HR Employee Analysis Project

Using Python- Visual basics

```
'''1st of all we have HR Employees dataset
in this Project, i will brakdown the dataset through
analysing the data seeking to predict the reasone or
the causing factor of the employees attrition
Education
1 'Below College'
2 'College'
3 'Bachelor'
4 'Master'
5 'Doctor'

EnvironmentSatisfaction
1 'Low'
2 'Medium'
3 'High'
4 'Very High'

JobInvolvement
1 'Low'
2 'Medium'
3 'High'
4 'Very High'

JobSatisfaction
1 'Low'
2 'Medium'
3 'High'
4 'Very High'

PerformanceRating
1 'Low'
2 'Good'
3 'Excellent'
4 'Outstanding'

RelationshipSatisfaction
1 'Low'
2 'Medium'
3 'High'
4 'Very High'

WorkLifeBalance
1 'Bad'
```

```
2 'Good'
3 'Better'
4 'Best'
'''
```

```python
# import libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
```

```python
df= pd.read_csv('C:/Users/LENOVO/Desktop/internship/HR-Employee.csv')
df.head()
```

✓ 0.1s

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Edu
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	

```python
#EDA Exploration Data Analysis
df.shape # 1470 Raws with 35 columns (factors)
nullValues = df.isnull().sum().sum()#EDA : is to identify the pattterns through different data visualization
nullValues #No null values in this dataset
```

[4] ✓ 0.0s

```
... 0
```

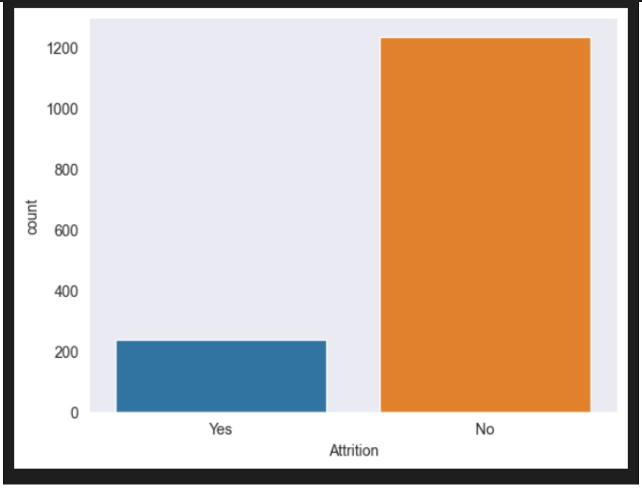
```python
duplicatedValues= df.duplicated().sum()
duplicatedValues# No duplcated values in this dataset
```

```
0.0s
0
```

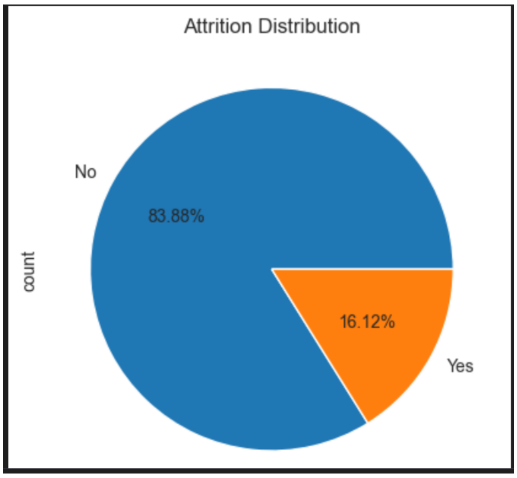
```
#compare btw count of employees who have attririon and who'r not
categorical_count = df['Attrition'].value_counts().to_frame()
categorical_count # by value_counts function i broke down the Attrition column into two values Yes and No to countvalues for each
```

	count
Attrition	
No	1233
Yes	237

```
sns.set_style('dark')
sns.countplot(x='Attrition',data=df) #Initial plot to compare btw yes and no who are left the company and who'r not. So we realized tht who were stayed more than who were left
```




```
#with pie chart
plt.title('Attrition Distribution')
df['Attrition'].value_counts().plot.pie(autopct='%1.2f%%')
#Initial Piechart we can see that 16.12% out of 83.88% of the employees who were left
```




```
# we can seperte each of yes or no in seperated dataframe
Attrition_yes = df[df['Attrition']=='Yes']
Attrition_no = df[df['Attrition']=='No']
Attrition_yes.shape #we can see that 237 employee had left the company while
1233 who are not

# No w
```

```
(237, 35)
```

```
# No we should moved into the visualization and see the correlation btw the
factors, and that must give us a clear insights about the independents
variables(factors)
#be4 that i will start with label the categorical columns  with numbers
# let us see the datatype 1st
df.dtypes #Attrition, BusinessTravel, Department, EducationField, Gender,
JobeRole, MaritalStatus, Over18, OverTime
```

```
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
```

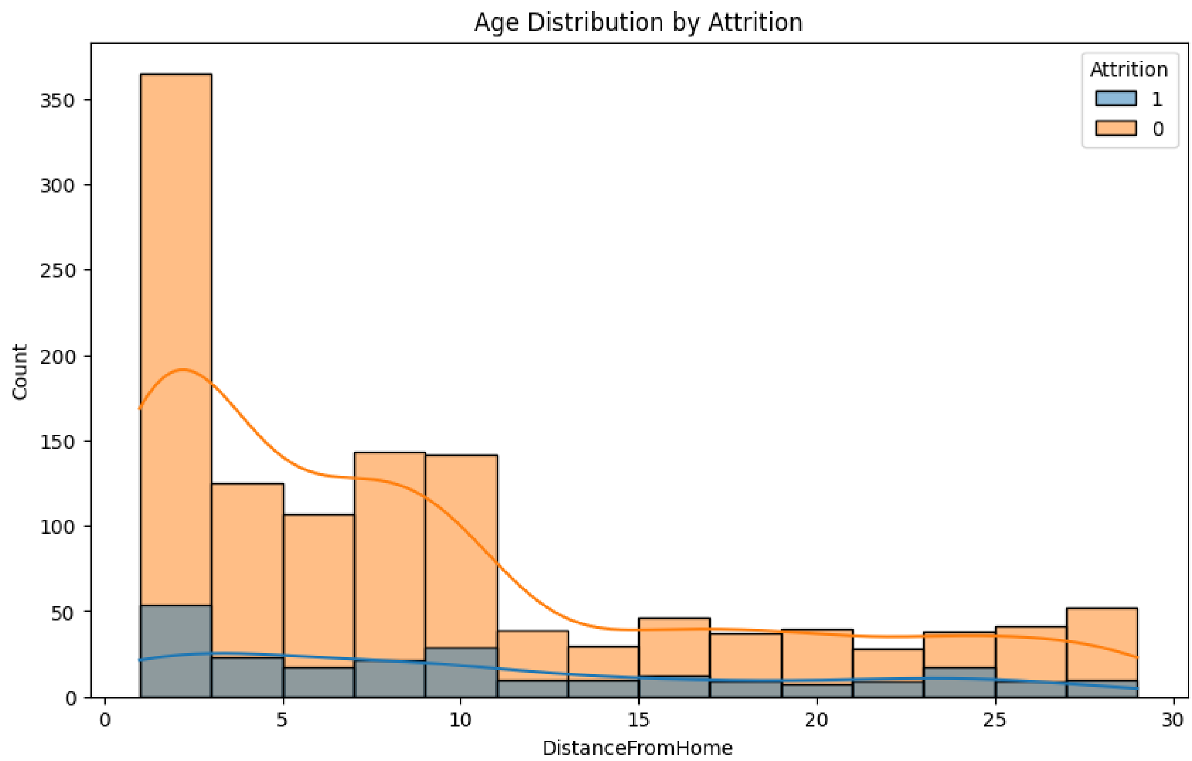
```
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
Over18                     object
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
...
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

