DIC-Project Phase -1

mbollepa-50441493

girijapo-50443049

Dataset name: HR_Employee_attrition.

PROBLEM STATEMENT:

Background of the problem:

For any company customers play a crucial role for the business to grow. But employees play the most crucial part because without employees there is no good future for the company. It is often seen that many employees tend to leave their current company due to various reasons such as poor work life balance, bored of current technology they are working on, or less promotions and some more.

So, we chose a dataset of a company that contains thousand employees, and it is noted that every year a few percentages of the employees are leaving the company.

Our main objective is to analyze what percentage of employees are leaving and the reasons why they are quitting their jobs. The data we have may help us to analyze and find the employee is likely to quit the job or not. So, if the reasons are known then we can atleast try not let this happen or reduce the percentage of employees leaving the company.

Contribution to the problem:

Employees are a great asset for the company and losing them would have many negative consequences for any company. For now, we took a dataset for just one company and applied few analyses to find out reasons for employees quitting. This analysis can be applied to many companies, and this can be great help since companies now have an idea as they will have the data of reasons why they are quitting. Now, companies can come up with strategies to stop this and reduce the percentage of their employees leaving.

DATA SOURCE:

Our dataset contains 4000 employees and various columns to analyze for employee attrition. We obtained this dataset from an online source named Kaggle.

These are the attributes of our dataset,

```
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
```

Libraries Importing and Loading of Data:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

```
data.head(10)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipS
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	32	No	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	8	...	
6	59	No	Travel_Rarely	1324	Research & Development	3	3	Medical	1	10	...	
7	30	No	Travel_Rarely	1358	Research & Development	24	1	Life Sciences	1	11	...	
8	38	No	Travel_Frequently	216	Research & Development	23	3	Life Sciences	1	12	...	
9	36	No	Travel_Rarely	1299	Research & Development	27	3	Medical	1	13	...	

The below figure outputs shape of the data.

```
In [4]: data.shape
        #the data consits of 1470 rows and 35 columns

Out[4]: (1470, 35)
```

The below figure describes the data

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

Out[5]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	14
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	

The below figure shows the datatypes of the attributes

```
In [6]: data.dtypes
#checking how many are numeric and how many are categorical
```

```
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
```

PRE-PROCESSING METHODS:

1. **Dropping unnecessary columns:**
   The first method we used is drop by which we can drop the unnecessary columns in the dataset if any exists. In below code we have dropped ‘Over18’ feature and shown it.

```
In [46]: cols = ['Over18']
         data1 = data.drop(cols, axis=1)
         #droppin unnecessary column in data.
```

```
In [47]: data1.head(1)
         #checking if its done or not.
```

Out[47]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	

1 rows × 34 columns

2. Dropping and Detecting NULL values :

Next step is to detect if any null values are present or not and then drop the null values.

```
In [48]: data1 = data1.dropna()
         #Dropping null valuesjust incase if any exists
```

```
In [10]: data1.isnull().sum()
         #checking if dataset consists of any null values or not
         #if present they can be either removed or filled.
         #observed that there are no null values
```

```
Out[10]: Age                         0
         Attrition                   0
         BusinessTravel              0
         DailyRate                   0
         Department                  0
         DistanceFromHome            0
         Education                   0
         EducationField              0
         EmployeeCount               0
         EmployeeNumber              0
         EnvironmentSatisfaction     0
         Gender                      0
         HourlyRate                  0
         JobInvolvement              0
         JobLevel                    0
         JobRole                     0
         JobSatisfaction             0
         MaritalStatus               0
         MonthlyIncome               0
         MonthlyRate                 0
         NumCompaniesWorked          0
         OverTime                    0
         PercentSalaryHike           0
         PerformanceRating           0
         RelationshipSatisfaction    0
         StandardHours               0

     StockOptionLevel            0
     TotalWorkingYears           0
     TrainingTimesLastYear       0
     WorkLifeBalance             0
     YearsAtCompany              0
     YearsInCurrentRole          0
     YearsSinceLastPromotion     0
     YearsWithCurrManager        0
```

As, our data didn't contain any null value we have not dropped any. But also, we applied the step of dropping null so that if atleast 1 exists it'll we dropped away.

3. Detecting for Duplicacy:

Next step is to check for duplicacy for this we use duplicated method.

```
In [12]: data1.duplicated()
         #checking for duplication.
```

```
Out[12]: 0       False
         1       False
         2       False
         3       False
         4       False
                 ...
         1465    False
         1466    False
         1467    False
         1468    False
         1469    False
         Length: 1470, dtype: bool
```

Observed that there was no duplicate value in our dataset.

4. Renaming:

Next step is we renamed an attribute. The below figure shows renaming an attribute Employee number with id.

