Import required packages

The dataset was created by IBM employees and was downloaded from Kaggle. The dataset is fiction actually represent any actual IBM employees.

Attrition: It is basically the turnover rate of employees inside an organization.

This can happen for many reasons:

Employees looking for better opportunities. A negative working environment. Bad management Sickr death) Excessive working hours

The objective is to see what influences the attrition

It starts from framing business question t

1. Import required packages

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

2. Data Extracting

load the dataset and have clear understanding of the dataset attributes

```
#reading CSV file
df = pd.read_csv('emp_attrition.csv')

#getting the first rows
```

```
df.head(10)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely	591	Research & Development	2	1
5	32	No	Travel_Frequently	1005	Research & Development	2	2
6	59	No	Travel_Rarely	1324	Research & Development	3	3
7	30	No	Travel_Rarely	1358	Research & Development	24	1
8	38	No	Travel_Frequently	216	Research & Development	23	3
9	36	No	Travel_Rarely	1299	Research & Development	27	3

```
#explore the sape of the dataset
print('Rows x Columns : ', df.shape[0], 'x', df.shape[1])
```

```
Rows x Columns :  1470 x 35
```

```
#read all coulmns names
print('Features: \n', df.columns.tolist())
```

```
Features: 
 ['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department', 'DistanceFromHome', '
```

```
#having a description of the dataset
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
None
```

```
#getting the unique values
print('\nUnique values:')
print(df.nunique())
for col in df.columns:
    print(col, ':', sorted(df[col].unique()))
```

```
Unique values:
Age                          43
Attrition                     2
BusinessTravel                3
DailyRate                   886
Department                    3
DistanceFromHome             29
Education                     5
EducationField                6
EmployeeCount                 1
EmployeeNumber             1470
EnvironmentSatisfaction       4
Gender                        2
HourlyRate                   71
JobInvolvement                4
JobLevel                      5
JobRole                       9
JobSatisfaction               4
MaritalStatus                 3
MonthlyIncome              1349
MonthlyRate                1427
NumCompaniesWorked           10
Over18                        1
OverTime                      2
PercentSalaryHike            15
PerformanceRating             2
RelationshipSatisfaction      4
StandardHours                 1
StockOptionLevel              4
TotalWorkingYears            40
TrainingTimesLastYear         7
WorkLifeBalance               4
YearsAtCompany               37
YearsInCurrentRole           19
YearsSinceLastPromotion      16
YearsWithCurrManager         18
dtype: int64
Age : [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 3
Attrition : ['No', 'Yes']
BusinessTravel : ['Non-Travel', 'Travel_Frequently', 'Travel_Rarely']
DailyRate : [102, 103, 104, 105, 106, 107, 109, 111, 115, 116, 117, 118, 119, 120, 121,
Department : ['Human Resources', 'Research & Development', 'Sales']
DistanceFromHome : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
Education : [1, 2, 3, 4, 5]
EducationField : ['Human Resources', 'Life Sciences', 'Marketing', 'Medical', 'Other', '
EmployeeCount : [1]
EmployeeNumber : [1, 2, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23,
EnvironmentSatisfaction : [1, 2, 3, 4]
Gender : ['Female', 'Male']
HourlyRate : [30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48
JobInvolvement : [1, 2, 3, 4]
JobLevel : [1, 2, 3, 4, 5]
JobRole : ['Healthcare Representative', 'Human Resources', 'Laboratory Technician', 'Mar
JobSatisfaction : [1, 2, 3, 4]
MaritalStatus : ['Divorced', 'Married', 'Single']
MonthlyIncome : [1009, 1051, 1052, 1081, 1091, 1102, 1118, 1129, 1200, 1223, 1232, 1261,
```

```
MonthlyRate : [2094, 2097, 2104, 2112, 2122, 2125, 2137, 2227, 2243, 2253, 2261, 2288, 2
NumCompaniesWorked : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Over18 : ['Y']
OverTime : ['No', 'Yes']
PercentSalaryHike : [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]
PerformanceRating : [3, 4]
RelationshipSatisfaction : [1, 2, 3, 4]
StandardHours : [80]
StockOptionLevel : [0, 1, 2, 3]
TotalWorkingYears : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1
TrainingTimesLastYear : [0, 1, 2, 3, 4, 5, 6]
WorkLifeBalance : [1, 2, 3, 4]
YearsAtCompany : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
YearsInCurrentRole : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
YearsSinceLastPromotion : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
YearsWithCurrManager : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
```

* As we can see “Over18”, “Standard Hours” and “Employee Count” contain the same valu which we do not need them in visualizing the dataset

2.Data Preparation

2.1 Data Cleaning

find missing data, remove data that will not assist with the visualization in analysis processing

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

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

```
df.count()
```

```
Age                         1470
Attrition                   1470
BusinessTravel              1470
DailyRate                   1470
Department                  1470
DistanceFromHome            1470
Education                   1470
EducationField              1470
EmployeeCount               1470
EmployeeNumber              1470
EnvironmentSatisfaction     1470
Gender                      1470
HourlyRate                  1470
JobInvolvement              1470
JobLevel                    1470
JobRole                     1470
JobSatisfaction             1470
MaritalStatus               1470
MonthlyIncome               1470
MonthlyRate                 1470
NumCompaniesWorked          1470
Over18                      1470
OverTime                    1470
PercentSalaryHike           1470
PerformanceRating           1470
RelationshipSatisfaction    1470
StandardHours               1470
StockOptionLevel            1470
TotalWorkingYears           1470
TrainingTimesLastYear       1470
WorkLifeBalance             1470
YearsAtCompany              1470
YearsInCurrentRole          1470
YearsSinceLastPromotion     1470
YearsWithCurrManager        1470
dtype: int64
```

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

```
False
```

* This result shows if we have any missing values we used different codes. And as we car values. Otherwise, we would have done some techniques, like dropping columns or rows, mising values by the mean, backward, or frontward values.

2.2 Remove unsuported columns

```
#drop unwanted columns

df = df.drop(['Over18','StandardHours','EmployeeCount'], axis=1)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
```

* Since “Over18”, “Standard Hours” and “Employee Count” has a static varible, we remove of processing the dataframe.

2.3 Mapping data

```
6   Education                 1470 non-null   int64
```

```
df.dtypes
```

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

```
df['Attrition'].unique()
```

```
array(['Yes', 'No'], dtype=object)
```

```
#Education map
Attrition_map = {"Yes" : 1, "No": 0}
print(Attrition_map)
df['Attrition']=df['Attrition'].map(Attrition_map)

Education_map = {1:"Below College", 2 :'College' ,3 : 'Bachelor' , 4 :'Master', 5 :'Doctor'
df['Education'] = df['Education'].map(Education_map)



EnvironmentSatisfaction_map = {1 :"Low", 2:"Medium", 3:"High", 4:"Very High"}
df["EnvironmentSatisfaction"] = df["EnvironmentSatisfaction"].map(EnvironmentSatisfaction_m

JobInvolvement_map = {1 :"Low", 2:"Medium", 3:"High", 4:"Very High"}
df["JobInvolvement"] = df["JobInvolvement"].map(JobInvolvement_map)

JobSatisfaction_map = {1 :"Low", 2:"Medium", 3:"High", 4:"Very High"}
df["JobSatisfaction"] = df["JobSatisfaction"].map(JobSatisfaction_map)

PerformanceRating_map = {1 :"Low", 2:"Medium", 3:"High", 4:"Outstanding"}
df["PerformanceRating"] = df["PerformanceRating"].map(PerformanceRating_map)

RelationshipSatisfaction_map = {1 :"Low", 2:"Medium", 3:"High", 4:"Outstanding"}
df["RelationshipSatisfaction"] = df["RelationshipSatisfaction"].map(RelationshipSatisfactic

WorkLifeBalance_map = {1 :"Low", 2:"Medium", 3:"High", 4:"Outstanding"}
df["WorkLifeBalance"] = df["WorkLifeBalance"].map(WorkLifeBalance_map)
```

```
{'Yes': 1, 'No': 0}
```

```
df['Attrition'].unique()
```

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

2.4 Grouping / Binning Ages

```
df["Age"].describe()
```

```
count    1470.000000
mean       36.923810
```

```
age_labels = ['18-24', '25-30', '31-35', '36-40', '41-45', '46-50', '51-55', '56-60']
df['age_group'] = pd.cut(df.Age, range(18, 61, 5), right=False, labels=age_labels)
```