```
df = df.replace(to_replace = ['Yes','No'],value = ['1','0'])
df = df.replace(to_replace = ['Travel_Rarely',
'Travel_Frequently','Non-Travel'],value = ['2','1','0'])
df = df.replace(to_replace = ['Married','Single','Divorced'],value =
['2','1','0'])
df = df.replace(to_replace = ['Male','Female'],value = ['1','0'])
#---
df = df.replace(to_replace = ['Human Resources','Research &
Development','Sales'],value = ['0','1','2'])
df = df.replace(to_replace = ['Human Resources','Life
Sciences','Marketing','Medical','Technical Degree','Other'],value =
['0','1','2','3','4','5'])
df = df.replace(to_replace = ['Healthcare Representative','Human
Resources','Laboratory Technician','Manager','Manufacturing
Director','Research Director','Research Scientist','Sales Executive','Sales
Representative'],value = [0,1,2,3,4,5,6,7,8])

df.head(5) #check the dataset again
```

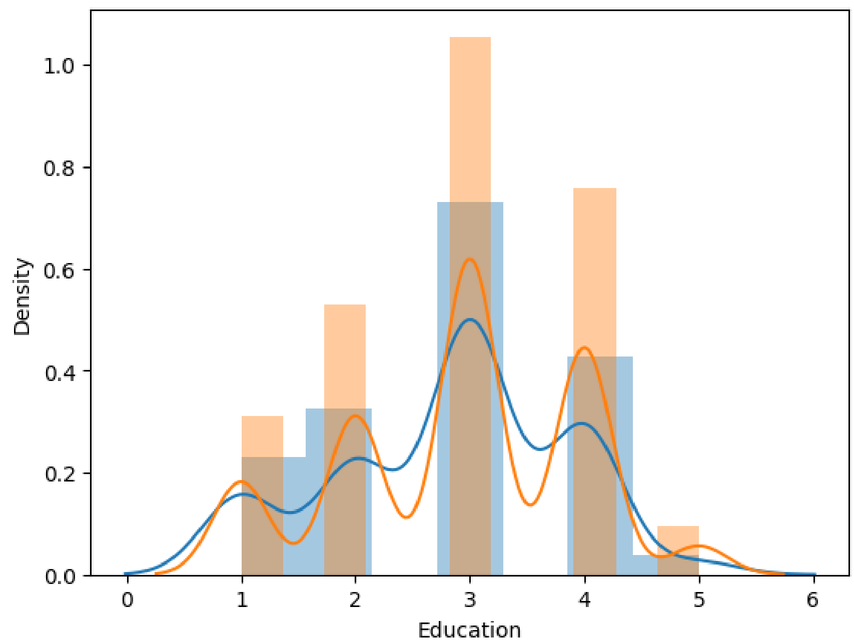
Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField
41	1	2	1102	2	1	2	1
49	0	1	279	1	8	1	1
37	1	2	1373	1	2	2	5
33	0	1	1392	1	3	4	1
27	0	2	591	1	2	1	3

ws × 35 columns

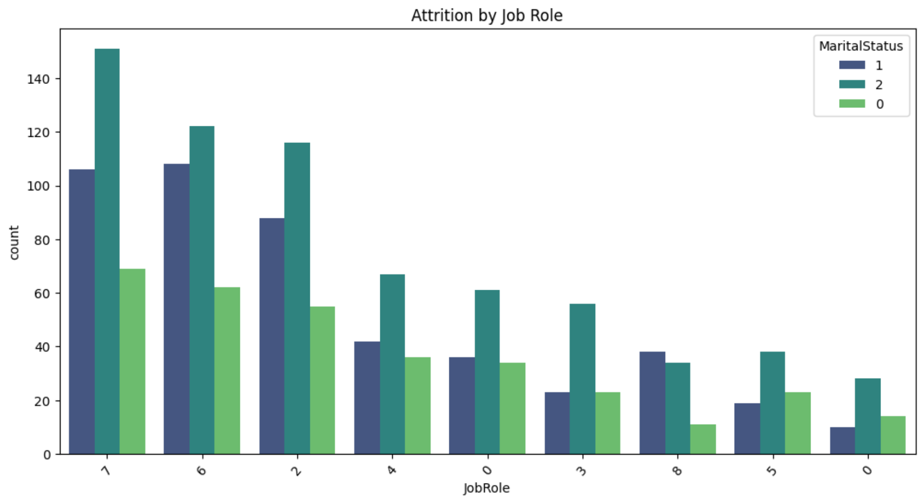
```
#DistanceFromHome and Attrition by histogram with boxplot
plt.figure(figsize=(10,6))
sns.histplot(data=df, x='DistanceFromHome', hue='Attrition',kde='True')
plt.title('Age Distribution by Attrition')
plt.show() #Here we can see the left employees have small distance numbers
from home and most of them their home very close from the company with numbers
histated with around 0-10 Klg
```



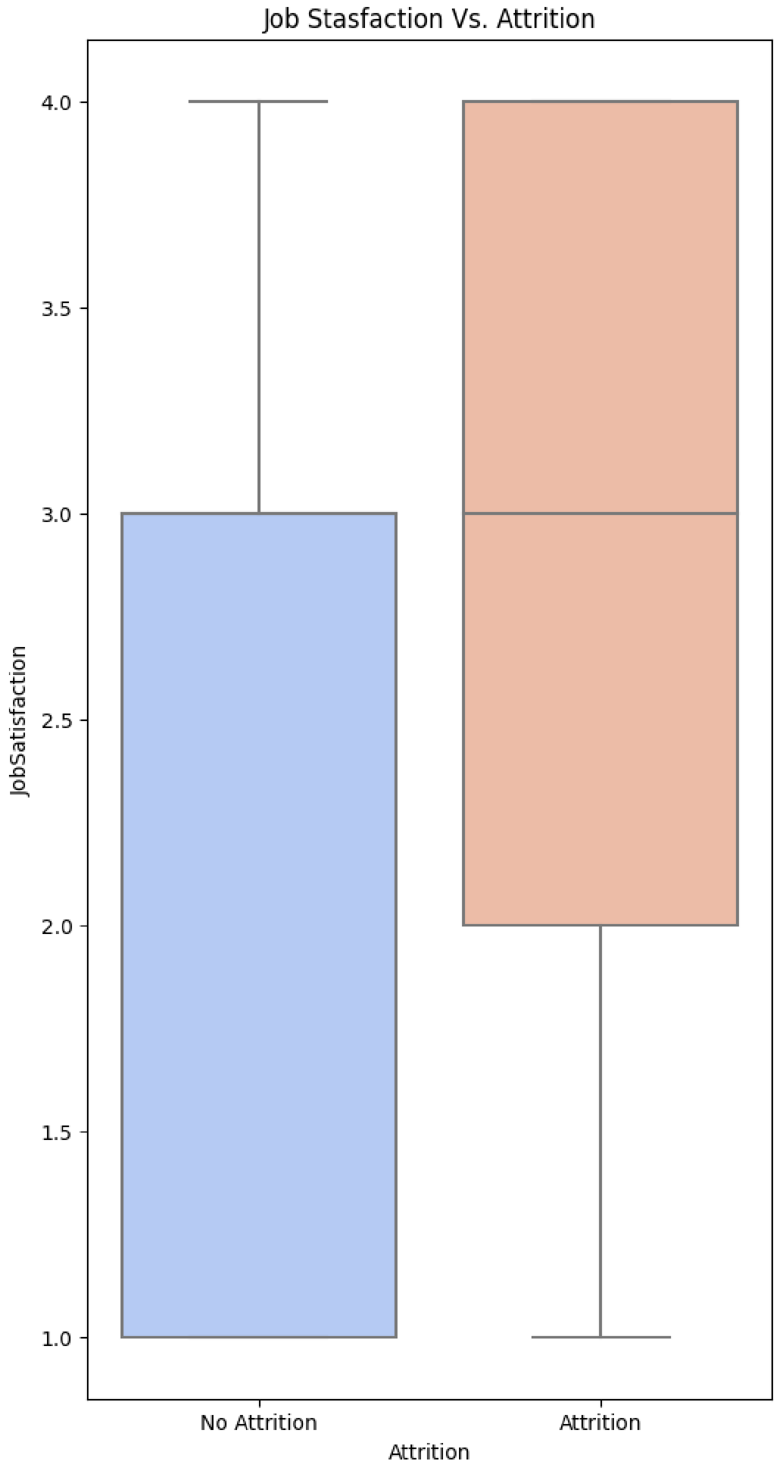