```
data1.rename({'EmployeeNumber': 'id'}, axis=1, inplace=True)
data1.head(10)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	id	...	RelationshipSatisfaction	S
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	
5	32	No	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	8	...	3	
6	59	No	Travel_Rarely	1324	Research & Development	3	3	Medical	1	10	...	1	
7	30	No	Travel_Rarely	1358	Research & Development	24	1	Life Sciences	1	11	...	2	
8	38	No	Travel_Frequently	216	Research & Development	23	3	Life Sciences	1	12	...	2	
9	36	No	Travel_Rarely	1299	Research & Development	27	3	Medical	1	13	...	2	

5. Skewing:

For the next step we used skewing method. Skewing is used to (fill it**)**

```
Age                         0.413286
DailyRate                  -0.003519
DistanceFromHome            0.958118
Education                  -0.289681
EmployeeCount               0.000000
id                          0.016574
EnvironmentSatisfaction    -0.321654
HourlyRate                 -0.032311
JobInvolvement             -0.498419
JobLevel                    1.025401
JobSatisfaction            -0.329672
MonthlyIncome               1.369817
MonthlyRate                 0.018578
NumCompaniesWorked          1.026471
PercentSalaryHike           0.821128
PerformanceRating           1.921883
RelationshipSatisfaction   -0.302828
StandardHours               0.000000
StockOptionLevel            0.968980
TotalWorkingYears           1.117172
TrainingTimesLastYear       0.553124
WorkLifeBalance            -0.552480
YearsAtCompany              1.764529
YearsInCurrentRole          0.917363
YearsSinceLastPromotion     1.984290
YearsWithCurrManager        0.833451
dtype: float64
```

I even observed skewness of 1 particularly feature name "MonthlyIncome" and observed that the skewness is rightmost for it. This feature even showed outliers

6. **Splitting data to numerical column and categorical column:**
   The next step we did was – we split the data into 2 types one is categorical column and the other is numerical column. In the below figure we can the list of categorical columns and numerical columns.

```
In [15]: categorical_cols=data1.select_dtypes(include=['object']).columns
         numerical_cols =data1.select_dtypes(include=np.number).columns.tolist()

In [16]: categorical_cols
         #List of categorical columns

Out[16]: Index(['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender',
                'JobRole', 'MaritalStatus', 'OverTime'],
               dtype='object')

In [17]: numerical_cols
         #List of numeric columns

Out[17]: ['Age',
          'DailyRate',
          'DistanceFromHome',
          'Education',
          'EmployeeCount',
          'id',
          'EnvironmentSatisfaction',
          'HourlyRate',
          'JobInvolvement',
          'JobLevel',
          'JobSatisfaction',
          'MonthlyIncome',
          'MonthlyRate',

         'NumCompaniesWorked',
         'PercentSalaryHike',
         'PerformanceRating',
         'RelationshipSatisfaction',
         'StandardHours',
         'StockOptionLevel',
         'TotalWorkingYears',
         'TrainingTimesLastYear',
         'WorkLifeBalance',
         'YearsAtCompany',
         'YearsInCurrentRole',
         'YearsSinceLastPromotion',
         'YearsWithCurrManager']
```

7. **Label Encoding:**

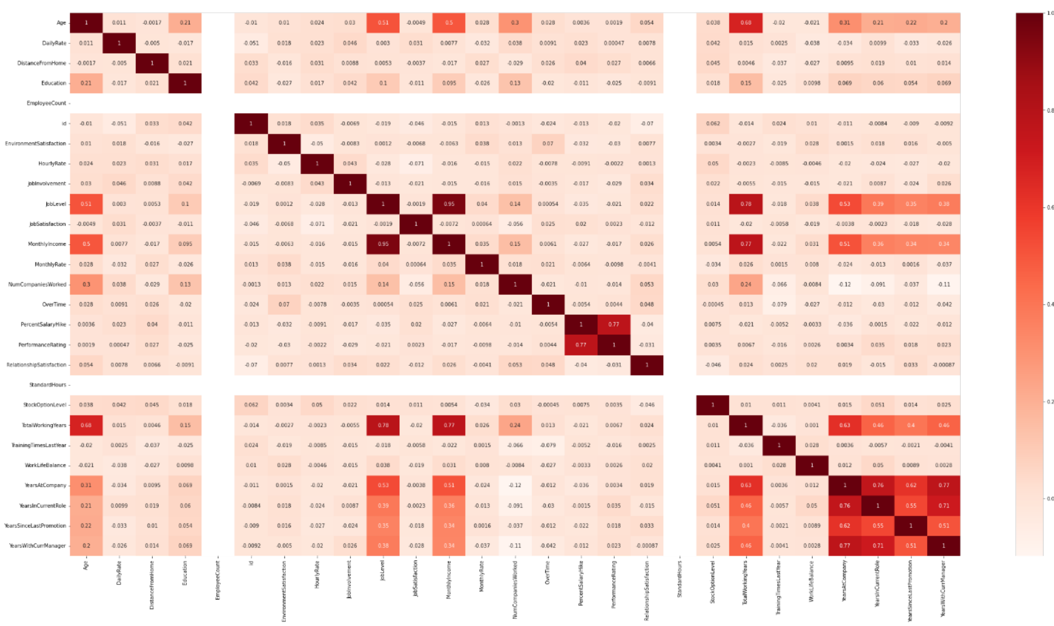
The next we did was that we converted categorical feature into a numerical feature using label encoder. Here we had a feature name ‘OverTime’ which consisted of yes/no using label encoding we have transformed it to numerical feature.