```
50%        36.000000
```

```
df.head(3)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	1	Travel_Rarely	1102	Sales	1	College
1	49	0	Travel_Frequently	279	Research & Development	8	Below College
2	37	1	Travel_Rarely	1373	Research & Development	2	College

3. Exploring statistics on the dataset

3.1 Descriptive statstic

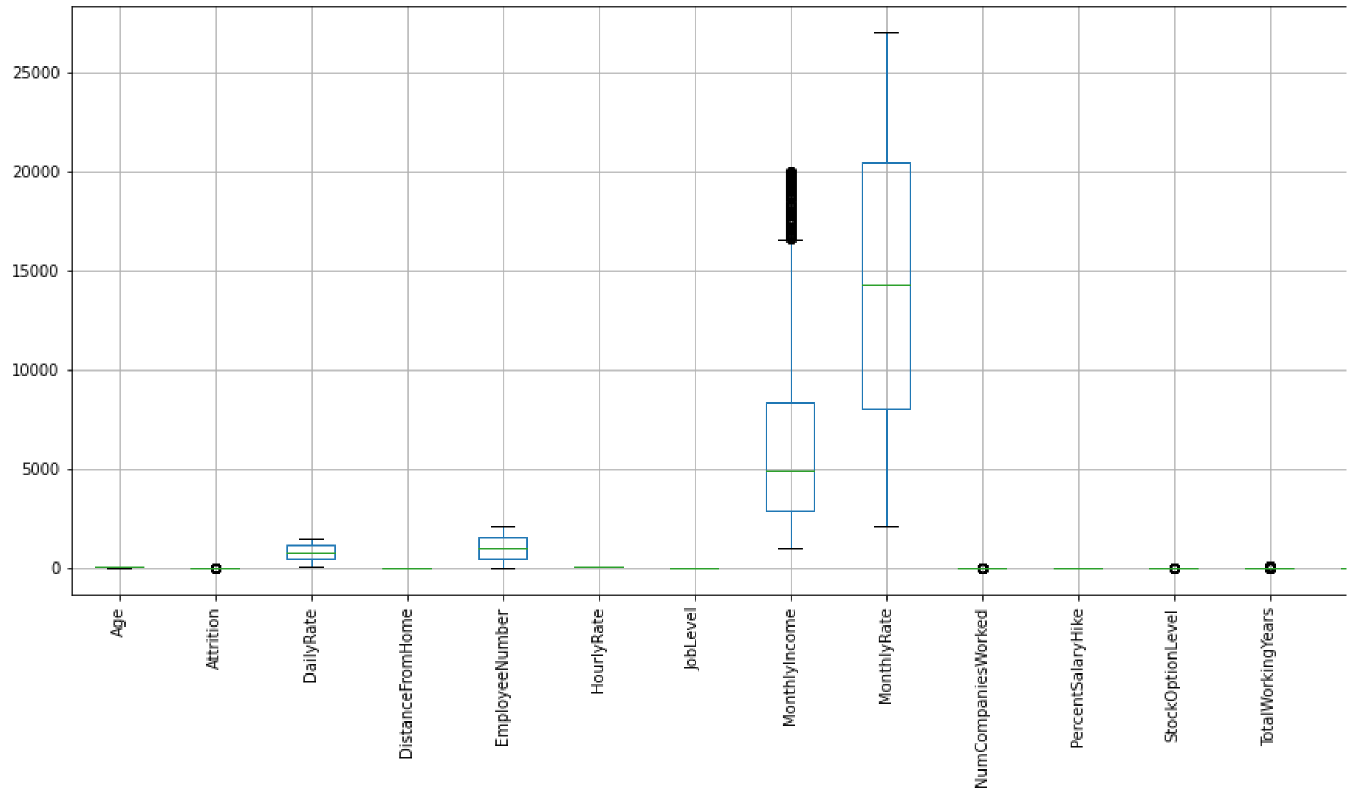
```
df.describe()
```

Age	Attrition	DailyRate	DistanceFromHome	EmployeeNumber	HourlyRa

3.2 Visualizing these statistics using boxplots

```
plt.rcParams["figure.figsize"] = (20,7)
df.boxplot()
plt.xticks(rotation=90)
```

```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18]), <a list of 18 Text major ticklabel objects>)
```



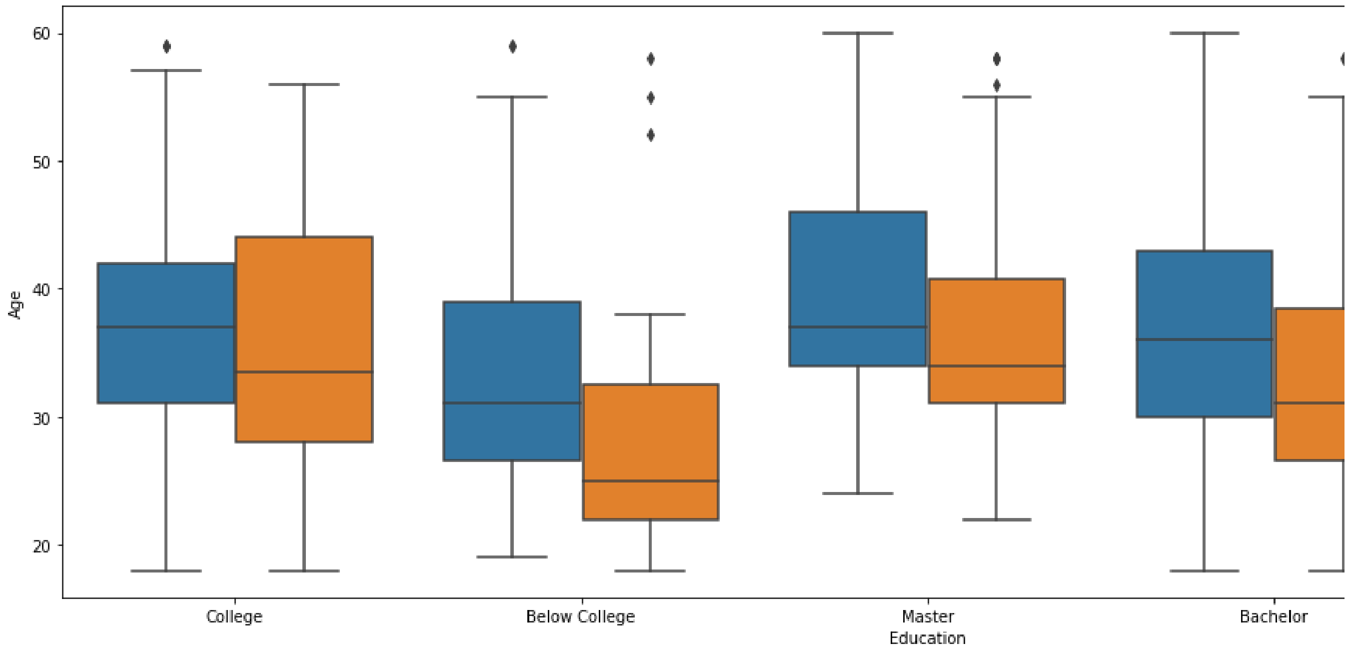

```
df["Attrition"].replace("Yes", 1, inplace = True)
df["Attrition"].replace("No", 0, inplace = True)

df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educati
0	41	1	Travel_Rarely	1102	Sales	1	Colle
1	49	0	Travel_Frequently	279	Research & Development	8	Bel Colle
2	37	1	Travel_Rarely	1373	Research & Development	2	Colle
3	33	0	Travel_Frequently	1392	Research & Development	3	Mas
4	27	0	Travel_Rarely	591	Research & Development	2	Bel Colle
...	...	...	...	...	...	...	
1465	36	0	Travel_Frequently	884	Research & Development	23	Colle
1466	39	0	Travel_Rarely	613	Research & Development	6	Bel Colle
1467	27	0	Travel_Rarely	155	Research & Development	4	Bache
1468	49	0	Travel_Frequently	1023	Sales	2	Bache
1469	34	0	Travel_Rarely	628	Research & Development	8	Bache

1470 rows × 33 columns

