

importing libraries

```
In [1]: import pandas as pd
pd.set_option('display.max_columns', None)
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: # Reading Dataset
df=pd.read_csv("Attrition data (1).csv")
```

```
In [3]: df.shape
```

```
Out[3]: (4410, 29)
```

```
In [4]: # displays the top 5 columns of the dataframe
df.head()
```

```
Out[4]:
```

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	Gender
0	1	51	No	Travel_Rarely	Sales	6	2	Life Sciences	Female
1	2	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	Female
2	3	32	No	Travel_Frequently	Research & Development	17	4	Other	Male
3	4	38	No	Non-Travel	Research & Development	2	5	Life Sciences	Male
4	5	32	No	Travel_Rarely	Research & Development	10	1	Medical	Male

```
In [5]: ## droping EmployeeID,StandardHours,Over18,EmployeeCount columns
df=df.drop(['EmployeeID','StandardHours','Over18','EmployeeCount'],axis=1)
df.head()
```

```
Out[5]:
```

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	Gender
0	51	No	Travel_Rarely	Sales	6	2	Life Sciences	Female
1	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	Female
2	32	No	Travel_Frequently	Research & Development	17	4	Other	Male
3	38	No	Non-Travel	Research & Development	2	5	Life Sciences	Male
4	32	No	Travel_Rarely	Research & Development	10	1	Medical	Male

Understanding the Dataset

```
In [6]: # no of rows and columns in the dataframe  
df.shape
```

```
Out[6]: (4410, 25)
```

```
In [7]: # column names of the dataframe  
df.columns
```

```
Out[7]: Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',  
               'Education', 'EducationField', 'Gender', 'JobLevel', 'JobRole',  
               'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',  
               'PercentSalaryHike', 'StockOptionLevel', 'TotalWorkingYears',  
               'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion',  
               'YearsWithCurrManager', 'EnvironmentSatisfaction', 'JobSatisfaction',  
               'WorkLifeBalance', 'JobInvolvement', 'PerformanceRating'],  
              dtype='object')
```

```
In [8]: # data types of each column in the dataframe  
df.dtypes
```

```
Out[8]: Age                  int64  
Attrition            object  
BusinessTravel        object  
Department            object  
DistanceFromHome     int64  
Education             int64  
EducationField        object  
Gender                object  
JobLevel              int64  
JobRole               object  
MaritalStatus         object  
MonthlyIncome          int64  
NumCompaniesWorked    float64  
PercentSalaryHike     int64  
StockOptionLevel      int64  
TotalWorkingYears     float64  
TrainingTimesLastYear int64  
YearsAtCompany         int64  
YearsSinceLastPromotion int64  
YearsWithCurrManager  int64  
EnvironmentSatisfaction float64  
JobSatisfaction       float64  
WorkLifeBalance        float64  
JobInvolvement         int64  
PerformanceRating     int64  
dtype: object
```

```
In [9]: # info of the dataframe  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 4410 entries, 0 to 4409  
Data columns (total 25 columns):  
 #   Column           Non-Null Count  Dtype     
---  --  
 0   Age              4410 non-null    int64    
 1   Attrition        4410 non-null    object    
 2   BusinessTravel   4410 non-null    object    
 3   Department       4410 non-null    object    
 4   DistanceFromHome 4410 non-null    int64    
 5   Education        4410 non-null    int64    
 6   EducationField   4410 non-null    object    
 7   Gender            4410 non-null    object    
 8   JobLevel          4410 non-null    int64    
 9   JobRole           4410 non-null    object    
 10  MaritalStatus    4410 non-null    object    
 11  MonthlyIncome    4410 non-null    int64    
 12  NumCompaniesWorked 4391 non-null    float64  
 13  PercentSalaryHike 4410 non-null    int64    
 14  StockOptionLevel  4410 non-null    int64    
 15  TotalWorkingYears 4401 non-null    float64  
 16  TrainingTimesLastYear 4410 non-null    int64    
 17  YearsAtCompany   4410 non-null    int64    
 18  YearsSinceLastPromotion 4410 non-null    int64    
 19  YearsWithCurrManager 4410 non-null    int64    
 20  EnvironmentSatisfaction 4385 non-null    float64  
 21  JobSatisfaction   4390 non-null    float64  
 22  WorkLifeBalance   4372 non-null    float64  
 23  JobInvolvement    4410 non-null    int64    
 24  PerformanceRating 4410 non-null    int64  
dtypes: float64(5), int64(13), object(7)  
memory usage: 861.5+ KB
```

```
In [10]: ## giving summary stats of dataset  
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%
Age	4410.0	36.923810	9.133301	18.0	30.0	36.0	43.0
DistanceFromHome	4410.0	9.192517	8.105026	1.0	2.0	7.0	14.0
Education	4410.0	2.912925	1.023933	1.0	2.0	3.0	4.0
JobLevel	4410.0	2.063946	1.106689	1.0	1.0	2.0	3.0
MonthlyIncome	4410.0	65029.312925	47068.888559	10090.0	29110.0	49190.0	83800.0
NumCompaniesWorked	4391.0	2.694830	2.498887	0.0	1.0	2.0	4.0
PercentSalaryHike	4410.0	15.209524	3.659108	11.0	12.0	14.0	18.0
StockOptionLevel	4410.0	0.793878	0.851883	0.0	0.0	1.0	1.0
TotalWorkingYears	4401.0	11.279936	7.782222	0.0	6.0	10.0	15.0
TrainingTimesLastYear	4410.0	2.799320	1.288978	0.0	2.0	3.0	3.0
YearsAtCompany	4410.0	7.008163	6.125135	0.0	3.0	5.0	9.0
YearsSinceLastPromotion	4410.0	2.187755	3.221699	0.0	0.0	1.0	3.0
YearsWithCurrManager	4410.0	4.123129	3.567327	0.0	2.0	3.0	7.0
EnvironmentSatisfaction	4385.0	2.723603	1.092756	1.0	2.0	3.0	4.0
JobSatisfaction	4390.0	2.728246	1.101253	1.0	2.0	3.0	4.0
WorkLifeBalance	4372.0	2.761436	0.706245	1.0	2.0	3.0	3.0
JobInvolvement	4410.0	2.729932	0.711400	1.0	2.0	3.0	3.0
PerformanceRating	4410.0	3.153741	0.360742	3.0	3.0	3.0	3.0



EDA(Exploratory Data Analysis)

```
In [11]: # counting null values in each column of the dataframe  
df.isnull().sum()
```

```
Out[11]: Age          0  
Attrition      0  
BusinessTravel  0  
Department      0  
DistanceFromHome 0  
Education        0  
EducationField    0  
Gender          0  
JobLevel         0  
JobRole          0  
MaritalStatus     0  
MonthlyIncome     0  
NumCompaniesWorked 19  
PercentSalaryHike 0  
StockOptionLevel   0  
TotalWorkingYears  9  
TrainingTimesLastYear 0  
YearsAtCompany     0  
YearsSinceLastPromotion 0  
YearsWithCurrManager 0  
EnvironmentSatisfaction 25  
JobSatisfaction    20  
WorkLifeBalance     38  
JobInvolvement      0  
PerformanceRating    0  
dtype: int64
```

```
In [12]: ## dropping null values  
df.dropna(how='any', inplace=True)
```

```
In [13]: ## checking for null values  
df.isnull().sum()
```

```
Out[13]: Age          0  
Attrition      0  
BusinessTravel  0  
Department      0  
DistanceFromHome 0  
Education        0  
EducationField    0  
Gender          0  
JobLevel         0  
JobRole          0  
MaritalStatus     0  
MonthlyIncome     0  
NumCompaniesWorked 0  
PercentSalaryHike 0  
StockOptionLevel   0  
TotalWorkingYears 0  
TrainingTimesLastYear 0  
YearsAtCompany    0  
YearsSinceLastPromotion 0  
YearsWithCurrManager 0  
EnvironmentSatisfaction 0  
JobSatisfaction    0  
WorkLifeBalance     0  
JobInvolvement      0  
PerformanceRating    0  
dtype: int64
```

```
In [14]: df.shape
```

```
Out[14]: (4300, 25)
```

```
In [15]: # checking duplicates if any  
df.duplicated().any()
```

```
Out[15]: True
```

```
In [16]: ## droping duplicates  
df.drop_duplicates(inplace=True)
```

```
In [17]: df.shape
```

```
Out[17]: (1470, 25)
```

```
In [18]: df.describe().T
```

Out[18]:

	count	mean	std	min	25%	50%	75%
Age	1470.0	36.923810	9.135373	18.0	30.0	36.0	43.0
DistanceFromHome	1470.0	9.192517	8.106864	1.0	2.0	7.0	14.0
Education	1470.0	2.912925	1.024165	1.0	2.0	3.0	4.0
JobLevel	1470.0	2.063946	1.106940	1.0	1.0	2.0	3.0
MonthlyIncome	1470.0	65029.312925	47079.567831	10090.0	29110.0	49190.0	83790.0
NumCompaniesWorked	1470.0	2.693197	2.498009	0.0	1.0	2.0	4.0
PercentSalaryHike	1470.0	15.209524	3.659938	11.0	12.0	14.0	18.0
StockOptionLevel	1470.0	0.793878	0.852077	0.0	0.0	1.0	1.0
TotalWorkingYears	1470.0	11.279592	7.780782	0.0	6.0	10.0	15.0
TrainingTimesLastYear	1470.0	2.799320	1.289271	0.0	2.0	3.0	3.0
YearsAtCompany	1470.0	7.008163	6.126525	0.0	3.0	5.0	9.0
YearsSinceLastPromotion	1470.0	2.187755	3.222430	0.0	0.0	1.0	3.0
YearsWithCurrManager	1470.0	4.123129	3.568136	0.0	2.0	3.0	7.0
EnvironmentSatisfaction	1470.0	2.721769	1.093082	1.0	2.0	3.0	4.0
JobSatisfaction	1470.0	2.728571	1.102846	1.0	2.0	3.0	4.0
WorkLifeBalance	1470.0	2.761224	0.706476	1.0	2.0	3.0	3.0
JobInvolvement	1470.0	2.729932	0.711561	1.0	2.0	3.0	3.0
PerformanceRating	1470.0	3.153741	0.360824	3.0	3.0	3.0	3.0



```
In [19]: df['Gender'].value_counts()
```

Out[19]: Male 882
Female 588
Name: Gender, dtype: int64

```
In [20]: categorical_features=[]
for i in df.columns:
    if(df[i].nunique()<15):
        categorical_features.append(i)
categorical_features
```

```
Out[20]: ['Attrition',
 'BusinessTravel',
 'Department',
 'Education',
 'EducationField',
 'Gender',
 'JobLevel',
 'JobRole',
 'MaritalStatus',
 'NumCompaniesWorked',
 'StockOptionLevel',
 'TrainingTimesLastYear',
 'EnvironmentSatisfaction',
 'JobSatisfaction',
 'WorkLifeBalance',
 'JobInvolvement',
 'PerformanceRating']
```

```
In [21]: numerical_features=[]
for i in df.columns:
    if(df[i].nunique()>=15):
        numerical_features.append(i)
numerical_features
```

```
Out[21]: ['Age',
 'DistanceFromHome',
 'MonthlyIncome',
 'PercentSalaryHike',
 'TotalWorkingYears',
 'YearsAtCompany',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

Data Visualization

```
In [22]: from scipy import stats
from statsmodels.api import qqplot
```

```
In [23]: # function takes a single dataframe column name as argument and
# plots a 1 x 2 subplot contains (pie plot and bar plot)
def plot_categorical(col_name):
    fig, ax = plt.subplots(
        nrows=1, ncols=2, figsize=(14, 5)
    ) # sets 1 x 2 subplots with figure size of (14,5)
    df[col_name].value_counts().plot.pie(
        autopct="%0f%%",
        labels=df[col_name].unique(),
        shadow=True,
        startangle=180,
        ax=ax[0],
    ) # plots pie chart of given column with % percentage

    df[col_name].value_counts().plot.bar(
        ax=ax[1]
    )

    fig.suptitle(col_name.title()) # set plot's title
    plt.show() # shows the plot
```

```
In [24]: # function takes a single dataframe column name as argument and
# plots a 2 x 2 subplot contains (histogram, quantile-quantile plot, kernel density estimate)
def plot_distribution(col_name):
    fig, axes = plt.subplots(
        2, 2, figsize=(15, 10)
    ) # sets 2 x 2 subplots with figure size of (14,10)
    sns.histplot(
        x=col_name, data=df, ax=axes[0][0], bins=20
    ) # displays histogram of the given dataframe column
    qqplot(
        data=df[col_name], ax=axes[0][1], line="45", fit=True, dist=stats.norm
    ) # displays quantile-quantile plot of the given dataframe column
    sns.kdeplot(
        x=col_name, data=df, ax=axes[1][0], fill=True
    ) # displays kernel density estimate plot of the given dataframe column
    sns.kdeplot(
        x=col_name, data=df, ax=axes[1][1], fill=True, cumulative=True
    ) # displays cumulative density plot of the given dataframe column
    plt.suptitle(
        "Distribution of " + col_name
    ) # sets the title for the subplot as "Distribution of <column name>"
    plt.show() # shows the plot
```

```
In [25]: # function takes a single dataframe column name as argument and
# plots a 2 x 2 subplot contains (boxplot, violinplot, swarmplot and barplot)
def plot_stats(col_name):
    fig, axes = plt.subplots(
        2, 2, figsize=(15, 10)
    ) # sets 2 x 2 subplots with figure size of (14,10)
    sns.boxplot(
        y=col_name, x="Attrition", data=df, ax=axes[0][0], palette="viridis_r"
    ) # displays boxplot of the given dataframe column
    sns.violinplot(
        y=col_name, x="Attrition", data=df, ax=axes[0][1], split=True, palette="vi
    ) # displays violinplot of the given dataframe column
    sns.swarmplot(
        y=col_name, x="Attrition", data=df, ax=axes[1][0], palette="viridis_r"
    ) # displays swarmplot of the given dataframe columns
    sns.barplot(
        x=col_name, y="Attrition", data=df, hue="Attrition", palette="viridis_r"
    ) # displays barplot of the given dataframe column respect to 'Attrition' col
    plt.suptitle(
        "Statistics of " + col_name
    ) # sets the title for the subplot as "Statistics of <column name>"
    plt.show() # shows the plot
```

