About Human Resource Analytics

The Human Resources department is in charge of taking care of employee's well being and ensuring they are happy with their position. With machine learning you can actually predict employee attrition to see what causes a valuable employee to leave or stay with a company. This is perfect for HR managers planning their hiring and auditing employee experience.

About the Dataset

This is a fictional data set created by IBM data scientists to uncover the factors that lead to employee attrition.

Goal of the Project

Univariate Analysis of different variables in the dataset, to perform preliminary observation on the Data.

Skills covered

Pandas, Numpy, Matplotlib, Seaborn, Data Cleaning and Wrangling, Null Values and Outlier Detection,Data Visualization and Exploratory Data Analysis.

Let's Get Started

We use Pandas and Numpy Libraries for data manipulation & calculation and Matplotlib and Seaborn libraries for data visualization.

1. Importing Libraies

In [1]:
```
# Pandas and Numpy for Data Manipulation & Calculation
# Matplotlib and Seaborn for Visualization

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

2. Loading Data into Pandas Dataframe

In [2]:
```
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
df.head()
```

Out[2]:

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

5 rows × 35 columns

3. Exploring the Dataset

3.1 Columns

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

Out[3]:
```
(1470, 35)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [5]:
```
df.dtypes
```

Out[5]:
```
Age                         int64
Attrition                  object
BusinessTravel             object
DailyRate                   int64
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                     object
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
Over18                     object
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

In [6]:
```
df.dtypes.groupby(df.dtypes.values).count()
```

Out[6]:
```
int64     26
object     9
dtype: int64
```

3.2 Statistical Summary

In [7]:
```
df.describe()
```

Out[7]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000

8 rows × 26 columns

In [8]:
```
df.describe(include='all')
```

Out[8]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
count	1470.000000	1470	1470	1470.000000	1470	1470.000000	1470.00
unique	NaN	2	3	NaN	3	NaN	
top	NaN	No	Travel_Rarely	NaN	Research & Development	NaN	
freq	NaN	1233	1043	NaN	961	NaN	
mean	36.923810	NaN	NaN	802.485714	NaN	9.192517	2.91
std	9.135373	NaN	NaN	403.509100	NaN	8.106864	1.02
min	18.000000	NaN	NaN	102.000000	NaN	1.000000	1.00
25%	30.000000	NaN	NaN	465.000000	NaN	2.000000	2.00
50%	36.000000	NaN	NaN	802.000000	NaN	7.000000	3.00
75%	43.000000	NaN	NaN	1157.000000	NaN	14.000000	4.00
max	60.000000	NaN	NaN	1499.000000	NaN	29.000000	5.00

11 rows × 35 columns

In [9]:
```
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
```

4. Data Preprocessing

Data preprocessing is a crucial step in preparing raw data for analysis or modeling. It involves identifying and handling issues such as missing values, outliers, and type inconsistencies, and transforming the data into a suitable format for analysis. The main goal is to ensure accurate and reliable results, which can significantly impact subsequent data analysis or modeling.

4.1 Null values analysis

In [10]:
```
df.isnull().sum()
```

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
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
```

In [11]:

```
df.nunique()
```

Out[11]:
```
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
```

Given that the columns Over18, EmployeeCount, and StandardHours only contain one type of value, they are not expected to provide any useful information for our analysis. Therefore, we have decided to remove these columns from the dataset to reduce complexity and improve the efficiency of subsequent analyses.

In [12]:
```
df.drop(['Over18', 'StandardHours', 'EmployeeCount'], axis = 1, inplace = True)
```

In [13]:
```
df.head()
```

Out[13]:

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

5 rows × 32 columns

The dataset contains employee information on 1470 observations, with each observation having 32 associated columns of data. Therefore, there are a total of 32 variables or features that can be used to study the characteristics and relationships among the observations in the dataset. Upon preliminary examination, no null values or type inconsistencies were detected. To gain a more comprehensive understanding of the data, we will conduct additional analyses, including univariate, bivariate, and multivariate approaches.

4.2 Writing the clean data to WA*Fn-UseC*-HR-Employee-Attrition-clean.csv

In [14]:
```
df.to_csv("WA_Fn-UseC_-HR-Employee-Attrition-clean.csv", header=True,index=False)
```

5. Univariate Analysis

Univariate analysis is a statistical technique used to examine the characteristics of a single variable or attribute. In the context of employee attrition, univariate analysis can be used to explore the distribution and frequency of various factors that may be contributing to attrition. Some examples of factors that may be explored using univariate analysis in the context of employee attrition include: age, gender, education level, job level, department, salary, work-life balance, job satisfaction, performance rating, and years at the company. This technique can help identify which factors are contributing more strongly to attrition and guide further analysis and modeling. It can also be used to identify outliers or anomalies in the data, and provide a basic summary of the data that can inform the choice of appropriate statistical tests or models.

In [58]:
```
# Using the colorblind palette from the Seaborn library for visualization
colors = sns.color_palette('colorblind')
```

5.1 Age

In [15]:
```
# creating a new DataFrame with the age counts and percentages
age_data = pd.DataFrame({'Counts': df['Age'].value_counts(), 'Percentages': df["Age"].
print(age_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
    Counts Percentages
35      78       5.31%
34      77       5.24%
36      69       4.69%
31      69       4.69%
29      68       4.63%
32      61       4.15%
30      60       4.08%
33      58       3.95%
38      58       3.95%
40      57       3.88%
37      50       3.40%
27      48       3.27%
28      48       3.27%
42      46       3.13%
39      42       2.86%
45      41       2.79%
41      40       2.72%
26      39       2.65%
44      33       2.24%
46      33       2.24%
43      32       2.18%
50      30       2.04%
25      26       1.77%
24      26       1.77%
49      24       1.63%
47      24       1.63%
55      22       1.50%
51      19       1.29%
53      19       1.29%
48      19       1.29%
54      18       1.22%
52      18       1.22%
22      16       1.09%
56      14       0.95%
23      14       0.95%
58      14       0.95%
21      13       0.88%
20      11       0.75%
59      10       0.68%
19       9       0.61%
18       8       0.54%
60       5       0.34%
57       4       0.27%
```

Age statistical summary

In [16]:
```
df['Age'].describe()
```

Out[16]:
```
count    1470.000000
mean       36.923810
std         9.135373
min        18.000000
25%        30.000000
50%        36.000000
75%        43.000000
max        60.000000
Name: Age, dtype: float64
```

Age Outlier Analysis

In [17]:
```
from scipy import stats
# Calculate z-scores of column 'Age'
z_scores = stats.zscore(df['Age'])

# Identify outliers with a z-score of greater than 3 or less than -3
outliers = df[(z_scores > 3) | (z_scores < -3)]
```

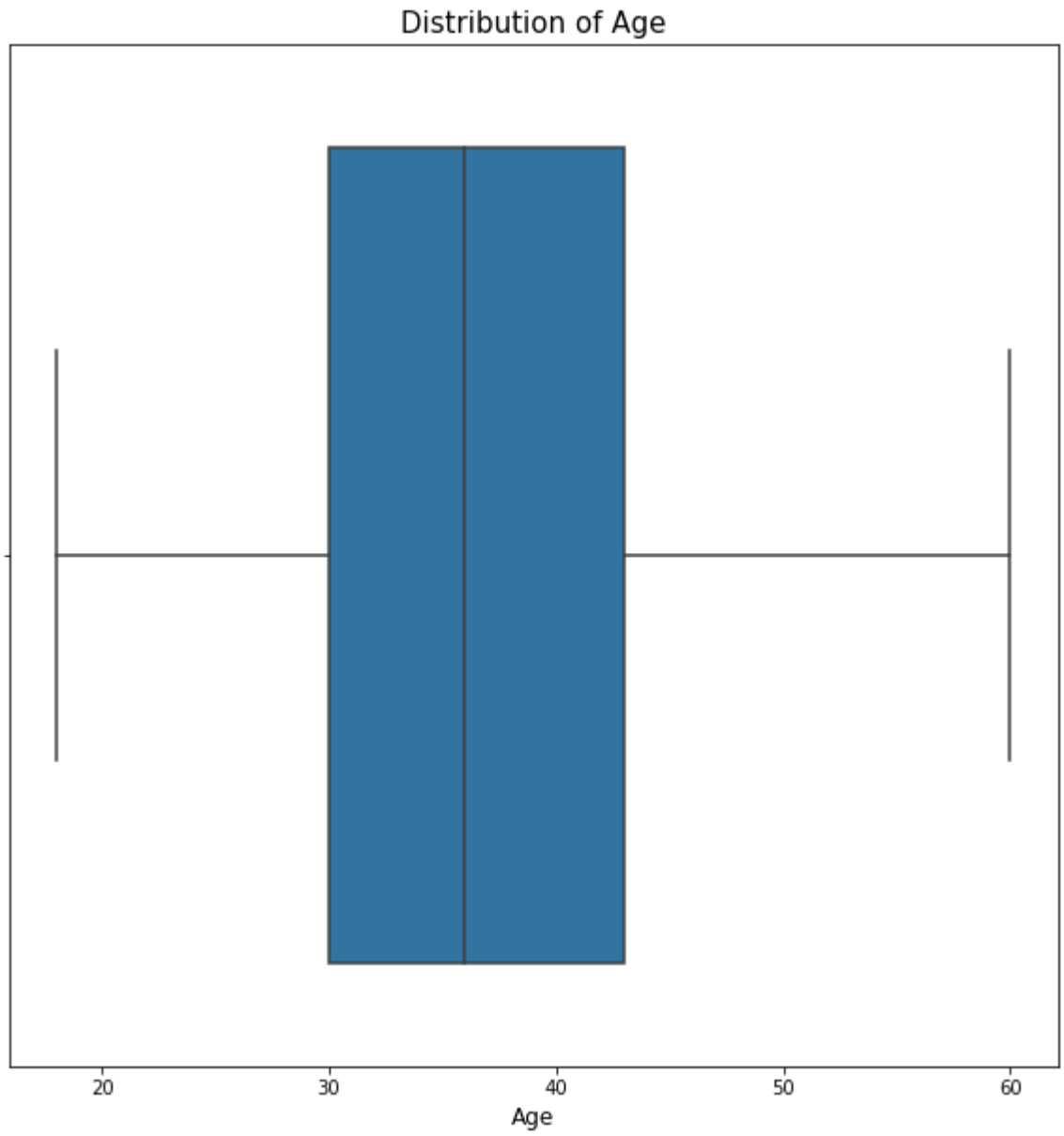
In [18]:
```
outliers.shape
```

Out[18]:
```
(0, 32)
```

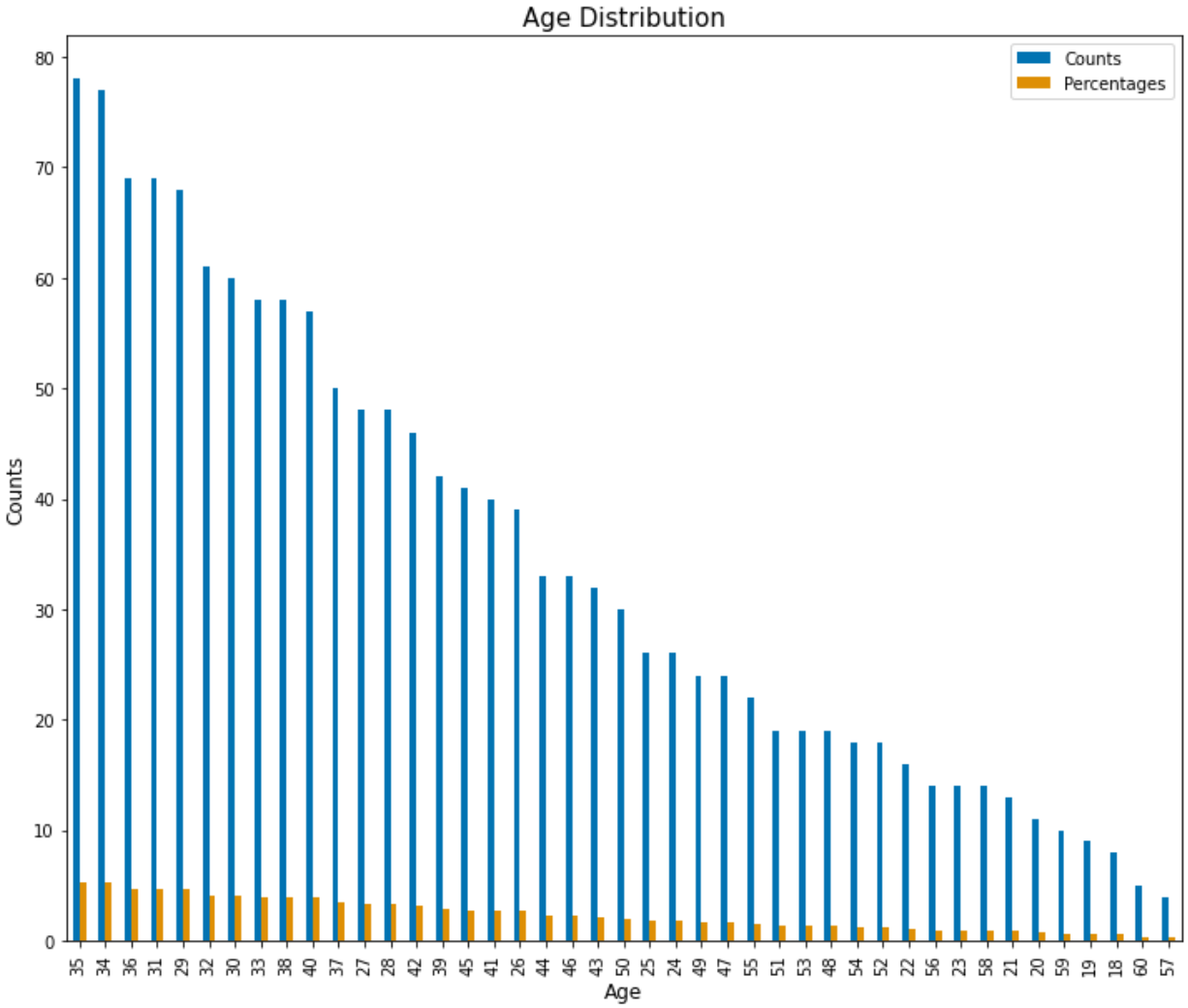
A z-score is a measure of how many standard deviations an observation is from the mean. In the context of outlier detection, an observation with a z-score greater than a certain threshold (3 is a common threshold). Based on the calculation of z-scores for the age column, we have found that all age values are within three standard deviations from the mean. This means that there are no age values that are too far from the average, and therefore, we can conclude that there are no outliers in the age column based on z-score.

Visualizing Age

In [60]:
```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(x=df['Age'], ax=ax)
ax.set_title('Distribution of Age', fontsize=15)
plt.xlabel('Age', fontsize=12)
# Show plot
plt.show()
```



In [61]:
```
age_data.plot(kind='bar', figsize=(12,10),color=colors)
# setting the chart title and axis labels
plt.title('Age Distribution', fontsize=15)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Counts',fontsize=12)

# show the chart
plt.show()
```



5.1 Observations of Age column

1. The range of ages from 18 to 60 is simply the difference between the smallest and largest age values in the dataset.
2. Maximum frequency of people are from age 35 and minimum frequency is of age 57
3. Mean age of the sample is 36.92 ~ 37 years.
4. The standard deviation of 9.14 measures how much the age values vary from the mean age.
5. The IQR is a measure of the range of ages that the middle 50% of the dataset falls within. It is calculated by taking the difference between the 75th and 25th percentiles of the age values. The IQR is a more robust measure of the spread of the data than the standard deviation, as it is less influenced by outliers in the dataset.

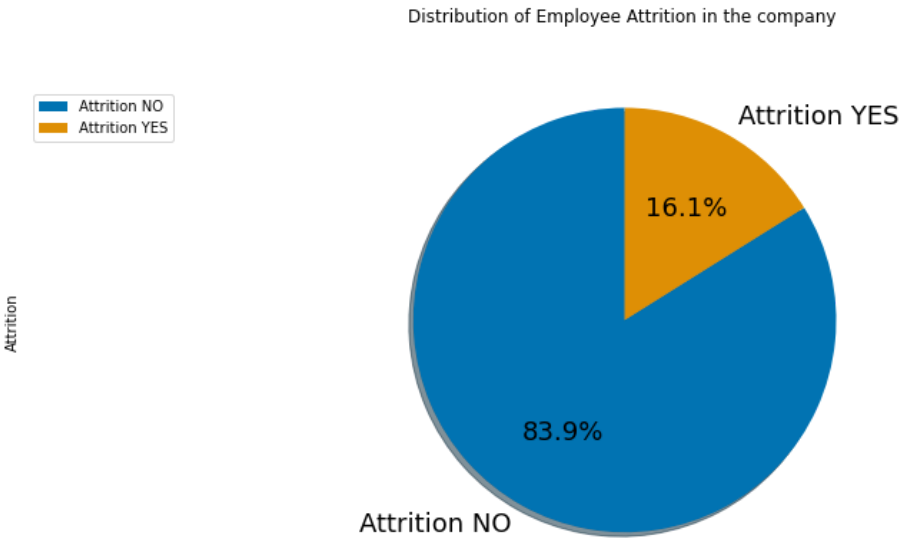
5.2 Attrition

In [21]:
```
# creating a new DataFrame with the attrition counts and percentages
attrition_data = pd.DataFrame({'Counts': df['Attrition'].value_counts(), 'Percentages'
print(attrition_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
     Counts Percentages
No     1233      83.88%
Yes     237      16.12%
```

Visualizing Attrition

In [64]:
```
labels = 'Attrition NO', 'Attrition YES'
df['Attrition'].astype(str).value_counts().plot(kind='pie',figsize = (15,6), fontsize=
plt.title('Distribution of Employee Attrition in the company ', y=1.12)
plt.axis('equal')
plt.legend(labels=labels, loc='upper left')
plt.show()
```



5.2 Observations of Attrition column

The attrition values represent the number and percentage of employees in the dataset who have either left the company (Yes) or stayed with the company (No). According to the data, there are 1233 employees who have not left the company, which represents 83.88% of the total number of employees in the dataset. This means that the majority of employees in the dataset have chosen to stay with the company. On the other hand, there are 237 employees who have left the company, representing 16.12% of the total number of employees in the dataset. This suggests that a smaller proportion of employees have decided to leave the company.These values are important for understanding the overall attrition rate and can provide insight into potential factors that may contribute to employee turnover in the company.

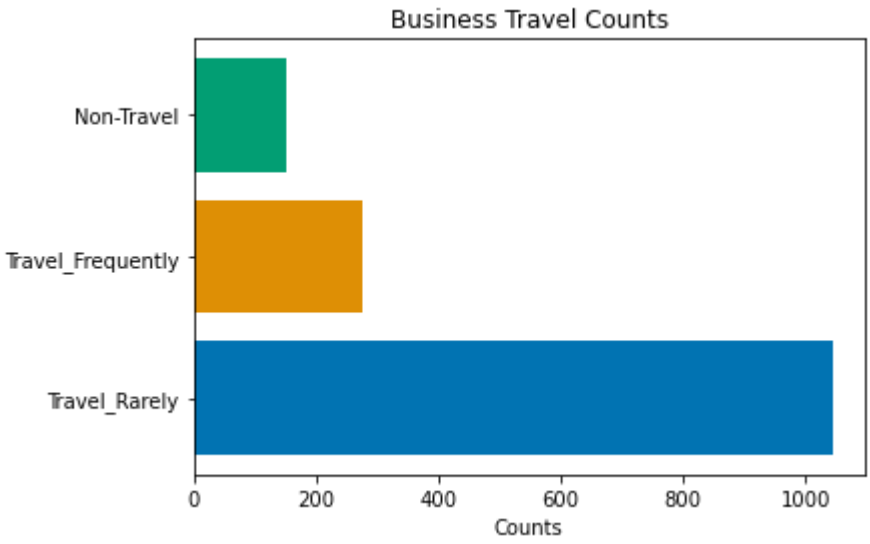
5.3 Business Travel

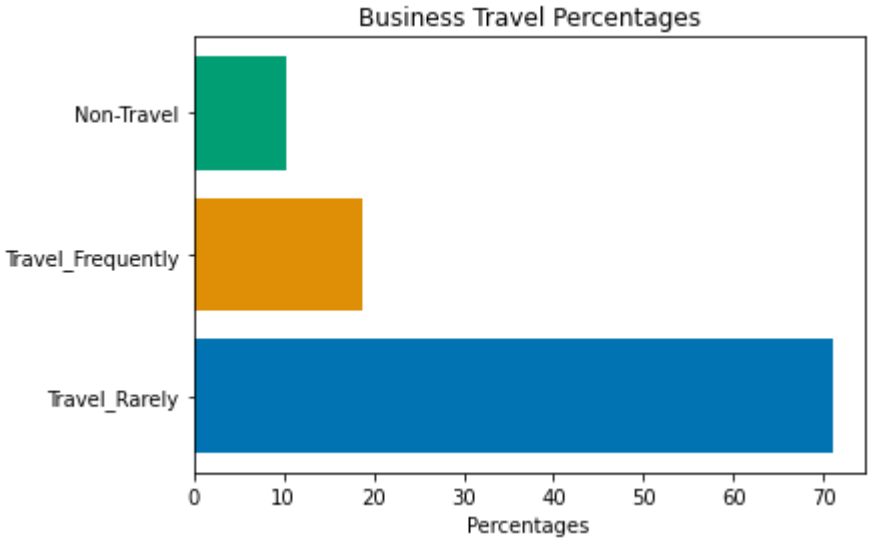
In [23]:
```
# creating a new DataFrame with the BusinessTravel counts and percentages
businessTravel_data = pd.DataFrame({'Counts': df['BusinessTravel'].value_counts(), 'Pe
print(businessTravel_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
                   Counts Percentages
Travel_Rarely        1043      70.95%
Travel_Frequently     277      18.84%
Non-Travel            150      10.20%
```

Visualizing BusinessTravel

In [65]:
```
# Creating a bar chart of counts
plt.barh(y=businessTravel_data.index, width=businessTravel_data['Counts'],color=colors
plt.xlabel('Counts')
plt.title('Business Travel Counts')
plt.show()

# Creating a bar chart of percentages
plt.barh(y=businessTravel_data.index, width=businessTravel_data['Percentages'],color=c
plt.xlabel('Percentages')
plt.title('Business Travel Percentages')
plt.show()
```





5.3 Observations of BusinessTravel column

The BussinessTravel data shows the proportion of employees in the dataset who travel for work, with 70.95% traveling rarely, 18.84% traveling frequently, and 10.20% not traveling at all. This information is useful for understanding the impact of travel on employee satisfaction and can inform company policies related to travel.

