attrition rate is rather very high = 31%

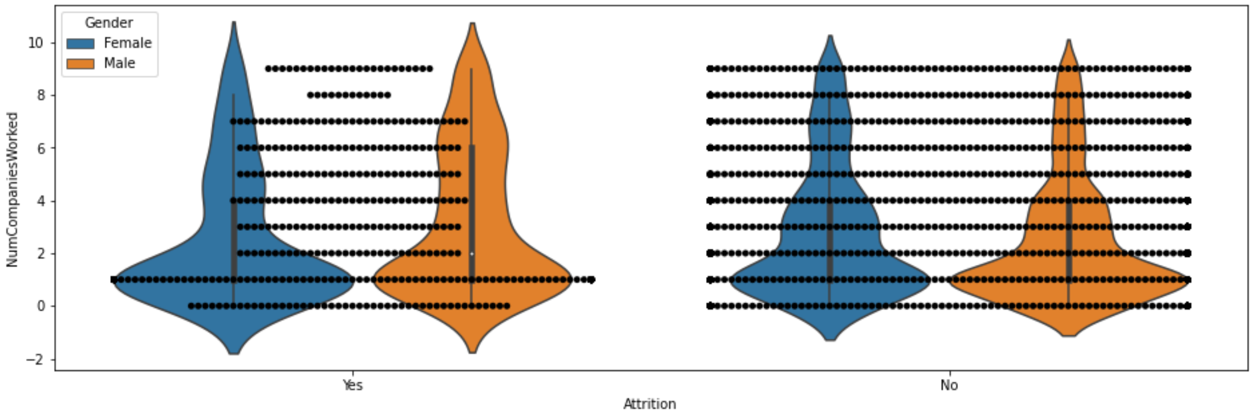
While for all the other levels, attrition is (statistically speaking) similar to each other.

In [236]:

```
# Attrion level with respect to the Number of companies employees have worked in and gender.

fig= plt.figure(figsize=(16,5))
sns.violinplot(x='Attrition',y='NumCompaniesWorked',data=data_copy,hue='Gender')
sns.swarmplot(x='Attrition',y='NumCompaniesWorked',data=data_copy,color='black')
```

Out[236]:

```
<AxesSubplot:xlabel='Attrition', ylabel='NumCompaniesWorked'>
```



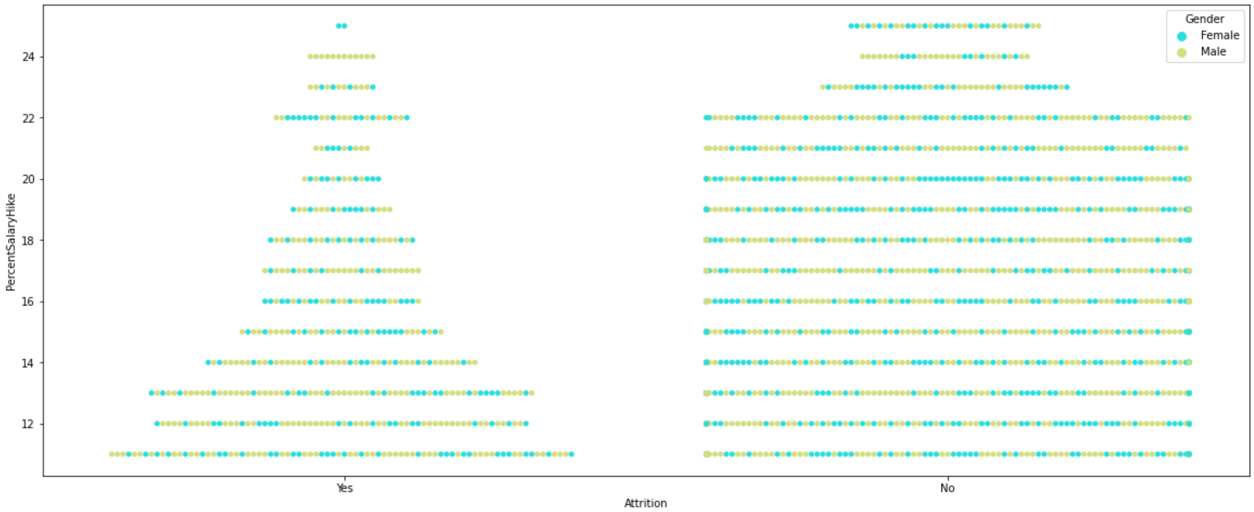
Insight 13

We can see from the above graph, female employees who have worked in 0-2 companies shifts more than the male counterpart. But Male gender also, switches maximum when the number of companies worked = 0-2.

Lesser number of employees who have worked in many companies, leave the organization.

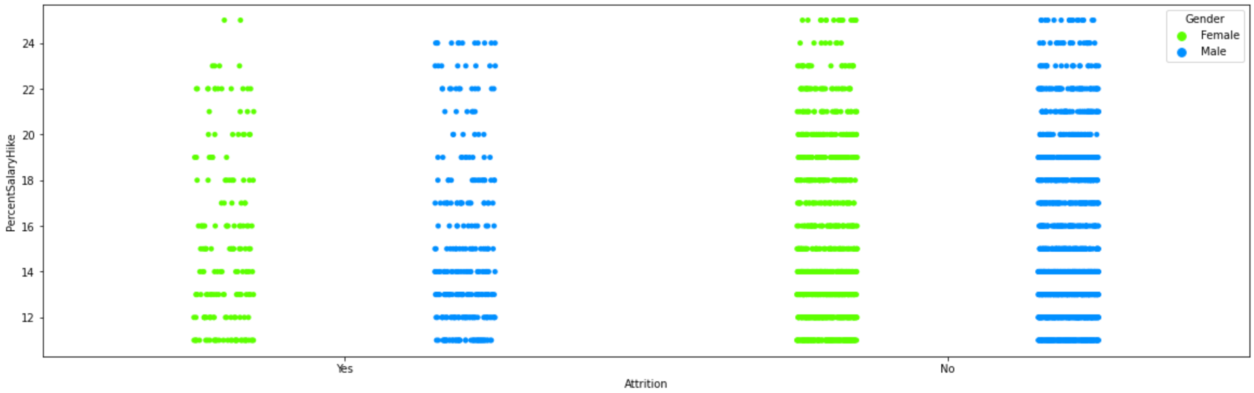
In [248]:

```
# Relationship between attirtion and Percent Salary hike given to the employees.

fig=plt.figure(figsize=(20,8))
pp = sns.swarmplot(x='Attrition',y='PercentSalaryHike',data=data_copy,hue='Gender',palette='rainbow')
# sns.stripplot(x='Attrition',y='PercentSalaryHike',data=data_copy,jitter = True,hue='Gender',dodge = True,palette='gist_rainbow'
```