```
sns.boxplot(x=df['Education'],y=df['Age'],data=df, hue=df["Attrition"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe556e09438>
```

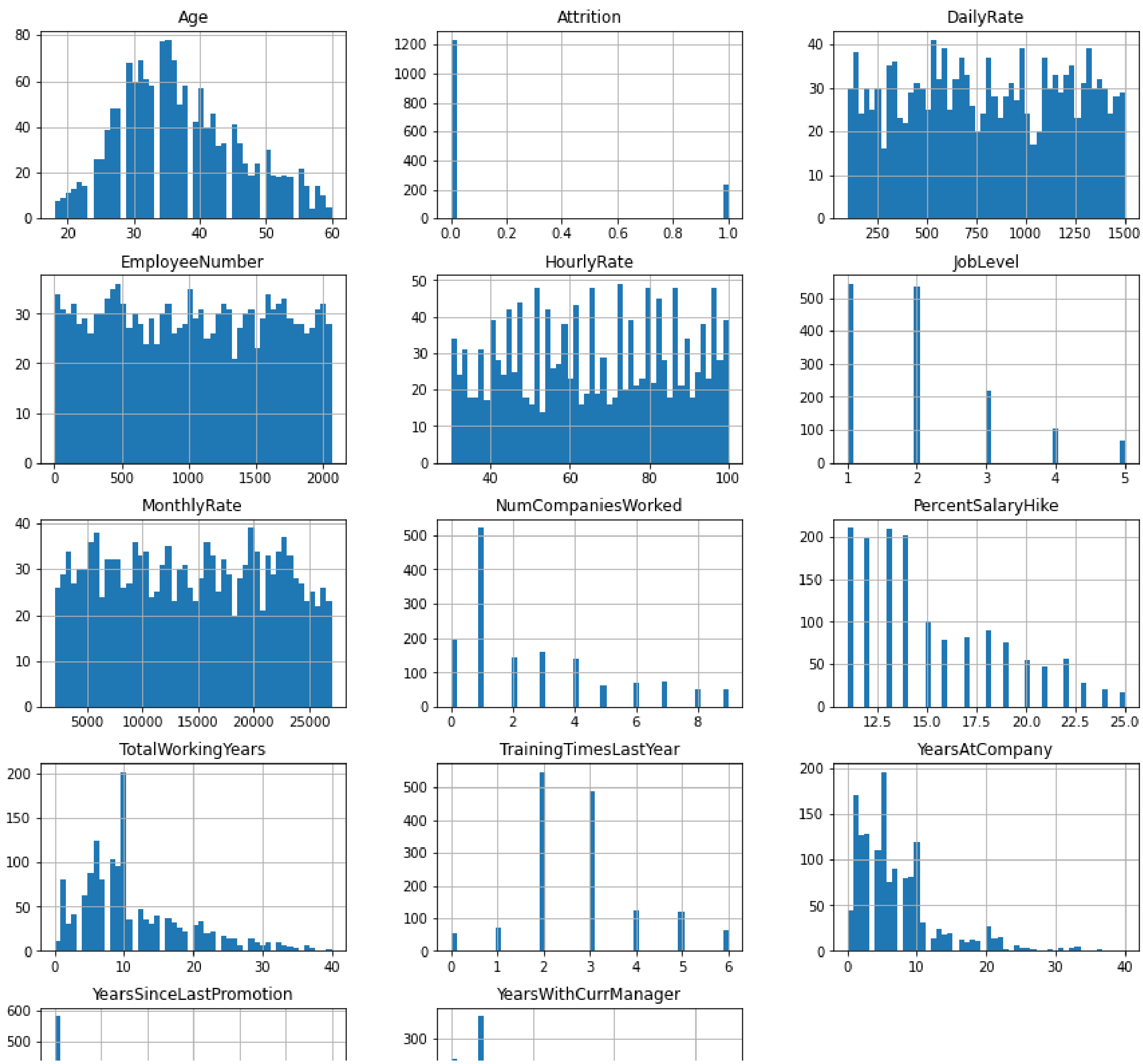


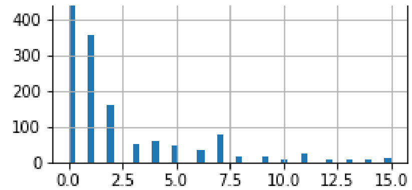
* It can be observed that the value ranges of columns (MonthlyIncome, MonthlyRate, Emp
significantly higher than the remaining numeric columns. This can be corrected using nor

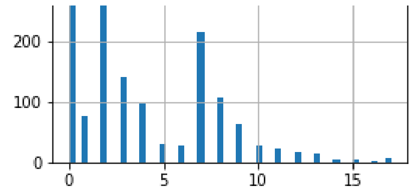
Visualizing the value distribution for each numeric column in t

```
df.hist(bins=50,figsize=(20,16))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fe5567b0080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe5567d49b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe556787c18>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe55673ee80>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe5566fd128>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe5566b4390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe5566645f8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe556696c88>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe556696cf8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe556608400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe5565b8780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe55656ab00>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe55651ce80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe5564de240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe55650c5c0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe5564bf940>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe556471cc0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe556431080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe5563e4400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe556417780>]],
      dtype=object)
```







4. Visualizing the value distributions for the individual variable and exp

4.1 Atrithion Rate

```
df.groupby(["Attrition"]).count()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Edu
Attrition							
0	1233	1233	1233	1233	1233	1233	
1	237	237	237	237	237	237	

```
df["Attrition"].value_counts()
```

```
0    1233
1     237
Name: Attrition, dtype: int64
```

```
emp_attrition = df[df["Attrition"] == 1]
emp_attrition = emp_attrition["Attrition"].count()
print ("The total number of employee who suffer from attrition are :" , emp_attrition)
```

```
The total number of employee who suffer from attrition are : 237
```

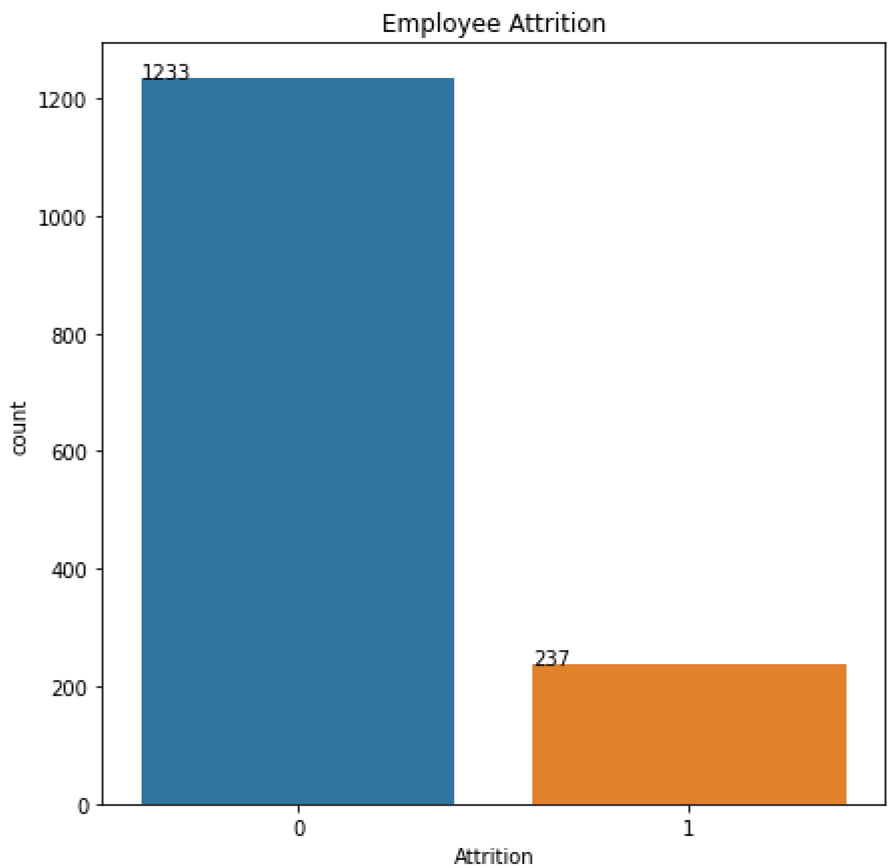
```
emp_no_attrition = df[df["Attrition"] == 0]
emp_no_attrition = emp_no_attrition["Attrition"].count()
print ("The total number of employee who is not suffer from attritionis :" , emp_no_attriti
```

```
The total number of employee who is not suffer from attritionis : 1233
```

```
# Show the percentage of each unique class label in the target Attrition column
df['Attrition'].value_counts()/len(df['Attrition'])*100
```

```
0    83.877551
1    16.122449
Name: Attrition, dtype: float64
```