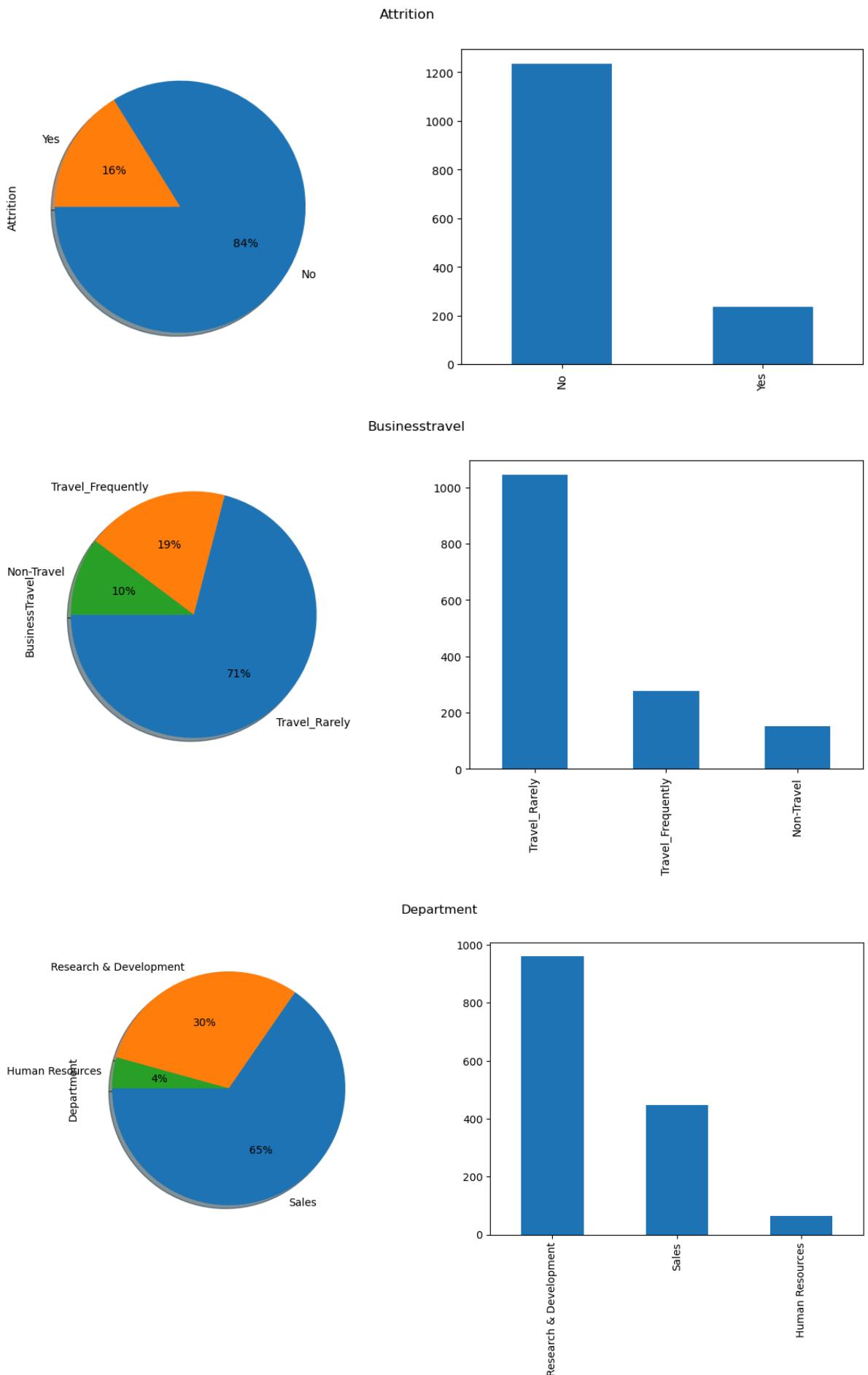
```
In [26]: # function takes a single dataframe column name as argument and
# plots a single plot contains kernel density estimate plot
def plot_kde(col_name):
    facet = sns.FacetGrid(
        df, hue="Attrition", aspect=4, palette="bright"
    ) # sets a seaborn FacetGrid with hue = "Attrition" and palette as "bright"
    facet.map(
        sns.kdeplot, col_name, shade=True
    ) # kernel density plot of the given dataframe column respect to the dependen
    facet.set(
        xlim=(0, df[col_name].max())
    ) # sets the x axis from 0 to maximum value of the column
    facet.add_legend() # add Legend to the plot
    plt.title(
        col_name + " respect to attrition"
    ) # sets the title for the plot as "<column name> respect to attrition"
    plt.show() # shows the plot
```

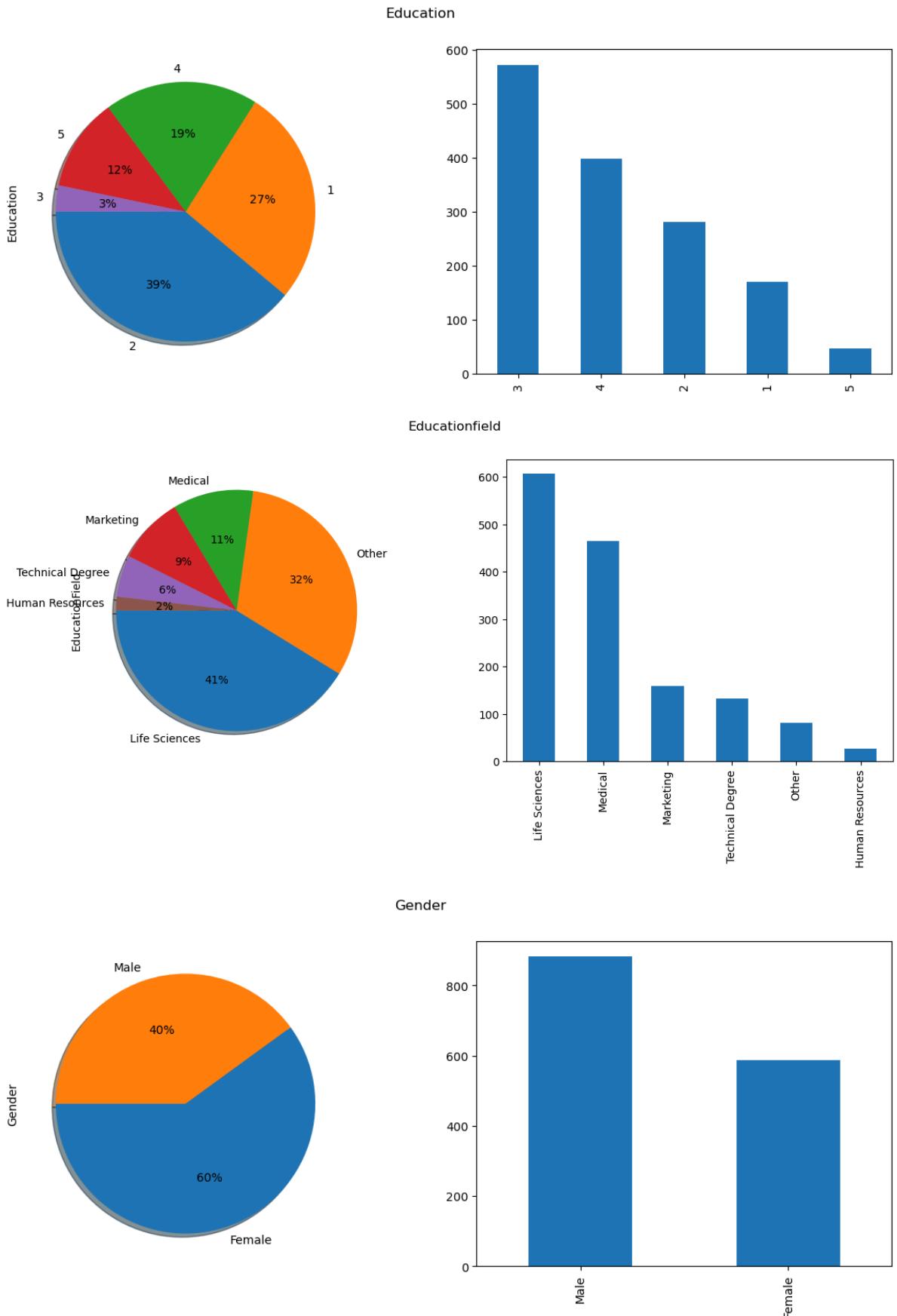
plots a 1 x 2 subplot contains (pie plot and bar plot) for each numerical columns

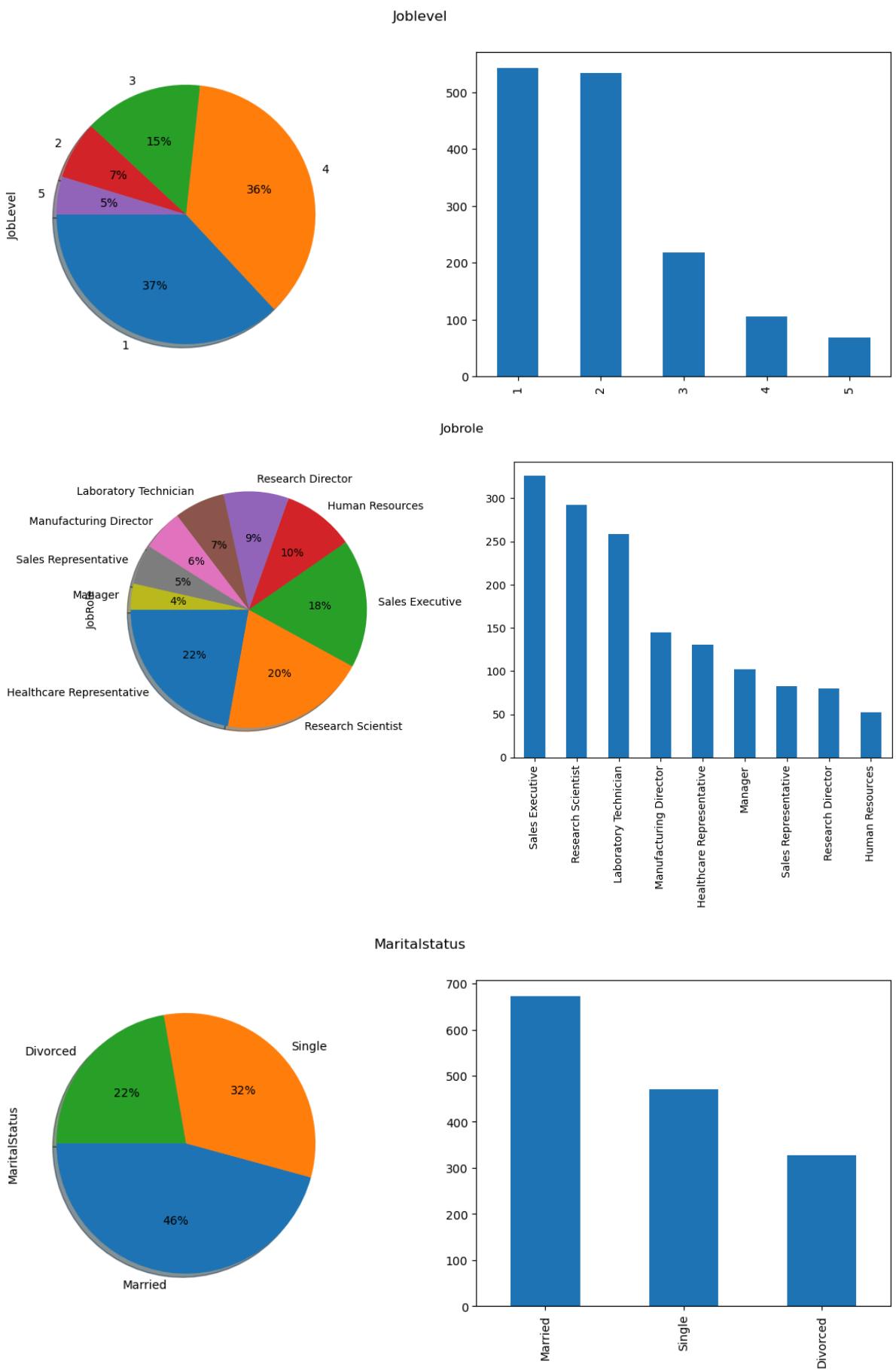
In [27]: categorical_features

Out[27]: ['Attrition',
 'BusinessTravel',
 'Department',
 'Education',
 'EducationField',
 'Gender',
 'JobLevel',
 'JobRole',
 'MaritalStatus',
 'NumCompaniesWorked',
 'StockOptionLevel',
 'TrainingTimesLastYear',
 'EnvironmentSatisfaction',
 'JobSatisfaction',
 'WorkLifeBalance',
 'JobInvolvement',
 'PerformanceRating']