In [249]:

```
#Relation between Attrition Level and PercentSalaryHike considering Gender.

fig = plt.figure(figsize=(20,6))
sns.stripplot(x='Attrition',y='PercentSalaryHike',data=data_copy,jitter = True,hue='Gender',dodge = True,palette='gist_rainbow')

plt.show()
```



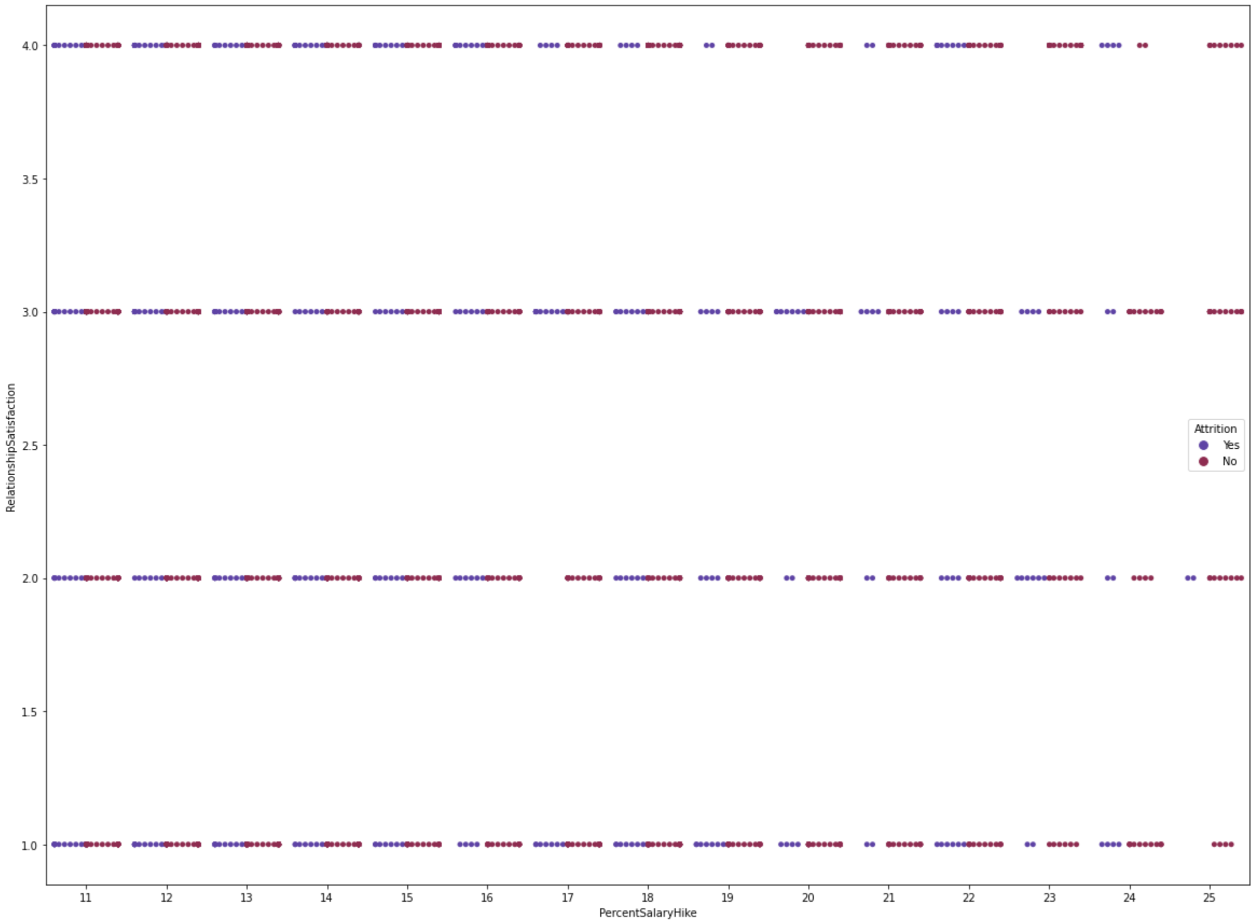
Insight 14

Grpahs in 248 and 249 are showing the replation between attrition and the percent salary hike. Clearly it show, Percent Salary hike between 10-14 shows higher attrition rate and if we see gender wise, male employees leave more than female employees as the graph is very dense for male employees.

One more very interesting pattern seen in sworm plot is -at 22% hike also, employees are showing a higher attrition rate. Although it is not very clear which gender shows greater number.