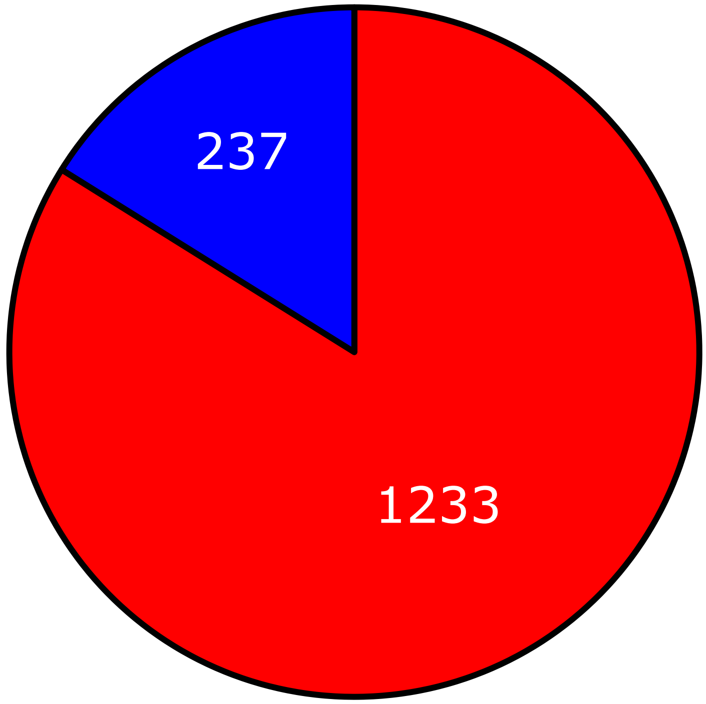
```
#Visualize the result
plt.rcParams["figure.figsize"] = (7,7)
ax = sns.countplot(x='Attrition', data=df)
for p in ax.patches:
    ax.annotate('{}'.format(p.get_height()), (p.get_x(), p.get_height()+1))

plt.title("Employee Attrition")
plt.show()
```



* The previous percentages show that almost 84% of the employees included in the datas attrition. Also, it can be observed that the data is imbalanced between the two class label for 'Yes') of the 'Attrition' target column. Thus, there is a need to balance the sampling rati of a classifier algorithm.

```
#using interactive graph
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objs as go
groups = df["Attrition"]
amount = df["Attrition"].value_counts()
colors = ['red', 'blue']
trace = go.Pie(labels=["No","Yes"], values=amount,
hoverinfo='label+percent', textinfo='value',
textfont=dict(size=25),
marker=dict(colors=colors,
```

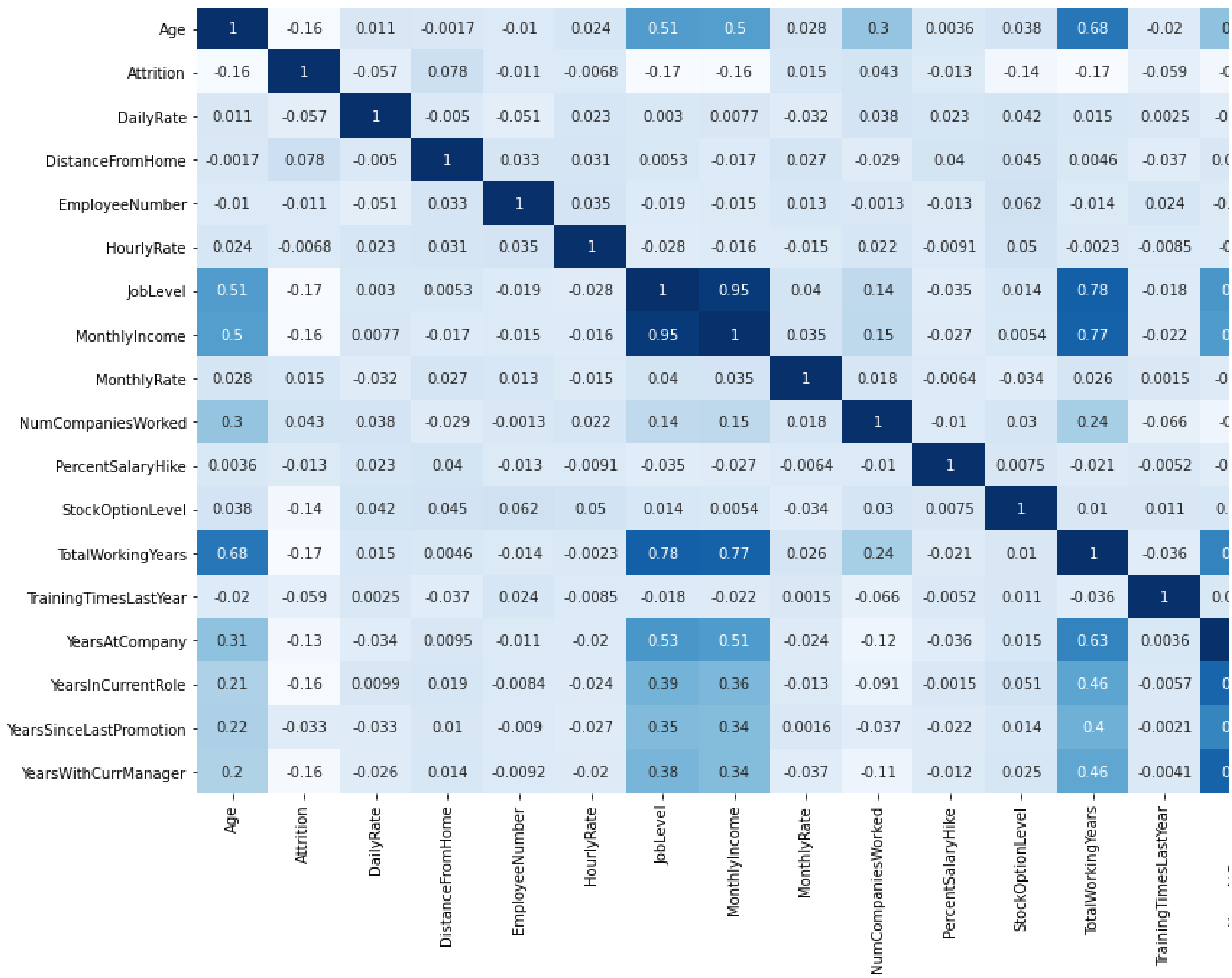
```
line=dict(color='#000000', width=3)))

# print ("it should be the obeset??")

iplot([trace])
```



4.2 Finding correlation between variables

```
data_correlation = df.corr()
plt.rcParams["figure.figsize"] = [20,10]
sns.heatmap(data_correlation,xticklabels=data_correlation.columns,yticklabels=data_correlat


print("How can we justify the numbers with boxs?")
```

```
How can we justify the numbers with boxs?
```



The correlation analysi shows interesting findings First, there is a high positive correlation between tl and the “JobLevel” and “MonthlyIncome”, which reflects a sort of fairness in promoting and paying p their experience level. Second, there was a high positive correlation between “PerformanceRating” ar which again confirms that the increase in salary is based on the increase in the performance level. Tl column does not have any correlation with the reminder of the numeric columns, which is somehow reasonable to have it increased with the increase in “MonthlyIncome” or “JobLevel” columns.