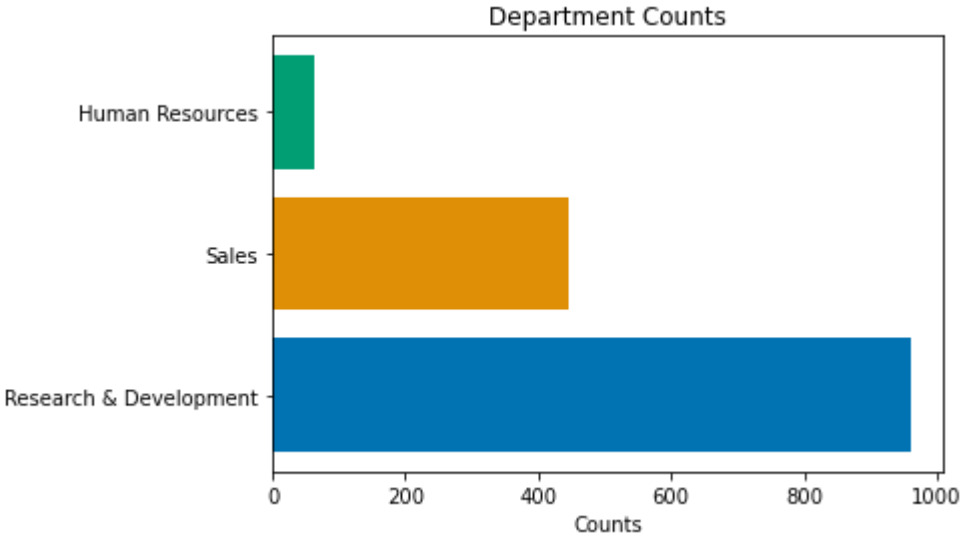
5.4 Department

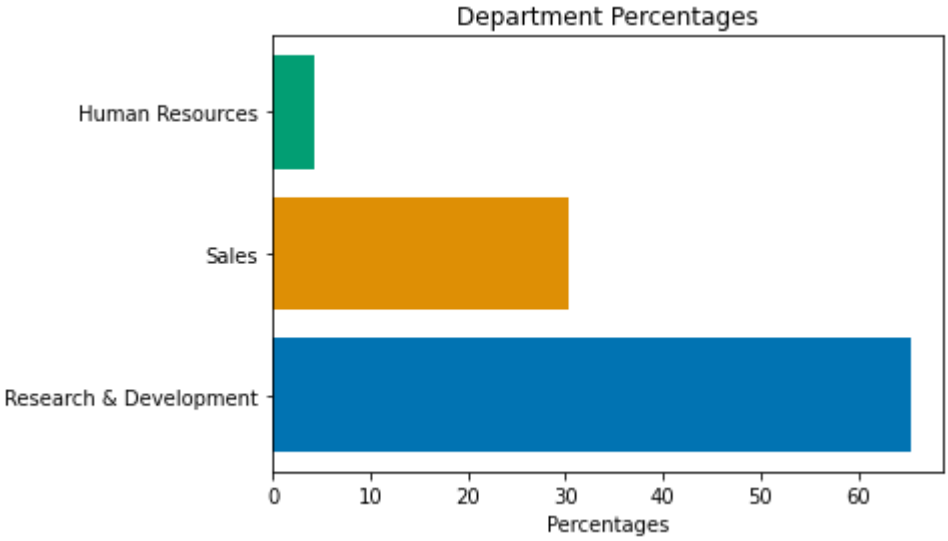
In [25]:
```
# creating a new DataFrame with the Department counts and percentages
dept_data = pd.DataFrame({'Counts': df['Department'].value_counts(), 'Percentages': df
print(dept_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
                        Counts Percentages
Research & Development     961      65.37%
Sales                      446      30.34%
Human Resources             63       4.29%
```

Visualizing the Department column

In [66]:
```
# Creating a bar chart of counts
plt.barh(y=dept_data.index, width=dept_data['Counts'],color=colors)
plt.xlabel('Counts')
plt.title('Department Counts')
plt.show()

# Creating a bar chart of percentages
plt.barh(y=dept_data.index, width=dept_data['Percentages'],color=colors)
plt.xlabel('Percentages')
plt.title('Department Percentages')
plt.show()
```





5.4 Observation of Department column

The R & D dept has more employees than any other departments which shows about the nature of the company is "Researh-Oriented".

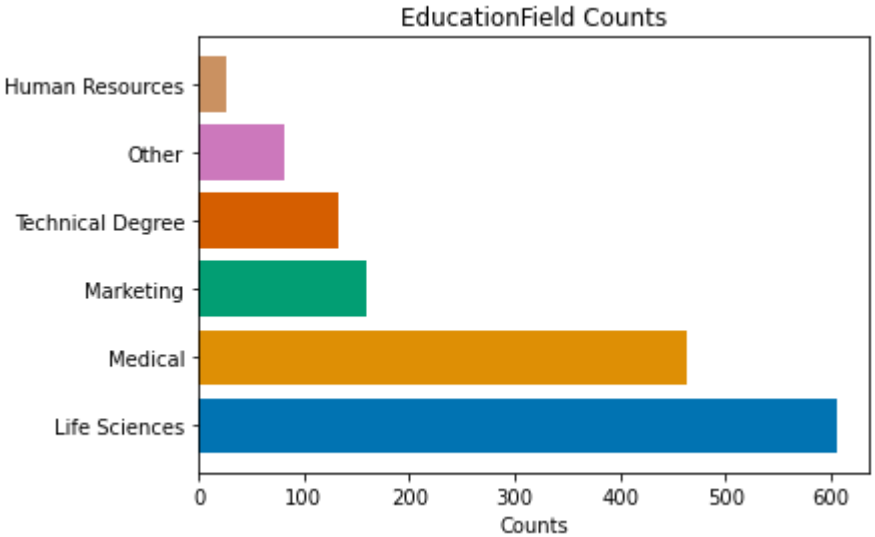
5.5 EducationField

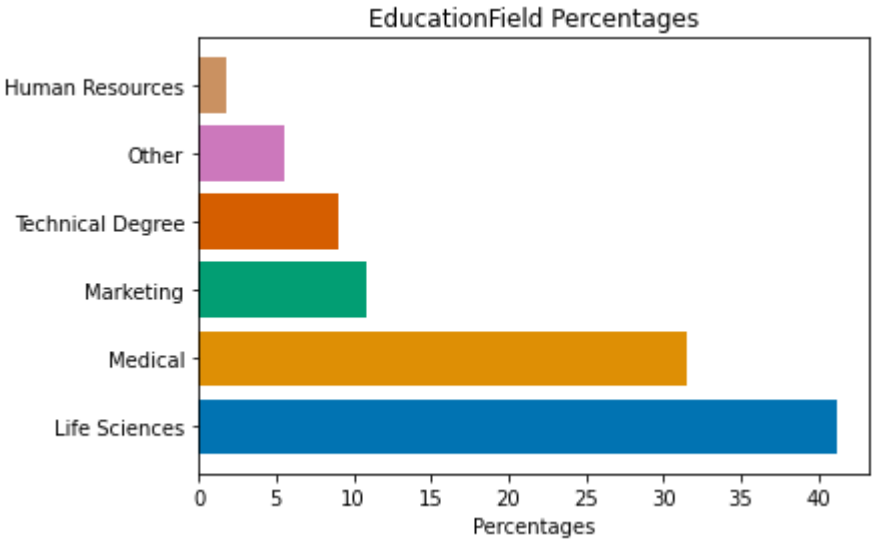
In [27]:
```
# creating a new DataFrame with the EducationField counts and percentages
EducationField_data = pd.DataFrame({'Counts': df['EducationField'].value_counts(), 'Pe
print(EducationField_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
                  Counts Percentages
Life Sciences        606      41.22%
Medical              464      31.56%
Marketing            159      10.82%
Technical Degree     132       8.98%
Other                 82       5.58%
Human Resources       27       1.84%
```

Visualizing the EducationField

In [67]:
```
# Creating a bar chart of counts
plt.barh(y=EducationField_data.index, width=EducationField_data['Counts'],color=colors
plt.xlabel('Counts')
plt.title('EducationField Counts')
plt.show()

# Creating a bar chart of percentages
plt.barh(y=EducationField_data.index, width=EducationField_data['Percentages'],color=c
plt.xlabel('Percentages')
plt.title('EducationField Percentages')
plt.show()
```





5.5 Observation of EducationField column

Based on the information provided, it appears that the company has a larger number of employees from a Life Sciences background and a smaller number of employees from an HR background. This suggests that the company places a greater emphasis on research in Life Sciences than in HR.

5.6 DistanceFromHome

In [69]:
```
# creating a new DataFrame with the DistanceFromHome counts and percentages
DistanceFromHome_data = pd.DataFrame({'Counts': df['DistanceFromHome'].value_counts(),
print(DistanceFromHome_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

	Counts	Percentages
2	211	14.35%
1	208	14.15%
10	86	5.85%
9	85	5.78%
3	84	5.71%
7	84	5.71%
8	80	5.44%
5	65	4.42%
4	64	4.35%
6	59	4.01%
16	32	2.18%
11	29	1.97%
24	28	1.90%
23	27	1.84%
29	27	1.84%
15	26	1.77%
18	26	1.77%
26	25	1.70%
25	25	1.70%
20	25	1.70%
28	23	1.56%
19	22	1.50%
14	21	1.43%
12	20	1.36%
17	20	1.36%
22	19	1.29%
13	19	1.29%
21	18	1.22%
27	12	0.82%

Statistical Analysis of DistanceFromHome

In [70]:
```
df["DistanceFromHome"].describe()
```

Out[70]:
```
count    1470.000000
mean        9.192517
std         8.106864
min         1.000000
25%         2.000000
50%         7.000000
75%        14.000000
max        29.000000
Name: DistanceFromHome, dtype: float64
```

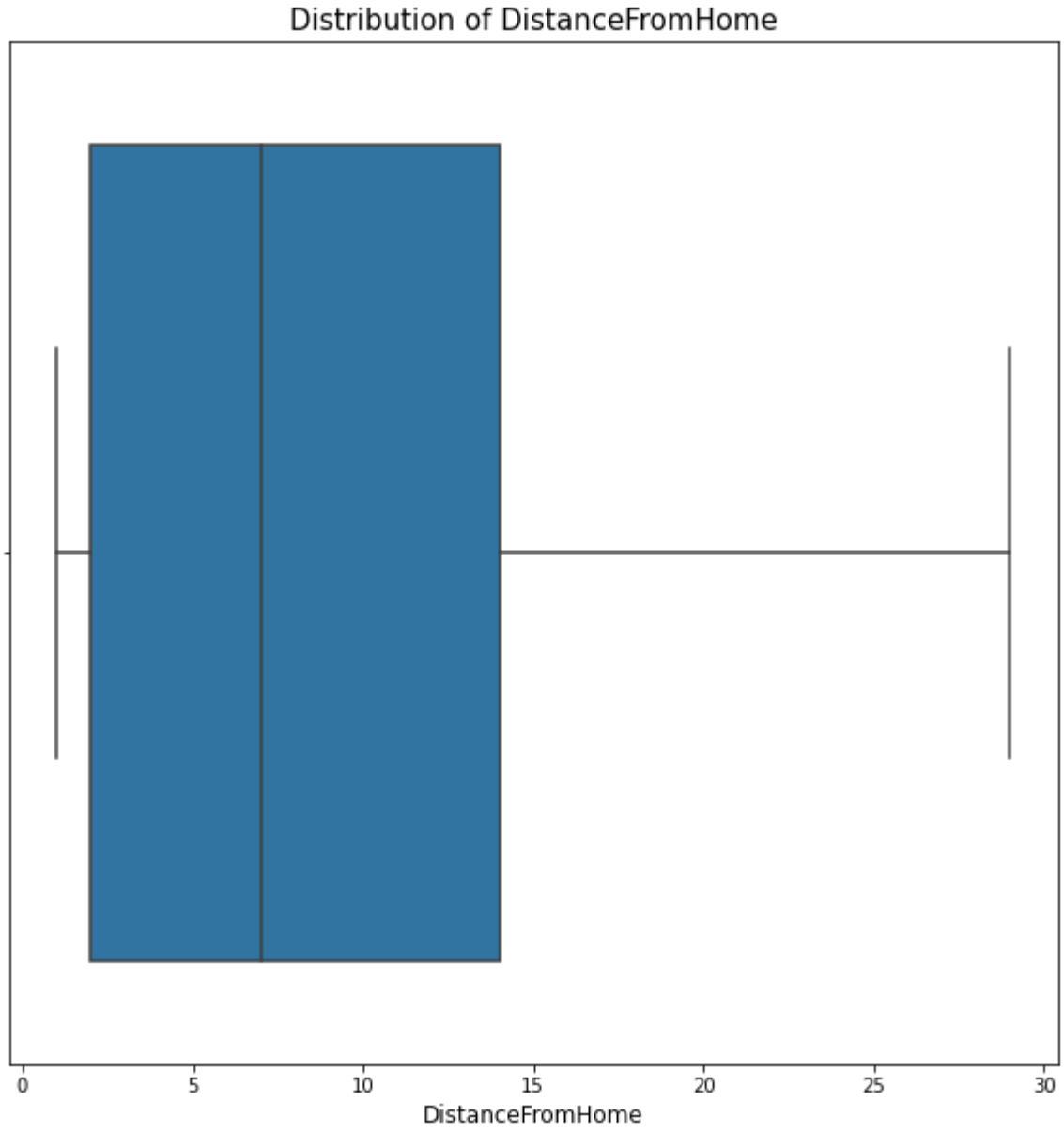
Outlier check for DistanceFromHome

In [71]:
```
from scipy import stats
# Calculate z-scores of column 'DistanceFromHome'
z_scores = stats.zscore(df['DistanceFromHome'])

# Identify outliers with a z-score of greater than 3 or less than -3
outliers = df[(z_scores > 3) | (z_scores < -3)]
outliers.shape
```

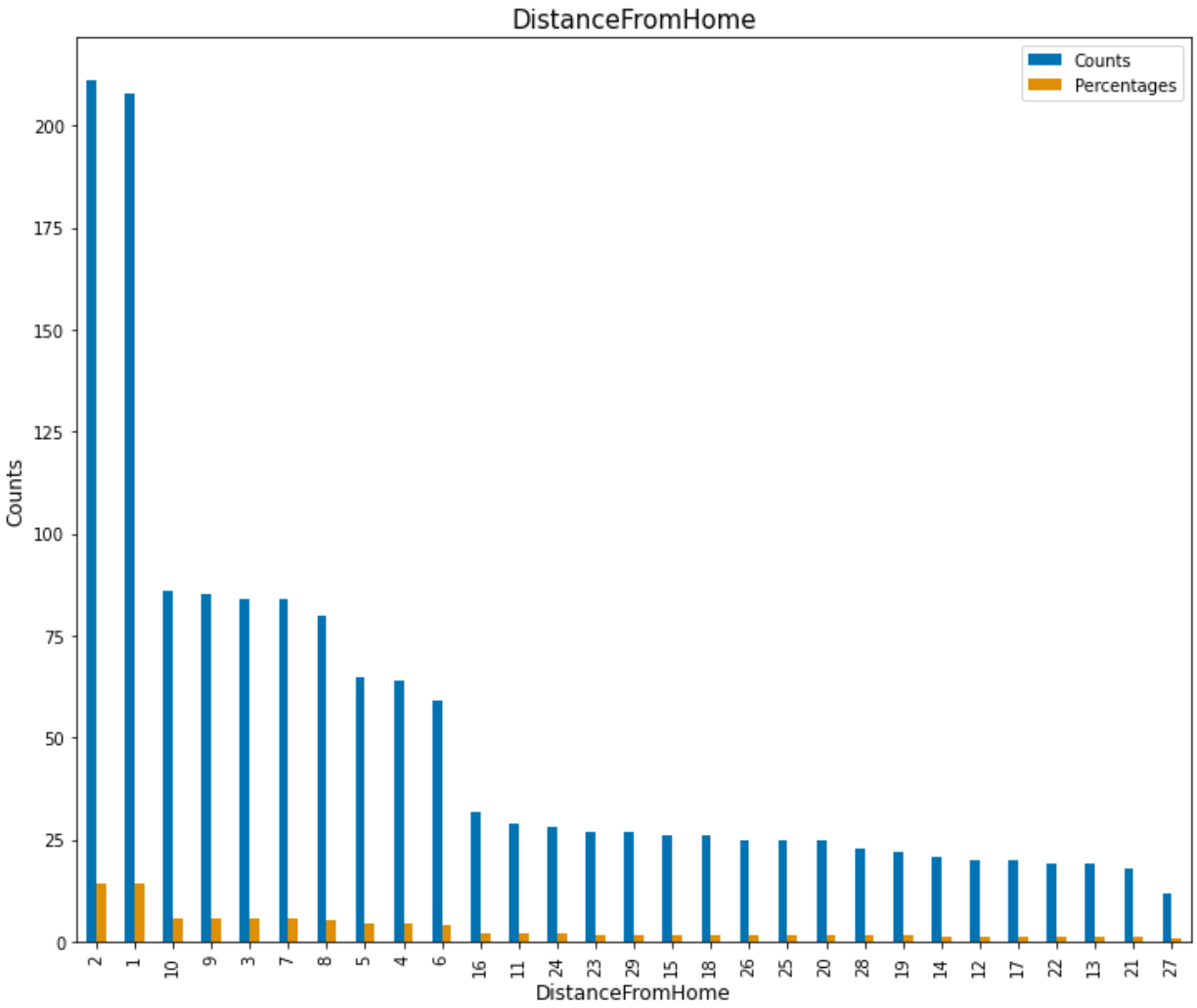
Out[71]:
```
(0, 32)
```

Visualizing DistanceFromHome

In [72]:
```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(x=df['DistanceFromHome'], ax=ax)
ax.set_title('Distribution of DistanceFromHome', fontsize=15)
plt.xlabel('DistanceFromHome', fontsize=12)
# Show plot
plt.show()
```



In [73]:
```
DistanceFromHome_data.plot(kind='bar', figsize=(12,10),color=colors)
# setting the chart title and axis labels
plt.title('DistanceFromHome', fontsize=15)
plt.xlabel('DistanceFromHome', fontsize=12)
plt.ylabel('Counts',fontsize=12)

# show the chart
plt.show()
```



5.6 Observation of DistanceFromHome column

The range of distances varies from a minimum of 1 km to a maximum of 29 km. The median distance is 7 km, indicating that half of the employees live within 7 km of the office. The mean distance is 9.19 km, and the standard deviation of 8.10 km indicates that the distance from home varies considerably among employees. 25% of employees live within 2 km and 75% of employees live within 14 km of the office of the office. The distance is an important factor for understanding the commuting patterns and needs of the employees, which may be helpful in addressing work-life balance issues.

5.7 EnvironmentSatisfaction

In [104…

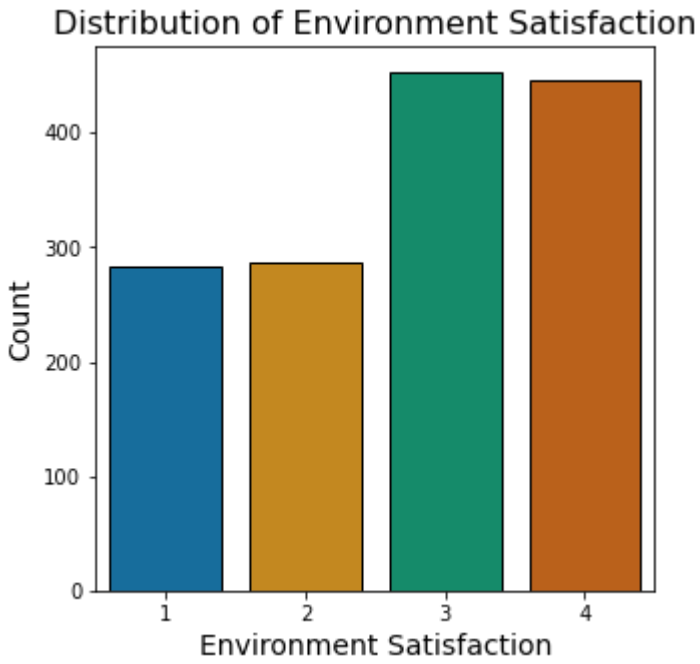
```
# creating a new DataFrame with the EnvironmentSatisfaction counts and percentages
EnvironmentSatisfaction_data = pd.DataFrame({'Counts': df['EnvironmentSatisfaction'].v
print(EnvironmentSatisfaction_data.to_string(formatters={'Percentages': '{:.2f}%'.form
```

```
   Counts Percentages
3     453      30.82%
4     446      30.34%
2     287      19.52%
1     284      19.32%
```

Visualizing the EnvironmentSatisfaction

In [126…

```
# Set colorblind-friendly palette
sns.set_palette("colorblind")

# Create count plot with edgecolor and figsize
plt.figure(figsize=(5,5))
ax = sns.countplot(x='EnvironmentSatisfaction', data=df, edgecolor="black")

# Add labels and title
ax.set_xlabel('Environment Satisfaction', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Distribution of Environment Satisfaction', fontsize=16)

# Show the plot
plt.show()
```



5.7 Observation of EnvironmentSatisfaction column

A significant number of respondents (over 800 employees) expressed satisfaction with the working environment by giving it a rating of 3 or 4 out of 4, which could indicate a positive workplace culture and conducive work environment.

5.8 Gender

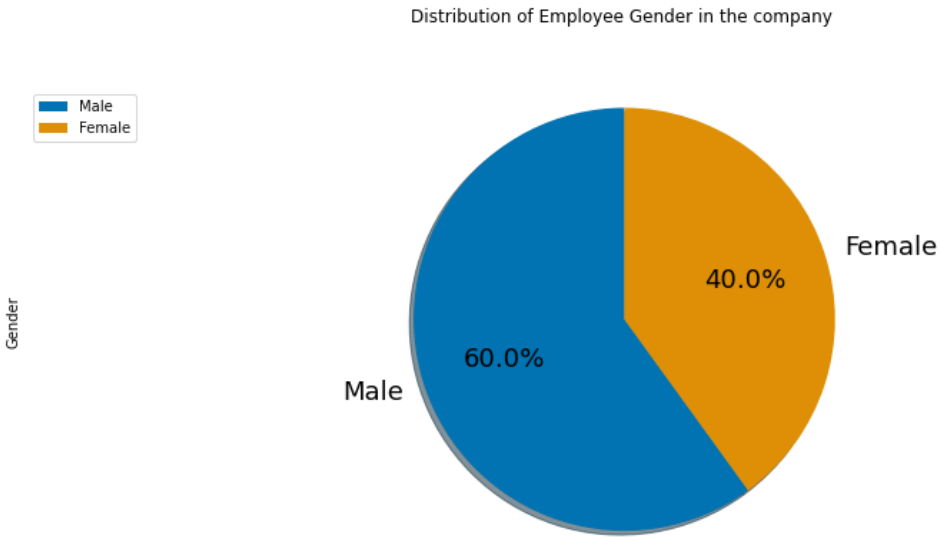
In [106…