In [274]:

```
# RelationshipSatisfaction
# Relation between attrtion and Relationship with the manager.

fig = plt.figure(figsize=(20,15))
sns.swarmplot(x='PercentSalaryHike',y='RelationshipSatisfaction',data=data_copy,hue='Attrition',dodge = True,palette='twilight')
plt.show()
```

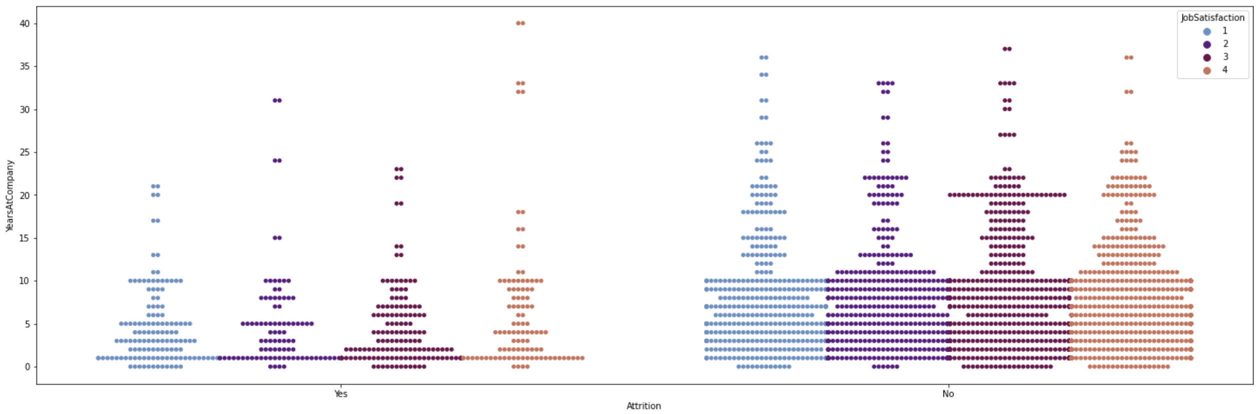


Relationship with the manager and attrition doesn't show much insight.

Tried a lot of permutation combinations, with gender/JobSatisfaction and few other variables and with different kinds of plots, but couldn't find any concrete results.

In [281]:

```
# YearsAtCompany

 # Relation between attrtion and Relationship with the manager.

fig = plt.figure(figsize=(25,8))
sns.swarmplot(y='YearsAtCompany',x='Attrition',data=data_copy,hue='JobSatisfaction',dodge = True,palette='twilight')
plt.show()
```



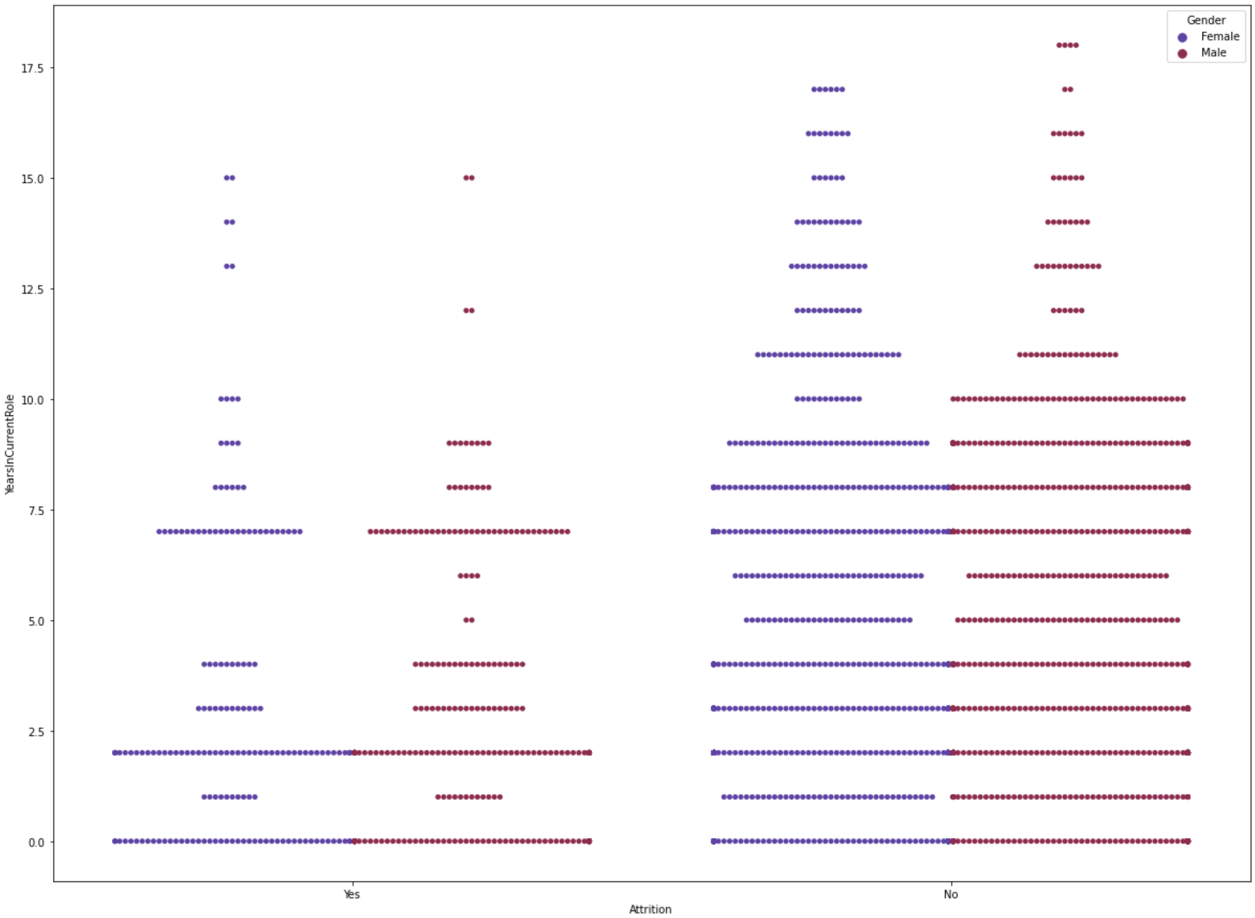
Insight 15

Job Satisfaction level is not at all affecting attrition level when the years in the company is 1. Attrition seems to be higher at that point.