Normalizing the dataset

before we go in deap in visualize the dataset, it is better to normalize it to avoid differnt variance

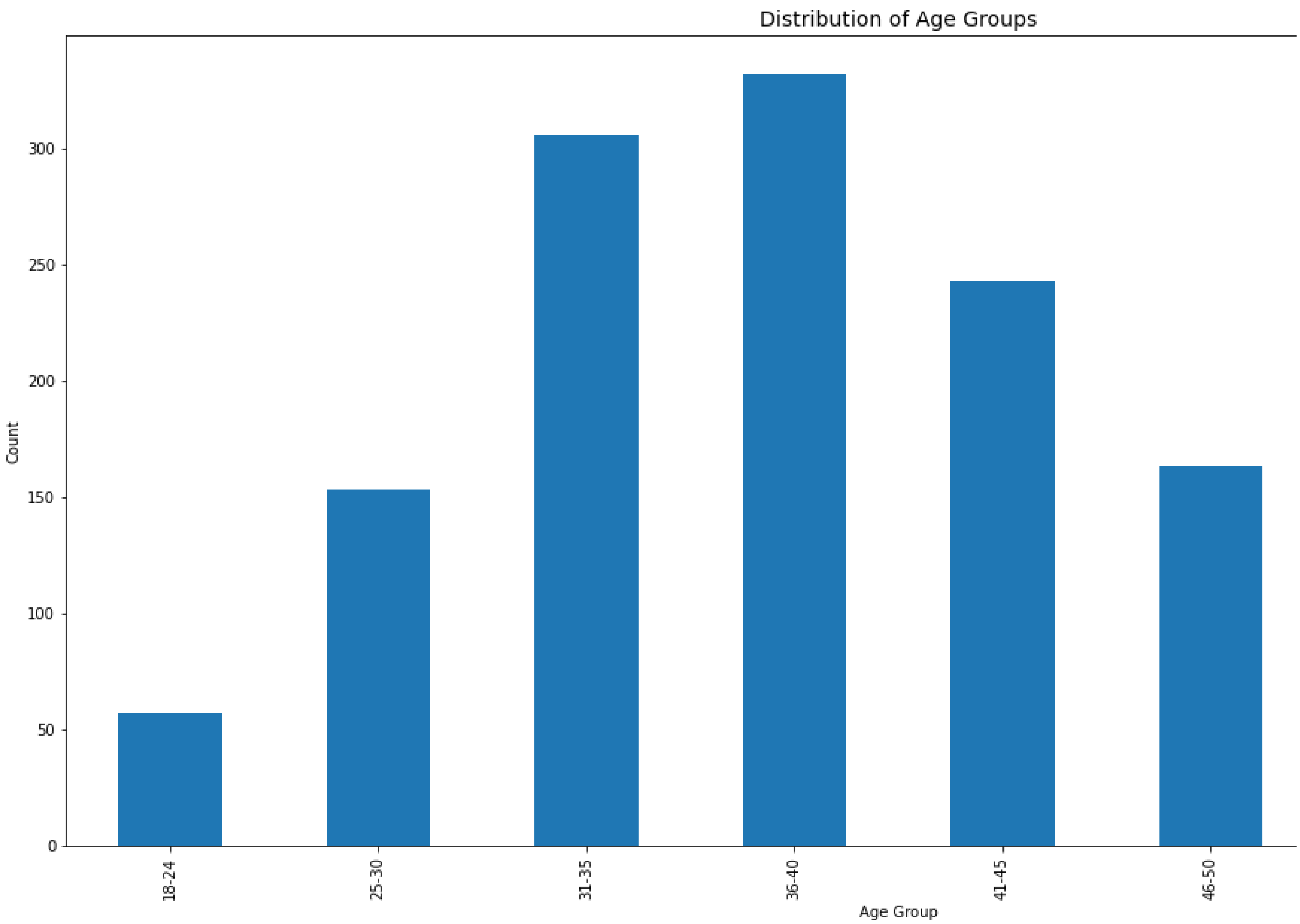
```
from sklearn.preprocessing import StandardScaler

standard=df.copy()
val=standard.select_dtypes("int64")

col_names=list(val.columns)

features =  val[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)

standard[col_names] = features
standard
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Ed
0	0.446350	2.280906	Travel_Rarely	0.742527	Sales	-1.010909	
1	1.322365	-0.438422	Travel_Frequently	-1.297775	Research & Development	-0.147150	
2	0.008343	2.280906	Travel_Rarely	1.414363	Research & Development	-0.887515	
3	-0.429664	-0.438422	Travel_Frequently	1.461466	Research & Development	-0.764121	
4	-1.086676	-0.438422	Travel_Rarely	-0.524295	Research & Development	-0.887515	
...	...	...	...	...	...	...	
1465	-0.101159	-0.438422	Travel_Frequently	0.202082	Research & Development	1.703764	
1466	0.227347	-0.438422	Travel_Rarely	-0.469754	Research & Development	-0.393938	
1467	-1.086676	-0.438422	Travel_Rarely	-1.605183	Research & Development	-0.640727	
1468	1.322365	-0.438422	Travel_Frequently	0.546677	Sales	-0.887515	
1469	-0.320163	-0.438422	Travel_Rarely	-0.432568	Research & Development	-0.147150	

1470 rows × 33 columns

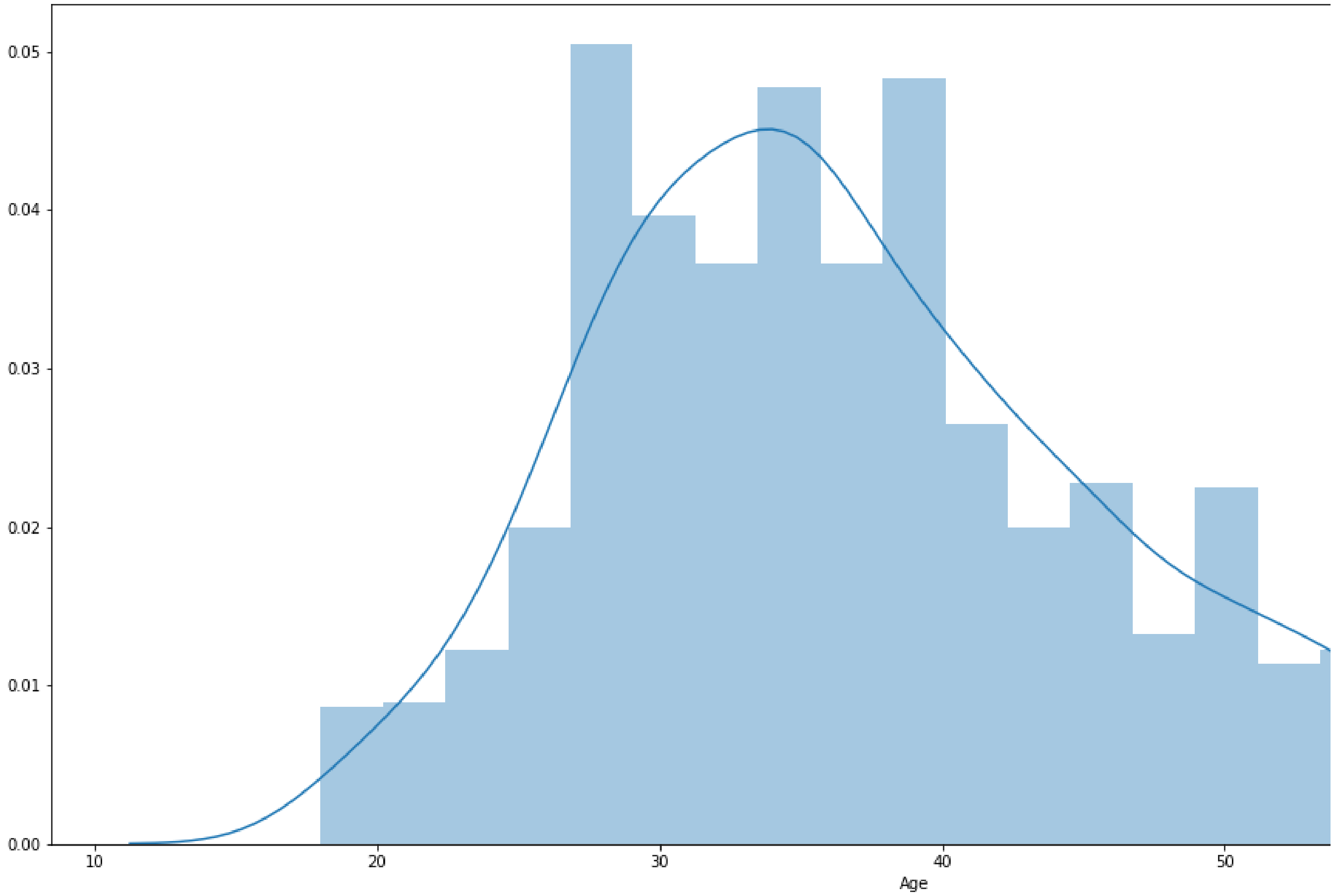
4.2 Relationship of Age Variable with Attrition

```
df.groupby(['age_group']).size().plot(kind='bar',stacked=True)
plt.title("Distribution of Age Groups",fontsize=14)
plt.ylabel('Count')
plt.xlabel('Age Group');
```




```
sns.distplot(df["Age"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe550719cc0>
```




```
youngest = df['Age'].min()
print(" The youngest employee in the company was in age : ", youngest)
```

```
 The youngest employee in the company was in age :  18
```

```
oldest = df['Age'].max()
print(" The oldest employee in the company was in age  ", oldest)
```