```
# creating a new DataFrame with the Gender counts and percentages
Gender_data = pd.DataFrame({'Counts': df['Gender'].value_counts(), 'Percentages': df['
print(Gender_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
        Counts Percentages
Male       882      60.00%
Female     588      40.00%
```

Visualizing the Gender

In [120…
```
labels = 'Male', 'Female'
df['Gender'].astype(str).value_counts().plot(kind='pie',figsize = (15,6), fontsize=18,
plt.title('Distribution of Employee Gender in the company ', y=1.12)
plt.axis('equal')
plt.legend(labels=labels, loc='upper left')
plt.show()
```



5.8 Observation of Gender column

The dataset has a larger proportion of males (60%) than females (40%), indicating a gender imbalance in the sample. However, it's important to note that this doesn't necessarily mean that the dataset is biased towards males or that males are overrepresented. Instead, it could simply reflect the underlying gender distribution of the population that the dataset was sampled from.

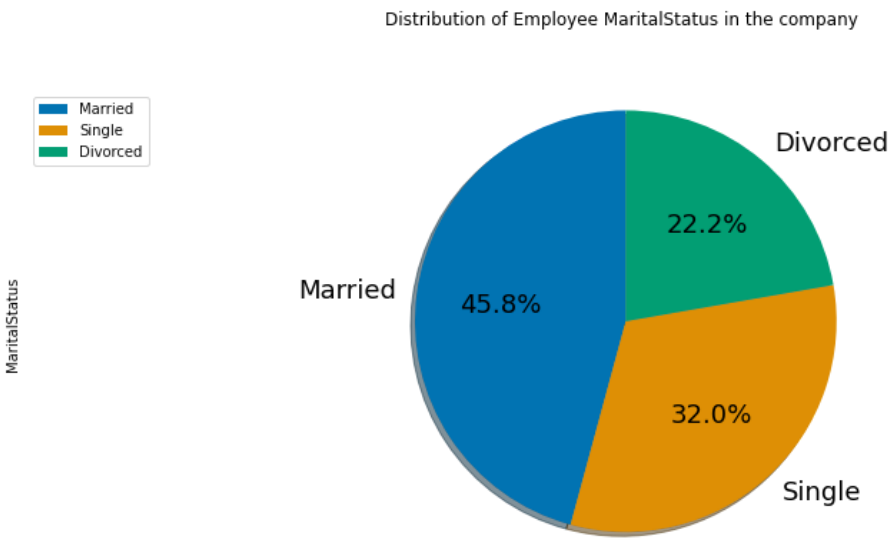
5.9 MaritalStatus

In [110…
```
# creating a new DataFrame with the MaritalStatus counts and percentages
MaritalStatus_data = pd.DataFrame({'Counts': df['MaritalStatus'].value_counts(), 'Perc
print(MaritalStatus_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
          Counts Percentages
Married      673      45.78%
Single       470      31.97%
Divorced     327      22.24%
```

Visualizing the MaritalStatus

In [121…
```
labels = 'Married', 'Single', 'Divorced'
df['MaritalStatus'].astype(str).value_counts().plot(kind='pie',figsize = (15,6), fonts
plt.title('Distribution of Employee MaritalStatus in the company ', y=1.12)
plt.axis('equal')
plt.legend(labels=labels, loc='upper left')
plt.show()
```



5.9 Observation of MaritalStatus column

The majority of individuals in the dataset are married (45.78%), followed by singles (31.97%), and divorced individuals (22.24%).

5.10 JobInvolvement

In [111…

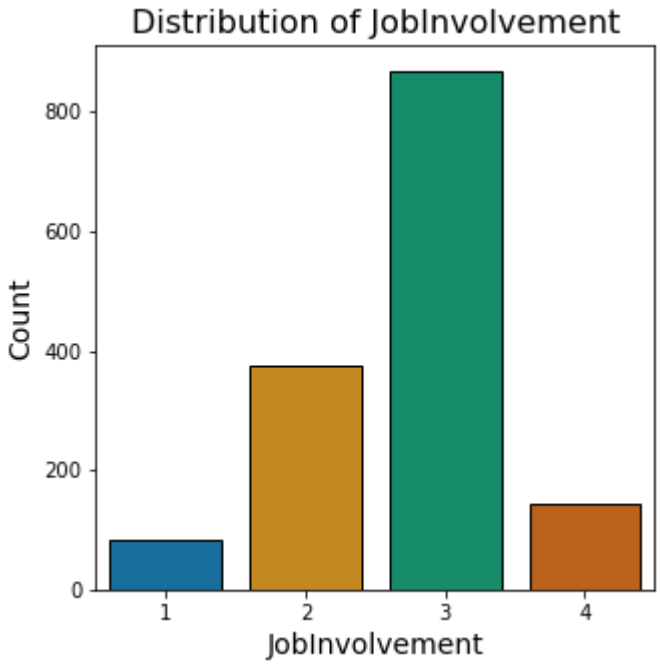
```
# creating a new DataFrame with the JobInvolvement counts and percentages
JobInvolvement_data = pd.DataFrame({'Counts': df['JobInvolvement'].value_counts(), 'Pe
print(JobInvolvement_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
   Counts Percentages
3     868      59.05%
2     375      25.51%
4     144       9.80%
1      83       5.65%
```

Visualizing the JobInvolvement

In [131…

```
# Set colorblind-friendly palette
sns.set_palette("colorblind")

# Create count plot with edgecolor and figsize
plt.figure(figsize=(5,5))
ax = sns.countplot(x='JobInvolvement', data=df, edgecolor="black")

# Add labels and title
ax.set_xlabel('JobInvolvement', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Distribution of JobInvolvement', fontsize=16)

# Show the plot
plt.show()
```



5.10 Observation of JobInvolvement column

We can say that more than 900 people in the dataset find themselves involved in their job, as indicated by their higher job involvement ratings.

5.11 JobSatisfaction

In [112…

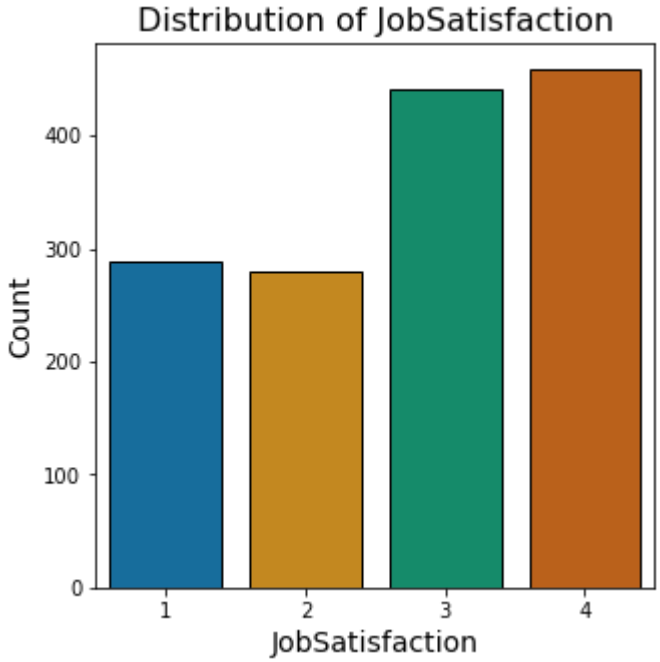
```
# creating a new DataFrame with the JobSatisfaction counts and percentages
JobSatisfaction_data = pd.DataFrame({'Counts': df['JobSatisfaction'].value_counts(), '
print(JobSatisfaction_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
   Counts Percentages
4     459      31.22%
3     442      30.07%
1     289      19.66%
2     280      19.05%
```

Visualizing the EnvironmentSatisfaction

In [128…

```
# Set colorblind-friendly palette
sns.set_palette("colorblind")

# Create count plot with edgecolor and figsize
plt.figure(figsize=(5,5))
ax = sns.countplot(x='JobSatisfaction', data=df, edgecolor="black")

# Add labels and title
ax.set_xlabel('JobSatisfaction', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Distribution of JobSatisfaction', fontsize=16)

# Show the plot
plt.show()
```



5.11 Observation of JobSatisfaction column

The majority of employees in the dataset are satisfied with their job, as indicated by the high count and percentage of individuals who rated their job satisfaction as 3 or 4 (30.07% and 31.22% respectively). Only a minority of individuals (19.66%) rated their job satisfaction as 1, indicating that they are unsatisfied with their job.

5.12 PerformanceRating

In [109…
```
# creating a new DataFrame with the PerformanceRating counts and percentages
PerformanceRating_data = pd.DataFrame({'Counts': df['PerformanceRating'].value_counts(
print(PerformanceRating_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
   Counts Percentages
3    1244      84.63%
4     226      15.37%
```

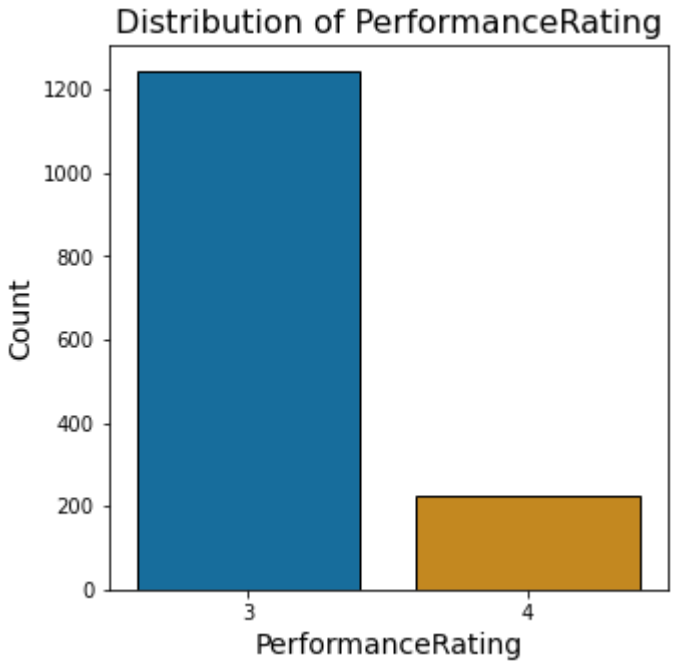
In [133…
```
df["PerformanceRating"].describe()
```

Out[133]:
```
count    1470.000000
mean        3.153741
std         0.360824
min         3.000000
25%         3.000000
50%         3.000000
75%         3.000000
max         4.000000
Name: PerformanceRating, dtype: float64
```

Visualizing the PerformanceRating

In [129…
```
# Set colorblind-friendly palette
sns.set_palette("colorblind")
```

```
# Create count plot with edgecolor and figsize
plt.figure(figsize=(5,5))
ax = sns.countplot(x='PerformanceRating', data=df, edgecolor="black")

# Add labels and title
ax.set_xlabel('PerformanceRating', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Distribution of PerformanceRating', fontsize=16)

# Show the plot
plt.show()
```



5.12 Observation of PerformanceRating column

Based on the provided statistical distribution, we can see that the average performance rating is 3.15, with a standard deviation of 0.36. The minimum rating is 3, while the maximum rating is 4. The 25th, 50th, and 75th percentiles all have a rating of 3, indicating that the majority of respondents rated performance as meeting expectations.

5.13 WorklifeBalance

In [108…

```
# creating a new DataFrame with the WorklifeBalance counts and percentages
WorkLifeBalance_data = pd.DataFrame({'Counts': df['WorkLifeBalance'].value_counts(),
print(WorkLifeBalance_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
   Counts Percentages
3     893      60.75%
2     344      23.40%
4     153      10.41%
1      80       5.44%
```

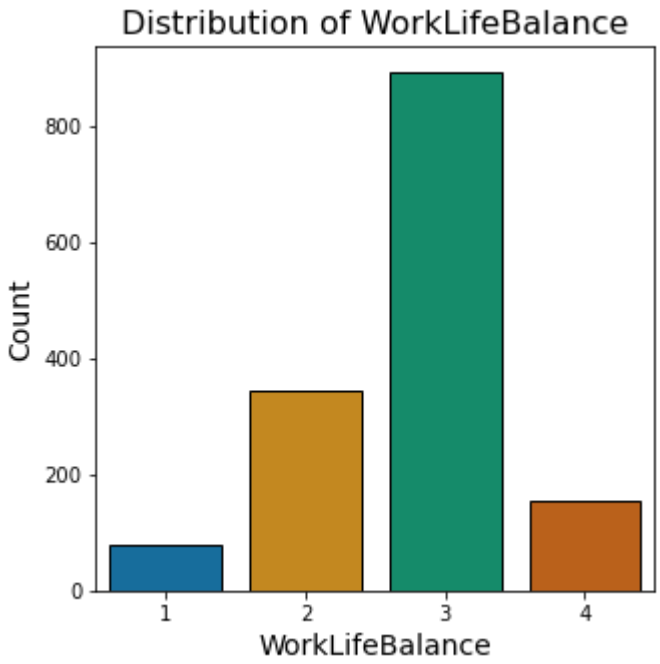
In [132…

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

Out[132]:
```
count    1470.000000
mean        2.761224
std         0.706476
min         1.000000
25%         2.000000
50%         3.000000
75%         3.000000
max         4.000000
Name: WorkLifeBalance, dtype: float64
```

Visualizing the WorkLifeBalance

In [135…
```
# Set colorblind-friendly palette
sns.set_palette("colorblind")

# Create count plot with edgecolor and figsize
plt.figure(figsize=(5,5))
ax = sns.countplot(x='WorkLifeBalance', data=df, edgecolor="black")

# Add labels and title
ax.set_xlabel('WorkLifeBalance', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Distribution of WorkLifeBalance', fontsize=16)

# Show the plot
plt.show()
```



5.13 Observation of WorklifeBalance column

Based on the provided statistical distribution, the average rating for work-life balance is 2.76 with a standard deviation of 0.71. The minimum rating is 1, while the maximum rating is 4. Additionally, 25% of the ratings are below 2, while 75% of the ratings are at or above 3. This suggests that the majority of respondents reported satisfactory work-life balance, but there is still room for improvement. It's important to note that individual perceptions of work-life

balance can vary greatly and may be influenced by various factors such as workload, work culture, and personal circumstances.

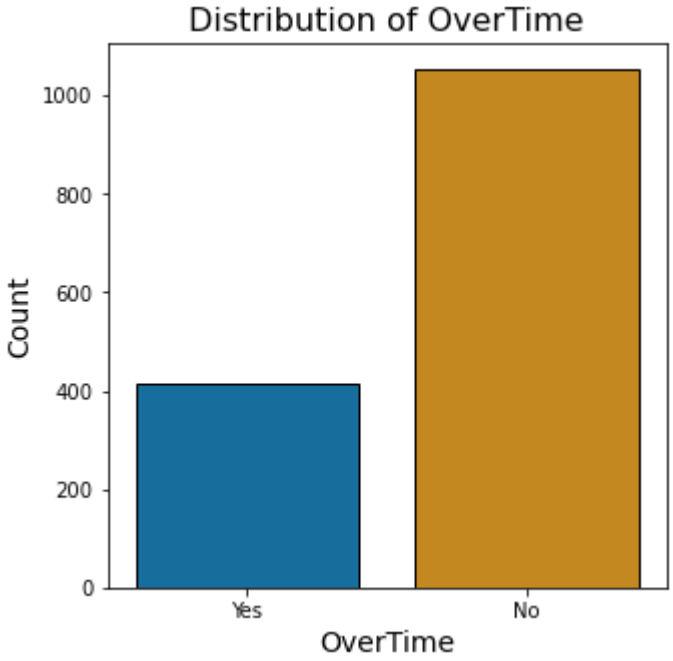
5.14 OverTime

In [134…
```
# creating a new DataFrame with the OverTime counts and percentages
OverTime_data = pd.DataFrame({'Counts': df['OverTime'].value_counts(), 'Percentages':
print(OverTime_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
     Counts Percentages
No     1054      71.70%
Yes     416      28.30%
```

Visualizing the OverTime

In [136…
```
# Set colorblind-friendly palette
sns.set_palette("colorblind")

# Create count plot with edgecolor and figsize
plt.figure(figsize=(5,5))
ax = sns.countplot(x='OverTime', data=df, edgecolor="black")

# Add labels and title
ax.set_xlabel('OverTime', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Distribution of OverTime', fontsize=16)

# Show the plot
plt.show()
```



5.14 Observation of OverTime column

The given counts and percentages suggest that a significant number of employees (28.3%) worked overtime. Although the majority of employees (71.7%) did not work overtime, it may be

important to investigate the reasons why employees worked overtime, the frequency of overtime work, and the impact of overtime on employee performance and well-being. Comparing attrition rates between employees who worked overtime and those who didn't can also help identify any correlation between overtime work and employee attrition.

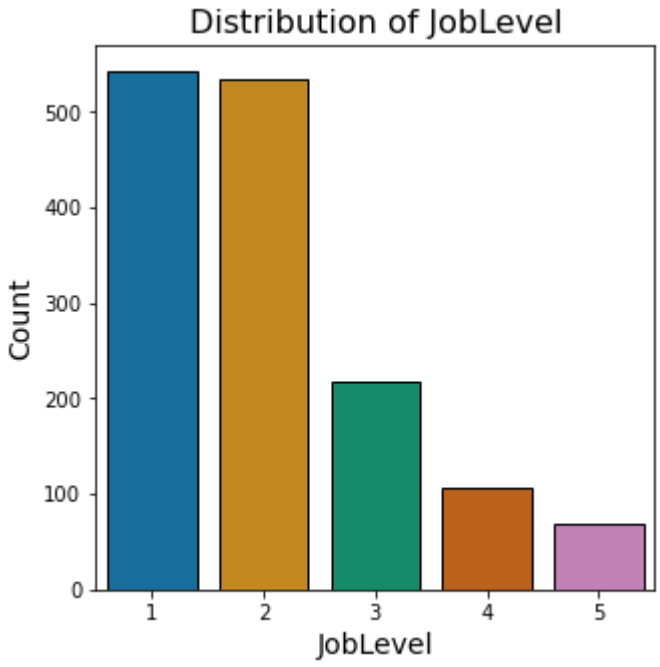
5.14 Job Level

In [193…
```
# creating a new DataFrame with the JobLevel counts and percentages
JobLevel_data = pd.DataFrame({'Counts': df['JobLevel'].value_counts(), 'Percentages':
print(JobLevel_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
   Counts Percentages
1     543      36.94%
2     534      36.33%
3     218      14.83%
4     106       7.21%
5      69       4.69%
```

Visualizing Job Level

In [194…
```
# Set colorblind-friendly palette
sns.set_palette("colorblind")

# Create count plot with edgecolor and figsize
plt.figure(figsize=(5,5))
ax = sns.countplot(x='JobLevel', data=df, edgecolor="black")

# Add labels and title
ax.set_xlabel('JobLevel', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Distribution of JobLevel', fontsize=16)

# Show the plot
plt.show()
```



5.14 Observation of Joblevel column

More than 1000 People are low level and only 10% are involved in tier 4 or 5 level jobs. It means that the organization may have a large number of entry-level or junior roles. These roles may require less experience or education, or they may serve as a starting point for employees to enter the company and gain experience before moving up the ranks.On the other hand, the statement also suggests that only 10% of the total population is involved in tier 4 or 5 level jobs. This could mean that these positions are relatively rare or hard to attain, and may require more specialized skills, education, or experience.

Overall, this type of distribution of job roles could indicate that the organization has a well-defined hierarchical structure, with clear paths for advancement from entry-level positions to higher-level ones.

6. Outliers Check for Numerical Columns

In [86]:
```
# select only integer type columns
int_cols = df.select_dtypes(include='int')
```

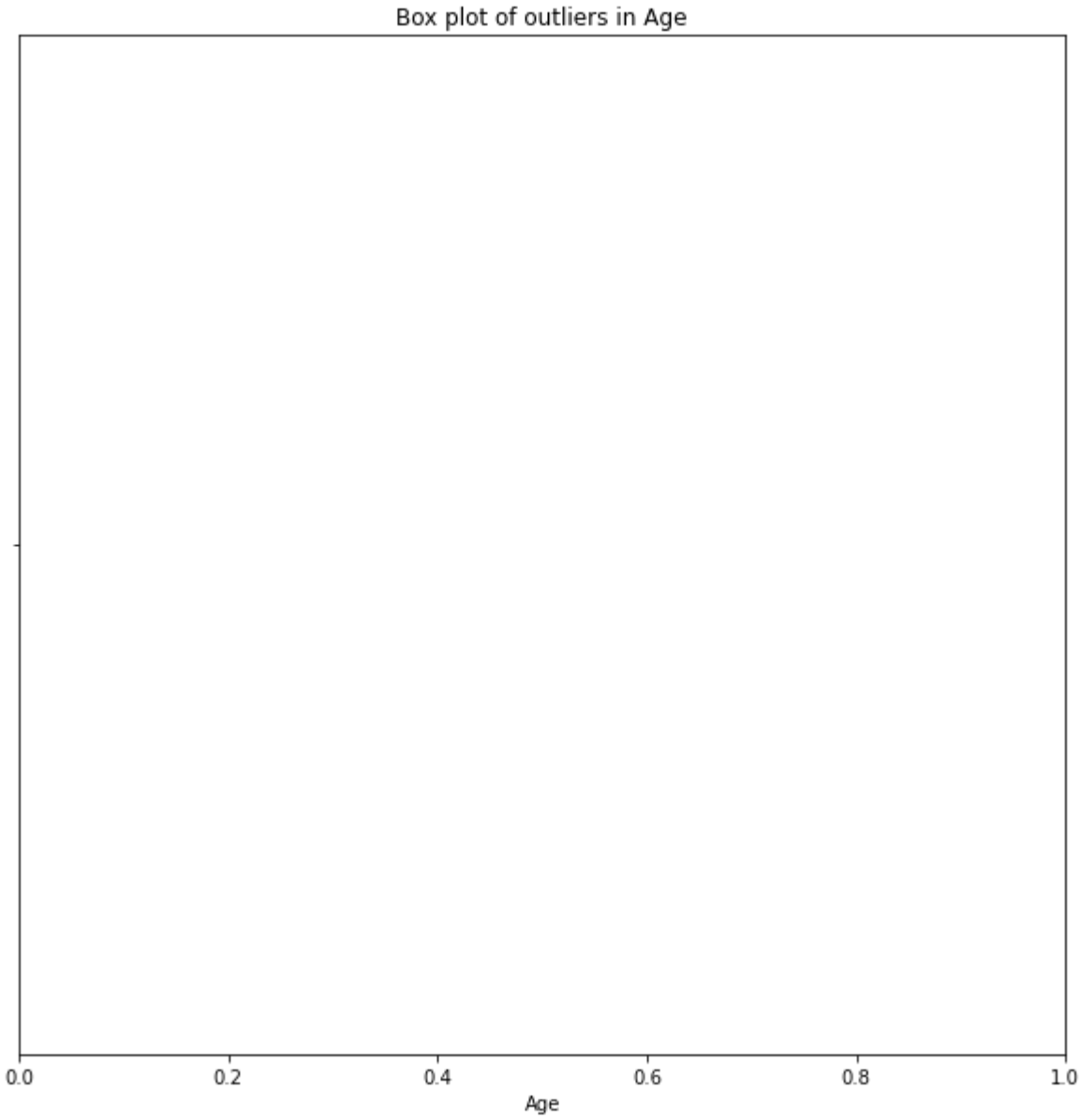
In [87]:
```
def check_Outliers(x):
    from scipy import stats
    for i in x:
    # Calculate z-scores of column 'DistanceFromHome'
        z_scores = stats.zscore(df[i])