(with others variable at Hue = PercentSalaryHike/Gender/Marital Status::: have been checked. SImilar kind of finding was seen.)

In [289]:

```
# YearsInCurrentRole
# Relation between attrtion and YearsInCurrentRole with the manager.

fig = plt.figure(figsize=(20,15))
sns.swarmplot(y='YearsInCurrentRole',x='Attrition',data=data_copy,hue='Gender',dodge = True,palette='twilight')
plt.show()
```

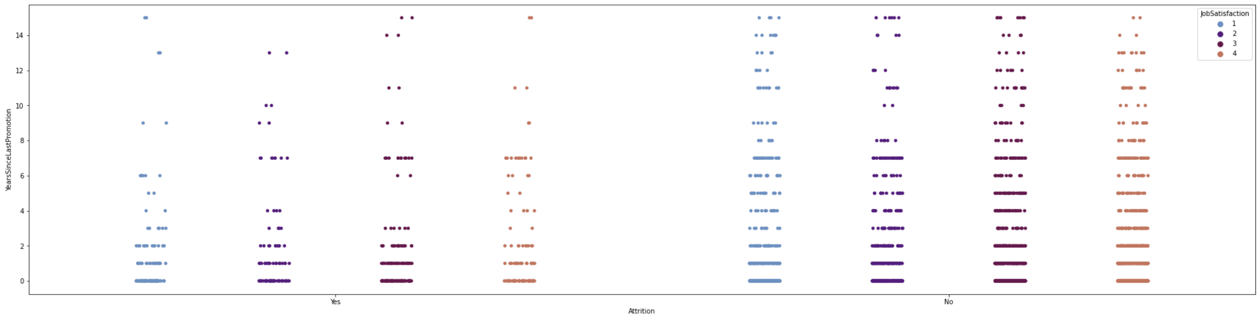


Insight 16

Employees with 0 to 2 years shows more attrition. Both genders shows similar kind of pattern.

In [298]:

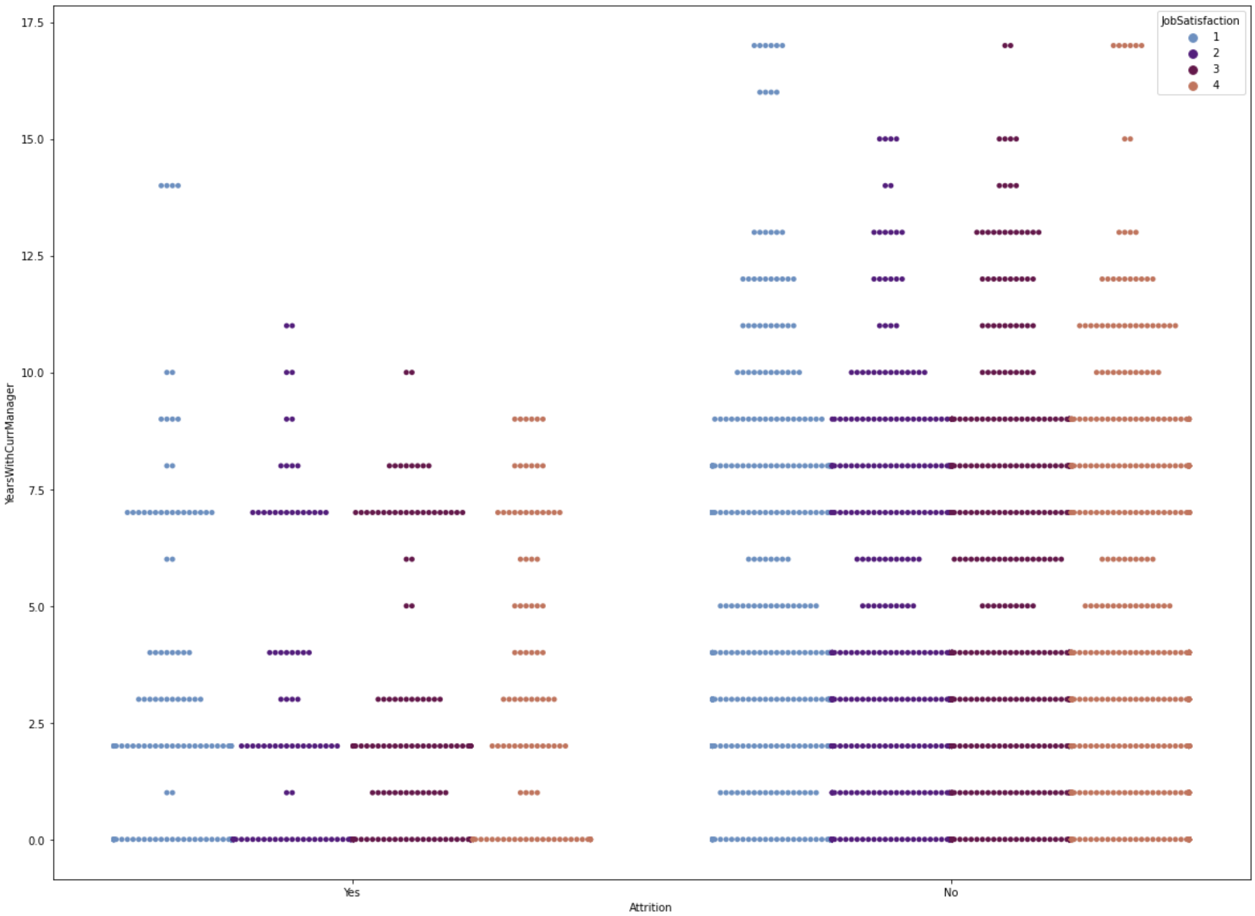
```
# YearsSinceLastPromotion
# Relation between attrtion and Relationship with the manager.

fig = plt.figure(figsize=(33,8))
sns.stripplot(x='Attrition',y='YearsSinceLastPromotion',data=data_copy,hue='JobSatisfaction',dodge = True,palette='twilight')
plt.show()
```



Employees with lesser year since promotion shows higher attrition rate.