```
 The oldest employee in the company was in age   60
```

```
#fining out who was the oldest employee
df.loc[oldest,:]
```

```
Age                                              32
Attrition                                         0
BusinessTravel                        Travel_Rarely
DailyRate                                       427
Department                   Research & Development
DistanceFromHome                                  1
Education                                  Bachelor
EducationField                              Medical
EmployeeNumber                                   78
EnvironmentSatisfaction                         Low
Gender                                         Male
HourlyRate                                       33
JobInvolvement                                 High
JobLevel                                          2
JobRole                      Manufacturing Director
JobSatisfaction                           Very High
MaritalStatus                               Married
MonthlyIncome                                  6162
MonthlyRate                                   10877
NumCompaniesWorked                                1
OverTime                                        Yes
PercentSalaryHike                                22
PerformanceRating                       Outstanding
RelationshipSatisfaction                     Medium
StockOptionLevel                                  1
TotalWorkingYears                                 9
TrainingTimesLastYear                             3
WorkLifeBalance                                High
YearsAtCompany                                    9
YearsInCurrentRole                                8
YearsSinceLastPromotion                           7
YearsWithCurrManager                              8
age_group                                     31-35
Name: 60, dtype: object
```

```
#fining out who was the youngest employee
df[df['Age']==youngest]
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educati
296	18	1	Travel_Rarely	230	Research & Development	3	Bache
301	18	0	Travel_Rarely	812	Sales	10	Bache
457	18	1	Travel_Frequently	1306	Sales	5	Bache
727	18	0	Non-Travel	287	Research & Development	5	Colle
828	18	1	Non-Travel	247	Research & Development	8	Bel Colle
972	18	0	Non-Travel	1124	Research & Development	1	Bache
1153	18	1	Travel_Frequently	544	Sales	3	Colle
1311	18	0	Non-Travel	1431	Research & Development	14	Bache

As we can see from the result above, the oldest employee was in his 60 years old, and he shows not
employee was in his 18, and he is attrition.

```
positive_attrition_df = df.loc[df['Attrition'] == 1]
negative_attrition_df = df.loc[df['Attrition'] == 0]
```

```
negative_attrition_df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
1	49	0	Travel_Frequently	279	Research & Development	8	Below College

```
positive_attrition_df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	1	Travel_Rarely	1102	Sales	1	College
2	37	1	Travel_Rarely	1373	Research & Development	2	College
14	28	1	Travel_Rarely	103	Research & Development	24	Bachelor
21	36	1	Travel_Rarely	1218	Sales	9	Master
24	34	1	Travel_Rarely	699	Research & Development	6	Below College

```
sns.distplot(negative_attrition_df['MonthlyIncome'], label='Negative attrition')
sns.distplot(positive_attrition_df['MonthlyIncome'], label='positive attrition')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fe550975278>
```

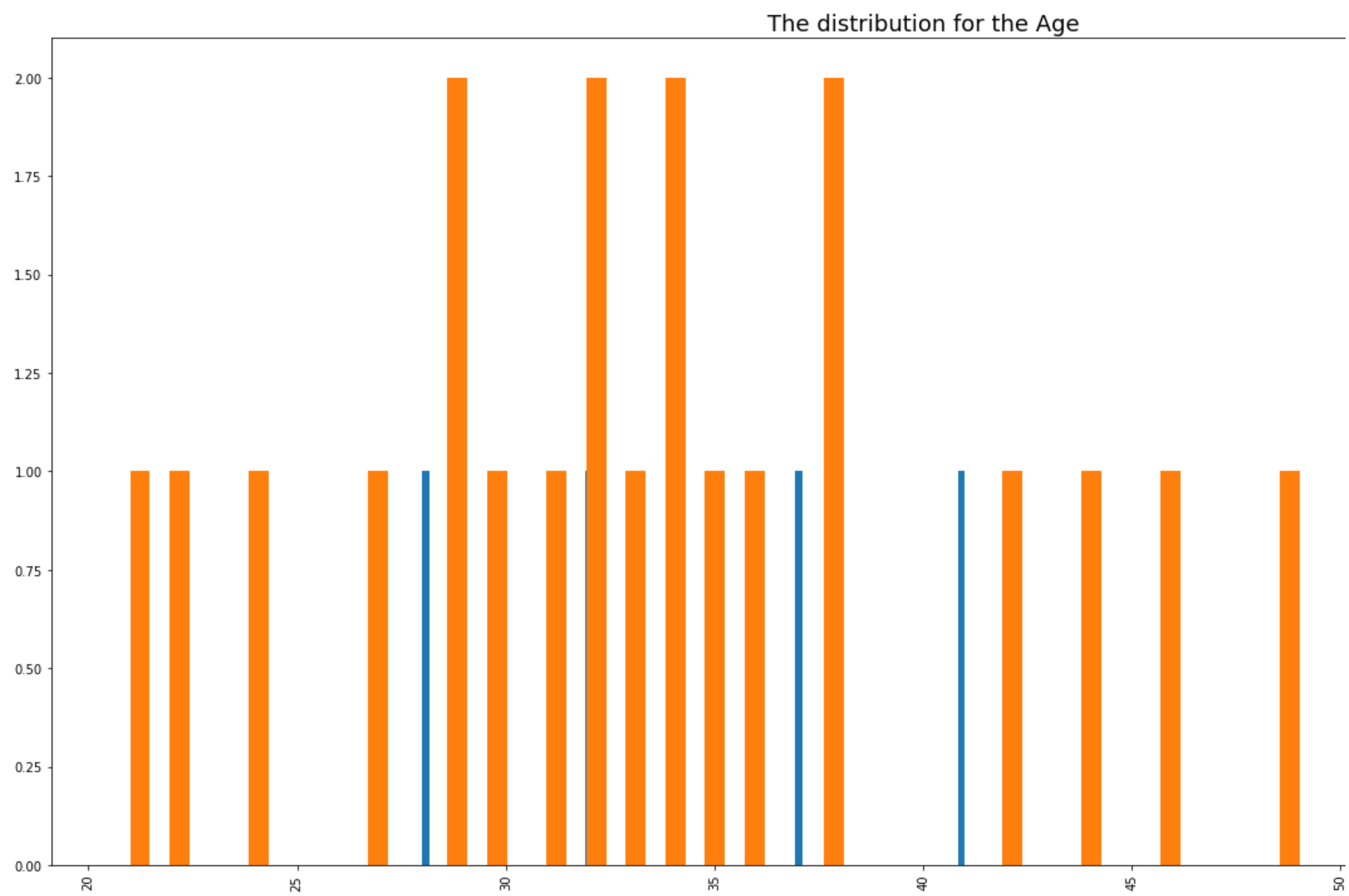

```
type(emp_attrition)
```

```
numpy.int64
```



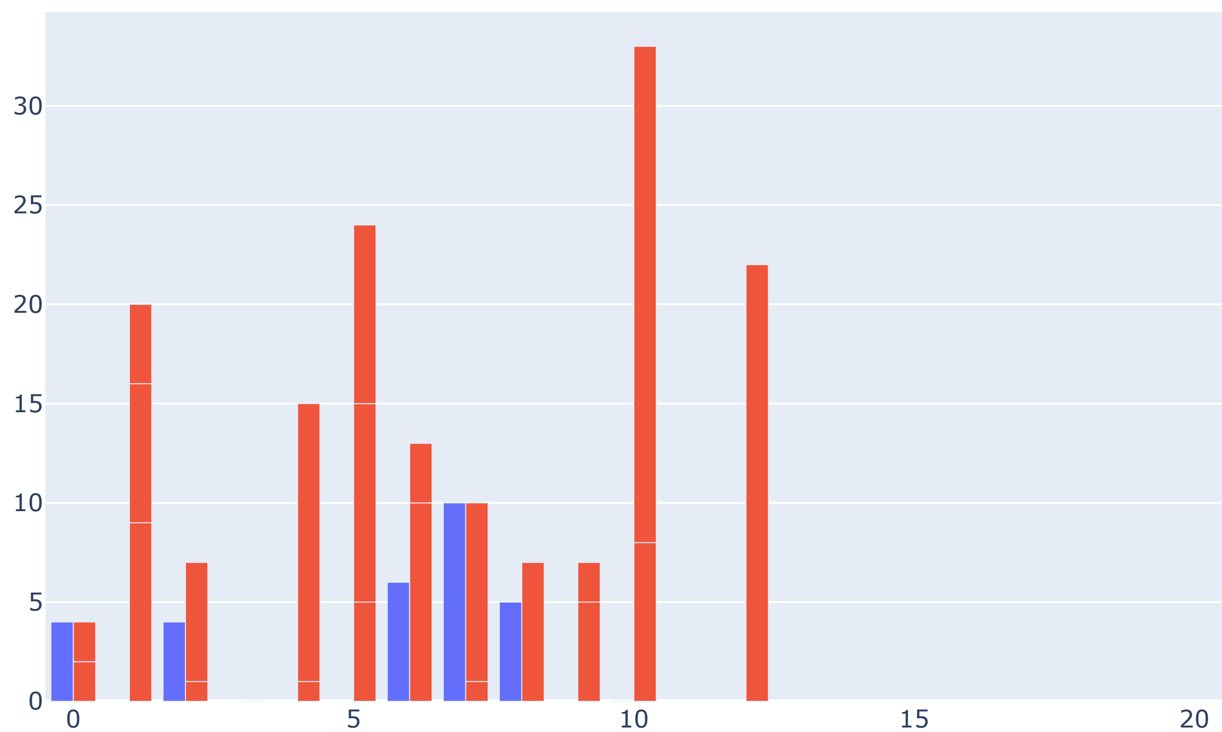
```
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objs as go

df= df.head(30)
trace1 = go.Bar(
# x = emp_attrition['Age'],
x = df['Age'],
y = df['Age'][df['Attrition']==1],
name= 'Yes')
trace2 = go.Bar(
# x = emp_no_attrition['Age'],
x = df['Age'],
y = df['Age'][df['Attrition']==0],
name= 'No')
data = [trace1, trace2]
layout = go.Layout(barmode='group')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```