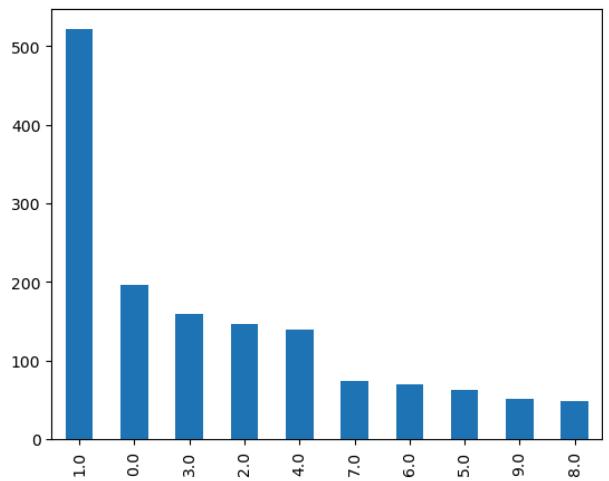
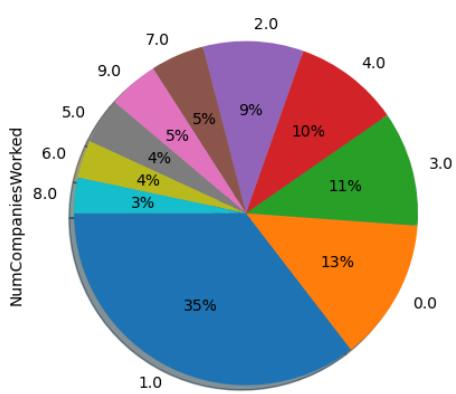
```
In [28]: for i in categorical_features:  
    plot_categorical(i)
```



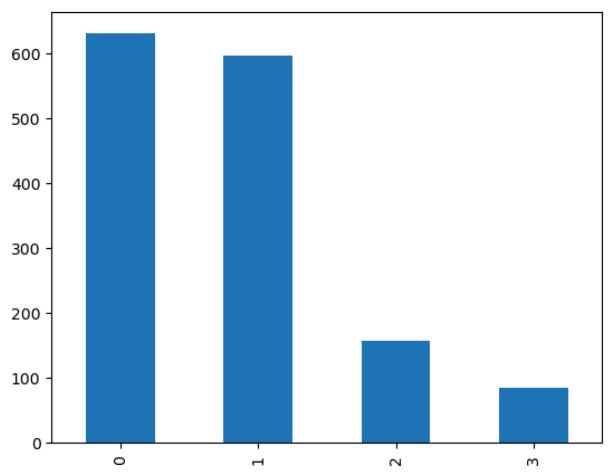
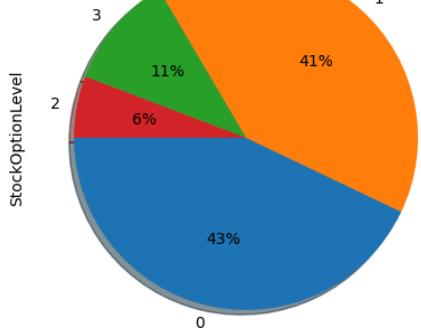




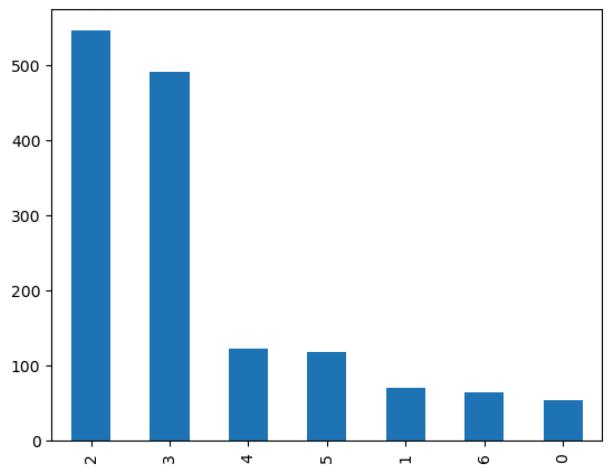
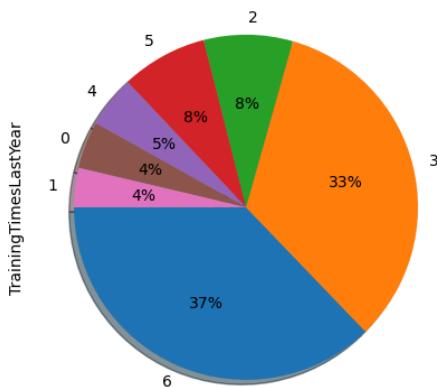
Numcompaniesworked



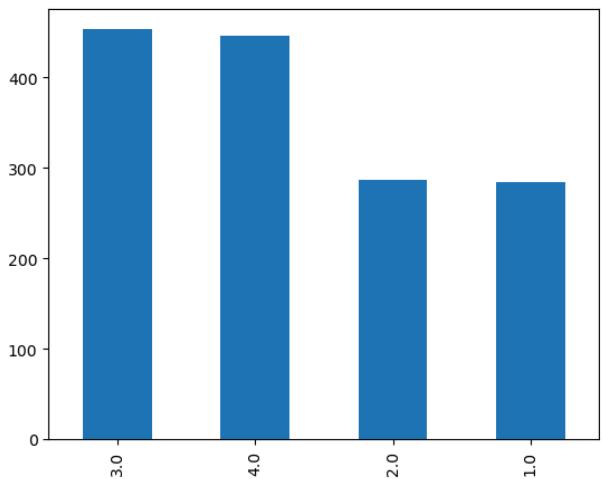
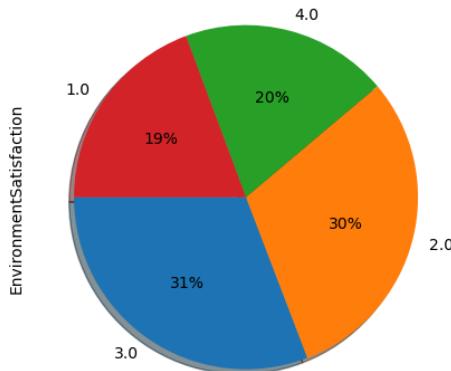
Stockoptionlevel



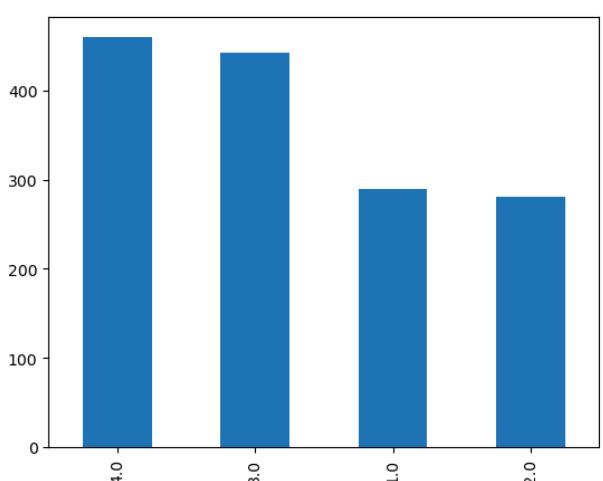
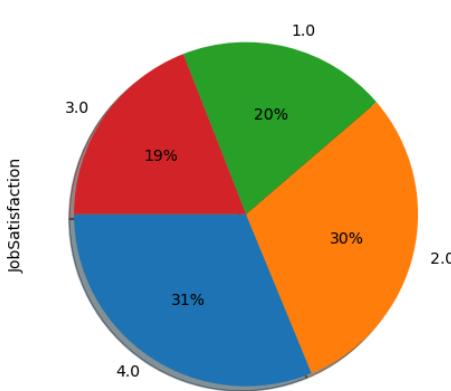
Trainingtimeslastyear



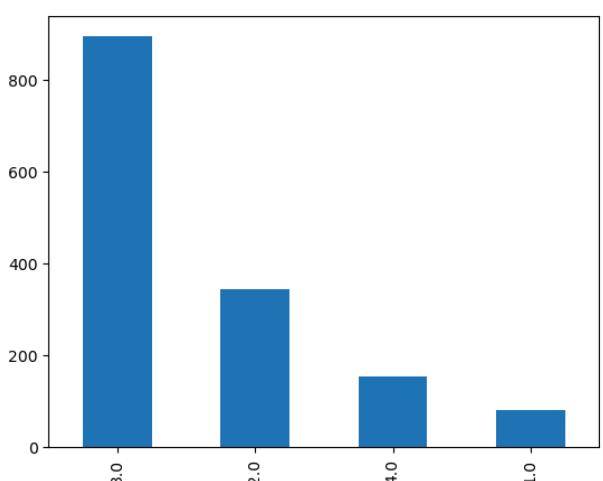
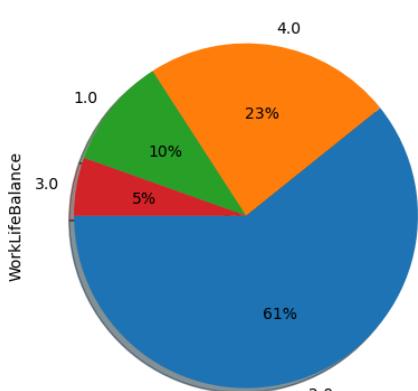
Environment satisfaction