        # Identify outliers with a z-score of greater than 3 or less than -3
        outliers = df[(z_scores > 3) | (z_scores < -3)]
        print("{} {}".format(i,outliers.shape))
```

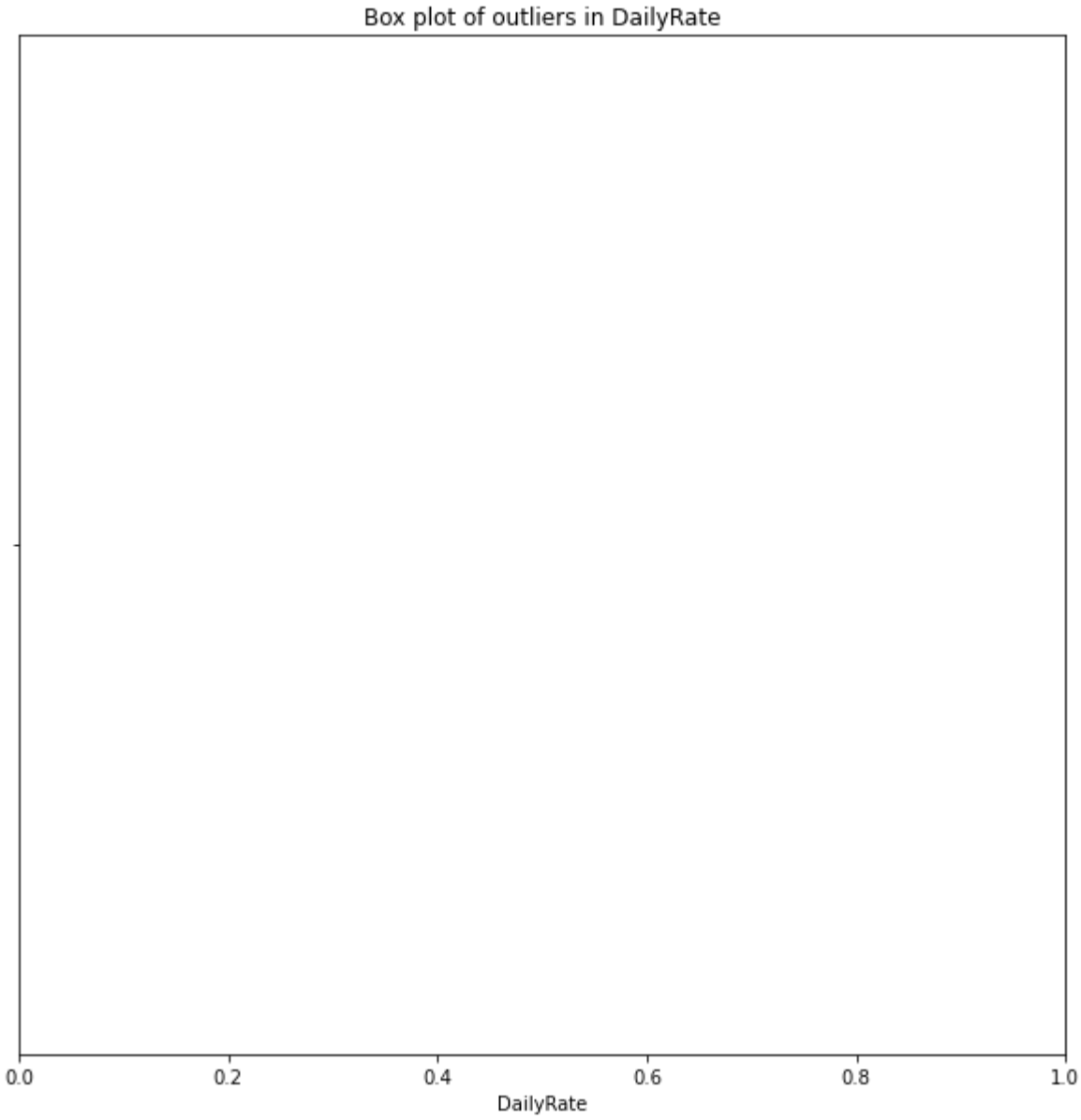
In [88]:
```
check_Outliers(int_cols)
```

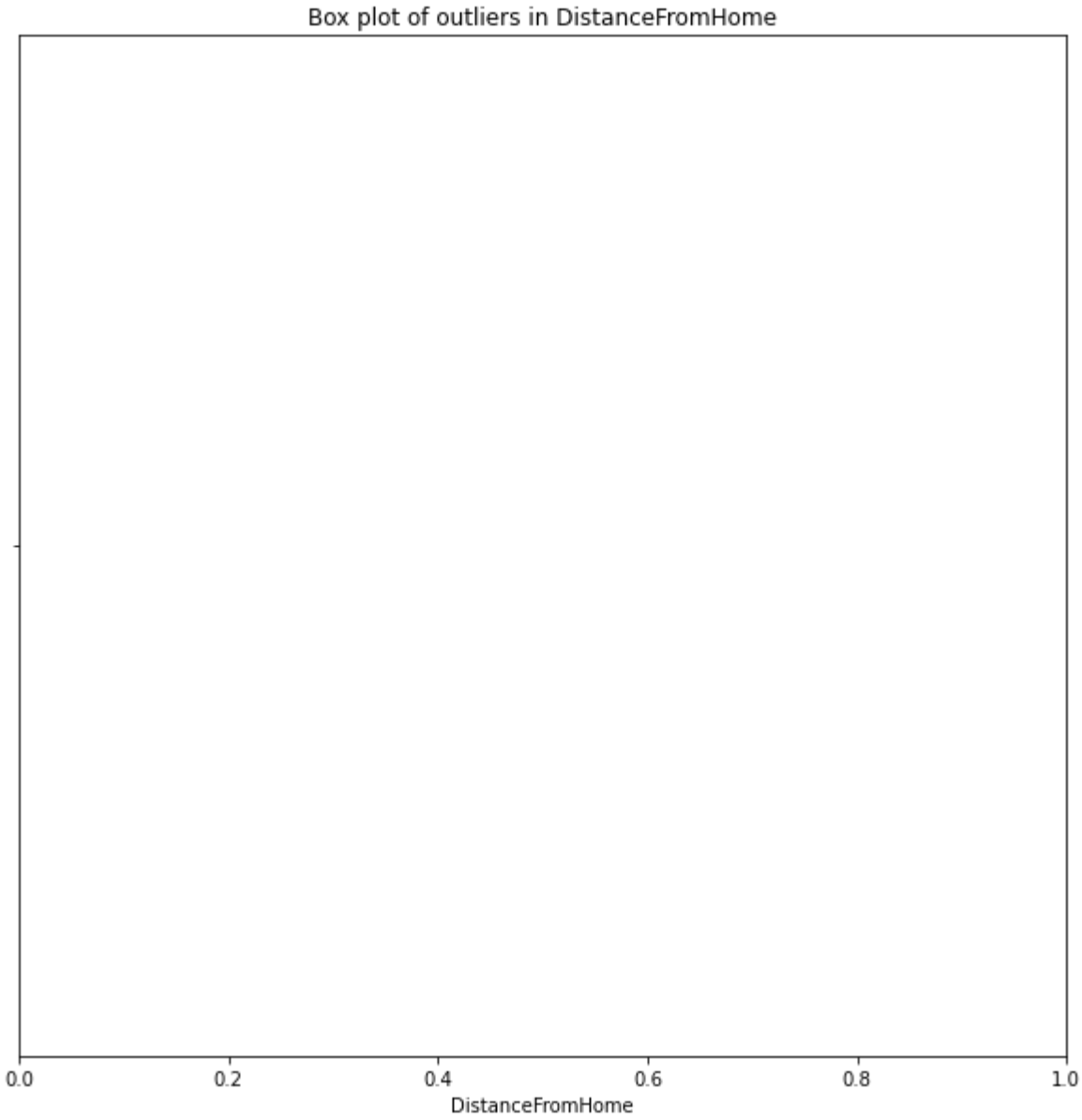
```
Age (0, 32)
DailyRate (0, 32)
DistanceFromHome (0, 32)
Education (0, 32)
EmployeeNumber (0, 32)
EnvironmentSatisfaction (0, 32)
HourlyRate (0, 32)
JobInvolvement (0, 32)
JobLevel (0, 32)
JobSatisfaction (0, 32)
MonthlyIncome (0, 32)
MonthlyRate (0, 32)
NumCompaniesWorked (0, 32)
PercentSalaryHike (0, 32)
PerformanceRating (0, 32)
RelationshipSatisfaction (0, 32)
StockOptionLevel (0, 32)
TotalWorkingYears (16, 32)
TrainingTimesLastYear (0, 32)
WorkLifeBalance (0, 32)
YearsAtCompany (25, 32)
YearsInCurrentRole (13, 32)
YearsSinceLastPromotion (42, 32)
YearsWithCurrManager (14, 32)
```

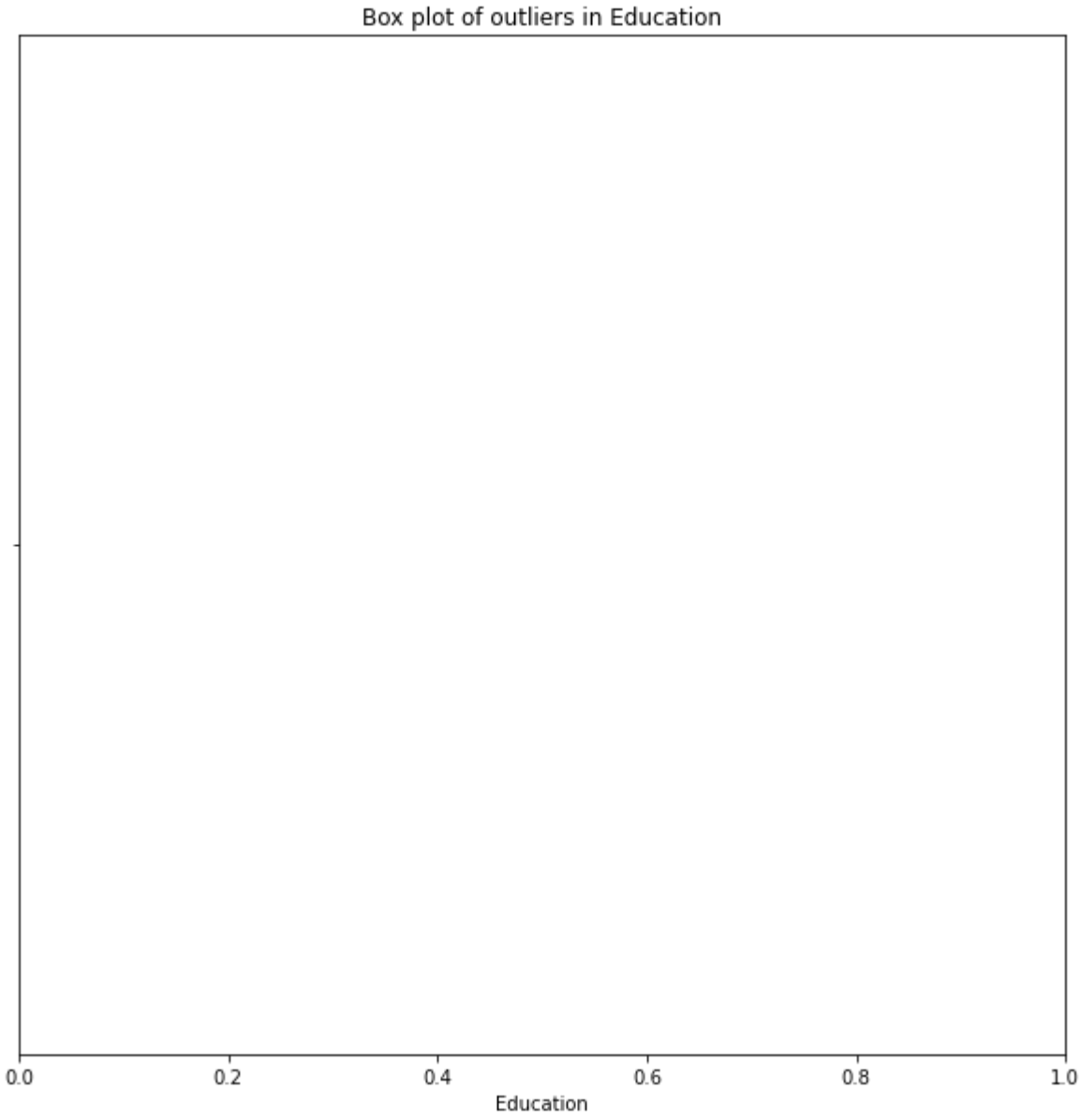
In [103…

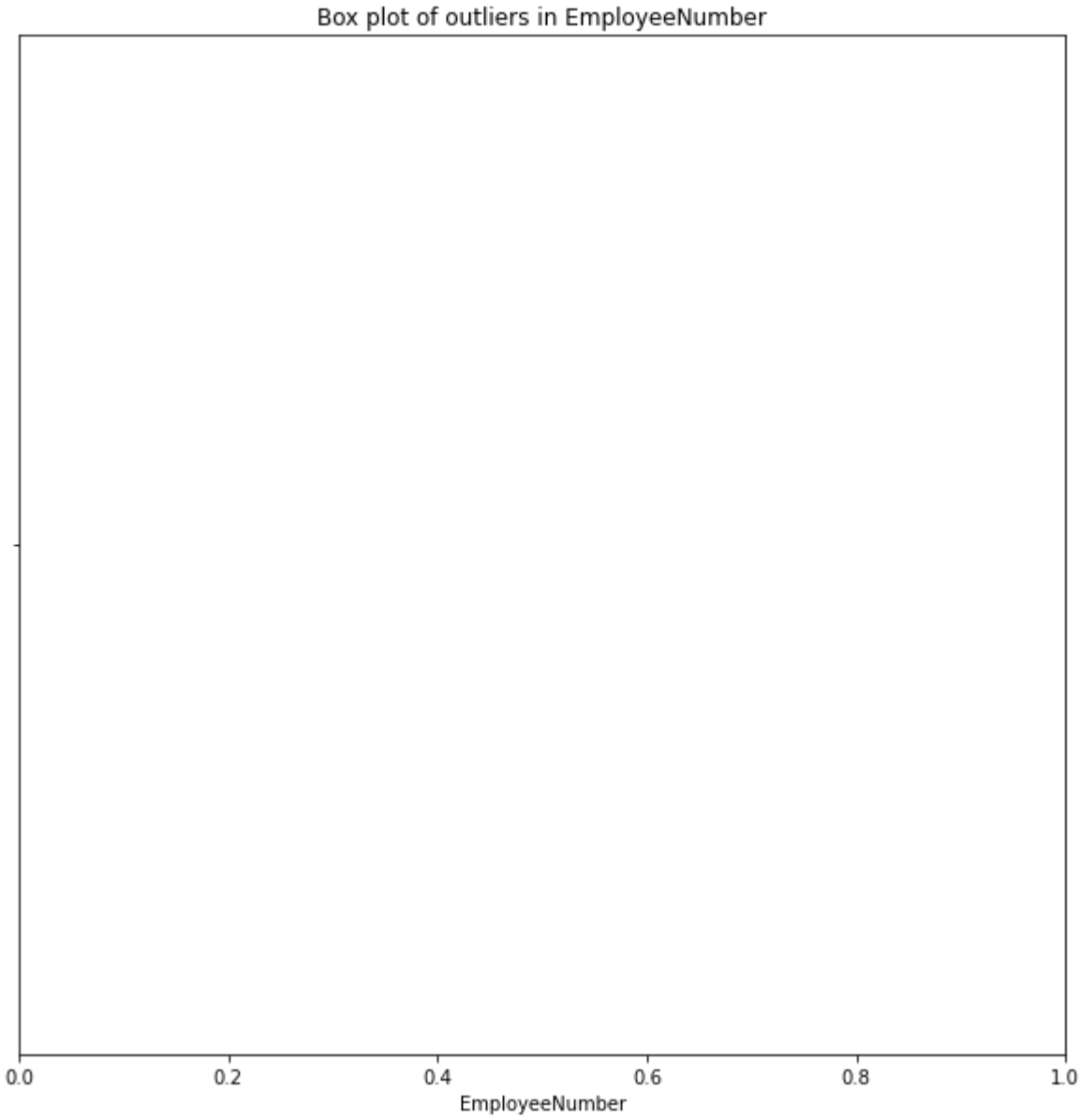
```
def addressing_Outliers(x, data):
    from scipy import stats
    for i in x:
        # Calculate z-scores of column 'DistanceFromHome'
        z_scores = stats.zscore(data[i])

        # Identify outliers with a z-score of greater than 3 or less than -3
        outliers = data[(z_scores > 3) | (z_scores < -3)]

        # Visualize the outliers using a box plot
        fig, ax = plt.subplots(figsize=(10, 10))
        sns.boxplot(x=outliers[i],ax=ax)
        plt.title("Box plot of outliers in {}".format(i))
        plt.show()