```
plt.hist(df['Age'][df["Attrition"]==1], bins= 80, histtype="bar")
plt.hist(df['Age'][df["Attrition"]==0], bins= 80, histtype="bar")
plt.legend("Age", loc='uper right')


plt.xlabel= ("Age")
plt.ylabel = ("Frequency")
plt.title('The distribution for the Age', fontsize = 18 )

plt.xticks(rotation=90)

plt.tight_layout()
plt.savefig('Age.png', dpi = 300)

plt.show()
```

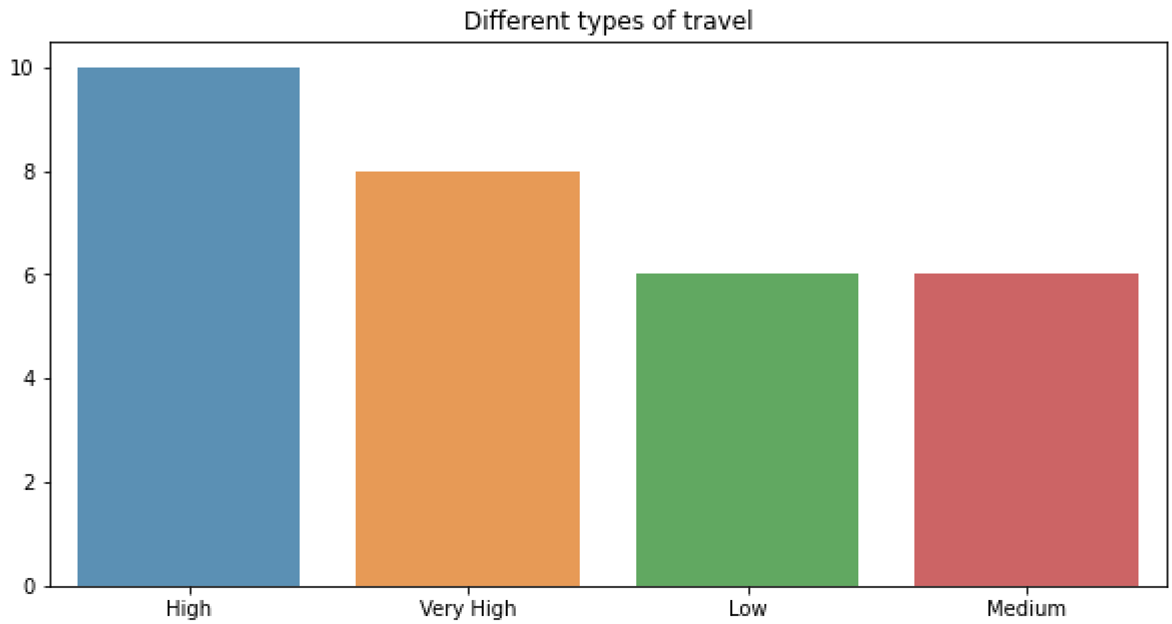



```
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objs as go

df= df.head(30)
```

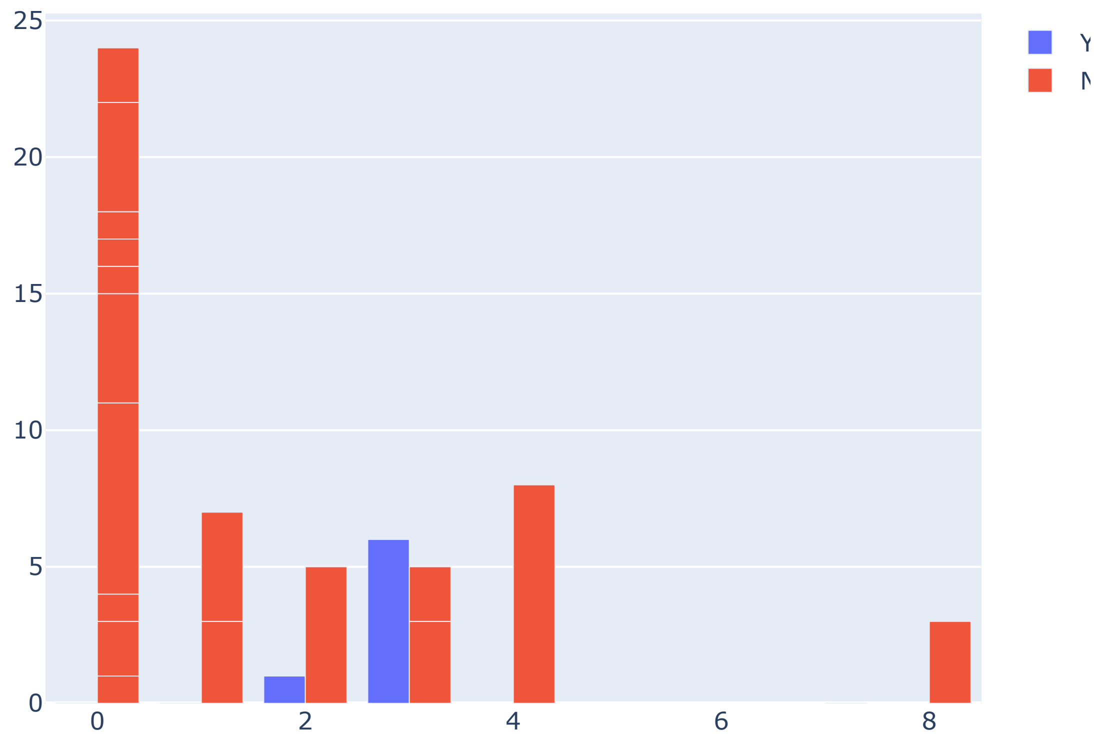
```
trace1 = go.Bar(
# x = emp_attrition['Age'],
x = df['YearsAtCompany'],
y = df['YearsAtCompany'][df['Attrition']==1],
name= 'Yes')
trace2 = go.Bar(
# x = emp_no_attrition['Age'],
x = df['YearsAtCompany'],
y = df['YearsAtCompany'][df['Attrition']==0],
name= 'No')
data = [trace1, trace2]
layout = go.Layout(barmode='group')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```