```
#
sns.distplot(df.loc[df['Attrition']=='1']['Education'])
sns.distplot(df.loc[df['Attrition']=='0']['Education']); #We can realised from this distribution plot that most of employees had Bachelor degree wether they were lefot or not, in the 2nd level they have Master degree

##
```



```python
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='JobRole', hue='MaritalStatus', palette='viridis')
plt.title('Attrition by Job Role')
plt.xticks(rotation=50)
plt.show()
''' we can see most married employees were worked
in Sales Executive postition with more then 140
person, then in the same position around 100 person
were single and around 60 were divorced. Its clear
that the top 3 positions played by most of employees
beside the Sales Executive were Research Scientist
and Lavoratory Technicuian while for the most
marital status for the employees was married then
single and divorsed employees were shaped the lowest
proprtion. Finally we can mentioned that the Humen
Resources position was the lowest role was played by
the employees '''
```

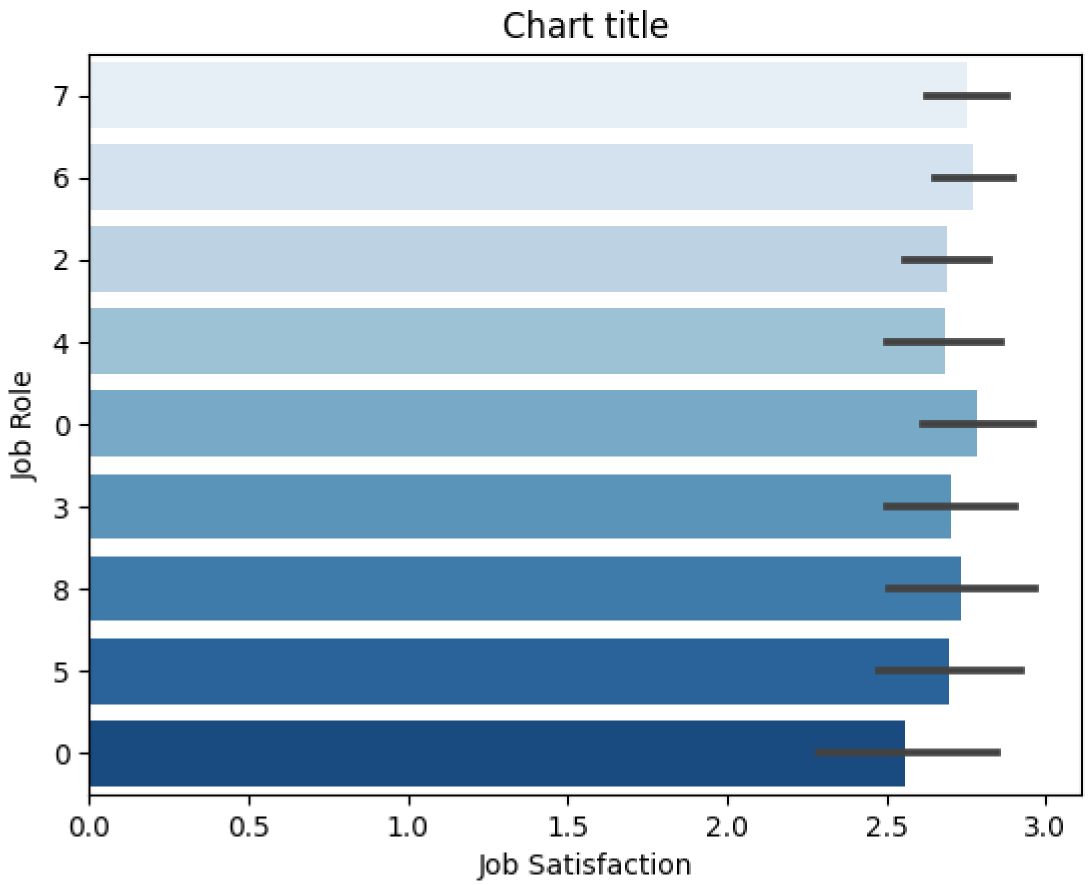