In [18]:
```
from sklearn.preprocessing import LabelEncoder
class_le = LabelEncoder()
data1['OverTime'] = class_le.fit_transform(data1['OverTime'].values)
#converting categorical feature to numerical using labelconding.
```

8. Checking for correction features:

The next step we did is we checked for existence of any correlation, and we observed that very few exists.

In [20]:
```
c=data1.corr()
plt.figure(figsize=(40,20))
sns.heatmap(c, annot=True, cmap="Reds")
plt.show()
#checking for correlation and observed in few
```



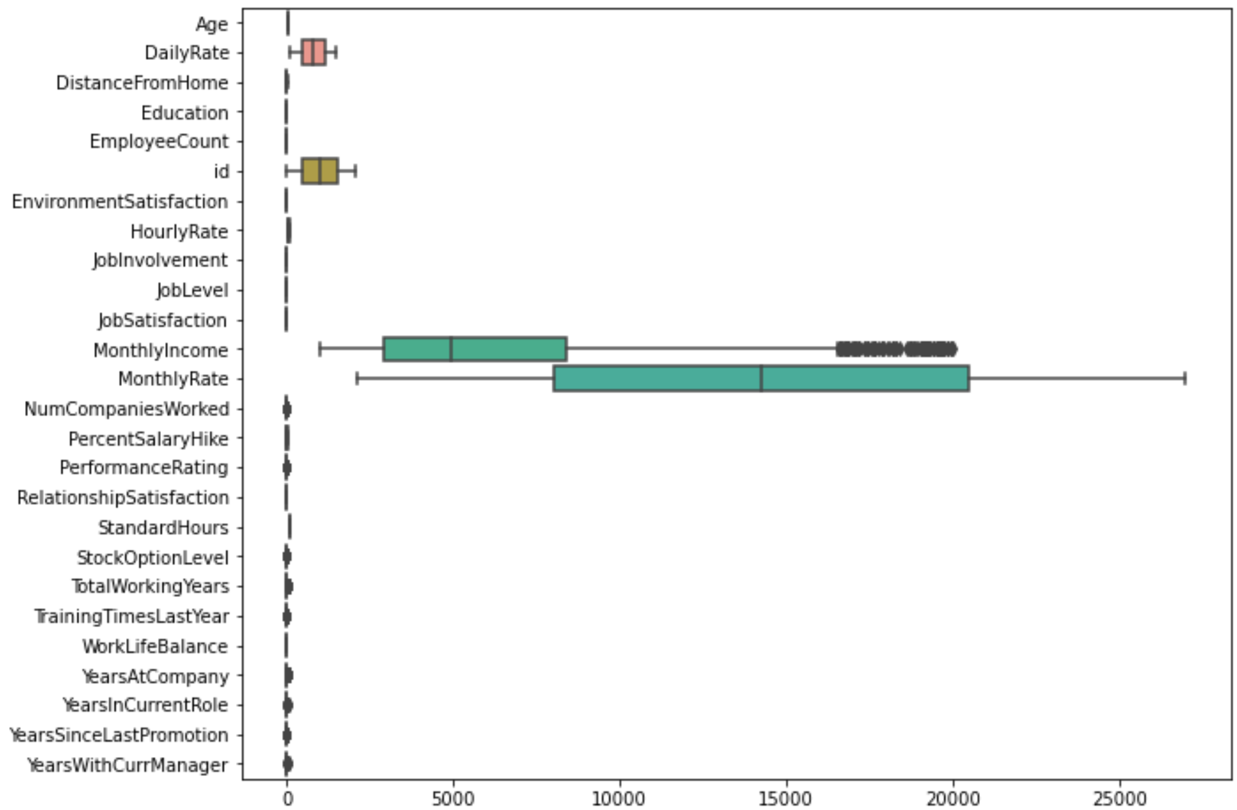
MonthlyIncome and JobLevel are most correlated features as observed.

9. Checking for Outliers:

The next step in the process was we checked for outliers and from the analysis it is observed that monthly income is impacted mostly.

In [21]:
```
plt.figure(figsize=(10,8))
sns.boxplot(data=data1[numerical_cols], orient="h")
#checking for outliers using boxplot for numerical columns and seen that monthly income is most impacted.
```

Out[21]: <AxesSubplot:>



As we see the boxplot of all numerical features, here it was observed that Monthly income has outliers and in next step we have done outlier treatment for this certain column.

10. Z-Score treatment:

The next step we did was Z-Score treatment for monthly income column as mentioned above.

In [22]:

```
print("Highest allowed",data1['MonthlyIncome'].mean() + 3*data1['MonthlyIncome'].std())
print("Lowest allowed",data1['MonthlyIncome'].mean() - 3*data1['MonthlyIncome'].std())
```

```
Highest allowed 20626.80164181099
Lowest allowed -7620.939056776979
```

zscore treatment

```
upper_limit = data1['MonthlyIncome'].mean() + 3*data1['MonthlyIncome'].std()
lower_limit = data1['MonthlyIncome'].mean() - 3*data1['MonthlyIncome'].std()
```

```
data1['MonthlyIncome'] = np.where(
    data1['MonthlyIncome']>upper_limit,
    upper_limit,
    np.where(
        data1['MonthlyIncome']<lower_limit,
        lower_limit,
        data1['MonthlyIncome']
    )
)
```

```
data1['MonthlyIncome'].describe()
```

```
count     1470.000000
mean      6502.931293
std       4707.956783
min       1009.000000
25%       2911.000000
50%       4919.000000
75%       8379.000000
max      19999.000000
Name: MonthlyIncome, dtype: float64
```