# apply the function to the dataframe
addressing_Outliers(int_cols, df)
```

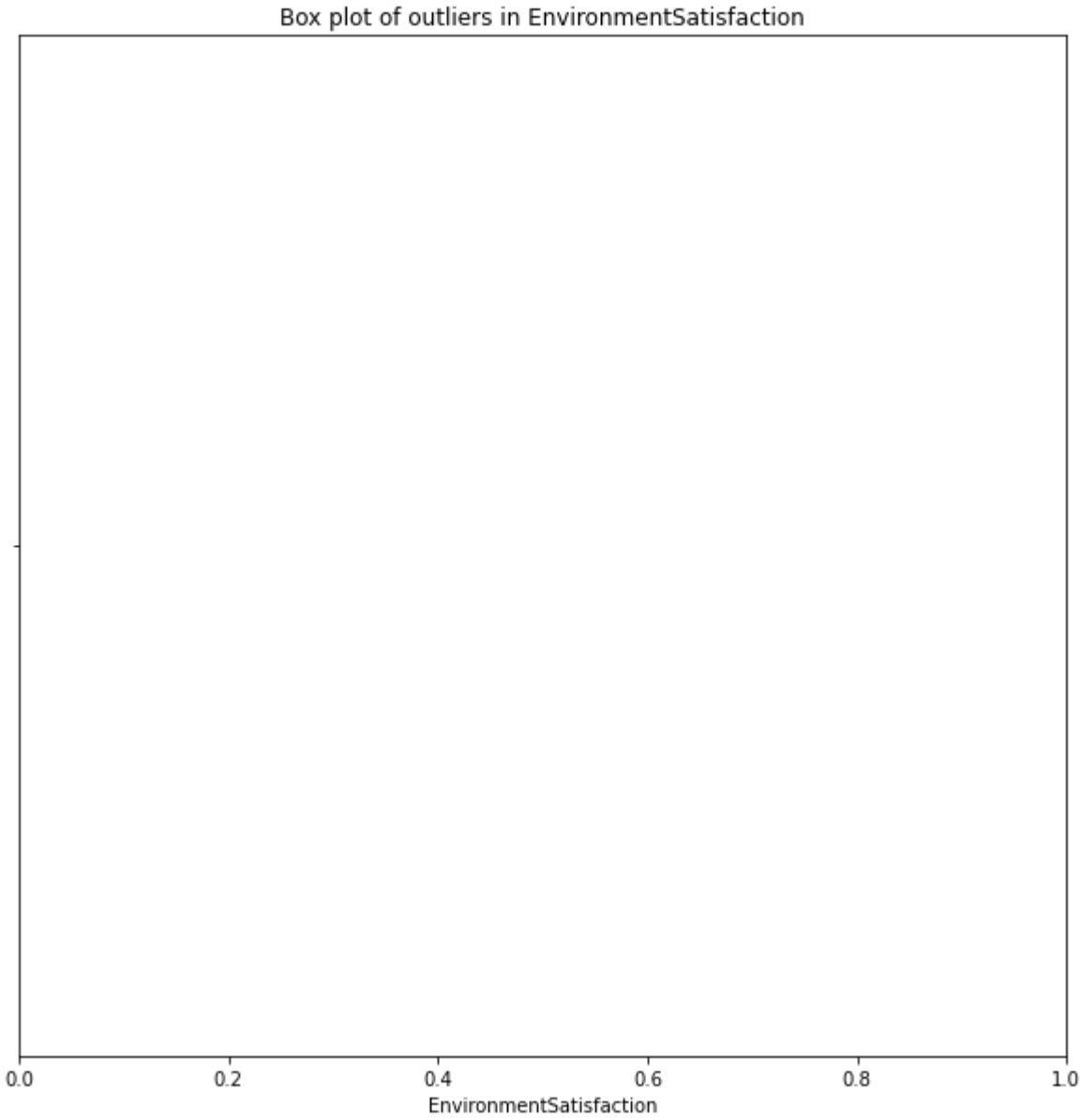

Box plot of outliers in Age
0.0
0.2
0.4
0.6
0.8
1.0
Age

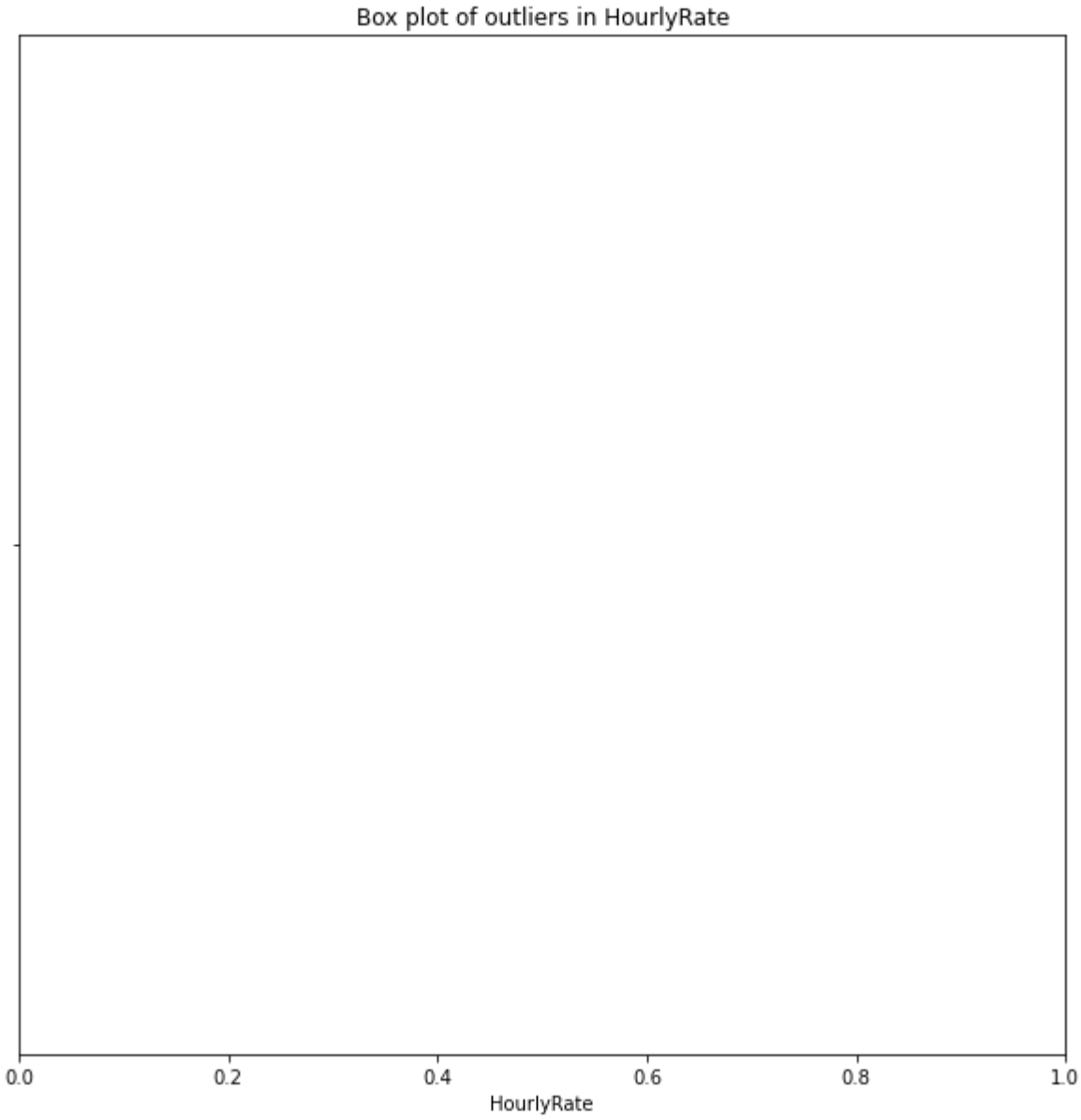

Box plot of outliers in DailyRate
0.0
0.2
0.4
0.6
0.8
1.0
DailyRate

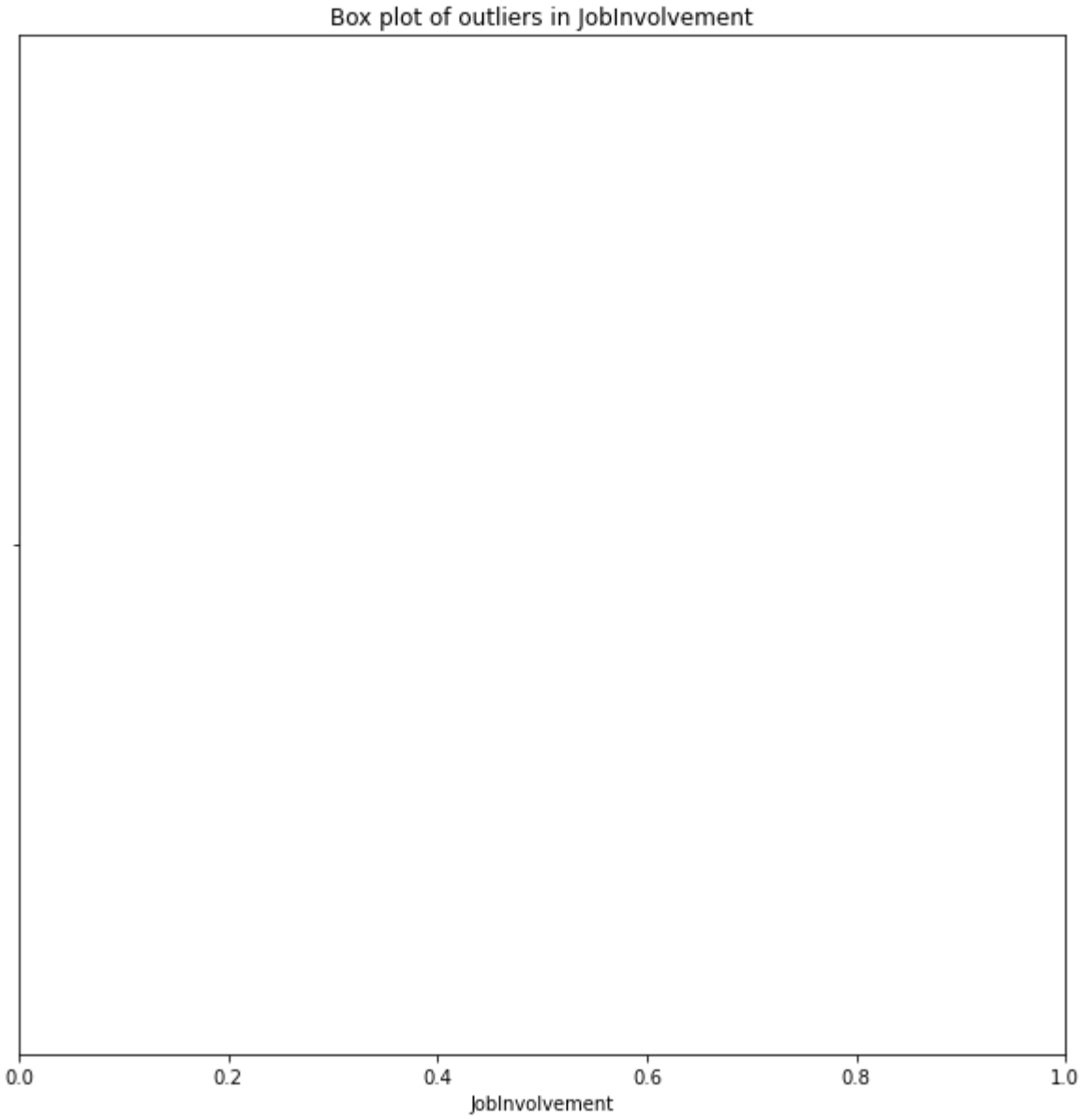

Box plot of outliers in DistanceFromHome
0.0
0.2
0.4
0.6
0.8
1.0
DistanceFromHome

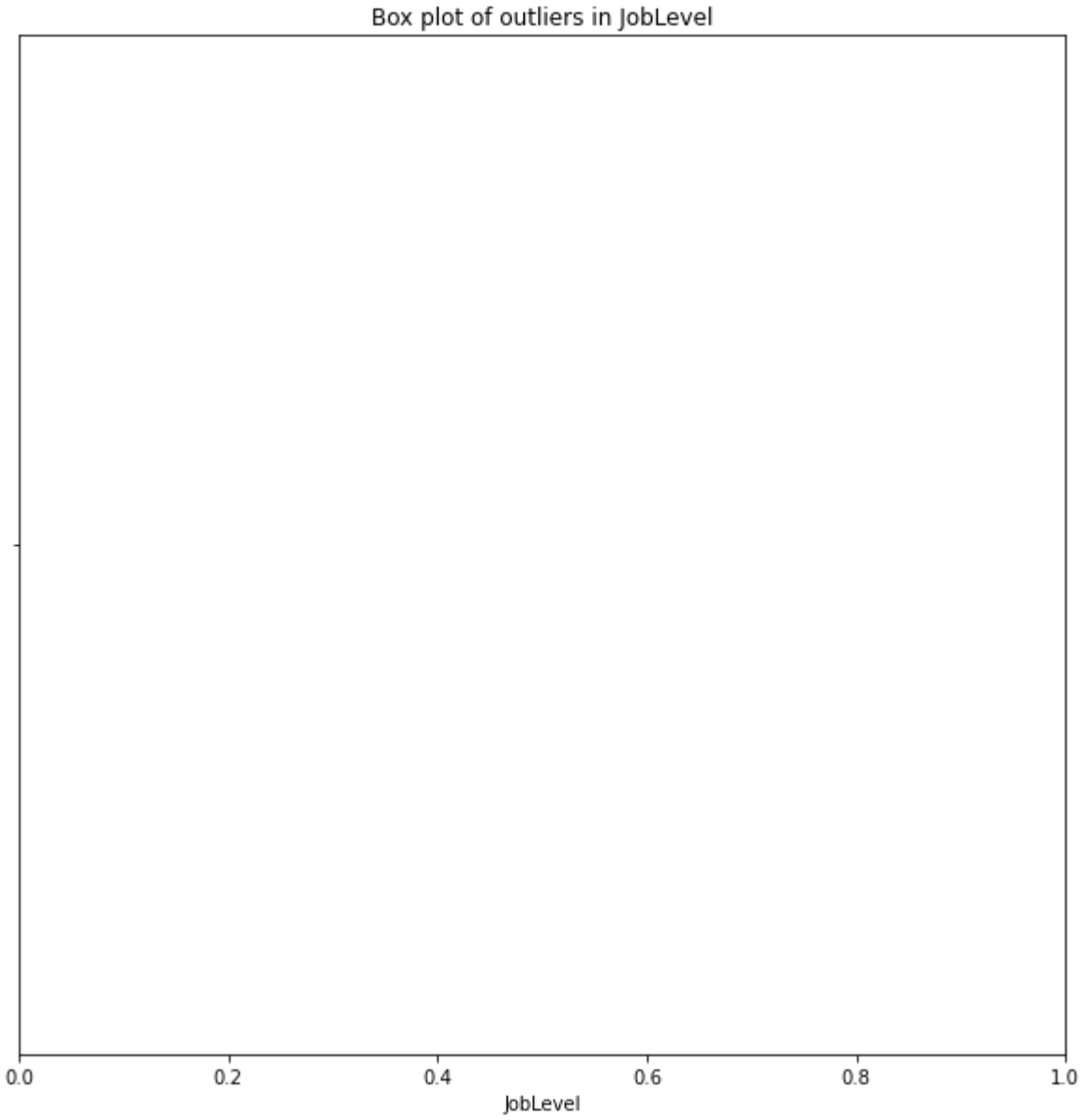

Box plot of outliers in Education
0.0
0.2
0.4
0.6
0.8
1.0
Education

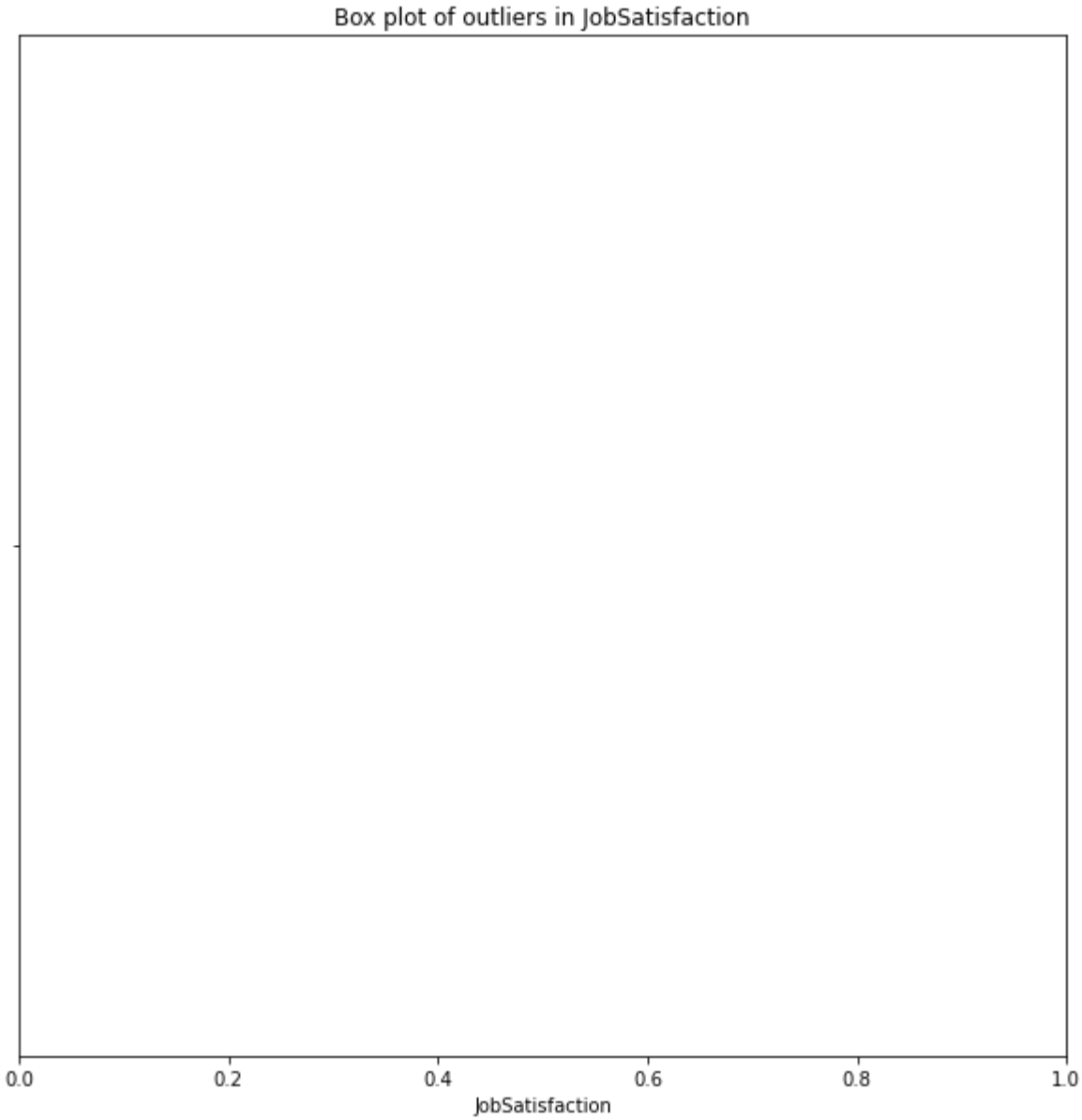

Box plot of outliers in EmployeeNumber
0.0
0.2
0.4
0.6
0.8
1.0
EmployeeNumber

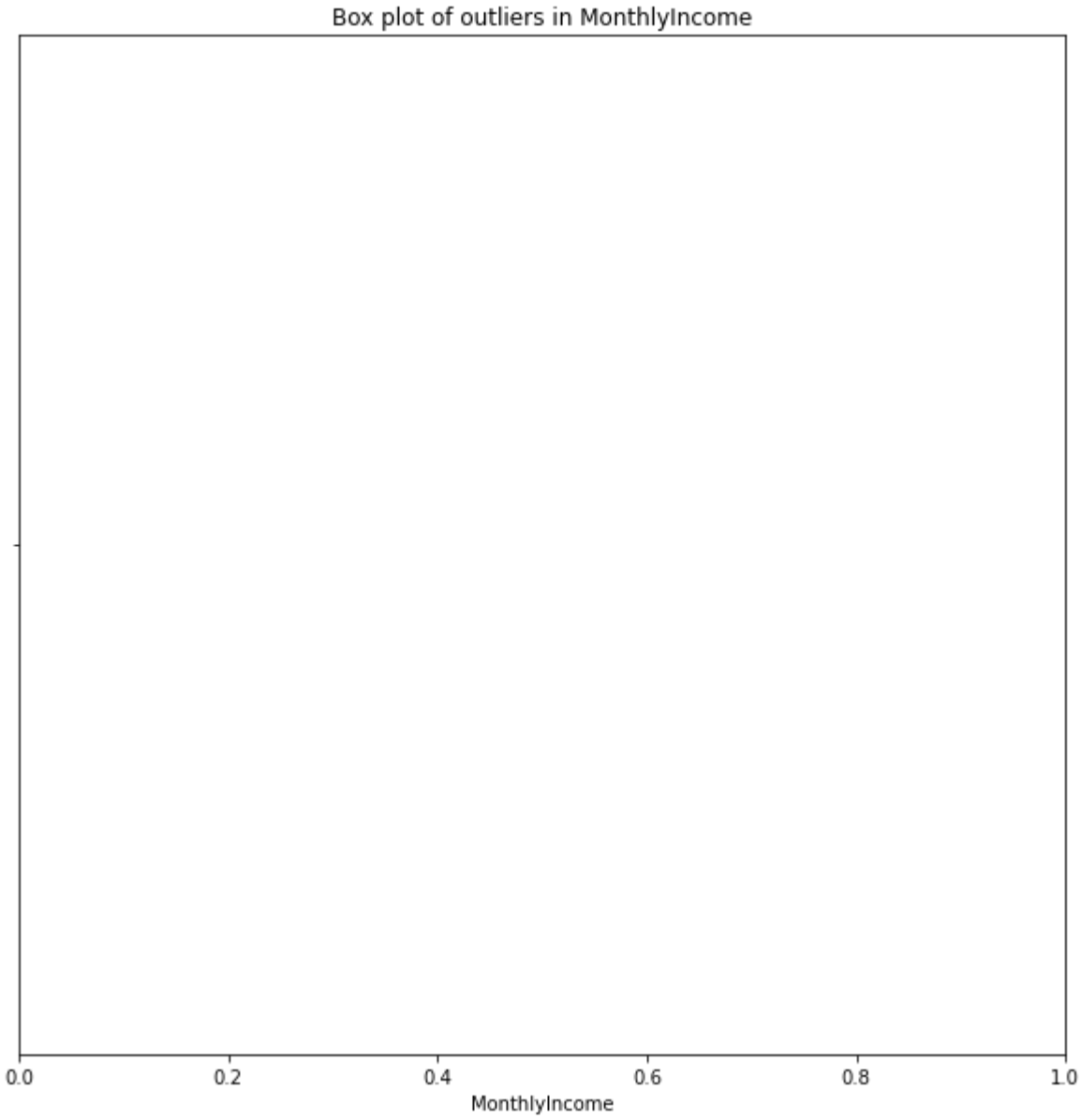

Box plot of outliers in EnvironmentSatisfaction
0.0
0.2
0.4
0.6
0.8
1.0
EnvironmentSatisfaction

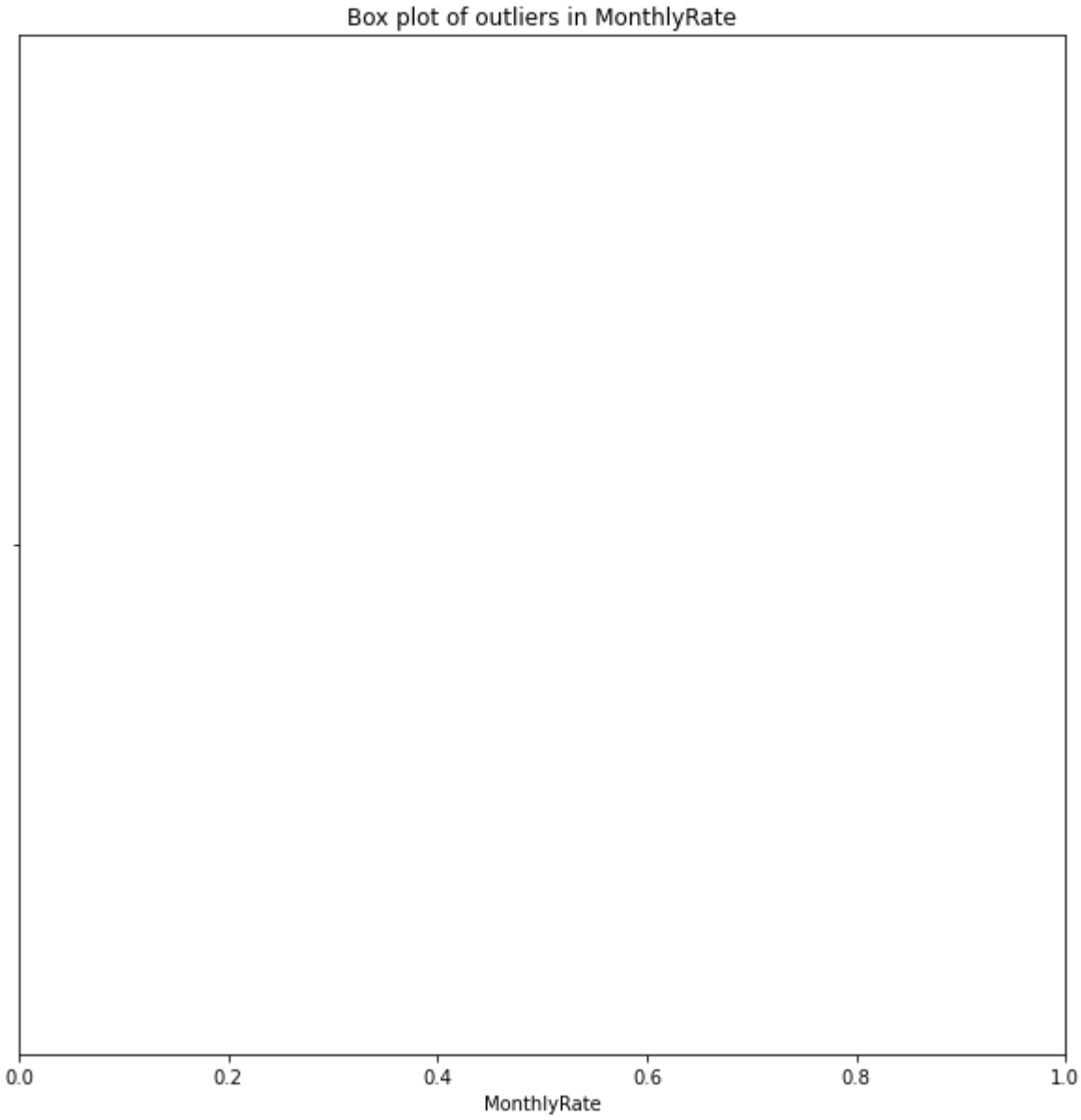

Box plot of outliers in HourlyRate
0.0
0.2
0.4
0.6
0.8
1.0
HourlyRate

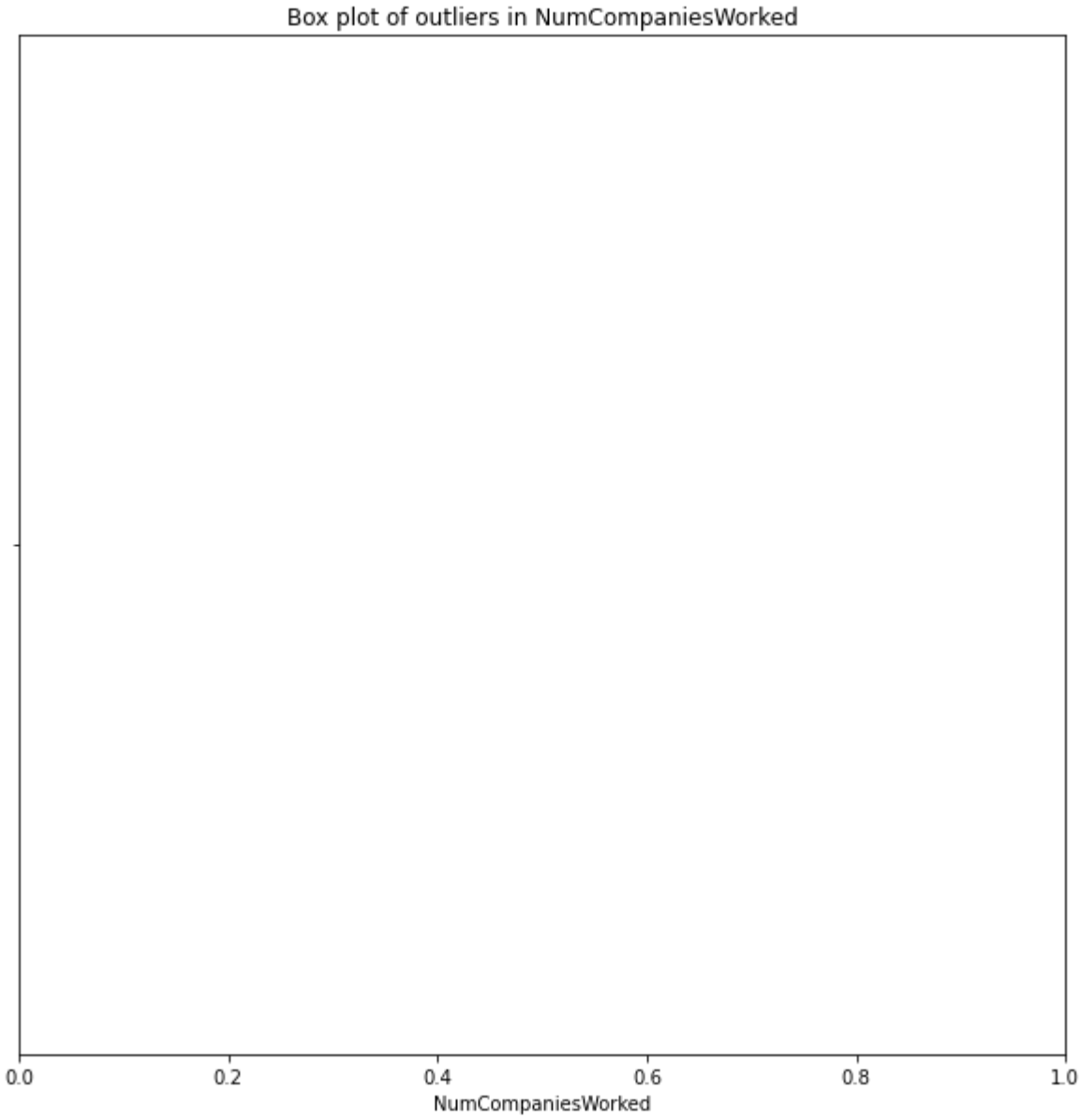

Box plot of outliers in JobInvolvement
0.0
0.2
0.4
0.6
0.8
1.0
JobInvolvement

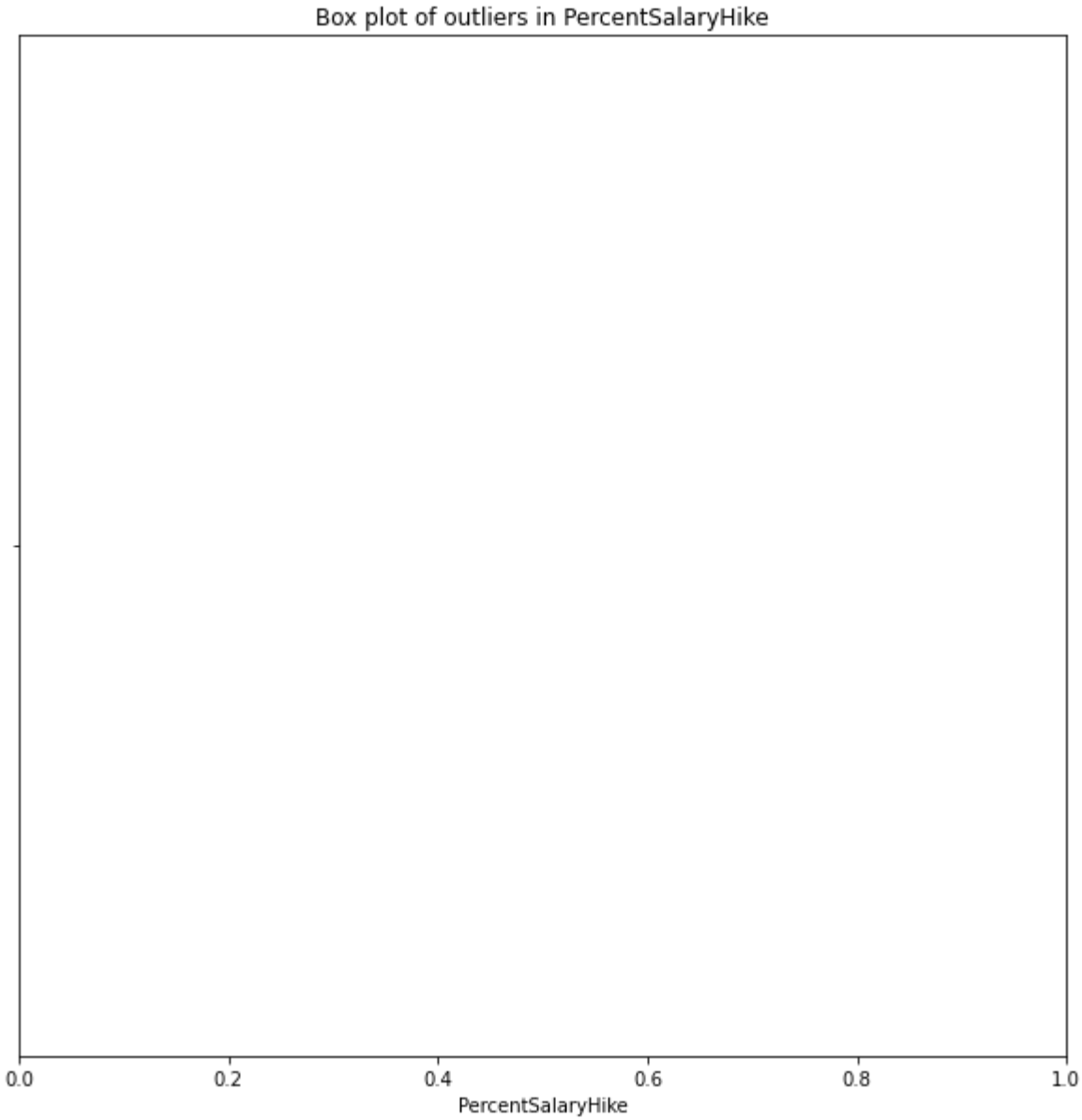

Box plot of outliers in JobLevel
0.0
0.2
0.4
0.6
0.8
1.0
JobLevel

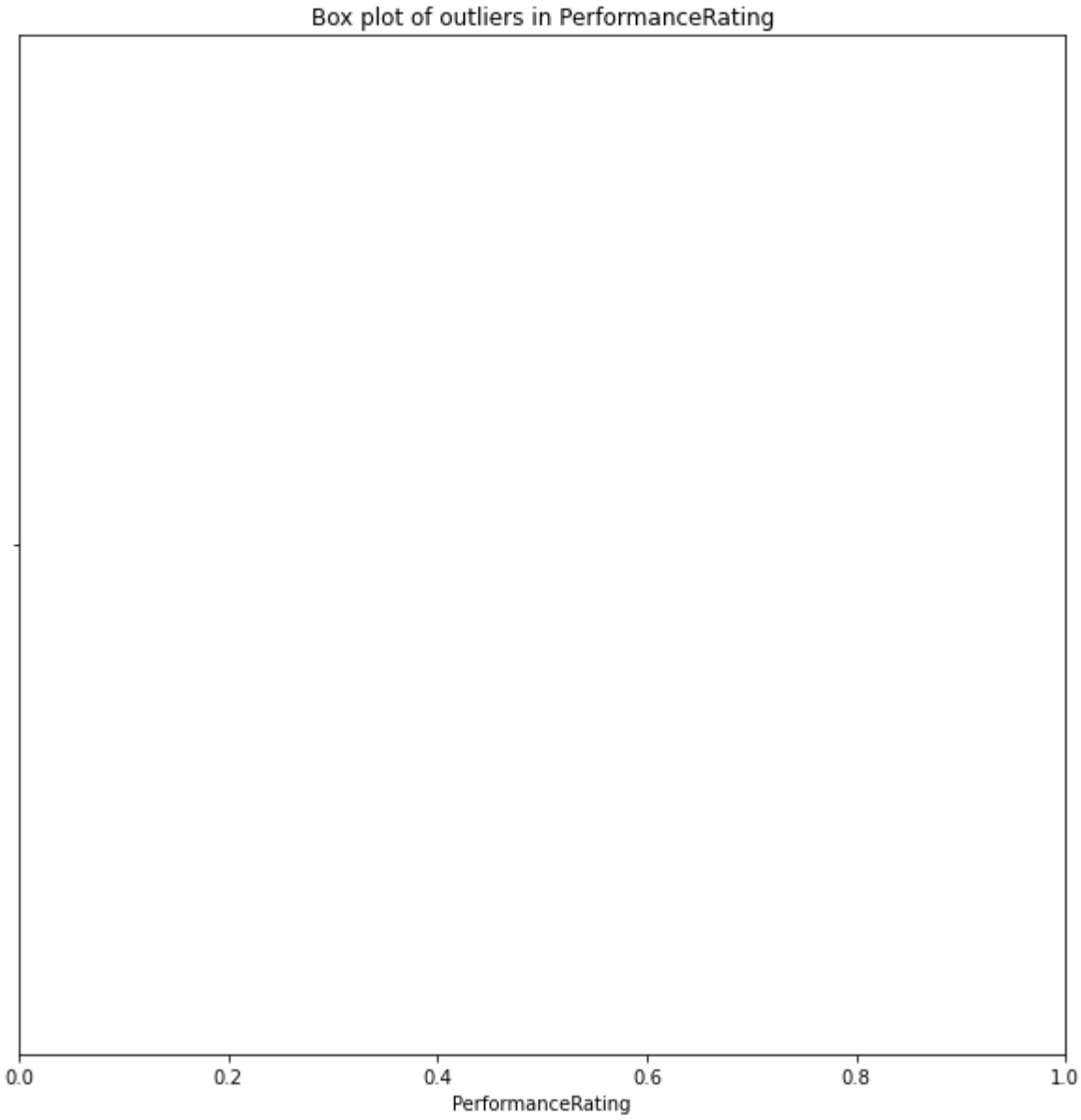

Box plot of outliers in JobSatisfaction
0.0
0.2
0.4
0.6
0.8
1.0
JobSatisfaction

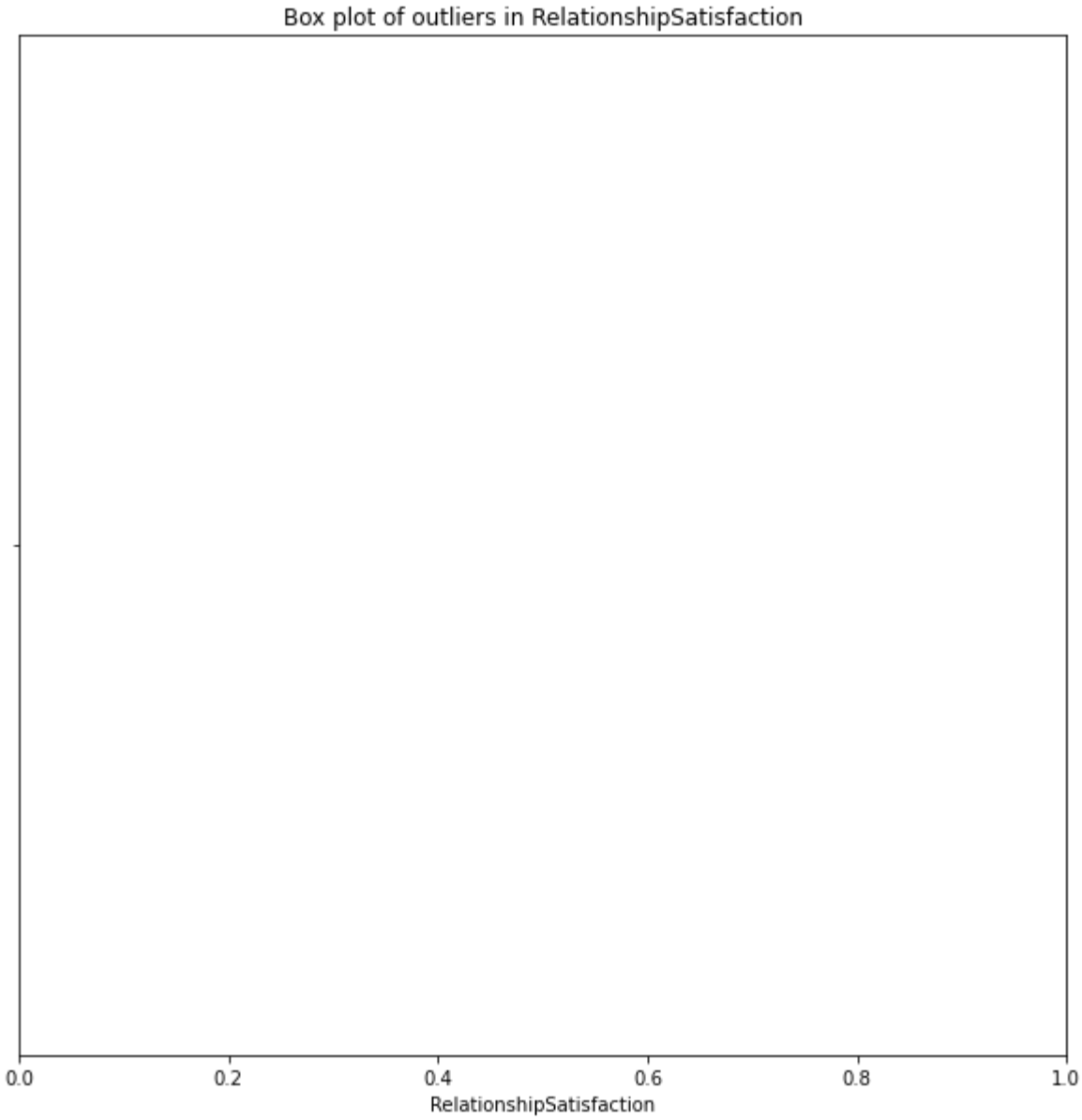

Box plot of outliers in MonthlyIncome
0.0
0.2
0.4
0.6
0.8
1.0
MonthlyIncome

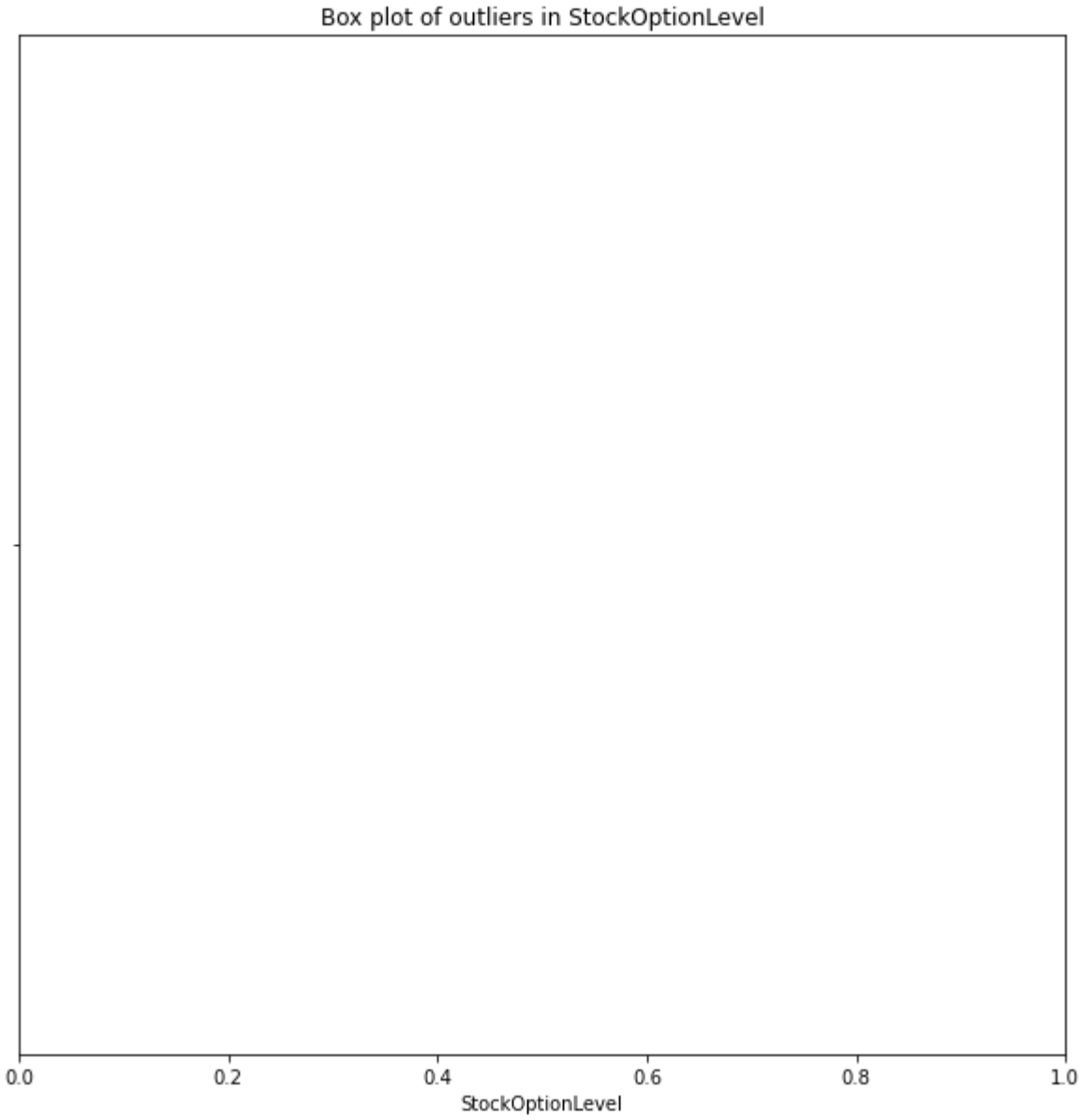

Box plot of outliers in MonthlyRate
0.0
0.2
0.4
0.6
0.8
1.0
MonthlyRate

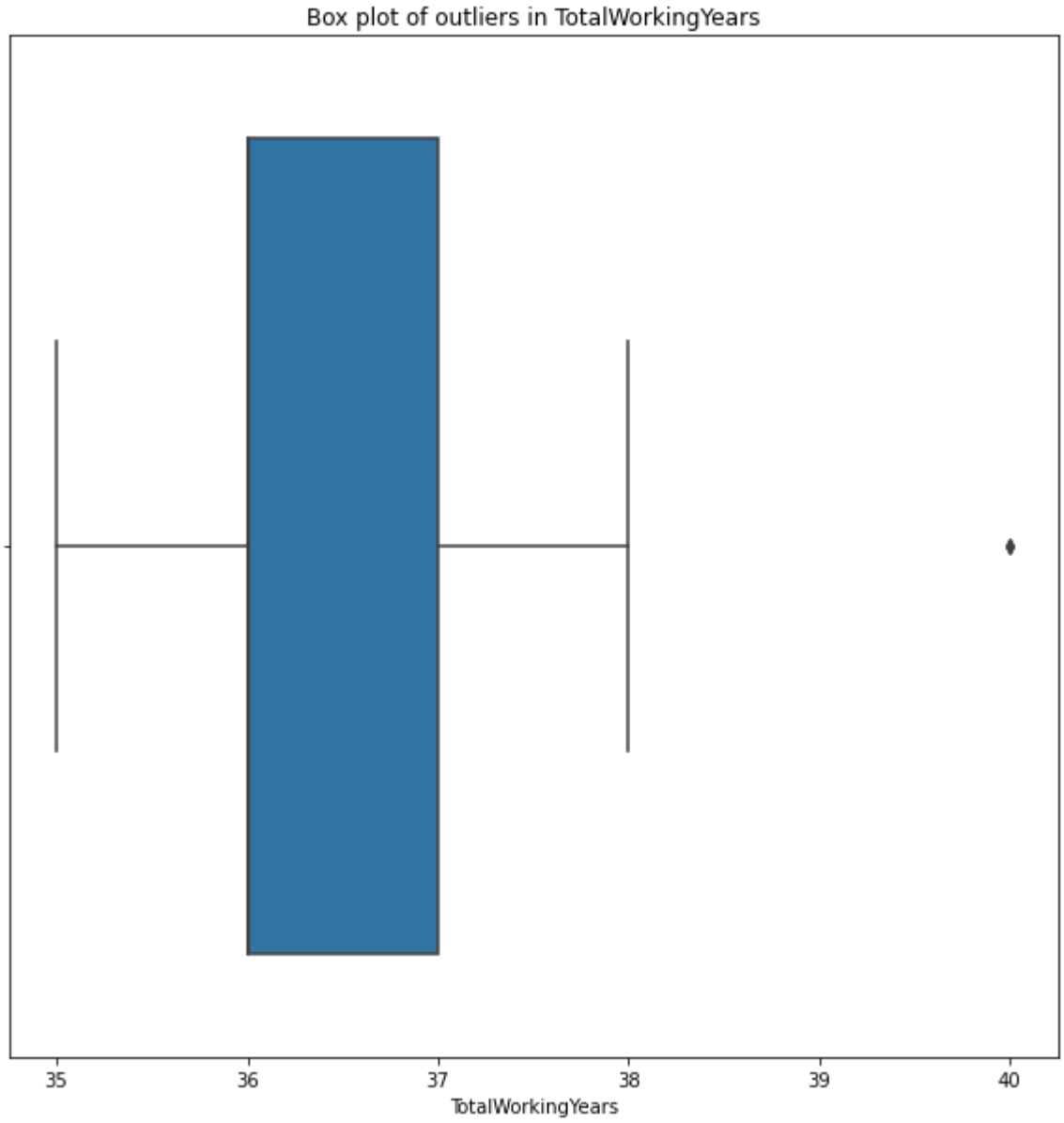

Box plot of outliers in NumCompaniesWorked
0.0
0.2
0.4
0.6
0.8
1.0
NumCompaniesWorked

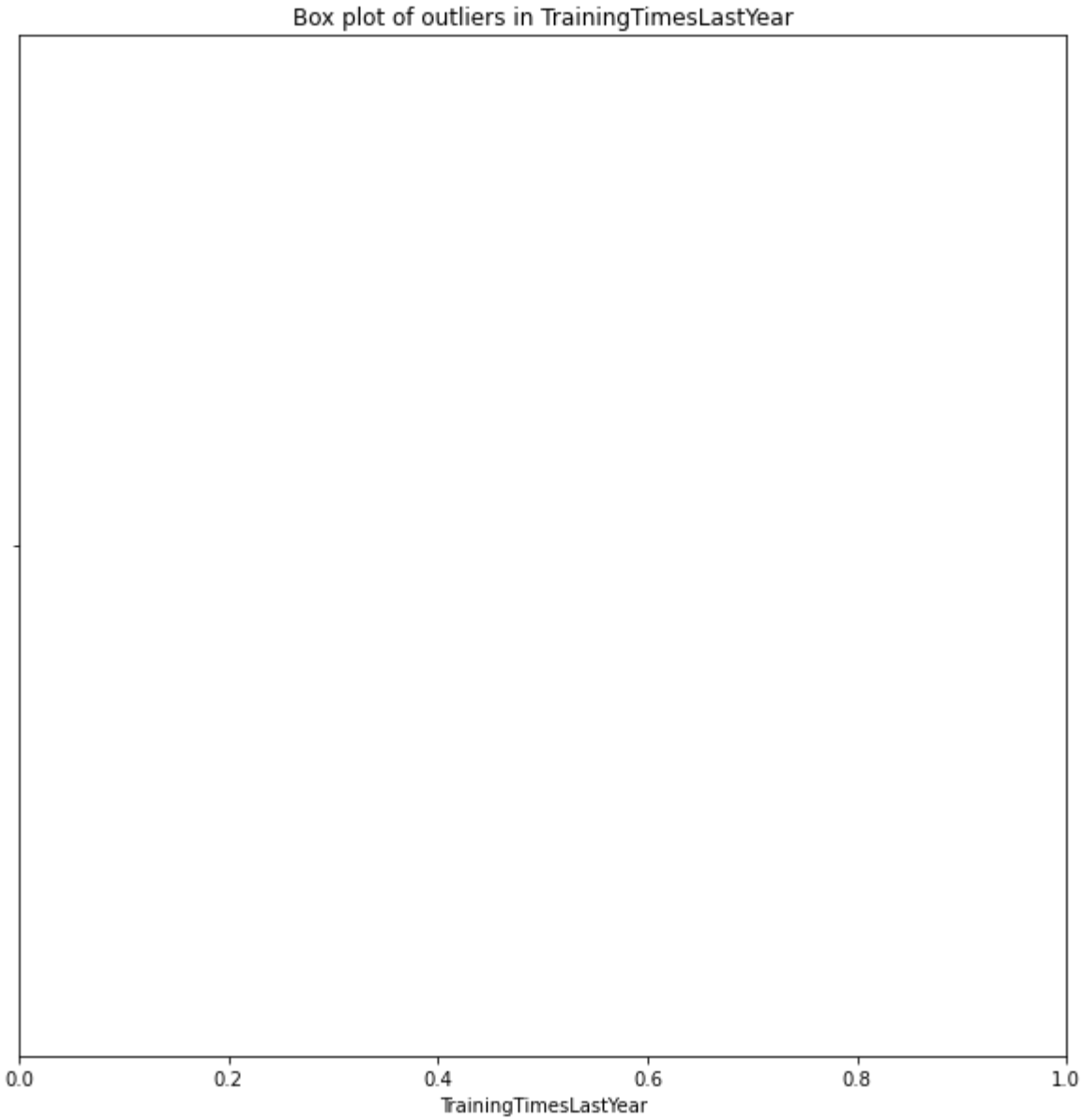

Box plot of outliers in PercentSalaryHike
0.0
0.2
0.4
0.6
0.8
1.0
PercentSalaryHike

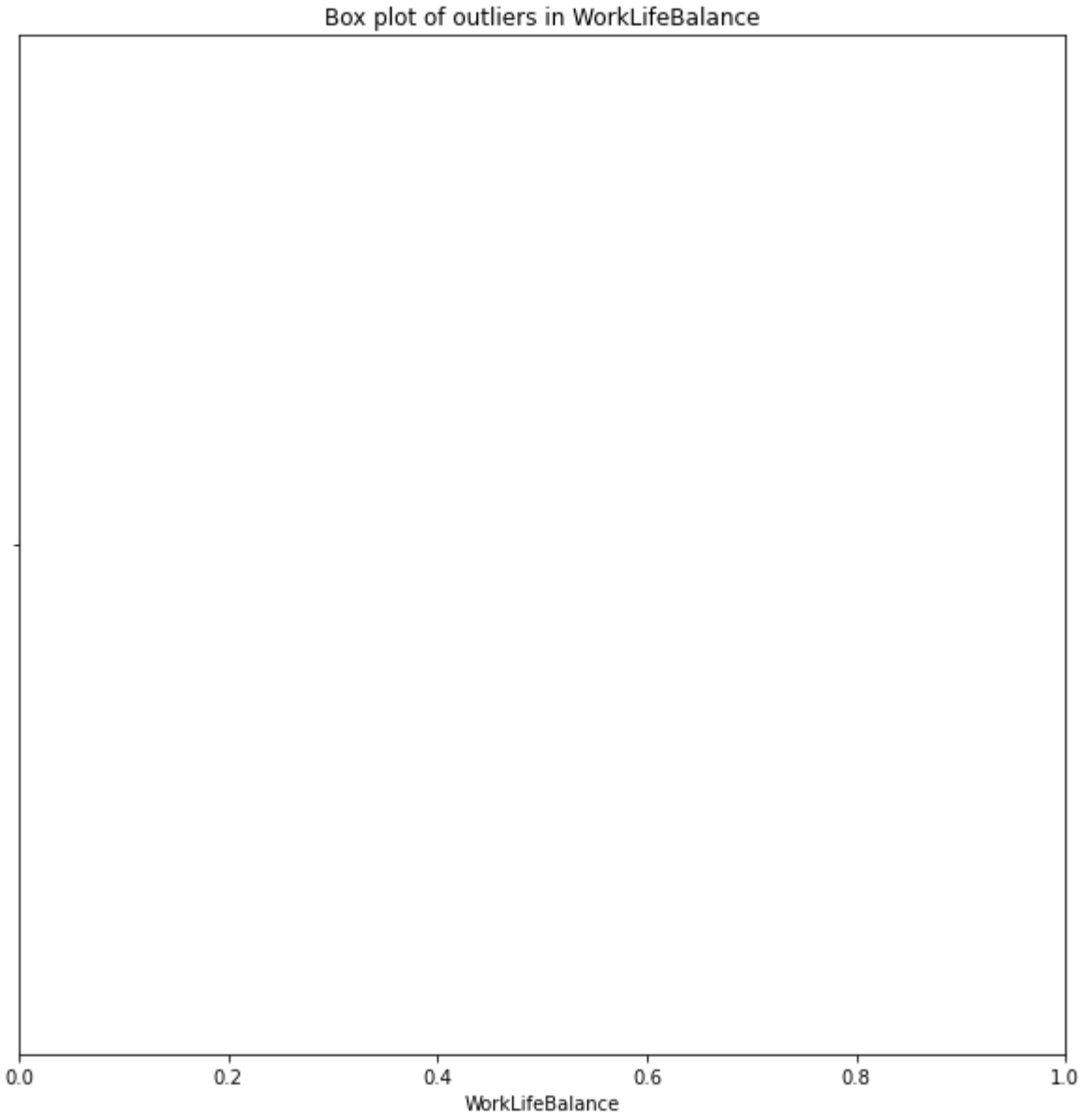

Box plot of outliers in PerformanceRating
0.0
0.2
0.4
0.6
0.8
1.0
PerformanceRating

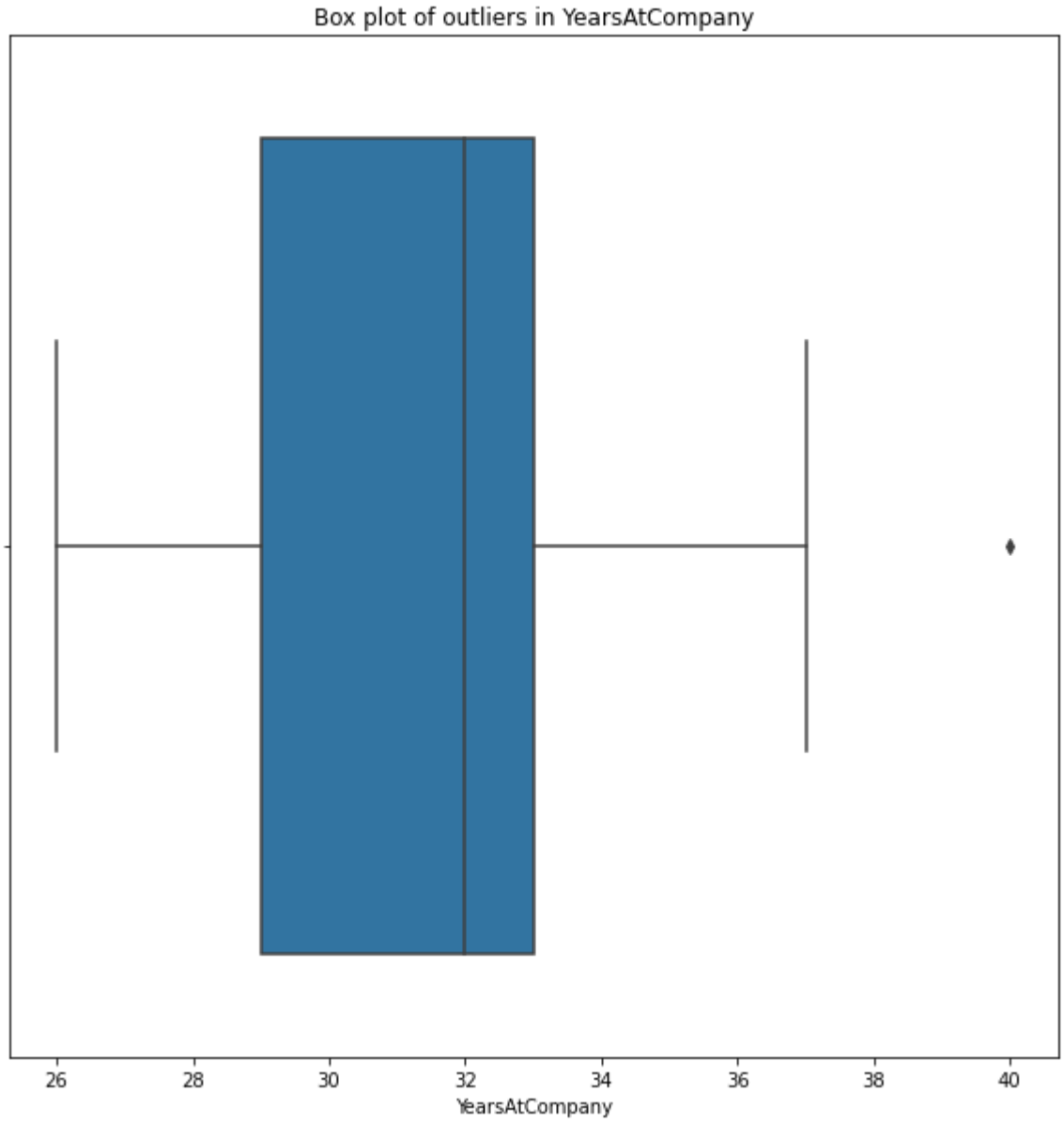

Box plot of outliers in RelationshipSatisfaction
0.0
0.2
0.4
0.6
0.8
1.0
RelationshipSatisfaction

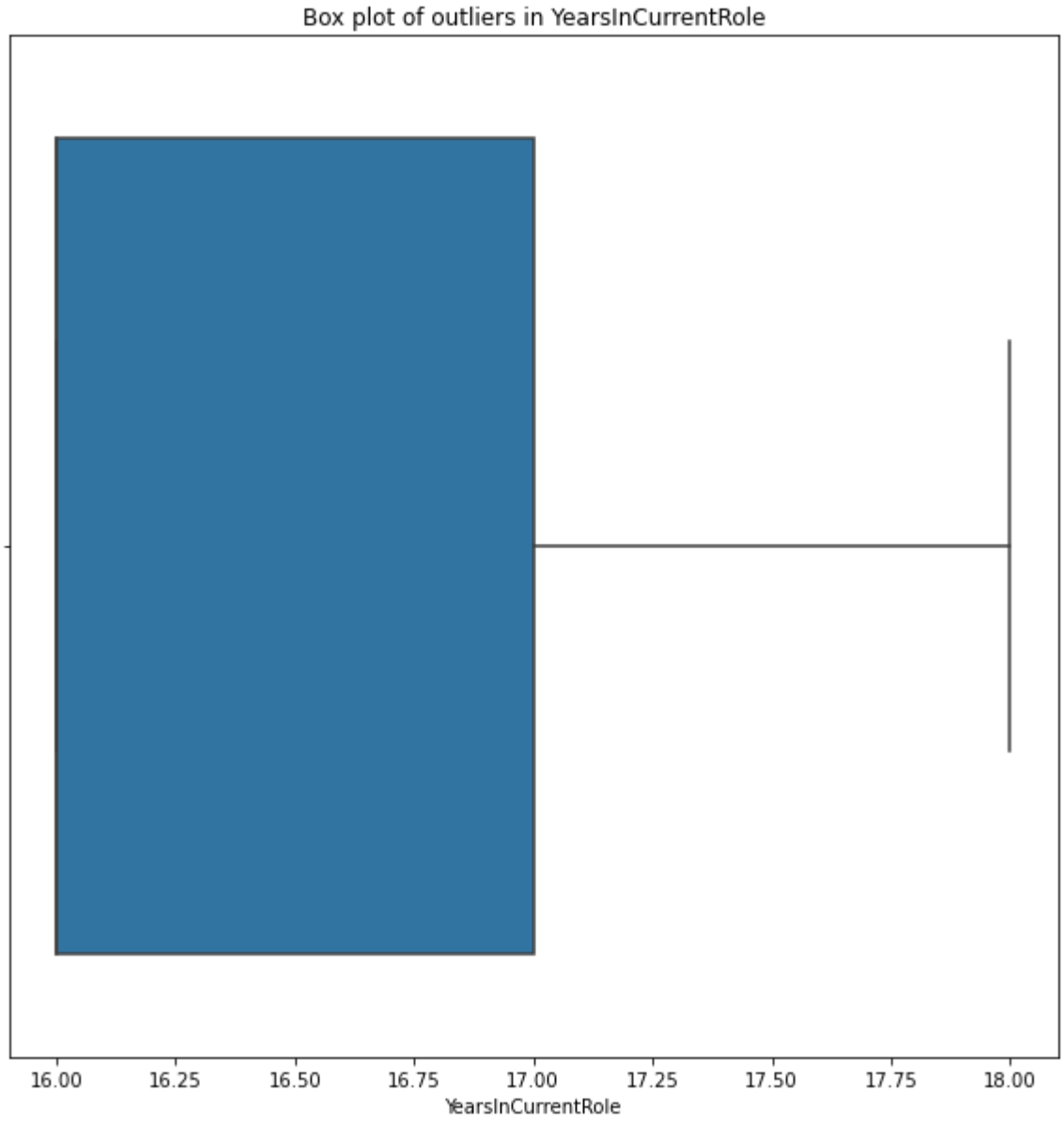

Box plot of outliers in StockOptionLevel
0.0
0.2
0.4
0.6
0.8
1.0
StockOptionLevel

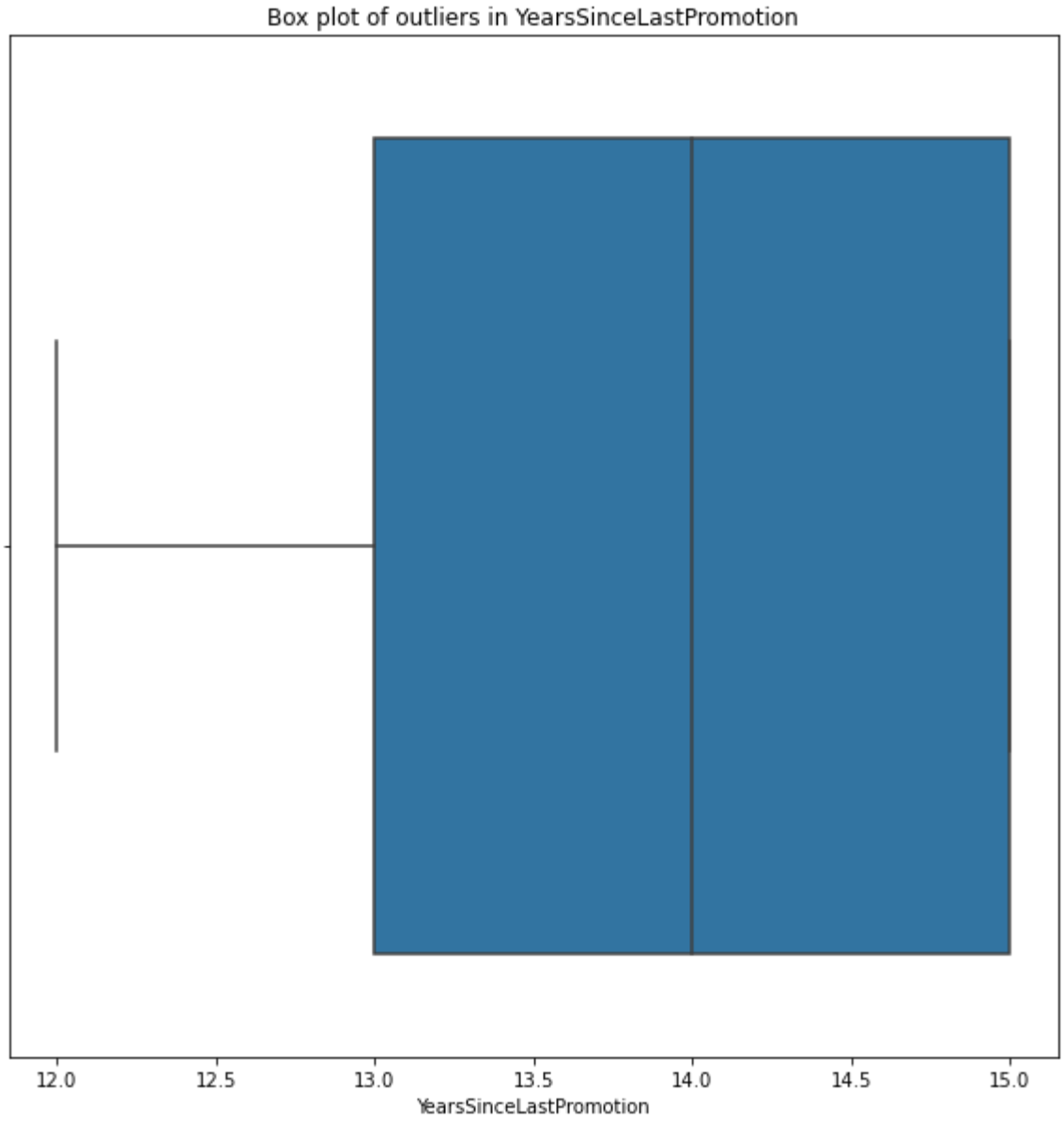

Box plot of outliers in TotalWorkingYears
35
36
37
38
39
40
TotalWorkingYears

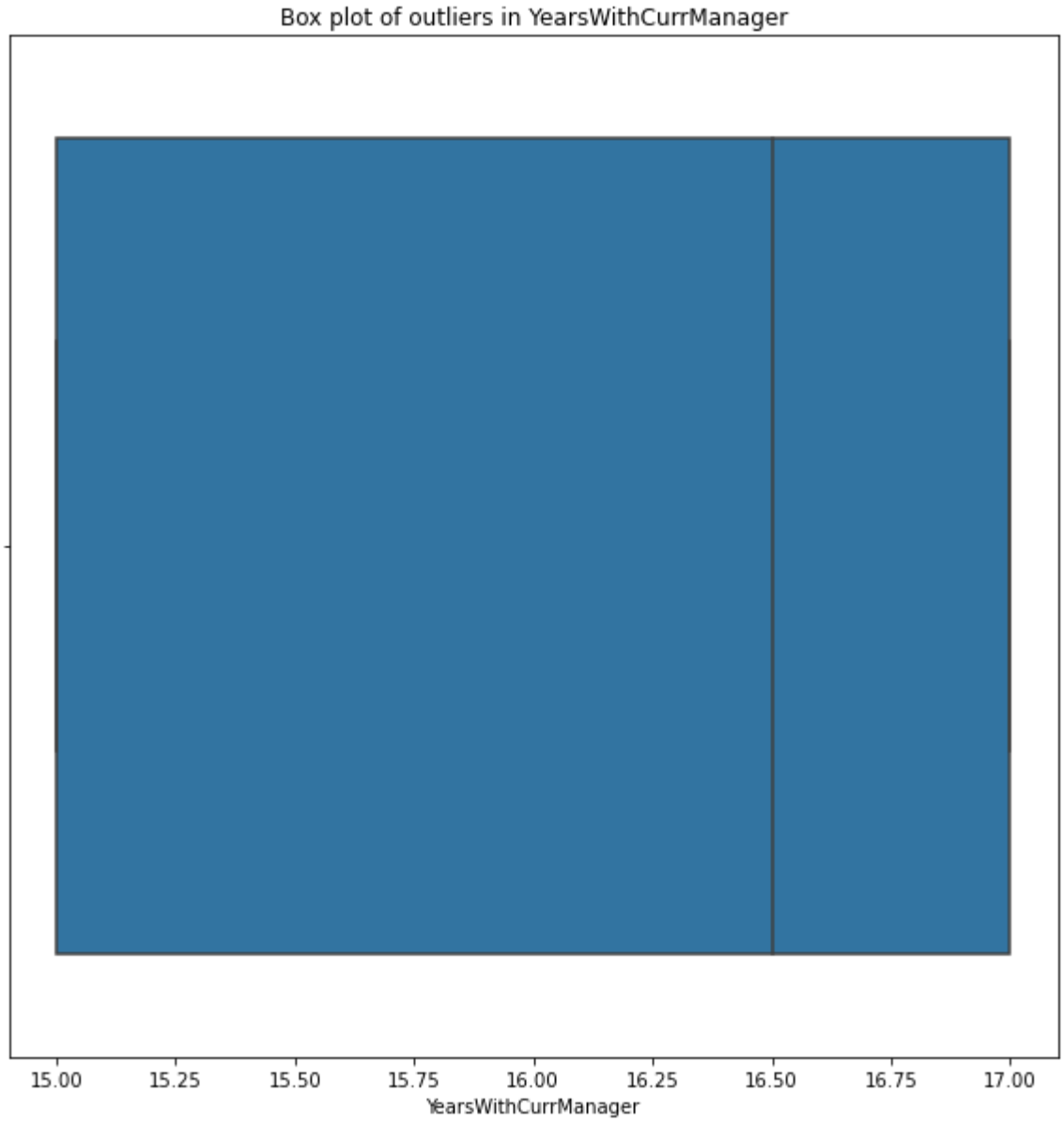

Box plot of outliers in TrainingTimesLastYear