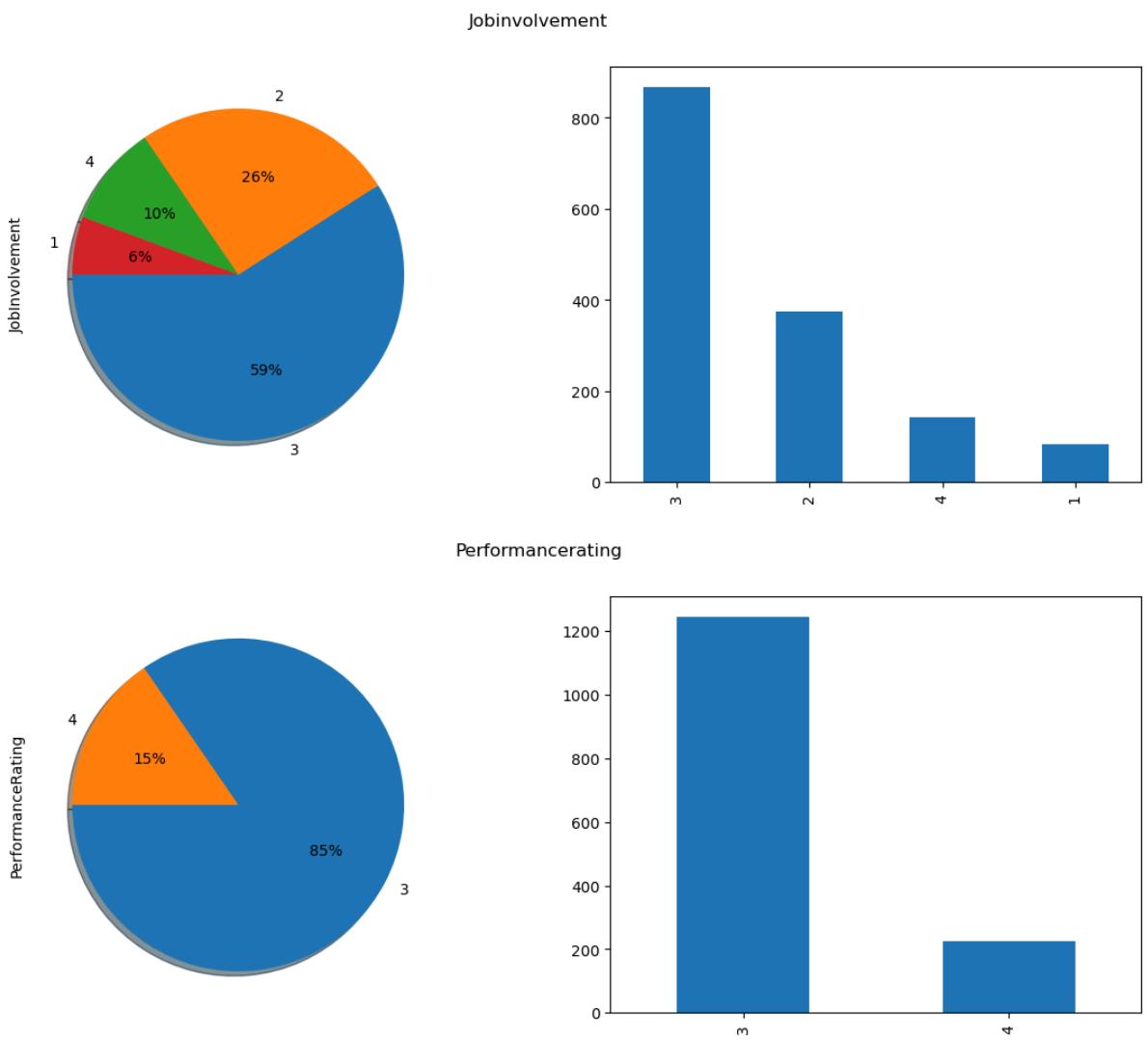


Job satisfaction



Work-life balance

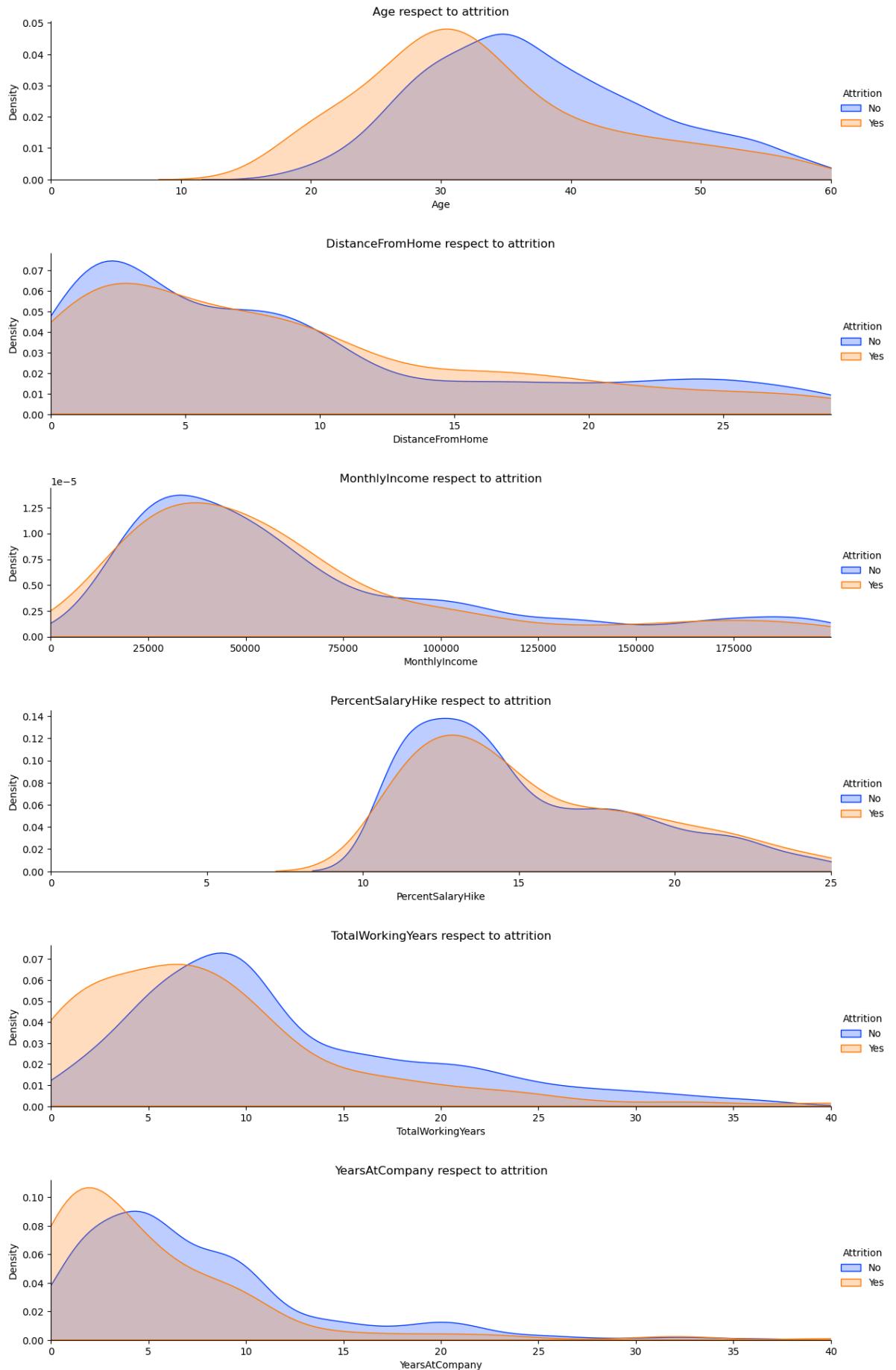


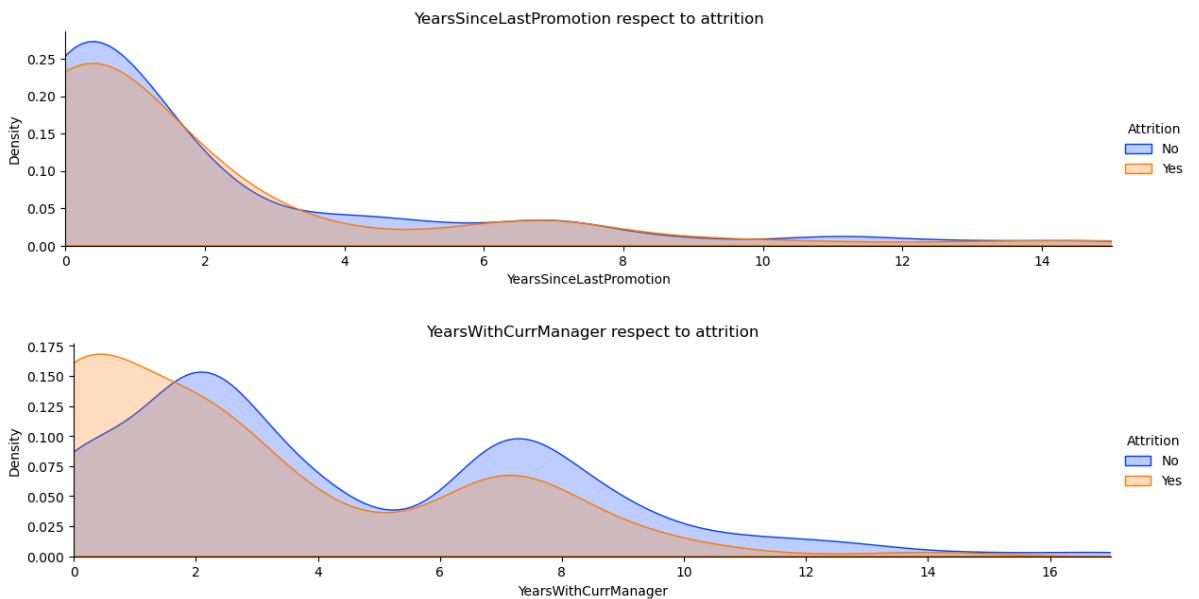


In []:

Kernel density estimate plot for each numerical features

```
In [29]: for i in numerical_features:  
    plot_kde(i)
```



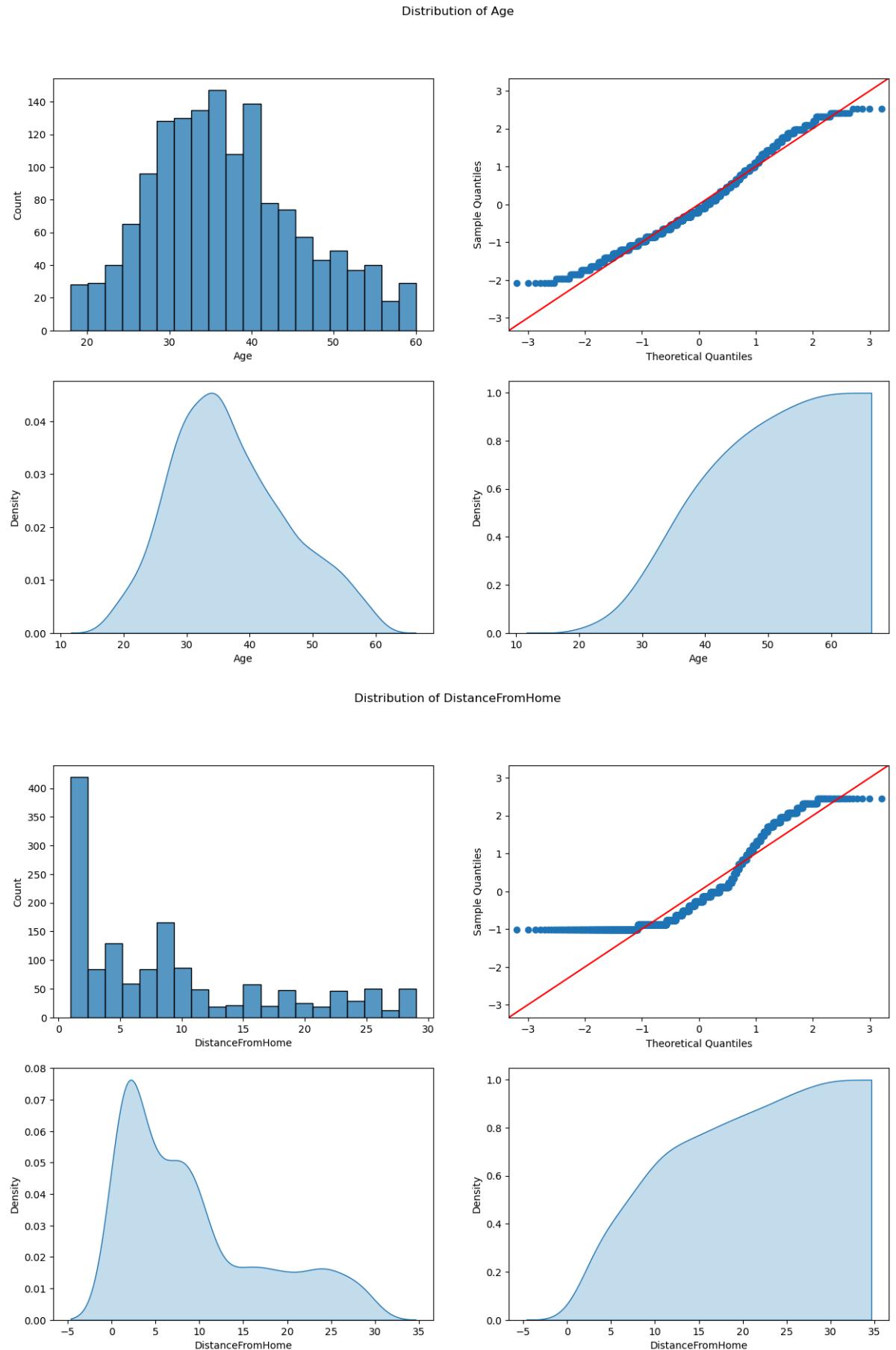


plots a 2 x 2 subplot contains (histogram, quantile-quantile plot, kernel density estimate plot and cumulative density plot) for each numerical feature

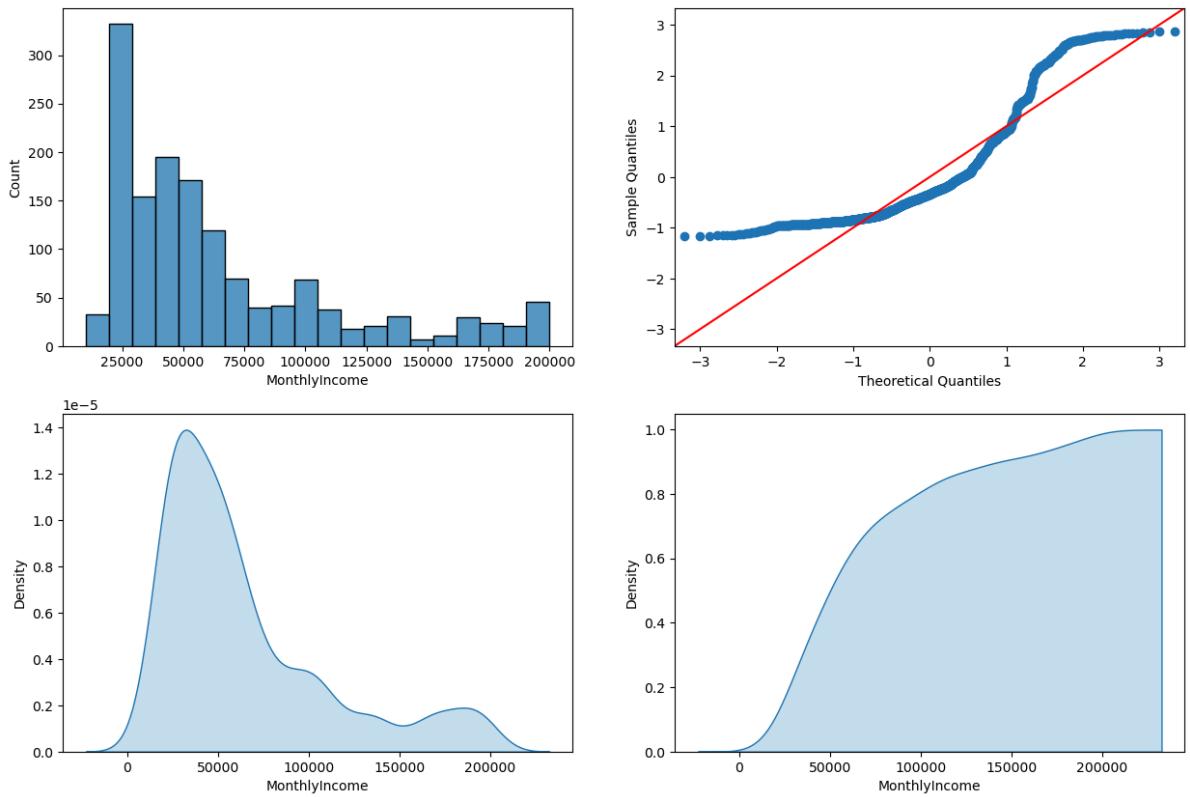
In [30]: numerical_features

Out[30]: ['Age',
 'DistanceFromHome',
 'MonthlyIncome',
 'PercentSalaryHike',
 'TotalWorkingYears',
 'YearsAtCompany',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']