11. Unique variable in categorical columns:

The next step we did is to check for unique variables in categorical columns

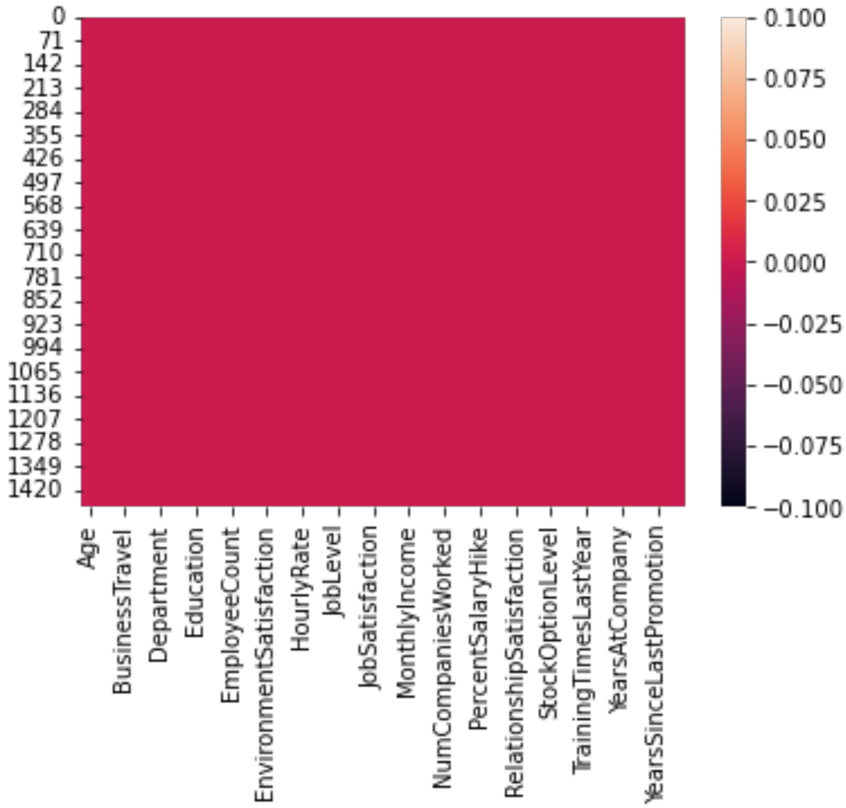
```
In [28]: data1['BusinessTravel'].unique()
         #checking what are unique in BusinessTrave

Out[28]: array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)

In [29]: data1['Attrition'].unique()

Out[29]: array(['Yes', 'No'], dtype=object)

In [30]: data1['EducationField'].unique()
         #checking what are unique

Out[30]: array(['Life Sciences', 'Other', 'Medical', 'Marketing',
                'Technical Degree', 'Human Resources'], dtype=object)

In [31]: data1['Department'].unique()
         #checking what are unique

Out[31]: array(['Sales', 'Research & Development', 'Human Resources'], dtype=object)

In [32]: data1['MaritalStatus'].unique()

Out[32]: array(['Single', 'Married', 'Divorced'], dtype=object)
```

So based on the uniqueness over here we have drawn a plot for most variable holding feature.

VISUALISATION:

1.Heatmap for NULL:

```
In [33]: sns.heatmap(data1.isnull(), cbar=True)

Out[33]: <AxesSubplot:>
```



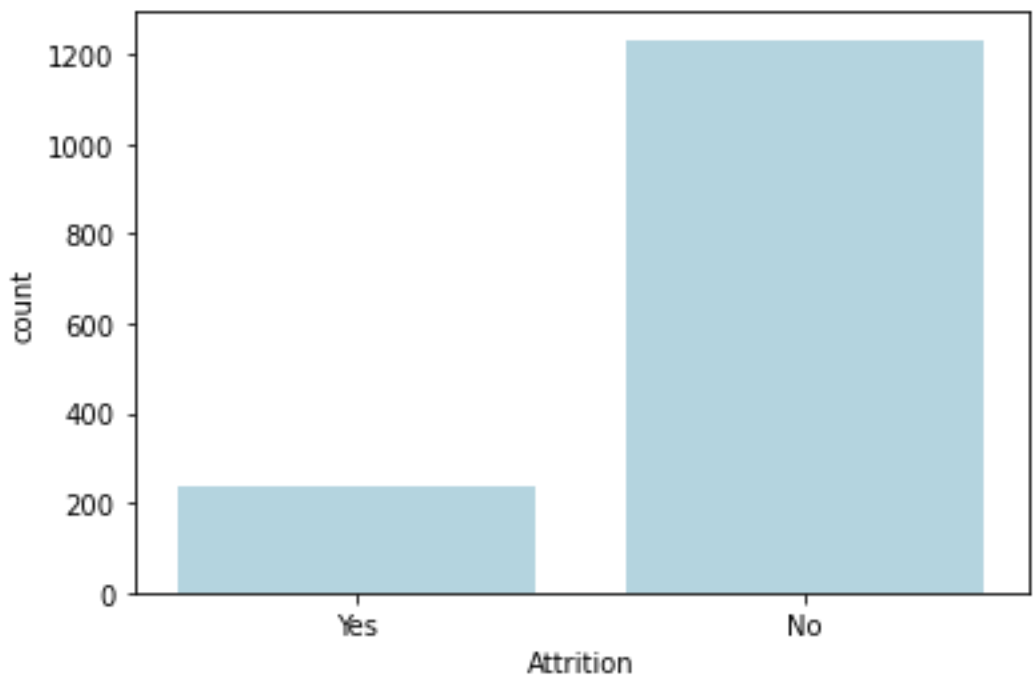
2.Correlation Plot:

We used heatmap for correlation plot. Refer above figure.

3.Count plot:

Next, we did the count plot to check how many employees are leaving the company and from the plot we observed that less than 400 are in line of leaving.

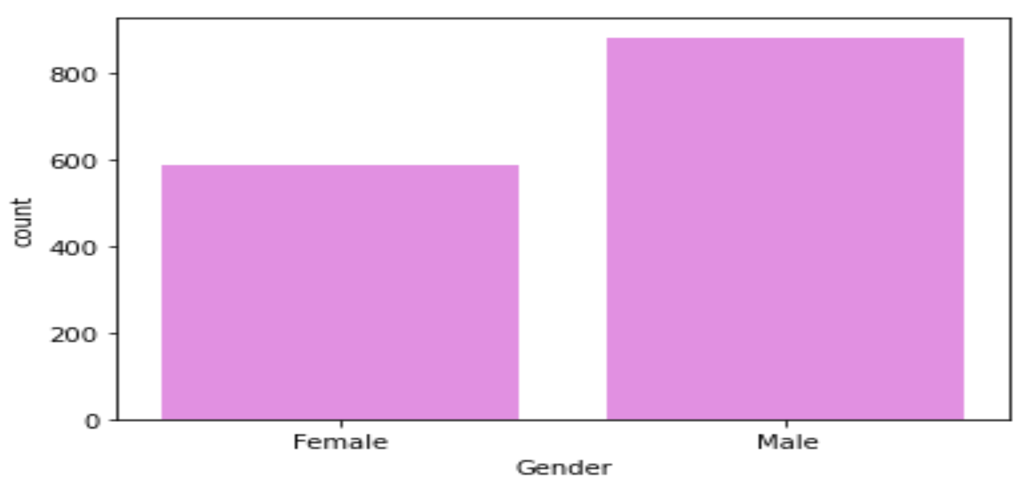
```
In [34]: sns.countplot(data1['Attrition'],color = 'lightblue')
         #checking how many are leaving company and observed less than 400 are in line of leaving.

C:\Users\Girija Polamreddy\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

Out[34]: <AxesSubplot:xlabel='Attrition', ylabel='count'>
```



3.Bar Plot:

Next, we did barplot to find gender of the most of employees working and we observed that most of them male.