```python
plt.figure(figsize=(6,12))
sns.boxplot(data=df, x='Attrition',y='JobSatisfaction',palette='coolwarm')
plt.title('Job Stasfaction Vs. Attrition')
plt.xticks([0,1],['No Attrition','Attrition'])
plt.show()
''' we can see that ppl who stay the company 50% of
them have medium to high level of satisfaction and
the rest have high to very high level of
sastisfaction. Then the left employees were all of
them had low to medium lecvel of satisfaction '''
```

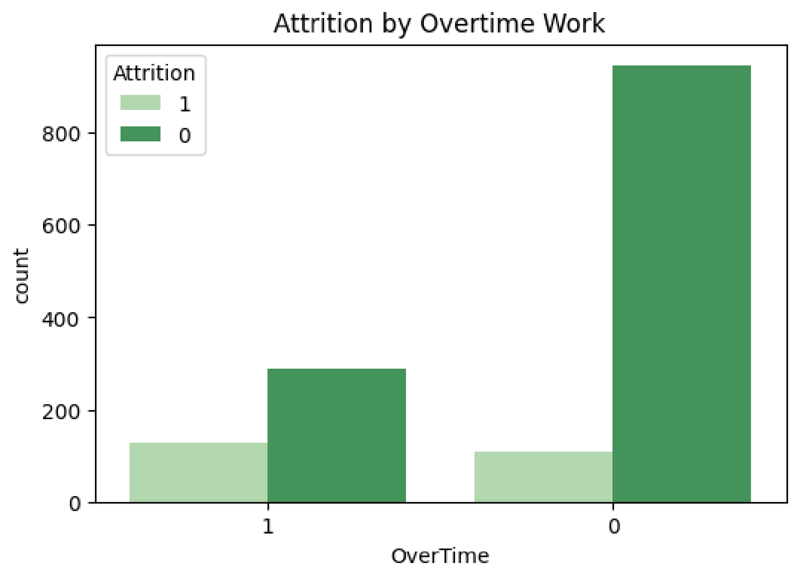



```
#Creating bar plot
sns.barplot(x = 'JobSatisfaction',y = 'JobRole',data = df ,palette = "Blues")
#Adding the aesthetics
plt.title('Chart title')
plt.xlabel('Job Satisfaction')
plt.ylabel('Job Role')
# Show the plot
plt.show()
''' we can see here part of workers in Humen Resources& workers in
```

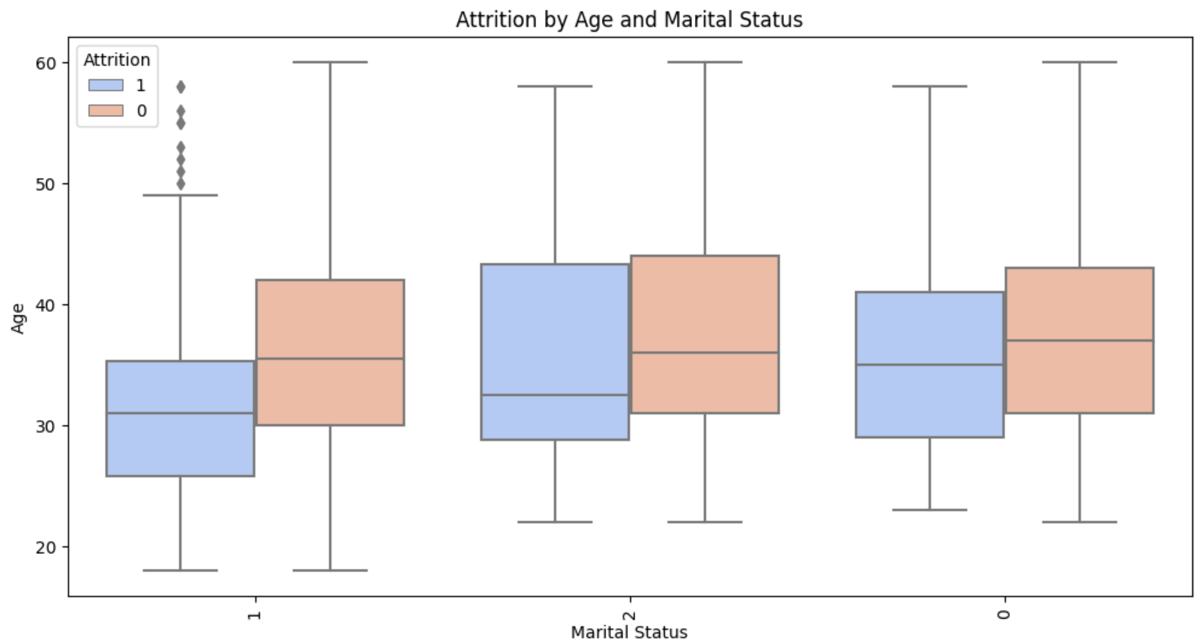
```
Sales Representative and Sales Scientist employees were vary satisfied. '''
```