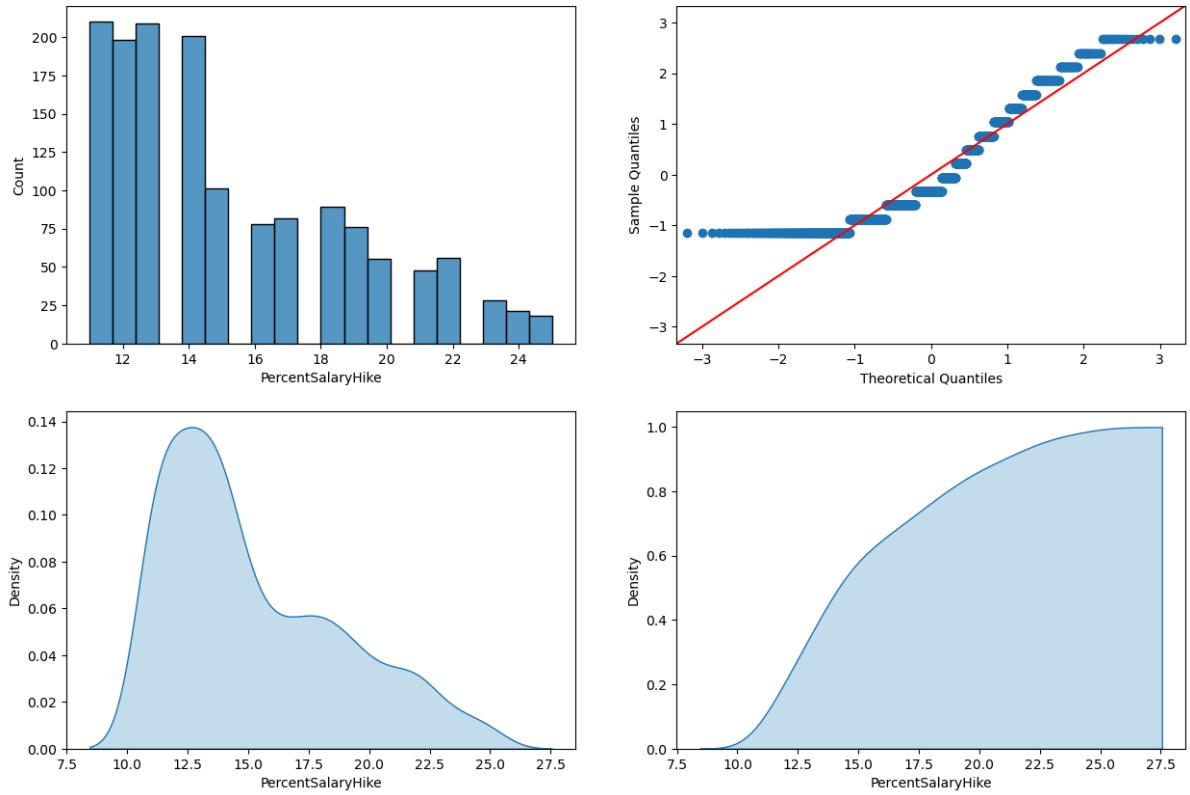
```
In [31]: for i in numerical_features:  
    plot_distribution(i)
```