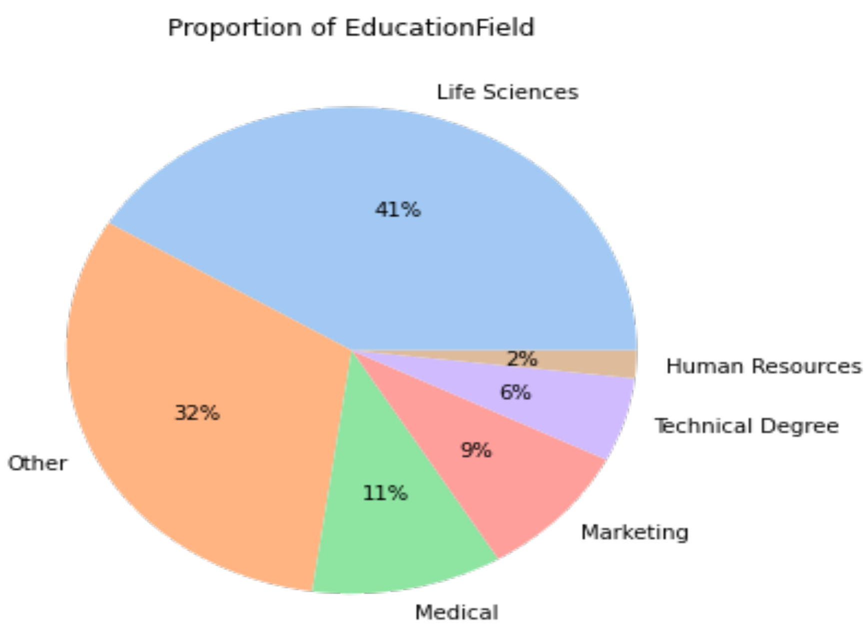
```
In [35]: data1['Gender'].unique()

Out[35]: array(['Female', 'Male'], dtype=object)

In [36]: sns.countplot(data1['Gender'],color = 'violet')
         #most of them are male working.

C:\Users\Girija Polamreddy\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

Out[36]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```



4.Pie Chart:

Next, we made a piechart to check the background of employees and observed that almost 41% employees are from Life Science background and Human Resource background was the least.

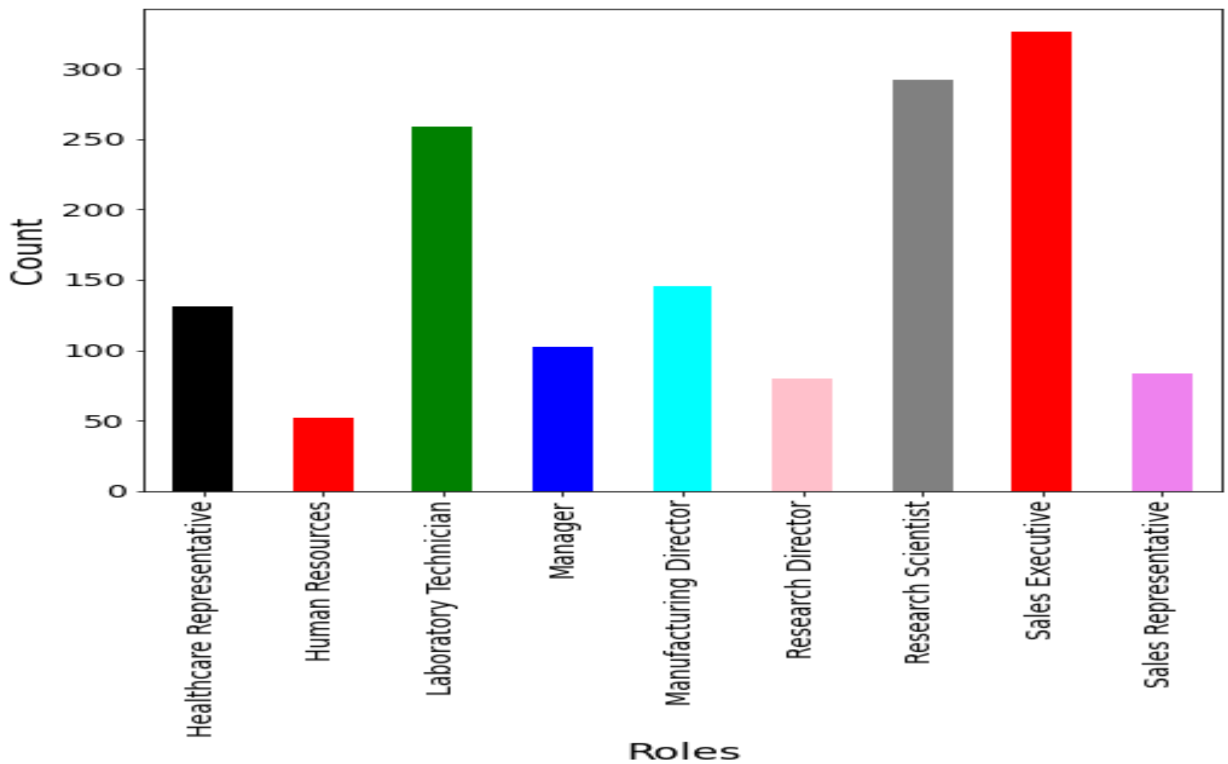
```
In [37]: plt.figure(figsize=[6,6])

         #Define column to use
         E = data1["EducationField"].value_counts(normalize=True)

         #Define labels
         labels = ['Life Sciences', 'Other', 'Medical', 'Marketing','Technical Degree', 'Human Resources']

         #Define color palette
         colors = sns.color_palette('pastel')

         #Create pie chart
         plt.pie(E,labels=labels,colors=colors, autopct='%.0f%%')
         plt.title("Proportion of EducationField")
         plt.show()
         #from this observed that 41 % are from Life science education background and least are from Human Resource background.
```



5.Plot for Job Role:

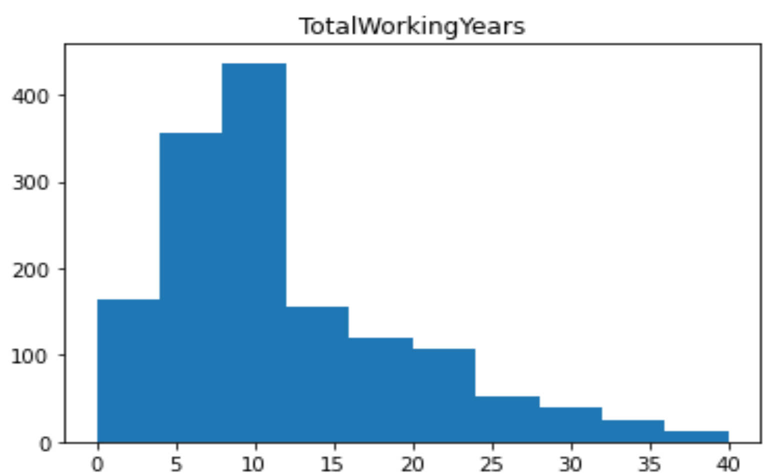
Next, we plotted for Job Role and from the plot we observed that most of them are sales executive employees.