0.0
0.2
0.4
0.6
0.8
1.0
TrainingTimesLastYear


Box plot of outliers in WorkLifeBalance
0.0
0.2
0.4
0.6
0.8
1.0
WorkLifeBalance


Box plot of outliers in YearsAtCompany
26
28
30
32
34
36
38
40
YearsAtCompany


Box plot of outliers in YearsInCurrentRole
16.00
16.25
16.50
16.75
17.00
17.25
17.50
17.75
18.00
YearsInCurrentRole


Box plot of outliers in YearsSinceLastPromotion
12.0
12.5
13.0
13.5
14.0
14.5
15.0
YearsSinceLastPromotion



6. Observation

Outliers are data points that significantly deviate from the rest of the data. In this case, the visualization suggests that there are some outliers in the following columns: TotalWorkingYears, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, and YearsWithCurrManager. The numbers in parentheses refer to the values of the outliers, which are considered to be extreme in comparison to the rest of the data.

TotalWorkingYears: The outliers in this column have values of 16 and 32, indicating that there are some employees who have either worked for a relatively short period of time or for a very long time.

YearsAtCompany: The outliers in this column have values of 25 and 32, suggesting that there are some employees who have either recently joined the company or have been with the company for a very long time.

YearsInCurrentRole: The outliers in this column have values of 13 and 32, indicating that there are some employees who have either recently joined their current role or have been in their current role for a very long time.

YearsSinceLastPromotion: The outliers in this column have values of 42 and 32, suggesting that there are some employees who have either recently received a promotion or have not received a promotion for a very long time.

YearsWithCurrManager: The outliers in this column have values of 14 and 32, indicating that there are some employees who have either recently started working with their current manager or have been working with their current manager for a very long time.

It's important to note that outliers can sometimes be a result of data entry errors, measurement errors, or genuine extreme values in the data. Therefore, it's important to carefully examine and understand the reasons behind the outliers before taking any actions based on them.

7. Examining Columns with outliers

7.1 TotalWorkingYears

In [182…