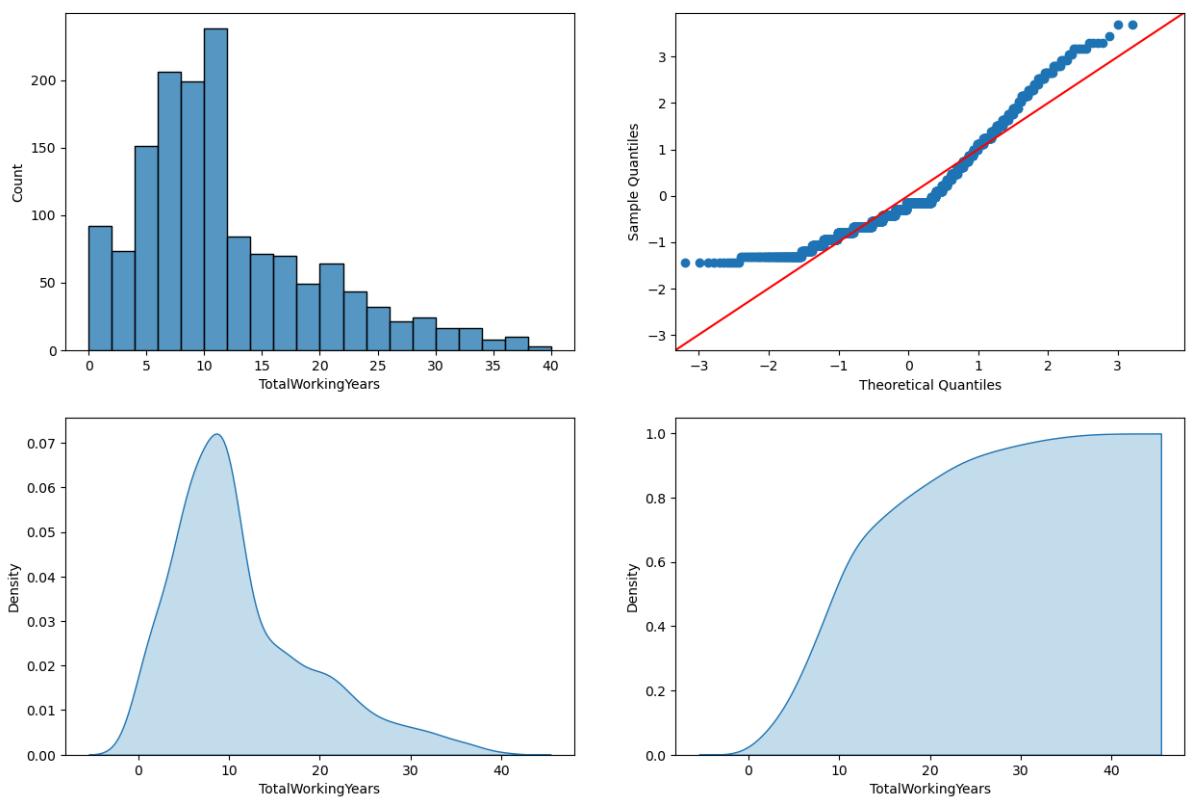
Distribution of MonthlyIncome



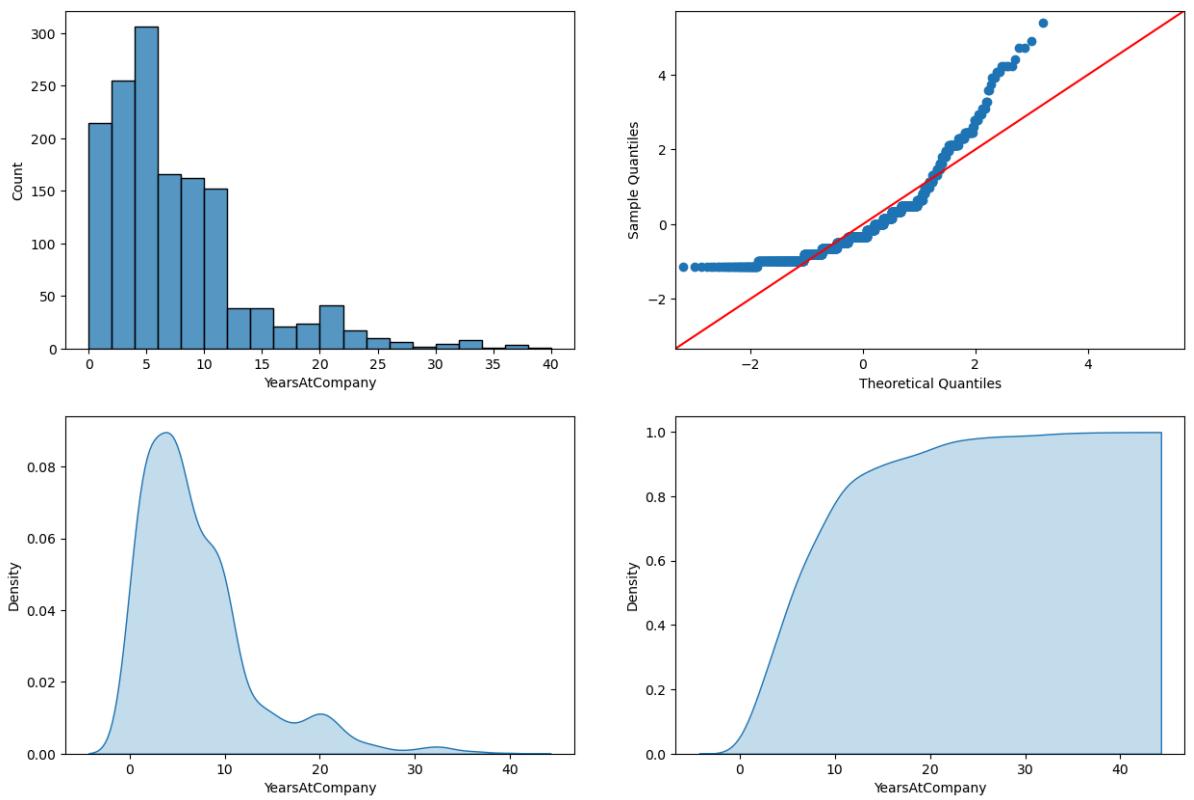
Distribution of PercentSalaryHike



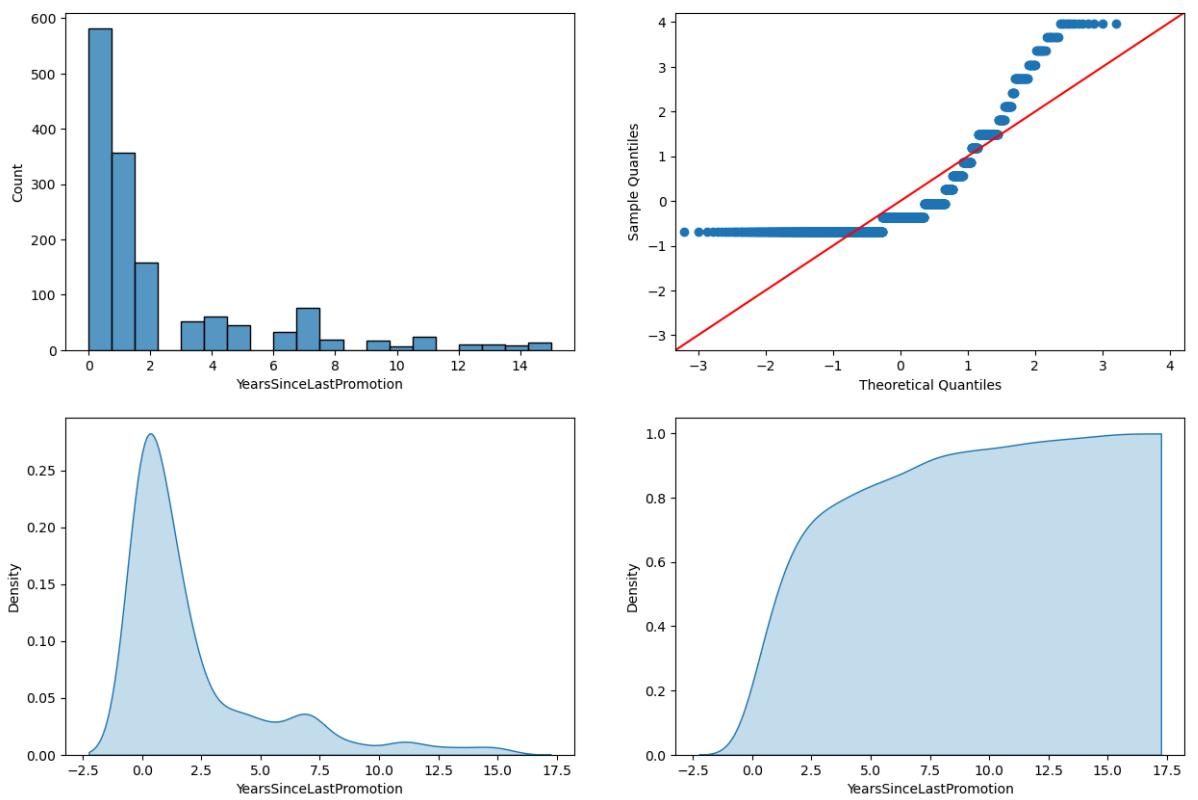
Distribution of TotalWorkingYears



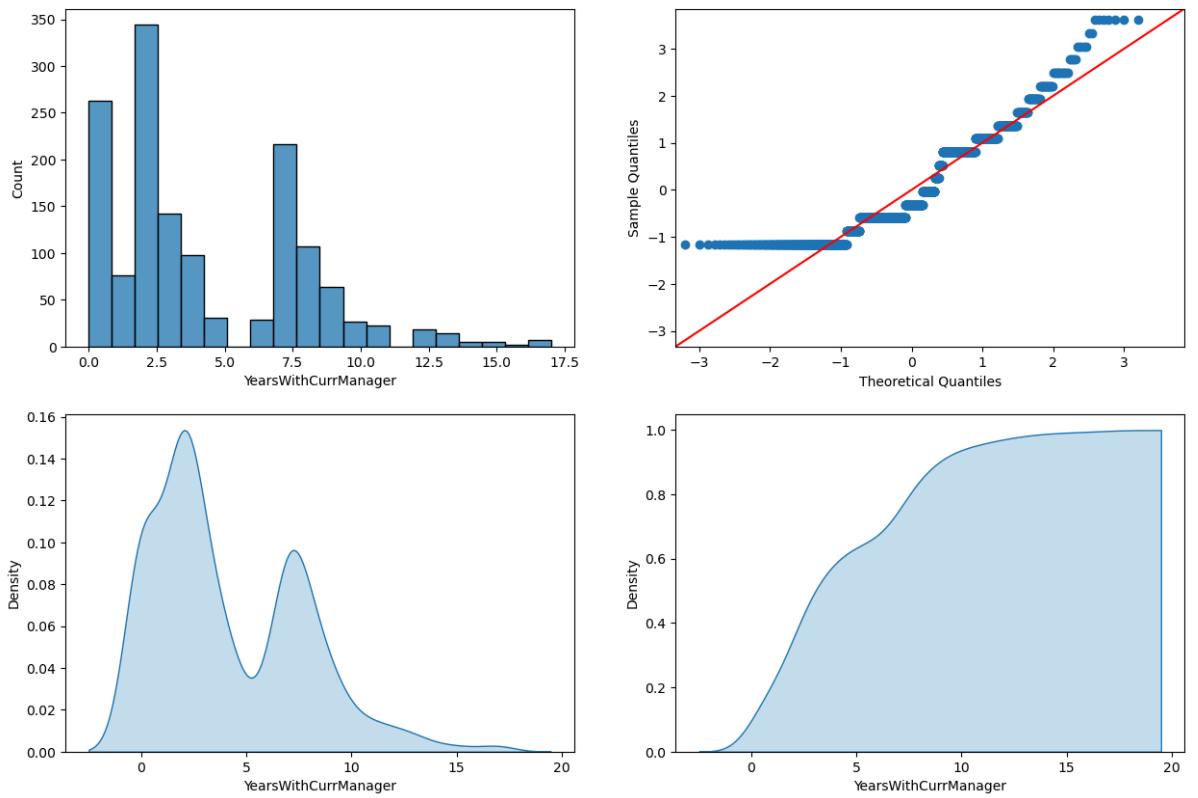
Distribution of YearsAtCompany



Distribution of YearsSinceLastPromotion

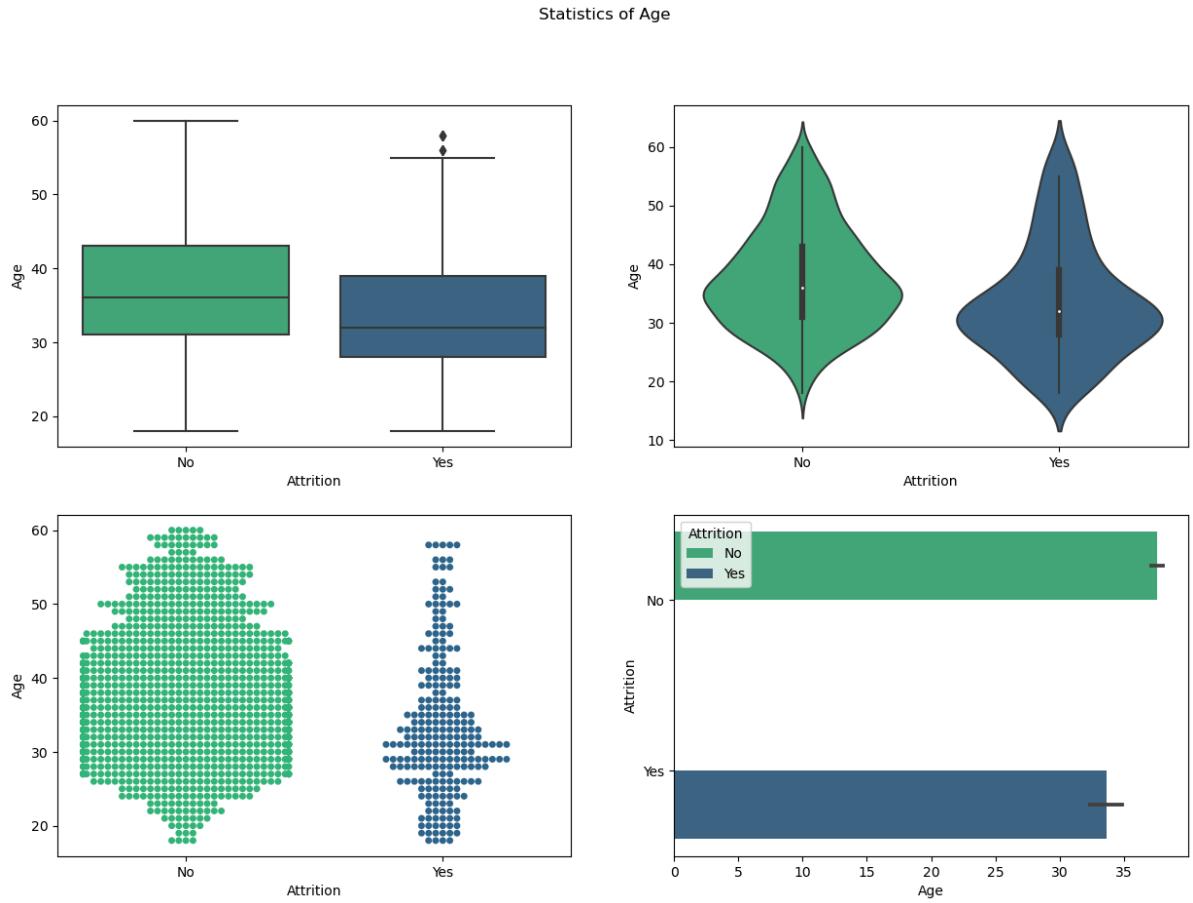


Distribution of YearsWithCurrManager

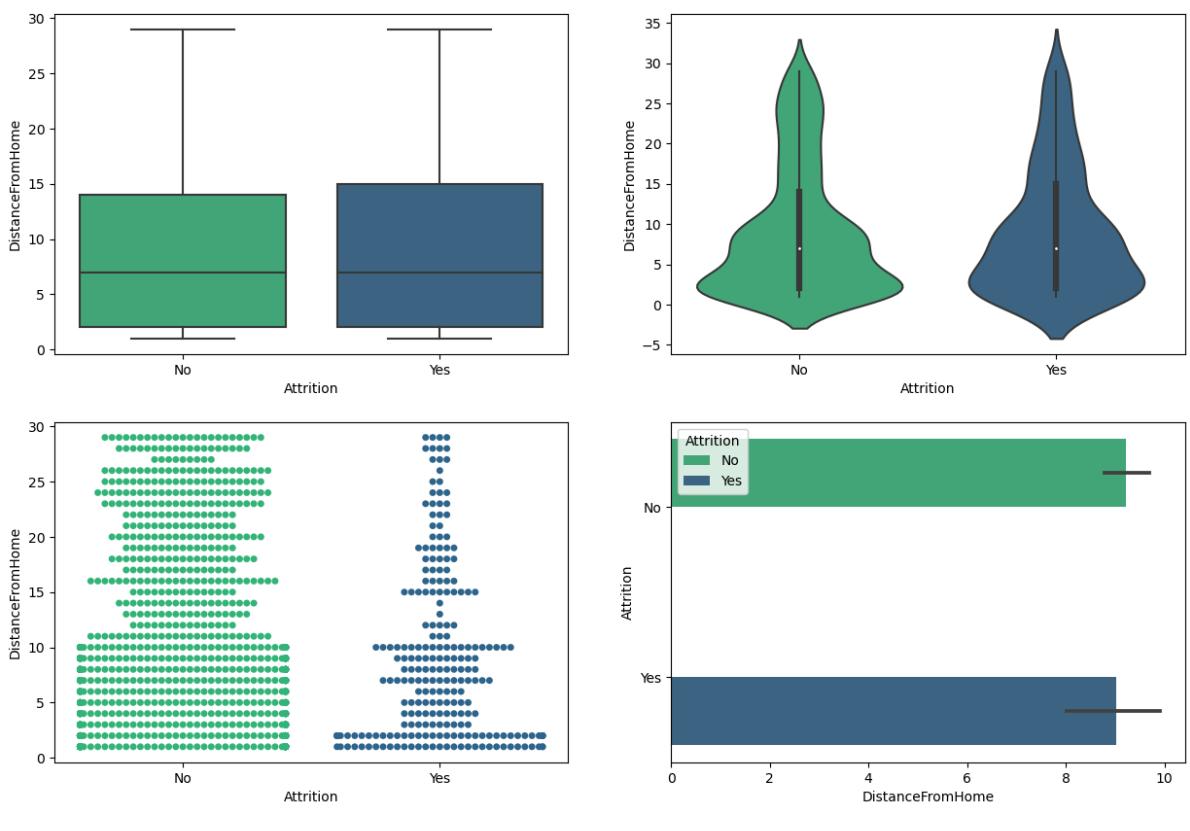


plots a 2 x 2 subplot contains (boxplot, violinplot, swarmplot and barplot) for each numerical features

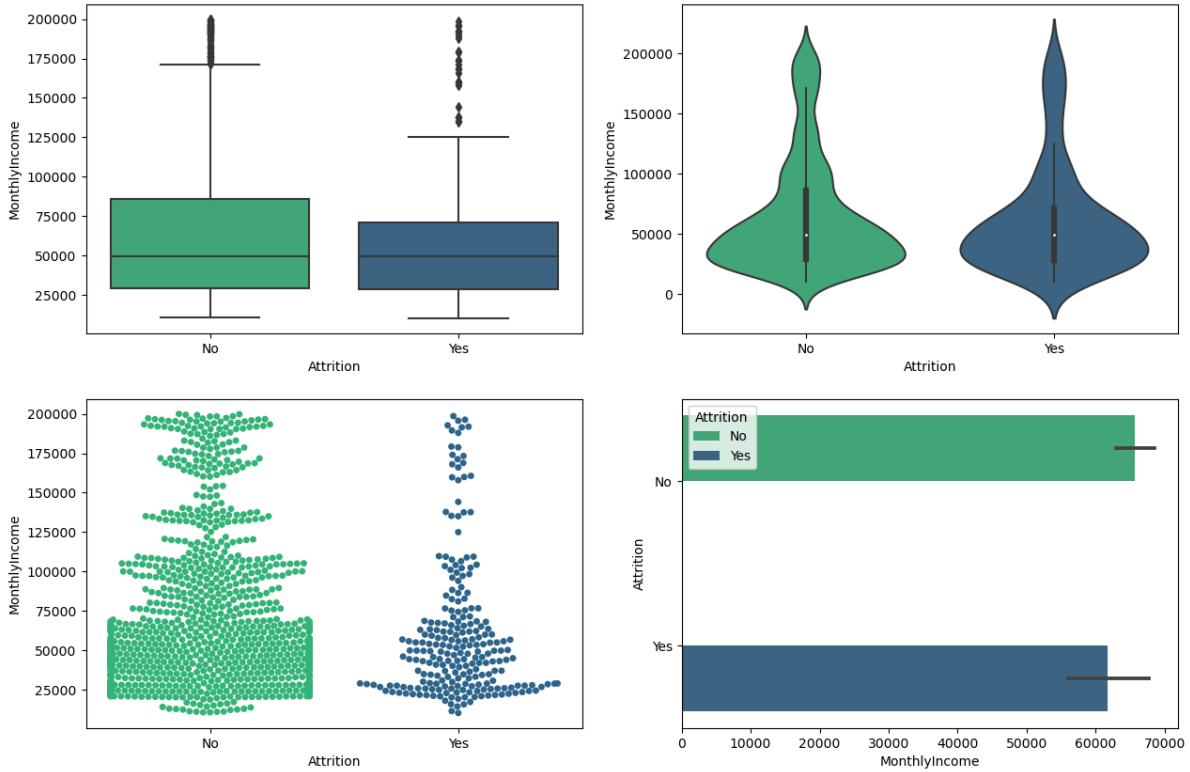
```
In [32]: for i in numerical_features:  
    plot_stats(i)
```