```
In [38]: data1['JobRole'].unique()

Out[38]: array(['Sales Executive', 'Research Scientist', 'Laboratory Technician',
                'Manufacturing Director', 'Healthcare Representative', 'Manager',
                'Sales Representative', 'Research Director', 'Human Resources'],
               dtype=object)

In [39]: b=data1['JobRole'].value_counts().sort_index().plot(kind='bar', fontsize=14, figsize=(8,7),color=['black', 'red', 'green', 'blue'
         b.set_xlabel('Roles', fontsize=18)
         b.set_ylabel('Count', fontsize=18);
         #There are many from Sales executive employees.
```



6.Hist:

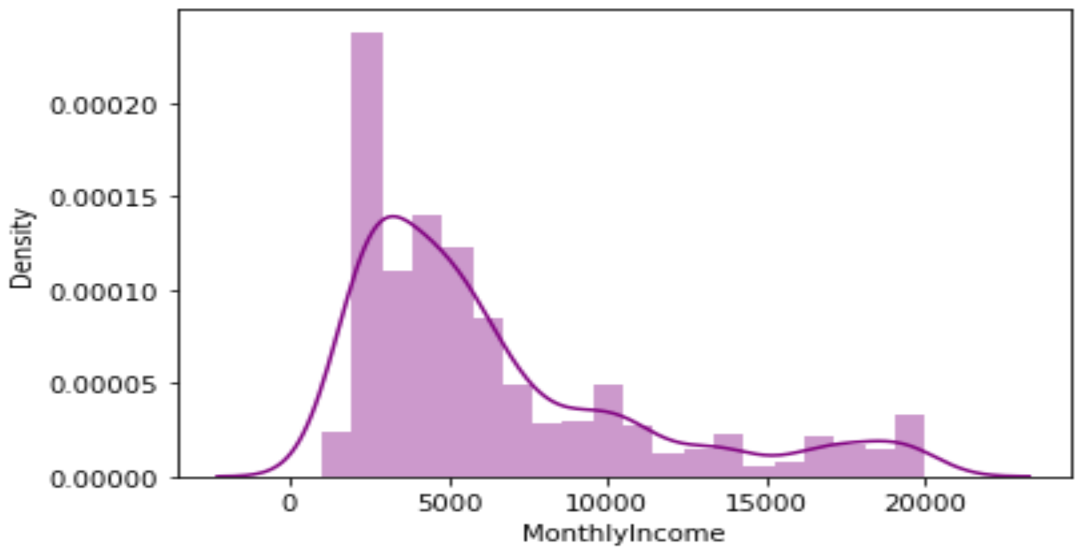
```
In [40]: plt.hist(data1['TotalWorkingYears'])
         plt.title("TotalWorkingYears")
         plt.show()
         #here we can observe that many less people have much workexperience, most of them have only 15years of experience
```



Most of them are working with more that 15 years of experience as observed.

7.Distplot for hist:

```
In [41]: sns.distplot(data1['MonthlyIncome'],color='purple')