```
#Attrition by overtime work
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='OverTime', hue='Attrition', palette= 'Greens')
plt.title('Attrition by Overtime Work')
plt.show()
''' We can noticed that between 200-400 left ppl
were having overtime work, while arund 100 dont have.
'''
```

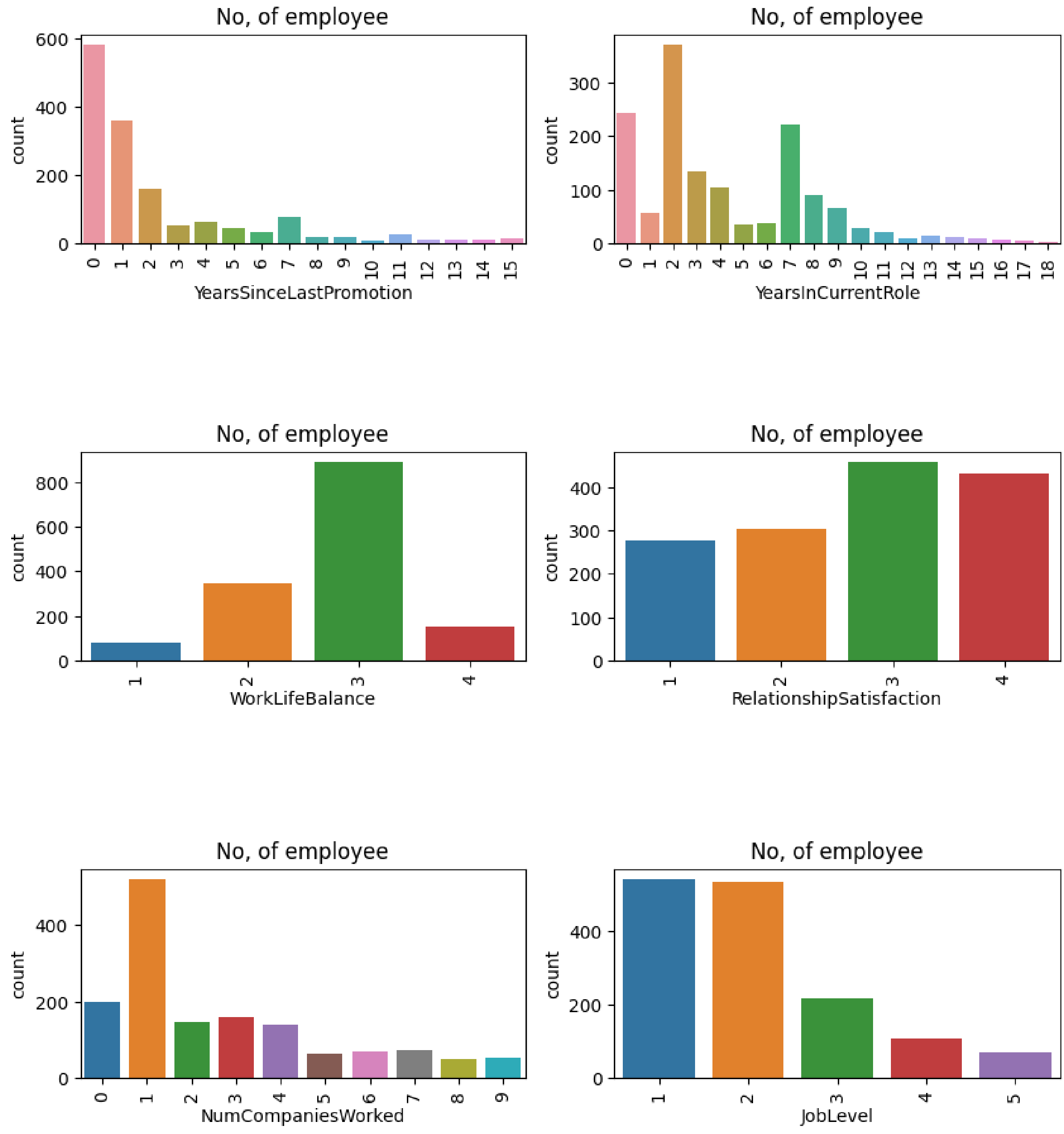



```
# Attrition by Marital status and Age
plt.figure(figsize=(12,6))
sns.boxplot(data=df, x='MaritalStatus',y='Age',hue='Attrition',
palette='coolwarm')
plt.title('Attrition by Age and Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Age')
plt.xticks(rotation=90)
plt.show()
```

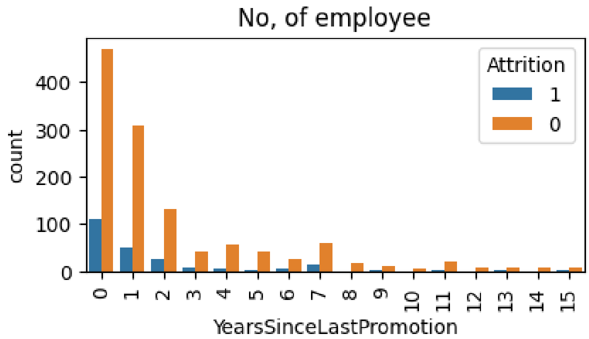


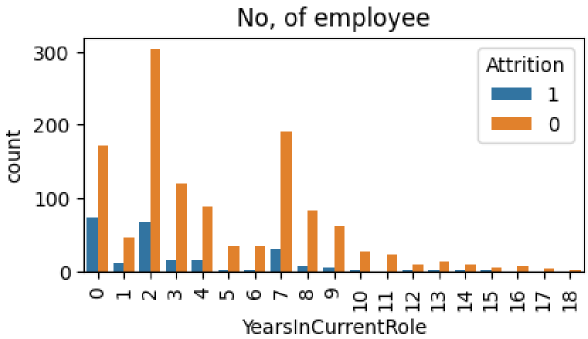

```
features= ['YearsSinceLastPromotion','YearsInCurrentRole',
'WorkLifeBalance','RelationshipSatisfaction','NumCompaniesWorked','JobLevel']
fig= plt.subplots(figsize=(10,15))
for i, j in enumerate(features):
    plt.subplot(4, 2, i+1)
```

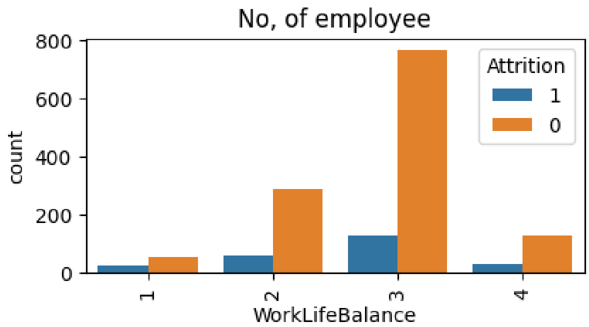
```
    plt.subplots_adjust(hspace=1.0)
    sns.countplot(x=j,data=df)
    plt.xticks(rotation=90)
    plt.title('No, of employee')
```

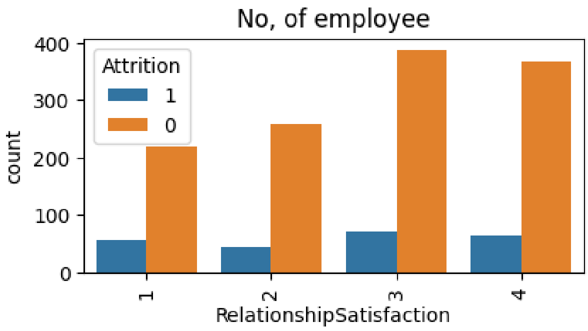


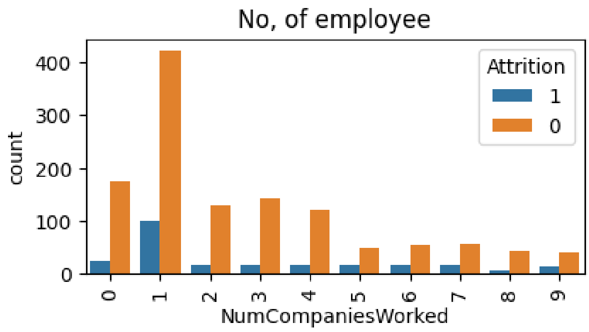