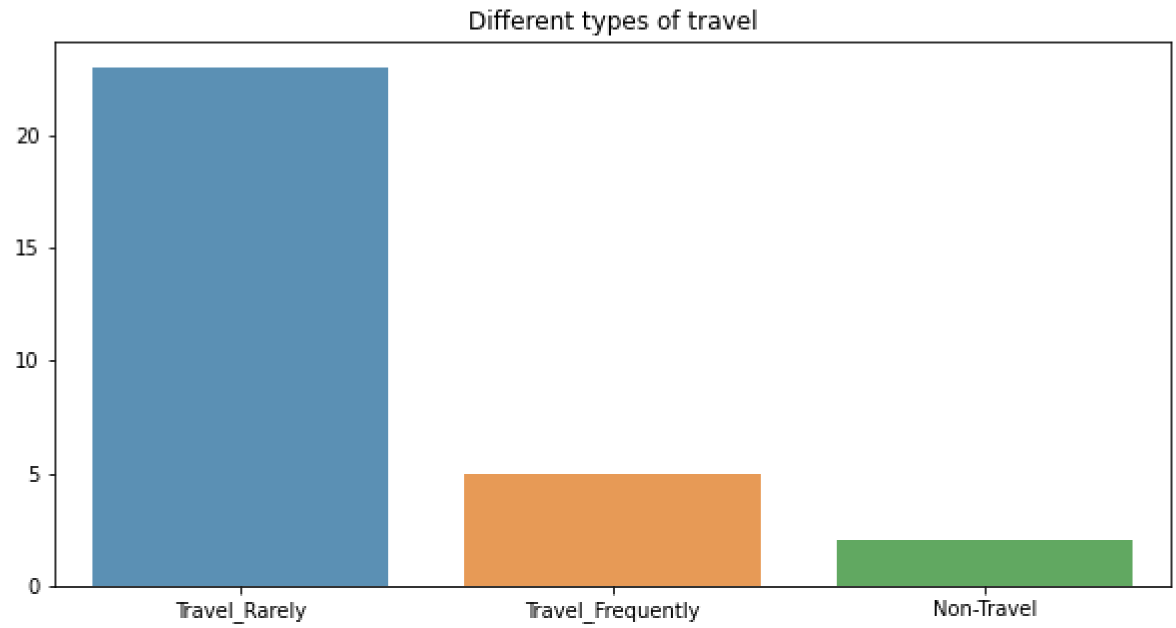

```
job  = df['JobSatisfaction'].value_counts()
plt.figure(figsize=(10,5))
sns.barplot(job.index, job.values, alpha=0.8)
plt.title('Different types of travel')
plt.show()
```




```
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objs as go

df= df.head(30)
trace1 = go.Bar(
# x = emp_attrition['Age'],
x = df['YearsSinceLastPromotion'],
y = df['YearsSinceLastPromotion'][df['Attrition']==1],
name= 'Yes')
trace2 = go.Bar(
# x = emp_no_attrition['Age'],
x = df['YearsSinceLastPromotion'],
y = df['YearsSinceLastPromotion'][df['Attrition']==0],
name= 'No')
data = [trace1, trace2]
layout = go.Layout(barmode='group')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```




```
business  = df['BusinessTravel'].value_counts()
plt.figure(figsize=(10,5))
sns.barplot(business.index, business.values, alpha=0.8)
plt.title('Different types of travel')
plt.show()
```


Different types of travel
20
15
10
5
0
Travel_Rarely
Travel_Frequently
Non-Travel

```
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	1	Travel_Rarely	1102	Sales	1	College
1	49	0	Travel_Frequently	279	Research & Development	8	Below College
2	37	1	Travel_Rarely	1373	Research & Development	2	College
3	33	0	Travel_Frequently	1392	Research & Development	3	Master
4	27	0	Travel_Rarely	591	Research & Development	2	Below College
5	32	0	Travel_Frequently	1005	Research & Development	2	College
6	59	0	Travel_Rarely	1324	Research & Development	3	Bachelor
7	30	0	Travel_Rarely	1358	Research & Development	24	Below College
8	38	0	Travel_Frequently	216	Research & Development	23	Bachelor
9	36	0	Travel_Rarely	1299	Research & Development	27	Bachelor
10	35	0	Travel_Rarely	809	Research & Development	16	Bachelor
11	29	0	Travel_Rarely	153	Research & Development	15	College
12	31	0	Travel_Rarely	670	Research & Development	26	Below College
13	34	0	Travel_Rarely	1346	Research & Development	19	College
14	28	1	Travel_Rarely	103	Research & Development	24	Bachelor
15	29	0	Travel_Rarely	1389	Research & Development	21	Master
16	32	0	Travel_Rarely	334	Research & Development	5	College
17	22	0	Non-Travel	1123	Research & Development	16	College
18	53	0	Travel_Rarely	1219	Sales	2	Master
19	38	0	Travel_Rarely	371	Research & Development	2	Bachelor

20	24	0	Non-Travel	673	Research & Development	11	College
21	36	1	Travel_Rarely	1218	Sales	9	Master
22	34	0	Travel_Rarely	419	Research & Development	7	Master
23	21	0	Travel_Rarely	391	Research & Development	15	College
24	34	1	Travel_Rarely	699	Research & Development	6	Below College
25	53	0	Travel_Rarely	1282	Research & Development	5	Bachelor
26	32	1	Travel_Frequently	1125	Research & Development	16	Below College
27	42	0	Travel_Rarely	691	Sales	8	Master
28	44	0	Travel_Rarely	477	Research & Development	7	Master
29	46	0	Travel_Rarely	705	Sales	2	Master

```
!pip install -c plotly chart-studio
```

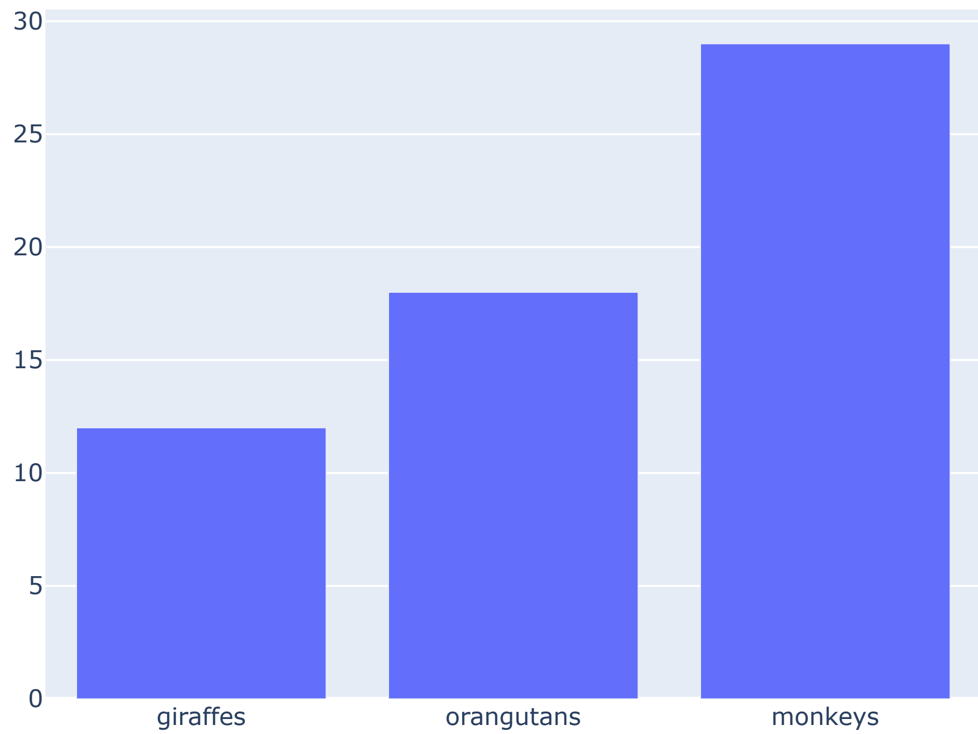
```
ERROR: Could not open requirements file: [Errno 2] No such file or directory: 'plotly'
```

```
x=['giraffes', 'orangutans', 'monkeys']
y=[12, 18, 29]
zoo=pd.DataFrame(x,columns=['animals'])
zoo['value']=y
zoo
```

	animals	value
0	giraffes	12
1	orangutans	18
2	monkeys	29

```
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objs as go

df= df.head(30)
trace4 = go.Bar(
x = zoo['animals'],
```

```
y = zoo['value'],
name= 'ZOO')
data = [trace4]
layout = go.Layout(barmode='group')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```



here i have worked on finding out the no.of animals that were present of a particular type. above i wa
two columns - animal name and the value_count of that spicies and with the help of these plots that i

```
print (df.groupby(['age_group']).Attrition.mean())
```

```
age_group
18-24    0.000
25-30    0.000
31-35    0.250
36-40    0.375
41-45    0.250
46-50    0.000
51-55    0.000
56-60    0.000
Name: Attrition, dtype: float64
```

Here I have tried to find out the mean value of attrition for a particular age group. i.e for eg- 36-40 is t
mean attrition means that the avg attrition value for the range is 0.375