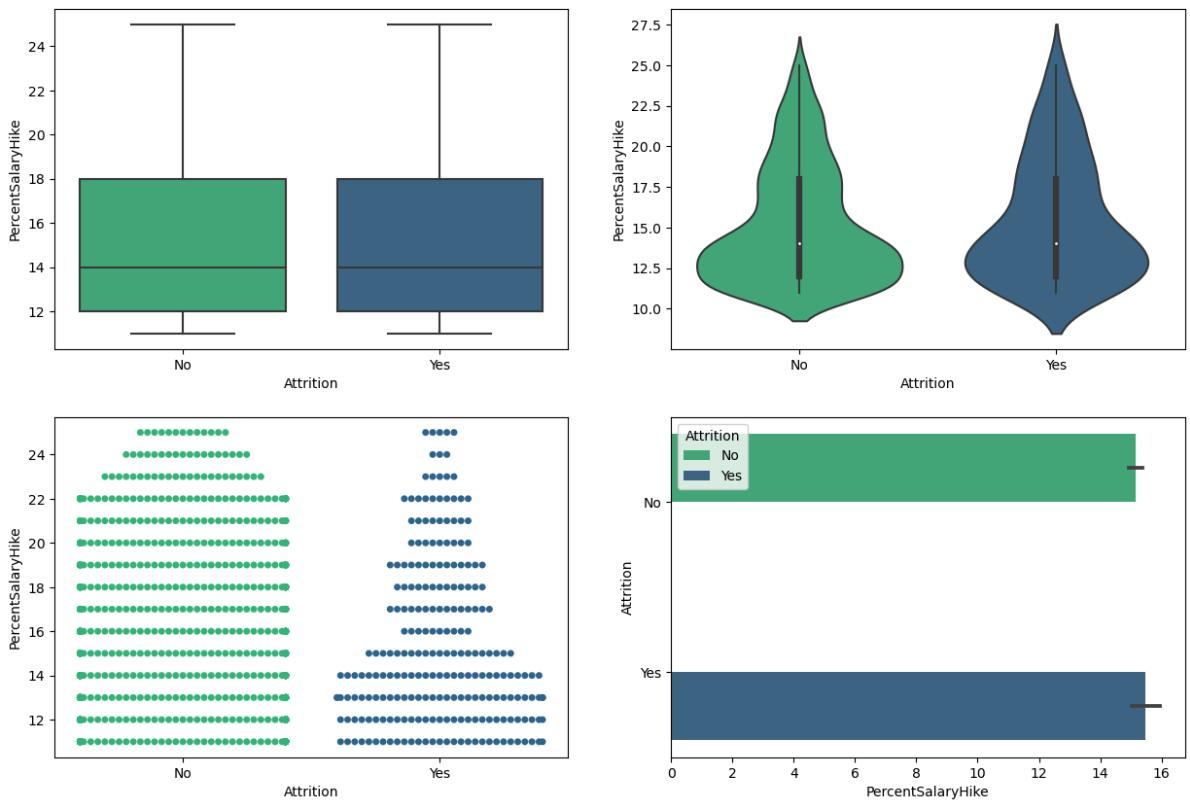
Statistics of DistanceFromHome



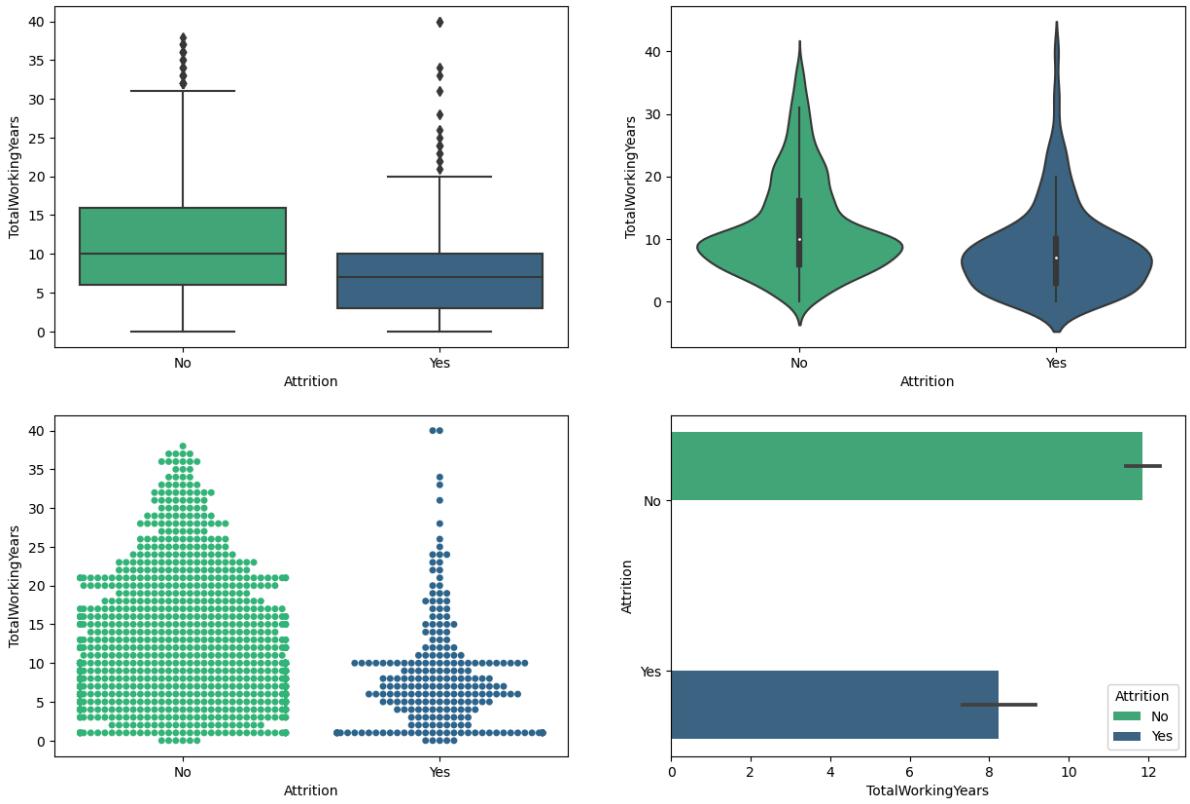
Statistics of MonthlyIncome



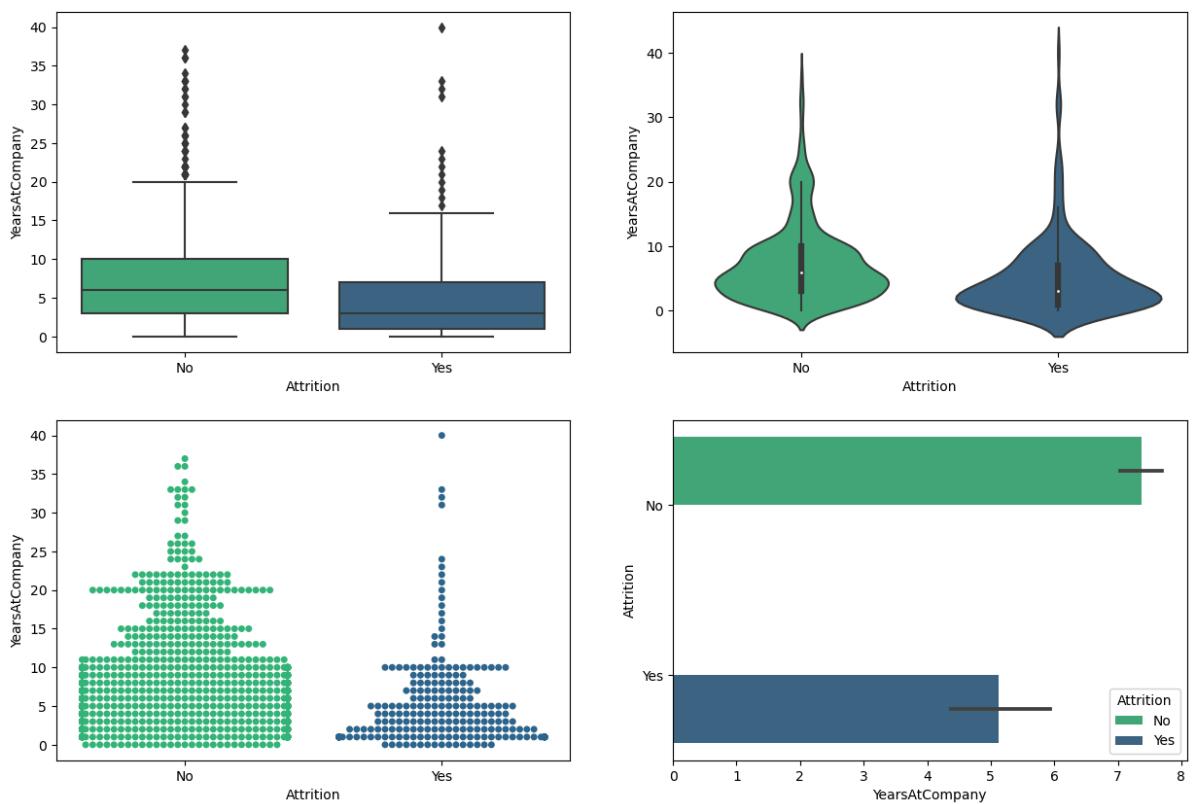
Statistics of PercentSalaryHike



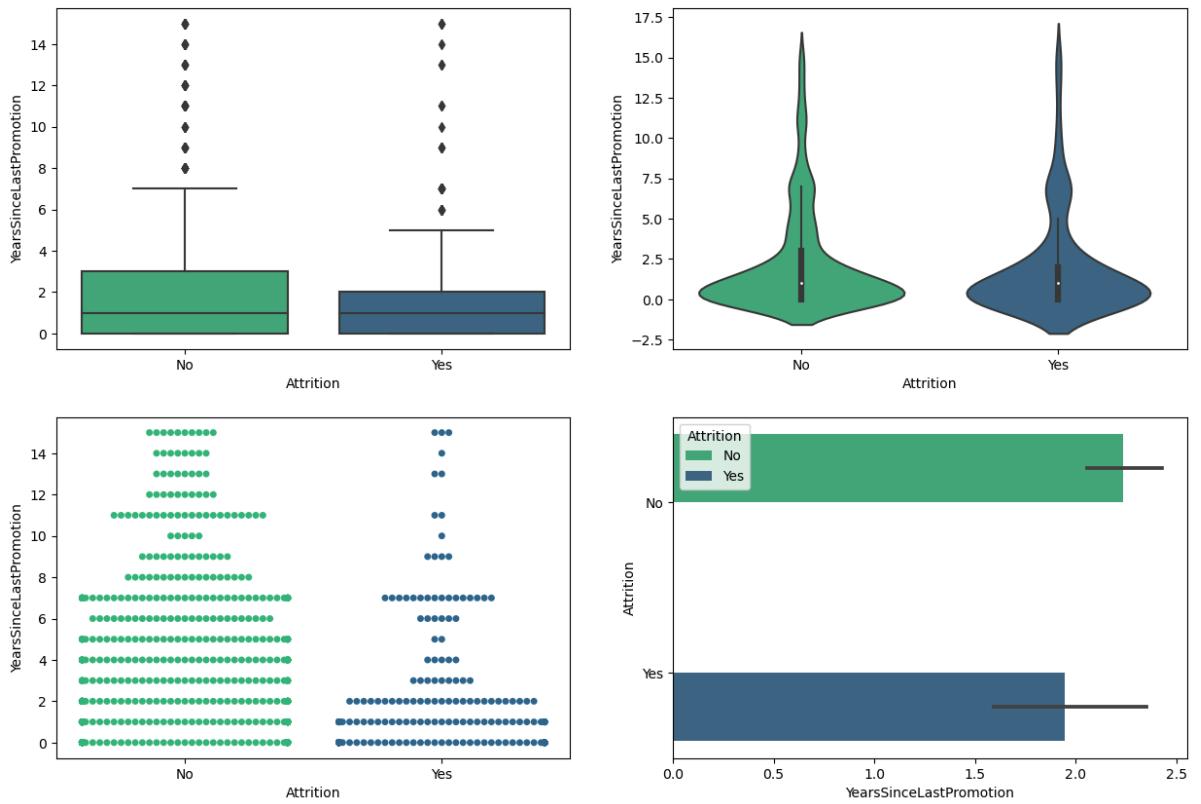
Statistics of TotalWorkingYears



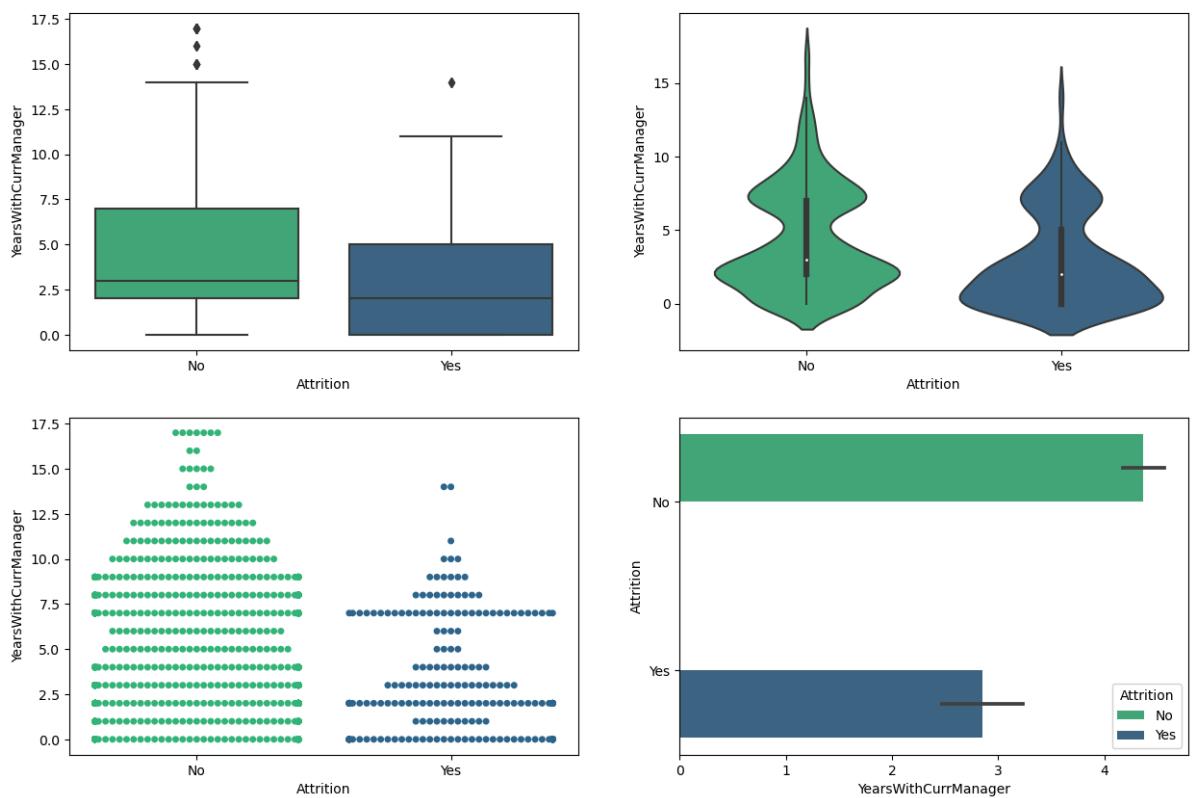
Statistics of YearsAtCompany



Statistics of YearsSinceLastPromotion



Statistics of YearsWithCurrManager

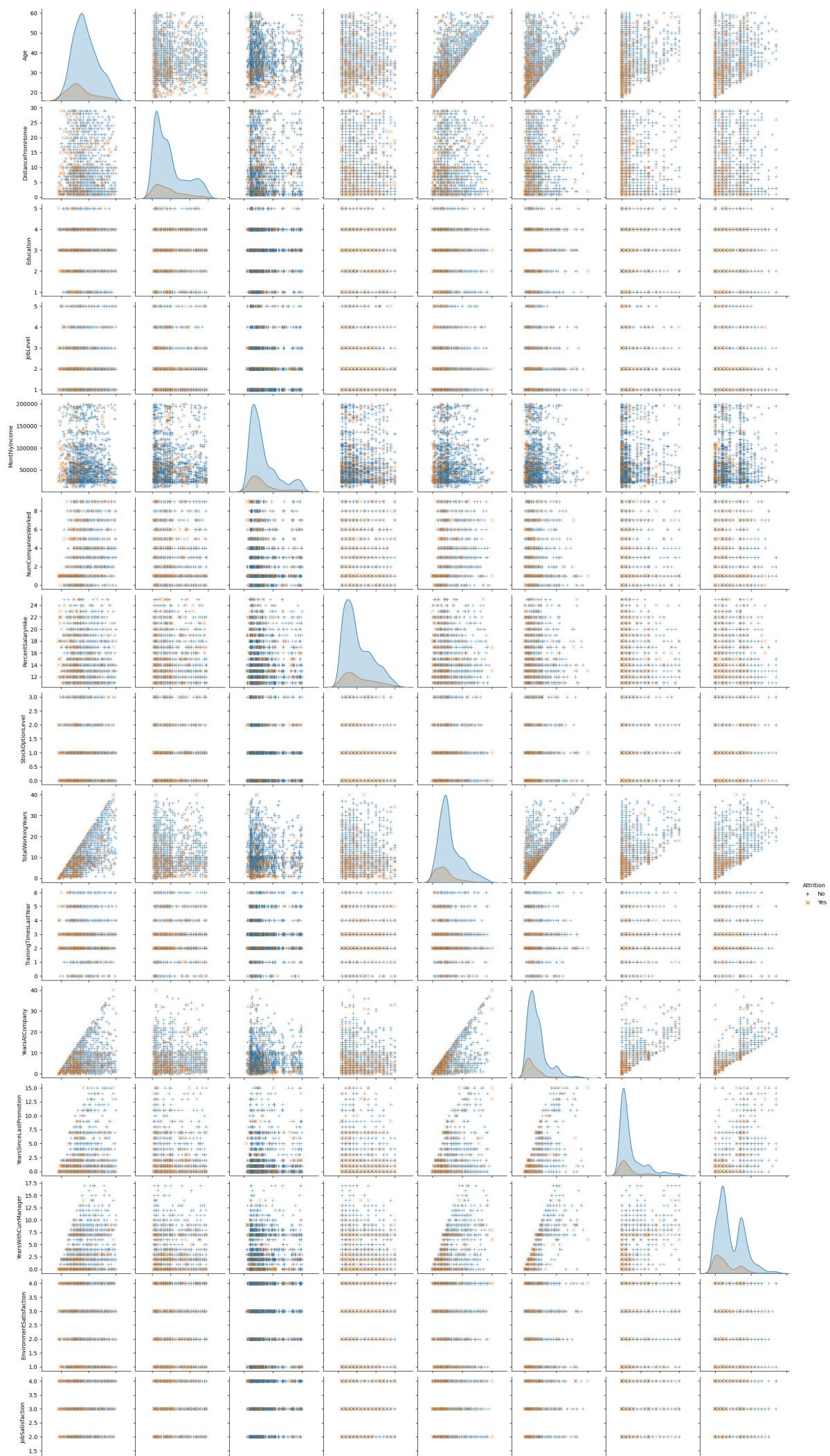


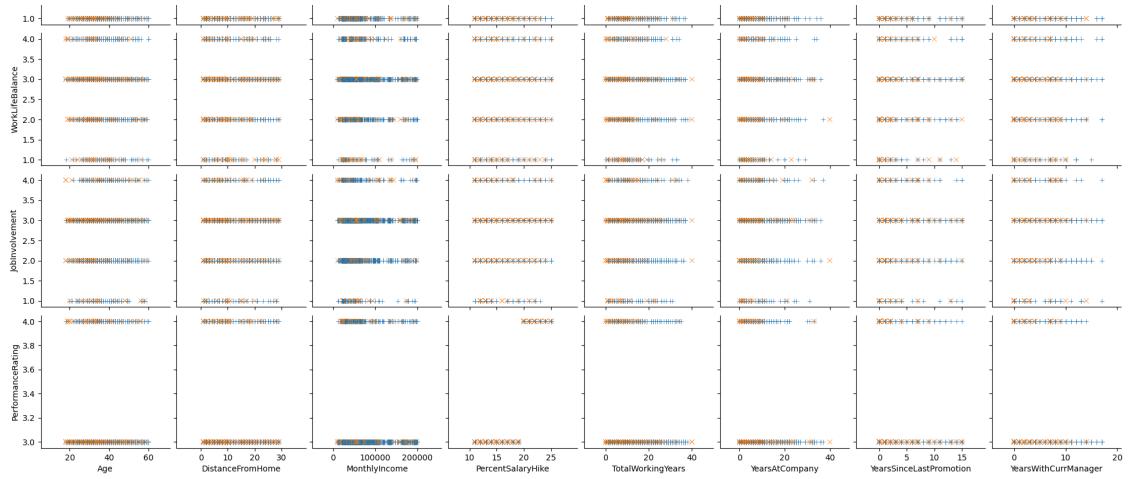
PairPlot

In [33]: numerical_features

Out[33]: ['Age',
 'DistanceFromHome',
 'MonthlyIncome',
 'PercentSalaryHike',
 'TotalWorkingYears',
 'YearsAtCompany',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']

```
In [34]: sns.pairplot(  
    df,  
    hue="Attrition",  
    x_vars=numerical_features,  
    markers=[ "+", "x" ],  
    ) # pairplot with given columns in x axis and all columns in y axis  
plt.show() # shows the plot
```





In []:

Observations after EDA:

Here are some key observations from the exploratory data analysis (EDA) related to employee attrition:

1. Gender Disparity: Female employees exhibit a higher attrition rate compared to their male counterparts.
2. Income Impact: Employees with low monthly income are more likely to leave the company.
3. Salary Percentage: A low percentage of salary relative to expectations can lead to employee resignation.
4. Manager Frequency: Frequent changes in managers correlate with higher attrition rates.
5. Performance Rating: Employee attrition is influenced by performance ratings.
6. Work Experience: The total number of working years at the company affects attrition, with longer tenure potentially reducing it.
7. Age Group: Most departing employees fall within the age range of 25 to 35.
8. Satisfaction Trends: Increasing employee satisfaction and improving the work environment can contribute to reducing attrition.

In []:

Changing the dtype

```
In [35]: ## Features to change
features_to_change=[]
for i in df.columns:
    if(df[i].dtype=='object'):
        features_to_change.append(i)
features_to_change
```

```
Out[35]: ['Attrition',
 'BusinessTravel',
 'Department',
 'EducationField',
 'Gender',
 'JobRole',
 'MaritalStatus']
```

```
In [36]: df['MaritalStatus'].value_counts()
```

```
Out[36]: Married      673  
Single       470  
Divorced     327  
Name: MaritalStatus, dtype: int64
```

```
In [37]: df['BusinessTravel'].value_counts()
```

```
Out[37]: Travel_Rarely      1043  
Travel_Frequently      277  
Non-Travel           150  
Name: BusinessTravel, dtype: int64
```

```
In [38]: ## transforming categorical features to numerical (using LabelEncoding technique)  
df1=df.copy()  
le=LabelEncoder()  
for i in features_to_change:  
    df1[i]=le.fit_transform(df[i])  
  
df1.head()
```

```
Out[38]:
```

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	Gender
0	51	0	2	2	6	2	1	0