```
# creating a new DataFrame with the TotalWorkingYears counts and percentages
TotalWorkingYears_data = pd.DataFrame({'Counts': df['TotalWorkingYears'].value_counts(
print(TotalWorkingYears_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

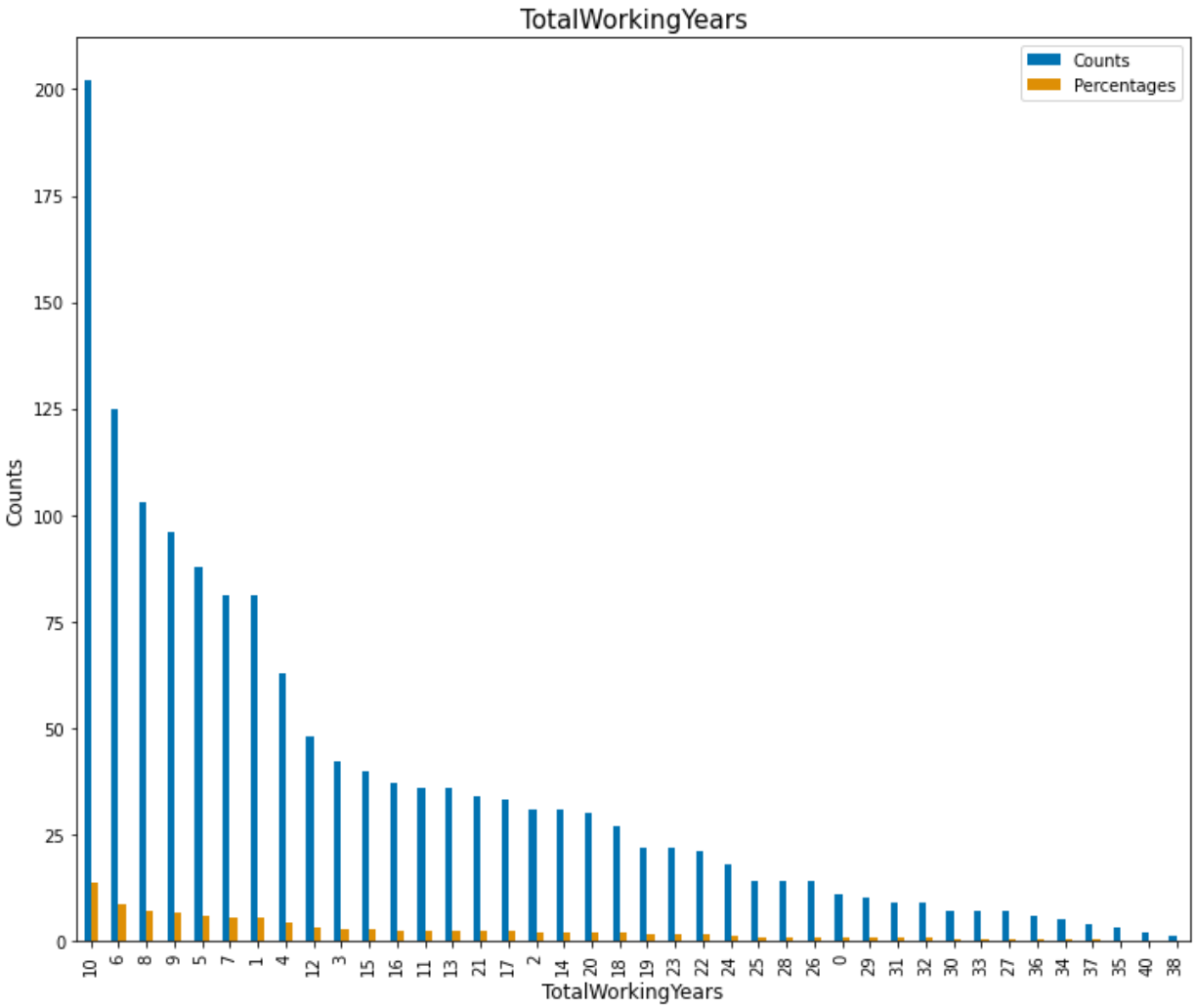
	Counts	Percentages
10	202	13.74%
6	125	8.50%
8	103	7.01%
9	96	6.53%
5	88	5.99%
7	81	5.51%
1	81	5.51%
4	63	4.29%
12	48	3.27%
3	42	2.86%
15	40	2.72%
16	37	2.52%
11	36	2.45%
13	36	2.45%
21	34	2.31%
17	33	2.24%
2	31	2.11%
14	31	2.11%
20	30	2.04%
18	27	1.84%
19	22	1.50%
23	22	1.50%
22	21	1.43%
24	18	1.22%
25	14	0.95%
28	14	0.95%
26	14	0.95%
0	11	0.75%
29	10	0.68%
31	9	0.61%
32	9	0.61%
30	7	0.48%
33	7	0.48%
27	7	0.48%
36	6	0.41%
34	5	0.34%
37	4	0.27%
35	3	0.20%
40	2	0.14%
38	1	0.07%