C:\Users\Girija Polamreddy\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

Out[41]: <AxesSubplot:xlabel='MonthlyIncome', ylabel='Density'>
```

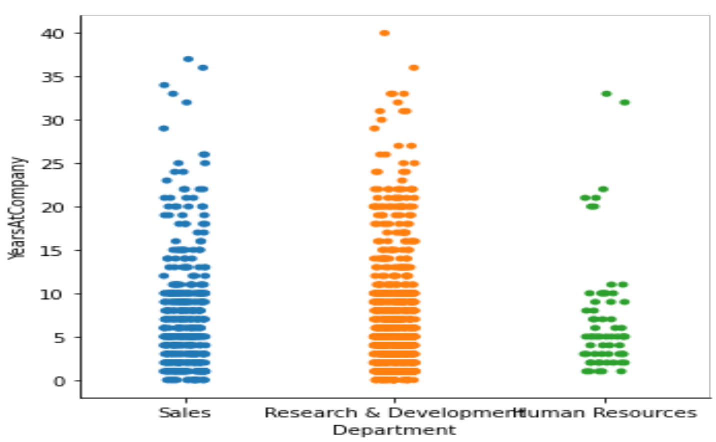


The distribution of income is observed.

8.Catplot:

Next, we used catplot to check employees from which department served most at company and it is observed that it is research and sales department.

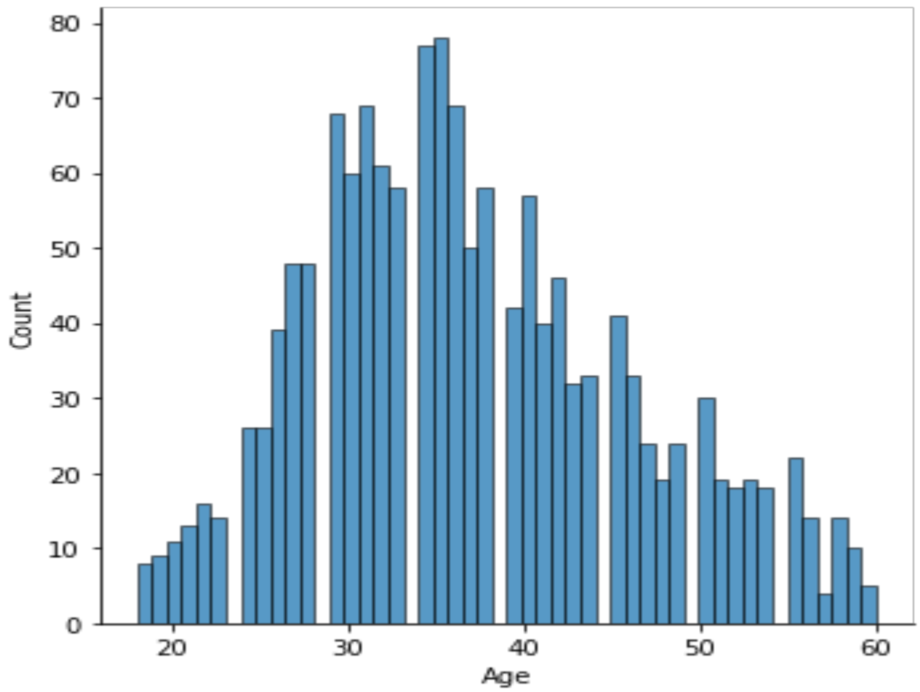
```
In [42]: sns.catplot(x='Department', y = 'YearsAtCompany', data = data1)
         #research &Sales department served most at firm is observed.
```



9.Density Plot:

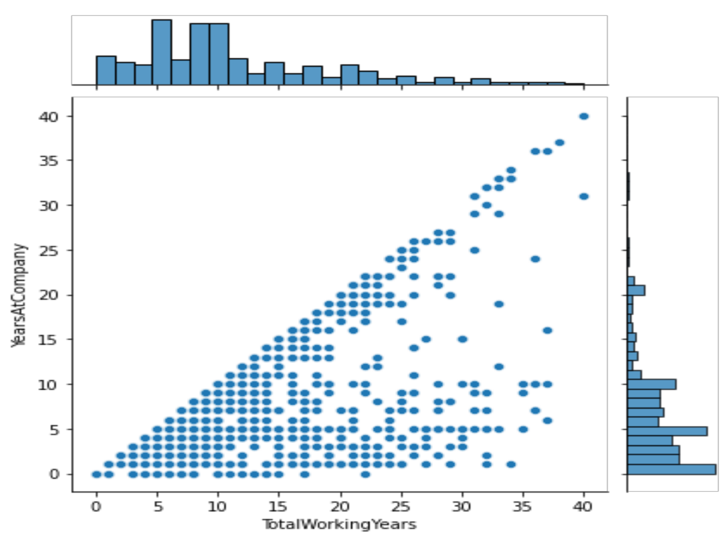
Next, we did density plot to find the range of age of employees and it is observed to be mid30-40.

```
In [43]: sns.displot(data1.Age, bins=50, kde=False)
         plt.show()
         #observed that most number of employess are mid 30-40
```



10.Joint Plot:

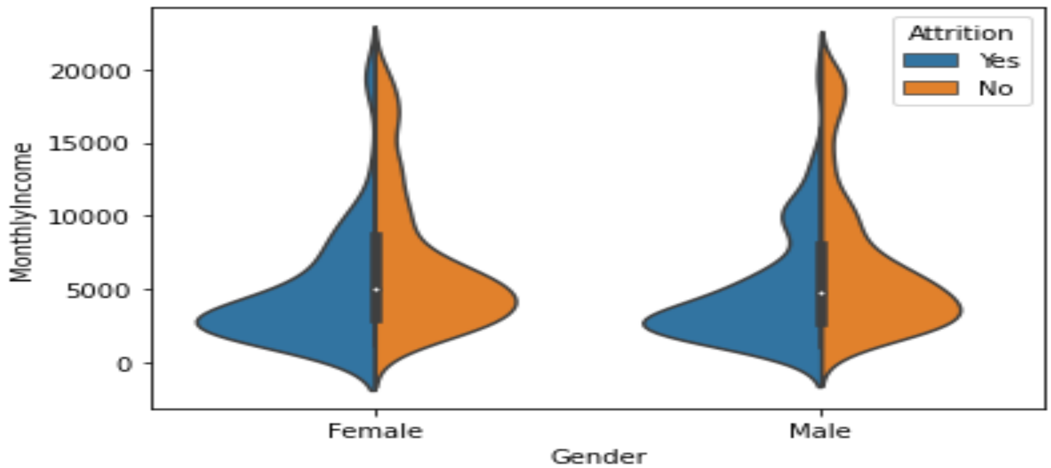
```
In [44]: sns.jointplot(x=data1["TotalWorkingYears"], y=data1["YearsAtCompany"], kind='scatter')
         plt.show()
         #drawn a plot on employees total working years with respect to the years that they worked at this particular company.
```



11.Violin Plot:

From the dataset monthly income and gender are most effective so violin plot is drawn to depict attrition and from the plot we found that most of them are males.

```
In [45]: sns.violinplot(x='Gender', y='MonthlyIncome',hue='Attrition',split=True,data=data1)
         #As observed from dataset monthly income and gender are most effective so violin plot is drawn to depict attrition and found most
```



Reference:

1)Dataset : [Kaggle]

2)[Pre-Processing reference]

3)[Pandas,Numpy]