1	31	1	1	1	10	1	1	0
2	32	0	1	1	17	4	4	1
3	38	0	0	1	2	5	1	1
4	32	0	2	1	10	1	3	1

In [39]: df1.corr()

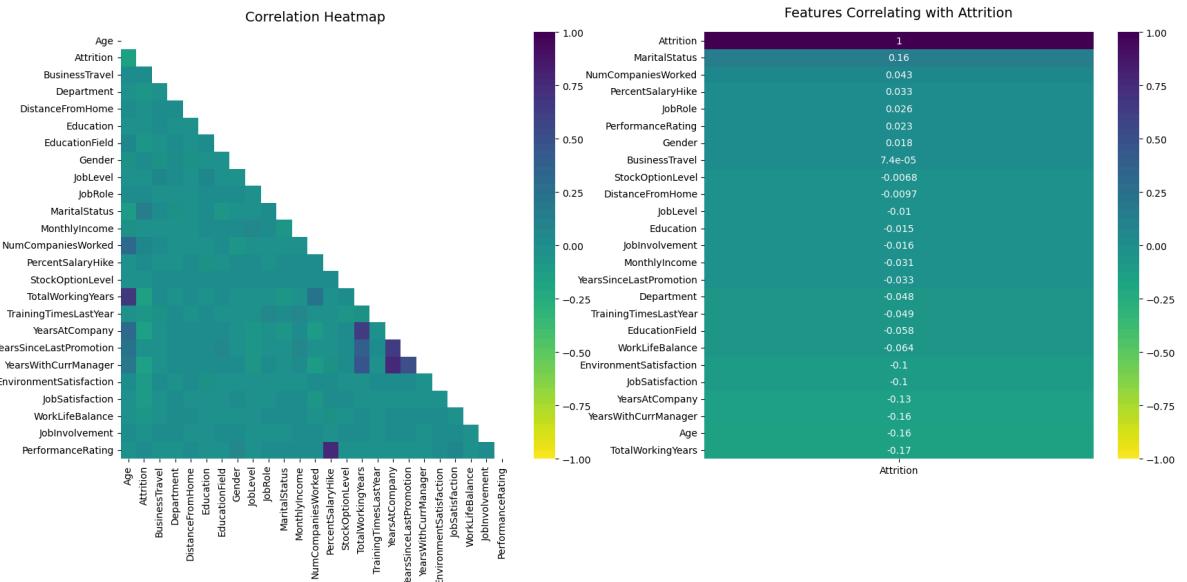
Out[39]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Educ
Age	1.000000	-0.159205	0.024751	-0.010846	0.006963	-0.03
Attrition	-0.159205	1.000000	0.000074	-0.048206	-0.009730	-0.01
BusinessTravel	0.024751	0.000074	1.000000	-0.010982	0.024616	0.00
Department	-0.010846	-0.048206	-0.010982	1.000000	0.012134	-0.00
DistanceFromHome	0.006963	-0.009730	0.024616	0.012134	1.000000	-0.00
Education	-0.035706	-0.015111	0.002755	-0.004597	-0.008638	1.00
EducationField	0.043194	-0.057940	-0.015462	0.013720	-0.024665	0.01
Gender	-0.039352	0.018125	-0.039243	0.003158	-0.044173	-0.01
JobLevel	-0.002884	-0.010290	0.046111	0.002923	-0.037329	0.04
JobRole	0.011422	0.025809	-0.027194	-0.025466	-0.010293	0.03
MaritalStatus	-0.095029	0.162070	0.024001	-0.044619	-0.027893	0.02
MonthlyIncome	-0.044314	-0.031176	-0.037597	-0.032237	-0.021607	0.00
NumCompaniesWorked	0.299635	0.043494	0.020875	-0.003870	-0.013822	-0.01
PercentSalaryHike	-0.033137	0.032533	-0.031054	-0.007840	0.038125	-0.04
StockOptionLevel	-0.031753	-0.006839	0.003683	0.010512	0.011169	0.00
TotalWorkingYears	0.680381	-0.171063	0.034226	-0.019408	0.010111	-0.01
TrainingTimesLastYear	-0.027308	-0.049431	-0.041888	0.012866	-0.009001	0.01
YearsAtCompany	0.311309	-0.134392	-0.014575	0.010078	0.031684	0.00
YearsSinceLastPromotion	0.216513	-0.033019	-0.032591	0.014845	0.002290	0.02
YearsWithCurrManager	0.202089	-0.156199	-0.022636	0.018016	0.021584	0.00
EnvironmentSatisfaction	0.010146	-0.103369	0.004174	-0.006416	0.020337	-0.04
JobSatisfaction	-0.004892	-0.103481	-0.033962	0.028018	-0.011663	-0.00
WorkLifeBalance	-0.021490	-0.063939	-0.011256	0.019080	0.010290	-0.00
JobInvolvement	0.018196	-0.015588	0.003121	0.006228	-0.001837	-0.01
PerformanceRating	-0.025563	0.023403	-0.017836	-0.024604	0.036418	-0.03

correlation

```
In [40]: fig, ax = plt.subplots(ncols=2, figsize=(20, 8)) # create a 1 x 2 subplot grid
resign_corr = df1.corr() # calculate correlation of the dataframe
mask = np.triu(np.ones_like(resign_corr, dtype=bool))
cat_heatmap = sns.heatmap(
    df1.corr(), mask=mask, vmin=-1, vmax=1, annot=False, ax=ax[0], cmap="viridis_r"
) # corner heatmap of the dataframe correlation
cat_heatmap.set_title(
    "Correlation Heatmap", fontdict={"fontsize": 14}, pad=12
) # sets the title for the heatmap with font size 14 and padding 12

heatmap = sns.heatmap(
    resign_corr[["Attrition"]].sort_values(by="Attrition", ascending=False),
    vmin=-1,
    vmax=1,
    annot=True,
    ax=ax[1],
    cmap="viridis_r",
) # create heatmap for features correlating with the dependant varibale 'Attrition'
heatmap.set_title(
    "Features Correlating with Attrition",
    fontdict={"fontsize": 14},
    pad=16, # sets the title for the heatmap with font size 14 and padding 16
)
plt.show() # shows the subplot
```



```
In [ ]:
```