```
fig= plt.subplots(figsize=(10,15))
for i, j in enumerate(features):
    plt.subplot(4, 2, i+1)
    plt.subplots_adjust(hspace=1.0)
    sns.countplot(x=j,data=DF, hue='Attrition')
    plt.xticks(rotation=90)
    plt.title('No, of employee'
```












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

DF.info()
''' This dataset had 1470 samples and 32 attributes,
(24 integer + 8 objects ) No variables have non null/
missing values'''
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 32 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   EmployeeNumber           1470 non-null   int64
 9   EnvironmentSatisfaction  1470 non-null   int64
```

```
 9   EnvironmentSatisfaction  1470 non-null   int64
 10  Gender                   1470 non-null   object
 11  HourlyRate               1470 non-null   int64
 12  JobInvolvement           1470 non-null   int64
 13  JobLevel                 1470 non-null   int64
 14  JobRole                  1470 non-null   object
 15  JobSatisfaction          1470 non-null   int64
 16  MaritalStatus            1470 non-null   object
 17  MonthlyIncome            1470 non-null   int64
 18  MonthlyRate              1470 non-null   int64
 19  NumCompaniesWorked       1470 non-null   int64
...
 30  YearsSinceLastPromotion  1470 non-null   int64
 31  YearsWithCurrManager     1470 non-null   int64
dtypes: int64(24), object(8)
```

```
DF.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSat
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	147
mean	36.923810	802.485714	9.192517	2.912925	1024.865306	
std	9.135373	403.509100	8.106864	1.024165	602.024335	
min	18.000000	102.000000	1.000000	1.000000	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	491.250000	
50%	36.000000	802.000000	7.000000	3.000000	1020.500000	
75%	43.000000	1157.000000	14.000000	4.000000	1555.750000	
max	60.000000	1499.000000	29.000000	5.000000	2068.000000	

- **Splitting the dataset into Training and test datasets:**

```
X= DF[['BusinessTravel','Department','DistanceFromHome','HourlyRate'
,'JobLevel', 'JobRole', 'JobSatisfaction', 'MonthlyIncome',
'OverTime','TotalWorkingYears','WorkLifeBalance', 'YearsAtCompany',
'YearsInCurrentRole',
       'YearsSinceLastPromotion']]
y= DF['Attrition']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y ,test_size=0.3,
random_state=42, )

-Modelling :

from sklearn.ensemble import GradientBoostingClassifier,
RandomForestClassifier
from sklearn.svm import SVC

#GB
Model1= GradientBoostingClassifier()
Model1.fit(x_train,y_train)
```

```
gb_y_pred= Model1.predict(x_test)

#RF
Model2= RandomForestClassifier()
Model2.fit(x_train,y_train)
rf_y_pred = Model2.predict(x_test)

#SVM

Model3=SVC()
Model3.fit(x_train,y_train)
svc_y_pred= Model3.predict(x_test)

#Evaluating
from sklearn.metrics import accuracy_score, precision_score, recall_score
Model1= GradientBoostingClassifier()
Model1.fit(x_train,y_train)
gb_y_pred= Model1.predict(x_test)

Models ={
    'GradientBoostingClassifier' : gb_y_pred,
    'RandomForestClassifier' : rf_y_pred,
    'SVM' : svc_y_pred
}

models= pd.DataFrame(Models)

for i in models:
    acc= accuracy_score(y_test, models[i])
    prec= precision_score(y_test, models[i],pos_label='1')
    recall= recall_score(y_test, models[i], pos_label='1')
    results= pd.DataFrame([[i,acc,prec,recall]],
                          columns = ['model','accuracy','precision','recall'
])
    print(results)
```

```
                        model  accuracy  precision    recall
  0  GradientBoostingClassifier  0.843537   0.357143  0.163934
                    model  accuracy  precision    recall
  0  RandomForestClassifier  0.861678        0.5  0.163934
    model  accuracy  precision  recall
  0   SVM  0.861678        0.0     0.0
```

Depending on the accuracy metric we will choose the RandomForest Classifier with high accuracy value (0.86)

Diabetes Analysis Project

Using Python- Visual basics

```
'''Pregnancies: Number of times pregnant
Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance
test
BloodPressure: Diastolic blood pressure (mm Hg)
SkinThickness: Triceps skin fold thickness (mm)
Insulin: 2-Hour serum insulin (mu U/ml)
BMI: Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction: Diabetes pedigree function
Age: Age (years)
Outcome: Class variable (0 or 1)'''
```

```
# import libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.plotting import plot_decision_regions
import missingno as msno
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler

df= pd.read_csv('C:/Users/LENOVO/Desktop/internship/diabetes.csv')
df.head()
```

✓ 0.0s

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

```
df.shape
```

✓ 0.0s

```
(768, 9)
```

```
nullValues = df.isnull().sum().sum()#EDA : is to identify the pattterns
through different data visualization
nullValues
```

✓ 0.0s