In [302]:

```
# YearsWithCurrManager
# Relation between attrtion and Relationship with the manager.

fig = plt.figure(figsize=(20,15))
sns.swarmplot(x='Attrition',y='YearsWithCurrManager',data=data_copy,hue='JobSatisfaction',dodge = True,palette='twilight')
# sns.violinplot(x='Attrition',y='YearsWithCurrManager',data=data_copy,hue='JobSatisfaction',dodge = True,palette='twilight')
# sns.boxplot(x='Attrition',y='YearsWithCurrManager',data=data_copy,hue='JobSatisfaction',dodge = True,palette='twilight')


plt.show()
```



Insight 17

Lesser the years with current Manager, more attrition level is seen. It is very interesting to note that at years with current manger is 7 years, attrition suddenly shoots up. Job Satisfaction level doesn't impact much to the existing trend.

Data Preprocessing

In [9]:

```
# The following columns were deleted. As the values in these columns were constant and doesn't impact the attrition rate in any w
# data_copy.drop(['EmployeeNumber','Over18','StandardHours','EmployeeCount'],axis=1,inplace=True)
data_copy.shape
```

Out[9]:

```
(2940, 31)
```

In [10]:

```
data_copy.columns.tolist()
```

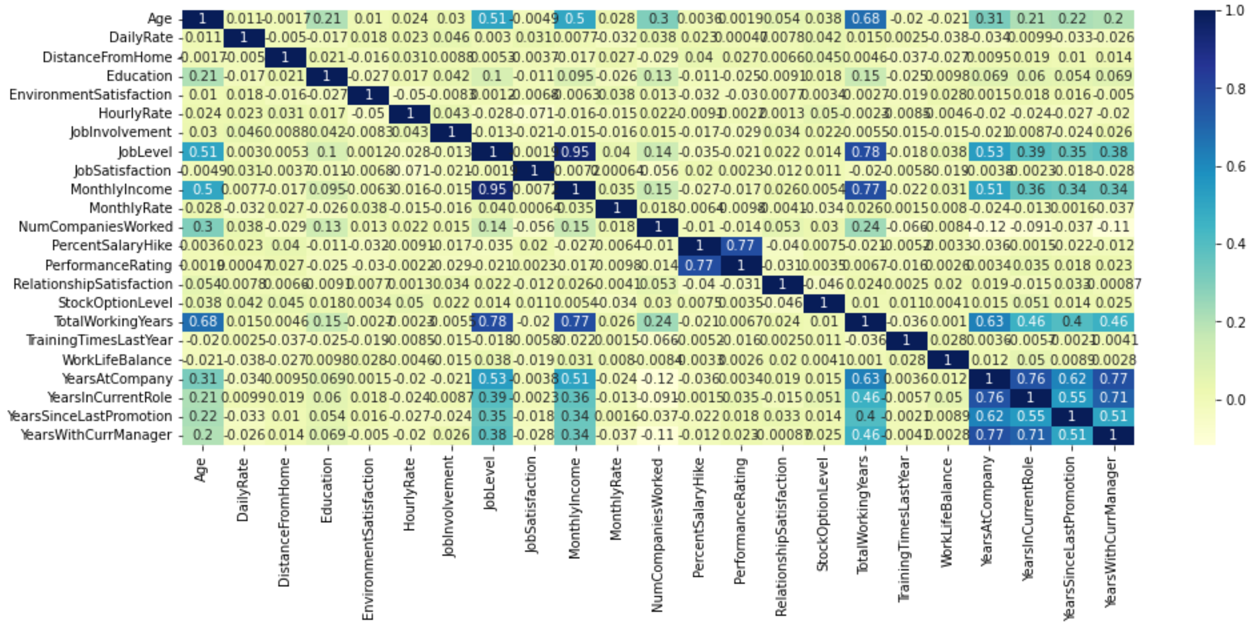
Out[10]:

```
['Age',
 'Attrition',
 'BusinessTravel',
 'DailyRate',
 'Department',
 'DistanceFromHome',
 'Education',
 'EducationField',
 'EnvironmentSatisfaction',
 'Gender',
 'HourlyRate',
 'JobInvolvement',
 'JobLevel',
 'JobRole',
 'JobSatisfaction',
 'MaritalStatus',
 'MonthlyIncome',
 'MonthlyRate',
 'NumCompaniesWorked',
 'OverTime',
 'PercentSalaryHike',
 'PerformanceRating',
 'RelationshipSatisfaction',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'WorkLifeBalance',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

Correlation Matrix

In [11]:

```
cor = data_copy.corr()
# print(cor) #only 23 rows and columns are there because only numerical columns are taken into consideration
fig=plt.figure(figsize=(16,6))
dataplot = sns.heatmap(cor, cmap="YlGnBu", annot=True)
plt.show()
```



Job Level, MonthlyIncome and TotaWorkingYears are highly correlated(greater than 75%). We can easily drop any two columns and can keep one of them.

Droping MonthlyIncome and TotaWorkingYears

In [12]:

```
data_copy.drop(['MonthlyIncome','TotalWorkingYears'],axis=1,inplace=True)
data_copy.shape
```

Out[12]:

```
(2940, 29)
```

Converting Categorical Variables data to Nominal data

In [13]:

```
def labelencoder(df):
    df_copy= df.copy()

    le = preprocessing.LabelEncoder()
    df_copy['Attrition']=le.fit_transform(df_copy['Attrition'])
    df_copy['BusinessTravel']=le.fit_transform(df_copy['BusinessTravel'])
    df_copy['Department']=le.fit_transform(df_copy['Department'])
    df_copy['EducationField']=le.fit_transform(df_copy['EducationField'])
    df_copy['Gender']=le.fit_transform(df_copy['Gender'])
    df_copy['JobRole']=le.fit_transform(df_copy['JobRole'])
    df_copy['MaritalStatus']=le.fit_transform(df_copy['MaritalStatus'])
    df_copy['OverTime']=le.fit_transform(df_copy['OverTime'])
    return df_copy

encoded_data=labelencoder(data_copy)
```

In [14]:

```
encoded_data.head(10)
```

Out[14]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	F
0	41	1	2	1102	2	1	2	1	2	0	...	
1	49	0	1	279	1	8	1	1	3	1	...	
2	37	1	2	1373	1	2	2	4	4	1	...	
3	33	0	1	1392	1	3	4	1	4	0	...	
4	27	0	2	591	1	2	1	3	1	1	...	
5	32	0	1	1005	1	2	2	1	4	1	...	
6	59	0	2	1324	1	3	3	3	3	0	...	
7	30	0	2	1358	1	24	1	1	4	1	...	
8	38	0	1	216	1	23	3	1	4	1	...	
9	36	0	2	1299	1	27	3	3	3	1	...	

10 rows × 29 columns

In [15]:

```
# Dividing the Independent and dependent variable data

y=encoded_data['Attrition'].values
x=encoded_data.drop(['Attrition'],axis=1)
```

In [16]:

```
#Splitting into training and testing dataset

X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.3)
```

In [17]:

```
X_train.shape
```

Out[17]:

```
(2058, 28)
```

Decision Tree

In [18]:

```
from sklearn import tree

dt_HR = tree.DecisionTreeClassifier(max_depth = 5) # Building the decision tree
dt_HR.fit(X_train,Y_train) #Training the model
dt_HR.score(X_test,Y_test) #Predicting the result
```

Out[18]:

```
0.8514739229024944
```

In [19]:

```
y_pred = dt_HR.predict(X_test)
dt_HR.score(X_test,Y_test)
```

Out[19]:

```
0.8514739229024944
```

In [20]:

```
y_pred = dt_HR.predict(X_test)

confusion_matrix(Y_test,y_pred)
```

Out[20]:

```
array([[711,  19],
       [112,  40]], dtype=int64)
```

Building Random Forest

In [21]:

```
model_HR = RandomForestClassifier(n_estimators = 100,random_state = 0)
model_HR.fit(X_train,Y_train)
model_HR_score_train = model_HR.score(X_train,Y_train)
print("Training Score is:",model_HR_score_train)
model_HR_score_test = model_HR.score(X_test,Y_test)
print("Testing Score is :",model_HR_score_test)
```

```
Training Score is: 1.0
Testing Score is : 0.9523809523809523
```

In [22]:

```
#Probability Calculation for calculating Area Under Curve(AUC) value

y_pred_prob = model_HR.predict_proba(X_test)[:, 1]
y_pred_prob
```

Out[22]:

```
array([0.72, 0.16, 0.04, 0.07, 0.52, 0.03, 0.74, 0.07, 0.03, 0.07, 0.36,
       0.01, 0.01, 0.42, 0.14, 0.  , 0.62, 0.01, 0.13, 0.06, 0.01, 0.04,
       0.4 , 0.07, 0.05, 0.05, 0.05, 0.04, 0.09, 0.18, 0.64, 0.1 , 0.  ,
       0.2 , 0.8 , 0.76, 0.1 , 0.04, 0.19, 0.01, 0.03, 0.9 , 0.2 , 0.03,
       0.14, 0.01, 0.02, 0.1 , 0.66, 0.08, 0.17, 0.09, 0.01, 0.73, 0.24,
       0.04, 0.04, 0.1 , 0.11, 0.08, 0.03, 0.14, 0.07, 0.85, 0.1 , 0.14,
       0.11, 0.  , 0.88, 0.02, 0.04, 0.02, 0.04, 0.73, 0.19, 0.01, 0.04,
       0.27, 0.  , 0.07, 0.36, 0.17, 0.18, 0.19, 0.03, 0.08, 0.01, 0.63,
       0.  , 0.03, 0.66, 0.03, 0.05, 0.29, 0.  , 0.02, 0.1 , 0.04, 0.01,
       0.07, 0.14, 0.05, 0.62, 0.03, 0.16, 0.2 , 0.26, 0.72, 0.04, 0.  ,
       0.93, 0.06, 0.05, 0.21, 0.02, 0.08, 0.07, 0.07, 0.12, 0.06, 0.21,
       0.11, 0.08, 0.11, 0.11, 0.77, 0.81, 0.68, 0.1 , 0.83, 0.41, 0.08,
       0.19, 0.69, 0.08, 0.02, 0.09, 0.92, 0.67, 0.02, 0.04, 0.17, 0.07,
       0.  , 0.16, 0.01, 0.03, 0.46, 0.09, 0.22, 0.08, 0.  , 0.05, 0.92,
       0.03, 0.7 , 0.1 , 0.76, 0.11, 0.01, 0.02, 0.04, 0.09, 0.62, 0.03,
       0.15, 0.07, 0.09, 0.04, 0.01, 0.03, 0.1 , 0.36, 0.84, 0.18, 0.06,
       0.  , 0.01, 0.12, 0.76, 0.03, 0.3 , 0.  , 0.04, 0.03, 0.04, 0.03,
       0.05, 0.06, 0.03, 0.03, 0.08, 0.19, 0.06, 0.26, 0.75, 0.1 , 0.09,
       0.06, 0.11, 0.08, 0.  , 0.01, 0.02, 0.06, 0.09, 0.01, 0.07, 0.08,
       0.11, 0.  , 0.02, 0.86, 0.05, 0.07, 0.05, 0.04, 0.03, 0.01, 0.12,
       0.04, 0.3 , 0.  , 0.  , 0.09, 0.05, 0.71, 0.04, 0.75, 0.05, 0.09,
       0.08, 0.1 , 0.  , 0.33, 0.17, 0.61, 0.2 , 0.04, 0.04, 0.03, 0.45,
       0.22, 0.  , 0.1 , 0.02, 0.  , 0.1 , 0.06, 0.74, 0.22, 0.02, 0.17,
       0.03, 0.08, 0.4 , 0.11, 0.67, 0.01, 0.17, 0.07, 0.06, 0.03, 0.01,
       0.01, 0.08, 0.02, 0.06, 0.81, 0.14, 0.04, 0.02, 0.12, 0.07, 0.08,
       0.02, 0.04, 0.  , 0.04, 0.07, 0.06, 0.04, 0.1 , 0.03, 0.18, 0.06,
       0.01, 0.09, 0.  , 0.06, 0.15, 0.  , 0.03, 0.09, 0.06, 0.46, 0.03,
       0.04, 0.77, 0.  , 0.4 , 0.14, 0.09, 0.2 , 0.1 , 0.29, 0.  , 0.03,
       0.07, 0.04, 0.06, 0.09, 0.02, 0.03, 0.03, 0.04, 0.19, 0.05, 0.1 ,
       0.12, 0.06, 0.29, 0.27, 0.05, 0.14, 0.29, 0.11, 0.  , 0.06, 0.29,
       0.39, 0.04, 0.02, 0.05, 0.11, 0.2 , 0.76, 0.01, 0.03, 0.1 , 0.01,
       0.03, 0.05, 0.24, 0.01, 0.05, 0.02, 0.06, 0.69, 0.06, 0.  , 0.19,
       0.  , 0.  , 0.14, 0.04, 0.04, 0.79, 0.01, 0.03, 0.11, 0.77, 0.63,
       0.03, 0.  , 0.05, 0.07, 0.06, 0.24, 0.78, 0.23, 0.75, 0.67, 0.63,
       0.07, 0.1 , 0.06, 0.1 , 0.01, 0.01, 0.05, 0.19, 0.29, 0.11, 0.06,
       0.01, 0.14, 0.73, 0.04, 0.12, 0.07, 0.06, 0.  , 0.03, 0.24, 0.14,
       0.09, 0.65, 0.09, 0.13, 0.03, 0.04, 0.18, 0.75, 0.83, 0.33, 0.06,
       0.04, 0.03, 0.78, 0.1 , 0.33, 0.74, 0.33, 0.  , 0.08, 0.18, 0.36,
       0.04, 0.73, 0.09, 0.01, 0.21, 0.02, 0.04, 0.09, 0.26, 0.1 , 0.03,
       0.03, 0.05, 0.18, 0.85, 0.81, 0.09, 0.76, 0.02, 0.12, 0.01, 0.15,
       0.05, 0.08, 0.06, 0.78, 0.02, 0.84, 0.75, 0.05, 0.02, 0.08, 0.24,
       0.05, 0.  , 0.08, 0.01, 0.75, 0.02, 0.04, 0.02, 0.46, 0.05, 0.23,
       0.03, 0.2 , 0.06, 0.04, 0.05, 0.22, 0.02, 0.68, 0.11, 0.01, 0.03,
       0.26, 0.07, 0.05, 0.06, 0.85, 0.03, 0.06, 0.77, 0.13, 0.02, 0.11,
       0.05, 0.06, 0.01, 0.  , 0.01, 0.16, 0.08, 0.09, 0.08, 0.04, 0.  ,
       0.04, 0.06, 0.08, 0.05, 0.35, 0.08, 0.08, 0.03, 0.04, 0.06, 0.37,
       0.09, 0.1 , 0.09, 0.01, 0.84, 0.  , 0.79, 0.13, 0.03, 0.04, 0.11,
       0.93, 0.22, 0.  , 0.76, 0.01, 0.62, 0.07, 0.07, 0.09, 0.04, 0.02,
       0.06, 0.65, 0.1 , 0.03, 0.01, 0.01, 0.01, 0.52, 0.12, 0.03, 0.03,
       0.02, 0.16, 0.36, 0.19, 0.06, 0.  , 0.33, 0.01, 0.05, 0.68, 0.01,
       0.73, 0.02, 0.16, 0.07, 0.75, 0.02, 0.03, 0.66, 0.06, 0.19, 0.07,
       0.41, 0.01, 0.03, 0.02, 0.04, 0.18, 0.04, 0.01, 0.73, 0.02, 0.1 ,
       0.01, 0.1 , 0.05, 0.13, 0.04, 0.03, 0.06, 0.33, 0.84, 0.07, 0.05,
       0.11, 0.08, 0.69, 0.02, 0.05, 0.01, 0.13, 0.13, 0.08, 0.08, 0.81,
       0.01, 0.02, 0.02, 0.22, 0.02, 0.02, 0.  , 0.43, 0.16, 0.  , 0.05,
       0.02, 0.  , 0.08, 0.03, 0.61, 0.  , 0.03, 0.06, 0.4 , 0.82, 0.68,
       0.37, 0.01, 0.13, 0.02, 0.01, 0.06, 0.03, 0.08, 0.  , 0.11, 0.02,
       0.08, 0.03, 0.17, 0.  , 0.14, 0.02, 0.57, 0.26, 0.06, 0.03, 0.09,
       0.03, 0.14, 0.08, 0.27, 0.12, 0.07, 0.19, 0.02, 0.07, 0.14, 0.85,
       0.04, 0.77, 0.77, 0.03, 0.06, 0.04, 0.05, 0.42, 0.06, 0.08, 0.14,
       0.16, 0.73, 0.92, 0.81, 0.03, 0.64, 0.01, 0.03, 0.02, 0.01, 0.03,
       0.01, 0.01, 0.02, 0.1 , 0.2 , 0.03, 0.14, 0.01, 0.01, 0.05, 0.25,
       0.07, 0.04, 0.39, 0.03, 0.43, 0.04, 0.12, 0.01, 0.03, 0.08, 0.03,
       0.  , 0.08, 0.35, 0.16, 0.02, 0.1 , 0.91, 0.18, 0.26, 0.  , 0.12,
       0.1 , 0.1 , 0.14, 0.12, 0.05, 0.8 , 0.78, 0.03, 0.  , 0.2 , 0.05,
       0.03, 0.01, 0.01, 0.  , 0.01, 0.13, 0.01, 0.06, 0.24, 0.07, 0.  ,
       0.08, 0.03, 0.06, 0.14, 0.03, 0.76, 0.64, 0.04, 0.08, 0.04, 0.07,
       0.02, 0.05, 0.62, 0.09, 0.14, 0.13, 0.08, 0.07, 0.17, 0.02, 0.71,
       0.73, 0.04, 0.02, 0.01, 0.01, 0.09, 0.  , 0.02, 0.  , 0.06, 0.04,
       0.01, 0.02, 0.01, 0.04, 0.02, 0.8 , 0.05, 0.01, 0.14, 0.03, 0.01,
       0.01, 0.02, 0.73, 0.21, 0.01, 0.01, 0.05, 0.65, 0.19, 0.1 , 0.05,
       0.1 , 0.08, 0.  , 0.05, 0.03, 0.02, 0.09, 0.01, 0.76, 0.11, 0.02,
       0.08, 0.11, 0.08, 0.25, 0.04, 0.66, 0.8 , 0.17, 0.19, 0.03, 0.01,
       0.03, 0.11, 0.07, 0.05, 0.02, 0.01, 0.06, 0.22, 0.03, 0.01, 0.04,
       0.1 , 0.12, 0.01, 0.03, 0.03, 0.12, 0.16, 0.1 , 0.01, 0.07, 0.08,
       0.02, 0.45, 0.64, 0.01, 0.  , 0.01, 0.19, 0.01, 0.57, 0.1 , 0.46,
       0.06, 0.18, 0.05, 0.04, 0.  , 0.04, 0.  , 0.06, 0.04, 0.05, 0.04,
       0.13, 0.09, 0.76, 0.01, 0.08, 0.05, 0.02, 0.08, 0.03, 0.03, 0.05,
       0.03, 0.03, 0.36, 0.81, 0.7 , 0.27, 0.22, 0.18, 0.01, 0.21, 0.02,
       0.16, 0.09, 0.15, 0.  , 0.68, 0.73, 0.03, 0.05, 0.03, 0.03, 0.84,
       0.03, 0.02])
```

In [23]:

```
#Predicting the Attrition for X_test

y_pred = model_HR.predict(X_test)
y_pred
```

Out[23]:

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

In [24]:

```
#Calculating the Model Accuracy

model_accu = (model_HR.score(X_test,Y_test))*100
recall_val = (recall_score(Y_test,y_pred))*100

print("Model accuracy is -: ",model_accu)
print("Model Recall Value is -:",recall_val)
```

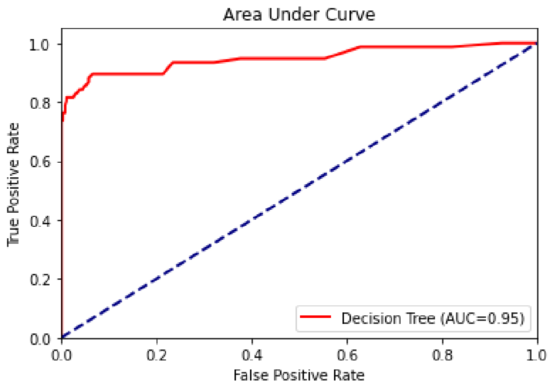
```
Model accuracy is -:  95.23809523809523
Model Recall Value is -: 73.68421052631578
```

In [25]:

```
#Area under Curve
# Flase positive rate, true positive rate calculation

fpr_dt, tpr_dt, _ =roc_curve(Y_test,y_pred_prob)
roc_auc_dt = auc(fpr_dt,tpr_dt)
```

In [26]:

```
plt.figure(1)
lw=2
plt.plot(fpr_dt,tpr_dt,color='red',lw=lw,
         label="Decision Tree (AUC=%0.2f)"%roc_auc_dt)
plt.plot([0,1],[0,1],color='navy',lw=lw,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("Area Under Curve")
plt.legend(loc=0)
plt.show()
```



In [27]:

```
print(confusion_matrix(Y_test,y_pred))
```

```
[[728   2]
 [ 40 112]]
```

Building Naive Bayes Classifier

In [28]:

```
from sklearn.naive_bayes import GaussianNB
nb_HR = GaussianNB()
nb_HR.fit(X_train,Y_train)
nb_HR.score(X_test,Y_test)
```

Out[28]:

```
0.8378684807256236
```

Building K-Nearest Classifier

In [29]:

```
from sklearn.neighbors import KNeighborsClassifier
knn_HR = KNeighborsClassifier(n_neighbors = 3)
knn_HR.fit(X_train,Y_train)
knn_HR.score(X_test,Y_test)
```

Out[29]:

```
0.7902494331065759
```

Building Logistic Regression Classifier

In [30]:

```
from sklearn.linear_model import LogisticRegression
lr_HR = LogisticRegression()
lr_HR.fit(X_train,Y_train)
lr_HR.score(X_test,Y_test)
```

Out[30]:

```
0.828798185941043
```

Building SVM CLassifier

In [ ]:

```
from sklearn.svm import SVC

sv_HR = SVC(probability=True, kernel='linear')
sv_HR.fit(X_train,Y_train)
sv_HR.score(X_test,Y_test)
```

Final Insight - Random Forest Model

On seeing all the models accuracy, Random forest gives the best result and accuracy.

So, we will go with the Random Forest Model.

(SVM took the longest time in execution. Rest all models executed in very less time.)