```
0
```

```
duplicatedValues= df.duplicated().sum()
duplicatedValues
```

✓ 0.0s

```
0
```

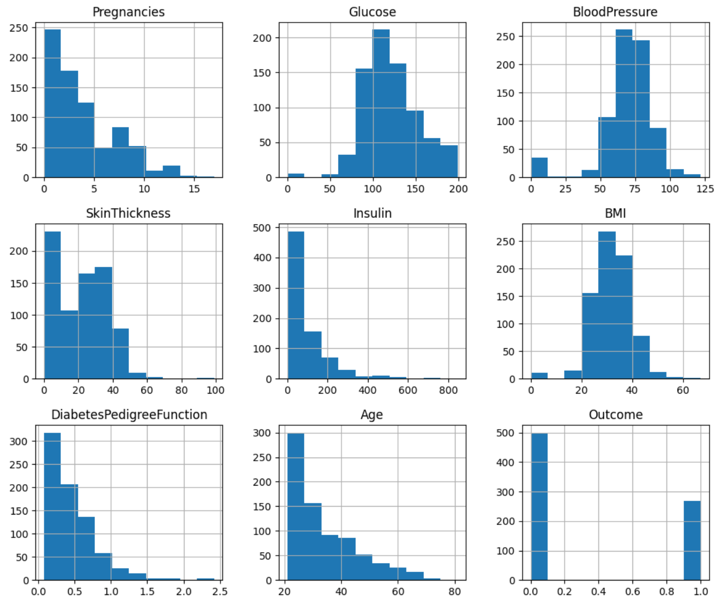
```
df.isnull().sum()
```

✓ 0.0s

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

After we discover about the null values and we did not find any, we should also replace 0 values with NAN. After we applied that we can see that in the result that there are many of null values.

```
df_copy = df.copy(deep=True)
df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']]=df_copy[[
'Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NAN)
p= df.hist(figsize=(12,10))
```



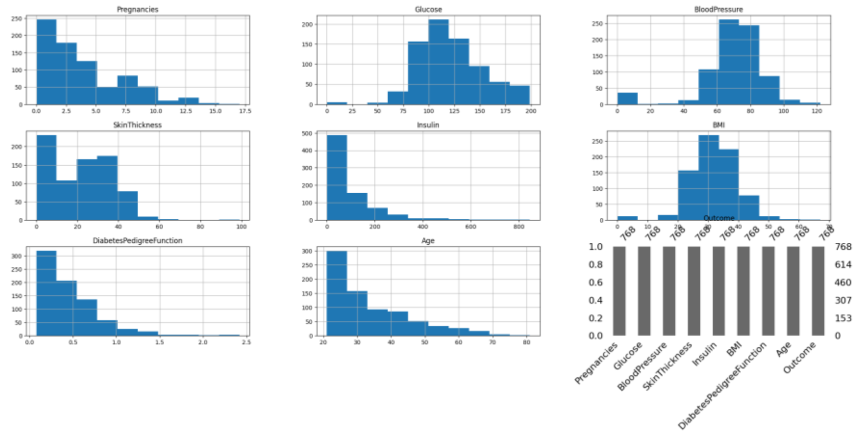
```
df_copy.isnull().sum()
```

✓ 0.0s

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

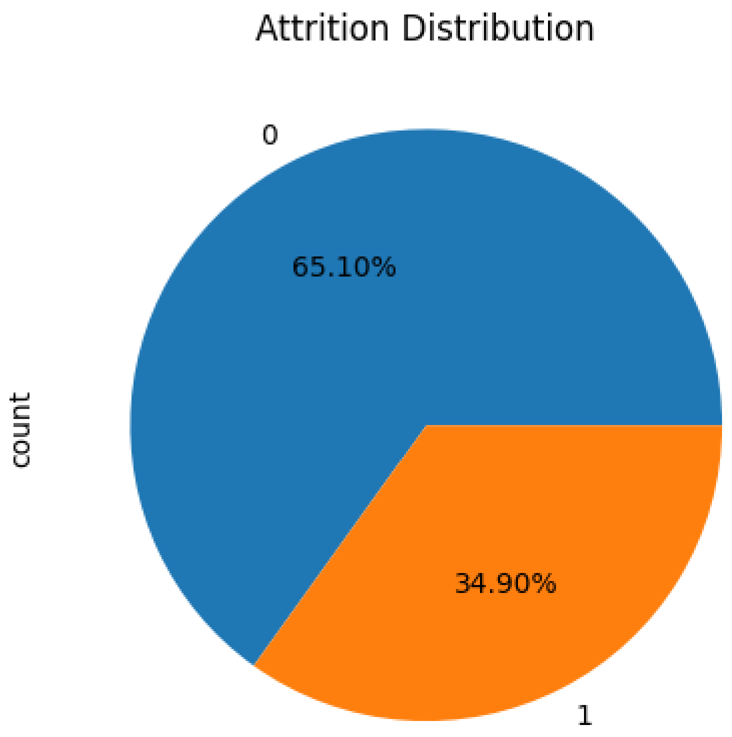
Now we should replace the null values with mean and median, we can see from the histogram plots that the normal histograms like BloodPressure and Glucose have normal distribution so we will impute the nulls with mean and the rest of the independents have skewed distribution so we'll replace it with mean.

```
df_copy['BloodPressure'].fillna(df_copy['BloodPressure'].mean(),inplace=True)
df_copy['Glucose'].fillna(df_copy['Glucose'].mean(),inplace=True)
df_copy['Insulin'].fillna(df_copy['Insulin'].median(),inplace=True)
df_copy['SkinThickness'].fillna(df_copy['SkinThickness'].median(),inplace=True
)
df_copy['BMI'].fillna(df_copy['BMI'].median(),inplace=True)
p= df.hist(figsize=(12,10))
p=msno.bar(df_copy)
```




```
'''
# one way to count values for our targer column here which is 'Out come'
there are two variables 0 for all who dont have the diabese , and 1 for who
have

color_wheel = {1: "£0392cf", 3: "£7bc043"}
colors= df['Outcome'].map(lambda x: color_wheel.get(x+1))
print(df.Outcome.value_counts())
p=df.Outcome.value_counts().plot(kind='bar')

'''
```

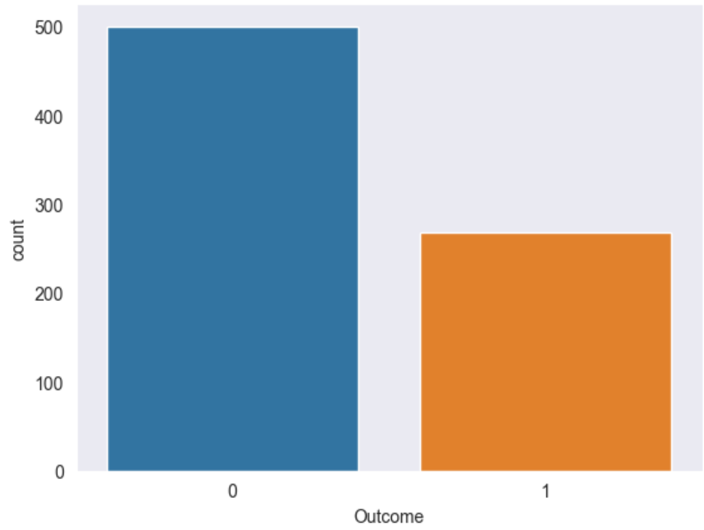
```
# Used way will be as follow:
plt.title('Attrition Distribution')
df['Outcome'].value_counts().plot.pie(autopct='%1.2f%%')
```




```
#compare btw count of employees who have attririon and who'r not
categorical_count = df['Outcome'].value_counts().to_frame()
categorical_count # by value_counts function i breakdown the Attrition column
into two values Yes and No to countvalues for each
```

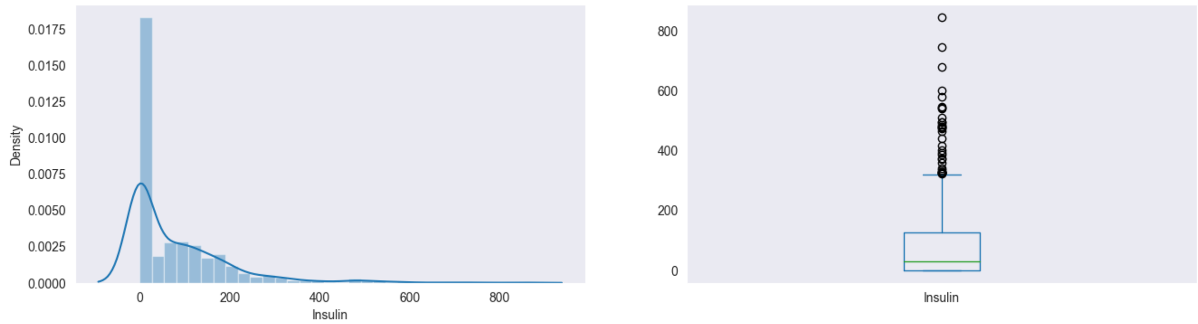
Outcome	count
0	500
1	268

```
sns.set_style('dark')
sns.countplot(x='Outcome',data=df) #Initial plot to compare btw yes and no who
are left the company and who'r not. So we realized tht who were stayed more
than who were left
```



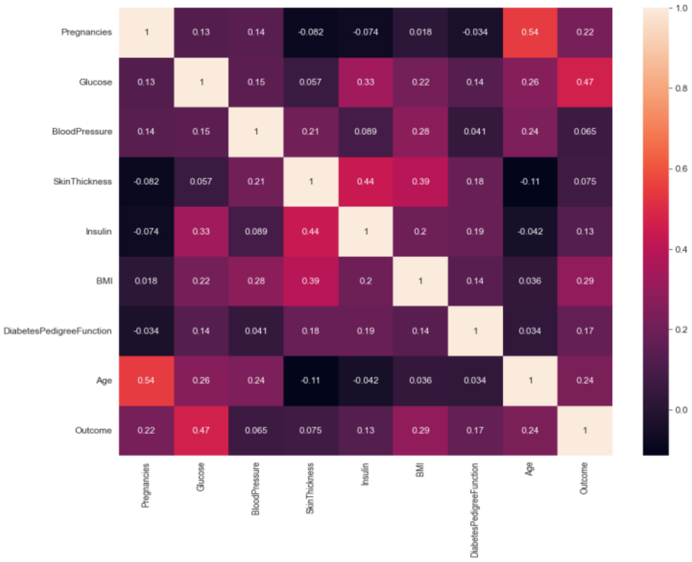
From these comparison plots, pie chart and histogram we can realize that diabetic proportion just shaped 34.90 which means half of the total patients

```
plt.subplot(121), sns.distplot(df['Insulin'])
plt.subplot(122), df['Insulin'].plot.box(figsize=(16,4))
```




```
‘’’So here we can see the distribution pplot for Insuline col, and also we can
realized that there are many  outliers from the other boxplot ’’’
```

```
plt.figure(figsize=(12, 10))
p= sns.heatmap(df.corr(), annot=True)
```



From this heatmap, we can see the correlation btw the features.

```
df_copy.head()
scale = StandardScaler()
x= pd.DataFrame(scale.fit_transform(df_copy.drop(['Outcome'],axis=1),),
columns=['Pregnancies','Insulin','Glucose',
'BloodPressure','SkinThickness','BMI','Age','DiabetesPedigreeFunction'])
```

Splitting and modeling:

```
X= df.drop('Outcome',axis=1)
y = df['Outcome']

x_train, x_test, y_train, y_test = train_test_split(X,y, test_size= 0.3,
random_state=10)
```

- **We will use two classifiers Random Forest and Decision Tree models:**

```
from sklearn.metrics import accuracy_score
#Random Forest :
from sklearn.ensemble import RandomForestClassifier
model1 = RandomForestClassifier(n_estimators=200)
model1.fit(x_train, y_train)
model_tr_pred = model1.predict(x_train)
acc_m1_tr=accuracy_score(model_tr_pred,y_train)
```

= 1.0 its over fitted, so now we will try now to predict depends on the test data:

```
#prediction for test data
```

```
✓ 0.1s

[[138  21]
 [ 28  44]]
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       159
           1       0.68      0.61      0.64        72

    accuracy                           0.79       231
   macro avg       0.75      0.74      0.75       231
weighted avg       0.78      0.79      0.78       231
```

Its also 0.1 for testing the model

```
#model2
from sklearn.tree import DecisionTreeClassifier
model2= DecisionTreeClassifier()
model2.fit(x_train,y_train)
model2_tra_pred=model2.predict(x_train)
acc_m2_tra= accuracy_score(y_train,model2_tra_pred)
acc_m2_tra

=0.1
#prediction for test data

model2_tes_pred = model2.predict(x_test)
model2_tes = accuracy_score(y_test,model2_tes_pred)
model2_tes# 0.67

from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, model2_tes_pred))
print(classification_report(y_test,model2_tes_pred))
```

```
...   [[124  35]
       [ 35  37]]
                  precision    recall  f1-score   support

               0       0.78      0.78      0.78       159
               1       0.51      0.51      0.51        72

        accuracy                           0.70       231
       macro avg       0.65      0.65      0.65       231
    weighted avg       0.70      0.70      0.70       231
```

We can conclude that the accuracy of the Random Forest model with 0.8 . So we can choose it as the best model for predicting.