In [177…
```
df.TotalWorkingYears.describe()
```

Out[177]:
```
count    1470.000000
mean       11.279592
std         7.780782
min         0.000000
25%         6.000000
50%        10.000000
75%        15.000000
max        40.000000
Name: TotalWorkingYears, dtype: float64
```

In [187…
```
TotalWorkingYears_data.plot(kind='bar', figsize=(12,10),color=colors)
# setting the chart title and axis labels
plt.title('TotalWorkingYears', fontsize=15)
plt.xlabel('TotalWorkingYears', fontsize=12)
plt.ylabel('Counts',fontsize=12)
```

```
# show the chart
plt.show()
```



In [192…
```
df.TotalWorkingYears.describe()
```

Out[192]:
```
count    1470.000000
mean       11.279592
std         7.780782
min         0.000000
25%         6.000000
50%        10.000000
75%        15.000000
max        40.000000
Name: TotalWorkingYears, dtype: float64
```

7.1.1 Observation of TotalWorkingYears

The Total Working Years data is right-skewed, indicating the majority of the samples have relatively low working experience, with a small percentage of individuals who have a lot of experience pulling the mean higher. The fact that more than 50% of samples are below the median also indicates that the distribution is not symmetrical, with more data points on one side of the median than the other. The mean is 11.28 years, and the standard deviation is 7.78 years. The 25th, 50th, and 75th percentiles are 6, 10, and 15 years, respectively. The interquartile

range (IQR) is 9 years, indicating that the majority of employees have worked for less than 15 years, with a few outliers who have been with the company for 15 years or more.

7.2 YearsAtCompany

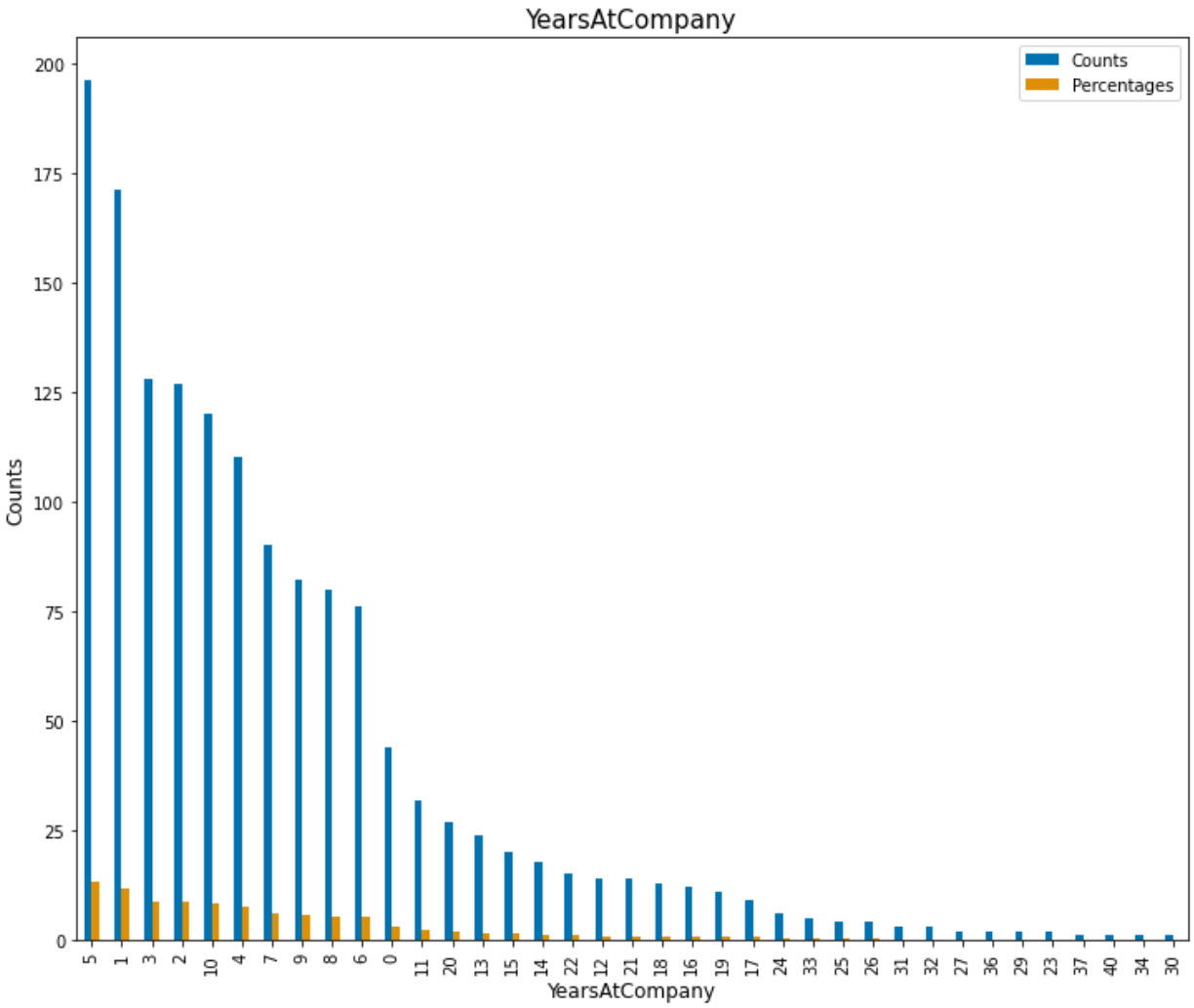
In [186…
```
# creating a new DataFrame with the YearsAtCompany counts and percentages
YearsAtCompany_data = pd.DataFrame({'Counts': df['YearsAtCompany'].value_counts(), 'Pe
print(YearsAtCompany_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

```
    Counts Percentages
5      196      13.33%
1      171      11.63%
3      128       8.71%
2      127       8.64%
10     120       8.16%
4      110       7.48%
7       90       6.12%
9       82       5.58%
8       80       5.44%
6       76       5.17%
0       44       2.99%
11      32       2.18%
20      27       1.84%
13      24       1.63%
15      20       1.36%
14      18       1.22%
22      15       1.02%
12      14       0.95%
21      14       0.95%
18      13       0.88%
16      12       0.82%
19      11       0.75%
17       9       0.61%
24       6       0.41%
33       5       0.34%
25       4       0.27%
26       4       0.27%
31       3       0.20%
32       3       0.20%
27       2       0.14%
36       2       0.14%
29       2       0.14%
23       2       0.14%
37       1       0.07%
40       1       0.07%
34       1       0.07%
30       1       0.07%
```

In [178…
```
df.YearsAtCompany.describe()
```

Out[178]:
```
count    1470.000000
mean        7.008163
std         6.126525
min         0.000000
25%         3.000000
50%         5.000000
75%         9.000000
max        40.000000
Name: YearsAtCompany, dtype: float64
```

In [188…
```
YearsAtCompany_data.plot(kind='bar', figsize=(12,10),color=colors)
# setting the chart title and axis labels
plt.title('YearsAtCompany', fontsize=15)
plt.xlabel('YearsAtCompany', fontsize=12)
plt.ylabel('Counts',fontsize=12)

# show the chart
plt.show()
```



7.2.1 Observation of TotalWorkingYears

The "YearsAtCompany" data is right-skewed because the mean (7 years) is greater than the median (5 years), and there are some employees who have served the company for a much longer time than the rest of the sample. The maximum value of 40 years is quite far from the mean, indicating that there are a small number of employees who have served the company for

a significantly longer time than the majority of the sample. This could be due to various reasons, such as employees who have reached retirement age but are still working, or employees who have been with the company for a long time and have advanced to senior positions.

In general, right-skewed distributions are common in many real-world scenarios, especially when dealing with data that has a natural lower bound (such as the minimum number of years an employee can work at a company), but no natural upper bound. It is important to recognize and account for the skewness of the distribution when analyzing the data, as it can impact the interpretation of statistical measures such as the mean and standard deviation.

7.3 YearsInCurrentRole

In [185…
```
# creating a new DataFrame with the YearsInCurrentRole counts and percentages
YearsInCurrentRole_data = pd.DataFrame({'Counts': df['YearsInCurrentRole'].value_count
print(YearsInCurrentRole_data.to_string(formatters={'Percentages': '{:.2f}%'.format}))
```

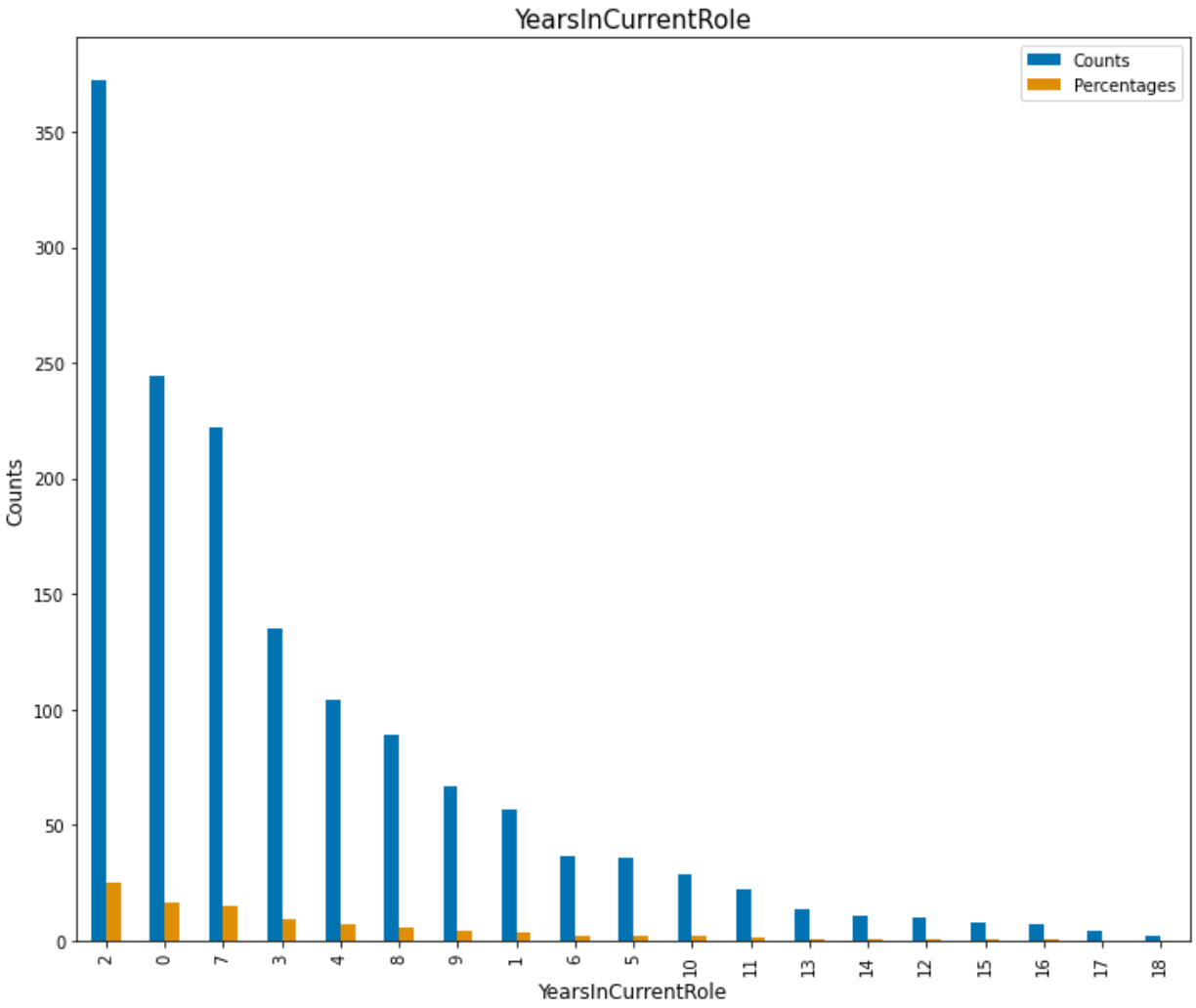
```
    Counts Percentages
2      372      25.31%
0      244      16.60%
7      222      15.10%
3      135       9.18%
4      104       7.07%
8       89       6.05%
9       67       4.56%
1       57       3.88%
6       37       2.52%
5       36       2.45%
10      29       1.97%
11      22       1.50%
13      14       0.95%
14      11       0.75%
12      10       0.68%
15       8       0.54%
16       7       0.48%
17       4       0.27%
18       2       0.14%
```

In [179…
```
df.YearsInCurrentRole.describe()
```

Out[179]:
```
count    1470.000000
mean        4.229252
std         3.623137
min         0.000000
25%         2.000000
50%         3.000000
75%         7.000000
max        18.000000
Name: YearsInCurrentRole, dtype: float64
```

In [189…
```
YearsInCurrentRole_data.plot(kind='bar', figsize=(12,10),color=colors)
# setting the chart title and axis labels
plt.title('YearsInCurrentRole', fontsize=15)
plt.xlabel('YearsInCurrentRole', fontsize=12)
plt.ylabel('Counts',fontsize=12)
```

```
# show the chart
plt.show()
```



7.3.1 Observation of YearsInCurrentRole

The median (50th percentile) is 3 years, which means that 50% of the employees have 3 or fewer years of experience in their current role. However, the mean (4.23 years) is greater than the median, indicating that the distribution is right-skewed. This means that there are some employees who have been in their current role for a much longer time, which is causing the tail of the distribution to be longer on the right side.

In this case, the maximum value is 18 years, which is a large difference from the median of 3 years. This suggests that there are a small number of employees who have been in their current role for a significantly longer time than the rest of the sample, and this is causing the distribution to be right-skewed.

7.4 YearsSinceLastPromotion

In [184…

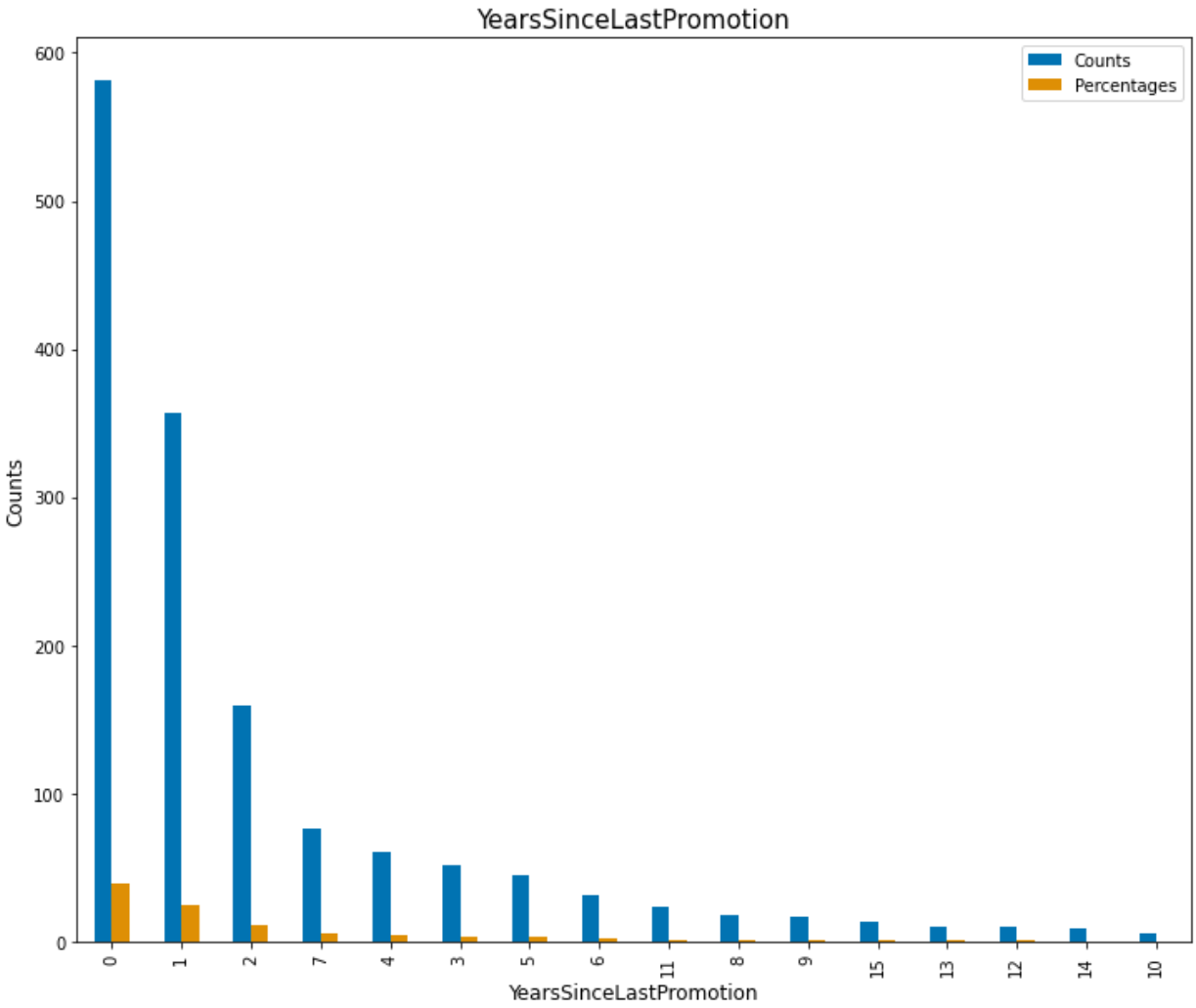
```
# creating a new DataFrame with the YearsSinceLastPromotion counts and percentages
YearsSinceLastPromotion_data = pd.DataFrame({'Counts': df['YearsSinceLastPromotion'].v
print(YearsSinceLastPromotion_data.to_string(formatters={'Percentages': '{:.2f}%'.form
```

	Counts	Percentages
0	581	39.52%
1	357	24.29%
2	159	10.82%
7	76	5.17%
4	61	4.15%
3	52	3.54%
5	45	3.06%
6	32	2.18%
11	24	1.63%
8	18	1.22%
9	17	1.16%
15	13	0.88%
13	10	0.68%
12	10	0.68%
14	9	0.61%
10	6	0.41%

In [180…
```
df.YearsSinceLastPromotion.describe()
```

Out[180]:
```
count    1470.000000
mean        2.187755
std         3.222430
min         0.000000
25%         0.000000
50%         1.000000
75%         3.000000
max        15.000000
Name: YearsSinceLastPromotion, dtype: float64
```

In [190…
```
YearsSinceLastPromotion_data.plot(kind='bar', figsize=(12,10),color=colors)
# setting the chart title and axis labels
plt.title('YearsSinceLastPromotion', fontsize=15)
plt.xlabel('YearsSinceLastPromotion', fontsize=12)
plt.ylabel('Counts',fontsize=12)

# show the chart
plt.show()
```



7.4.1 Observation of YearsSinceLastPromotion

"YearsSinceLastPromotion" provides information on how long it has been since the employee was last promoted. The most common value in the column is 0, which means that 581 employees (39.52% of the total) have never been promoted. This is followed by 1 year since the last promotion, with 357 employees (24.29% of the total). The third most common value is 2 years since the last promotion, with 159 employees (10.82% of the total).

From this data, we can infer that a large percentage of employees have not been promoted in recent years. The fact that 39.52% of employees have never been promoted suggests that there may be limited opportunities for career advancement in the company. Additionally, the fact that the majority of employees have not been promoted in the last two years may indicate a lack of emphasis on employee development and retention by the company. It's important to note that while the data does suggest a lack of recent promotions, it may not necessarily indicate a lack of opportunities or a lack of emphasis on employee development and retention by the company. There could be other reasons why employees have not been promoted, such as a lack of qualified candidates for open positions or a company culture that values stability and longevity over rapid career advancement.

7.5 YearsWithCurrManager

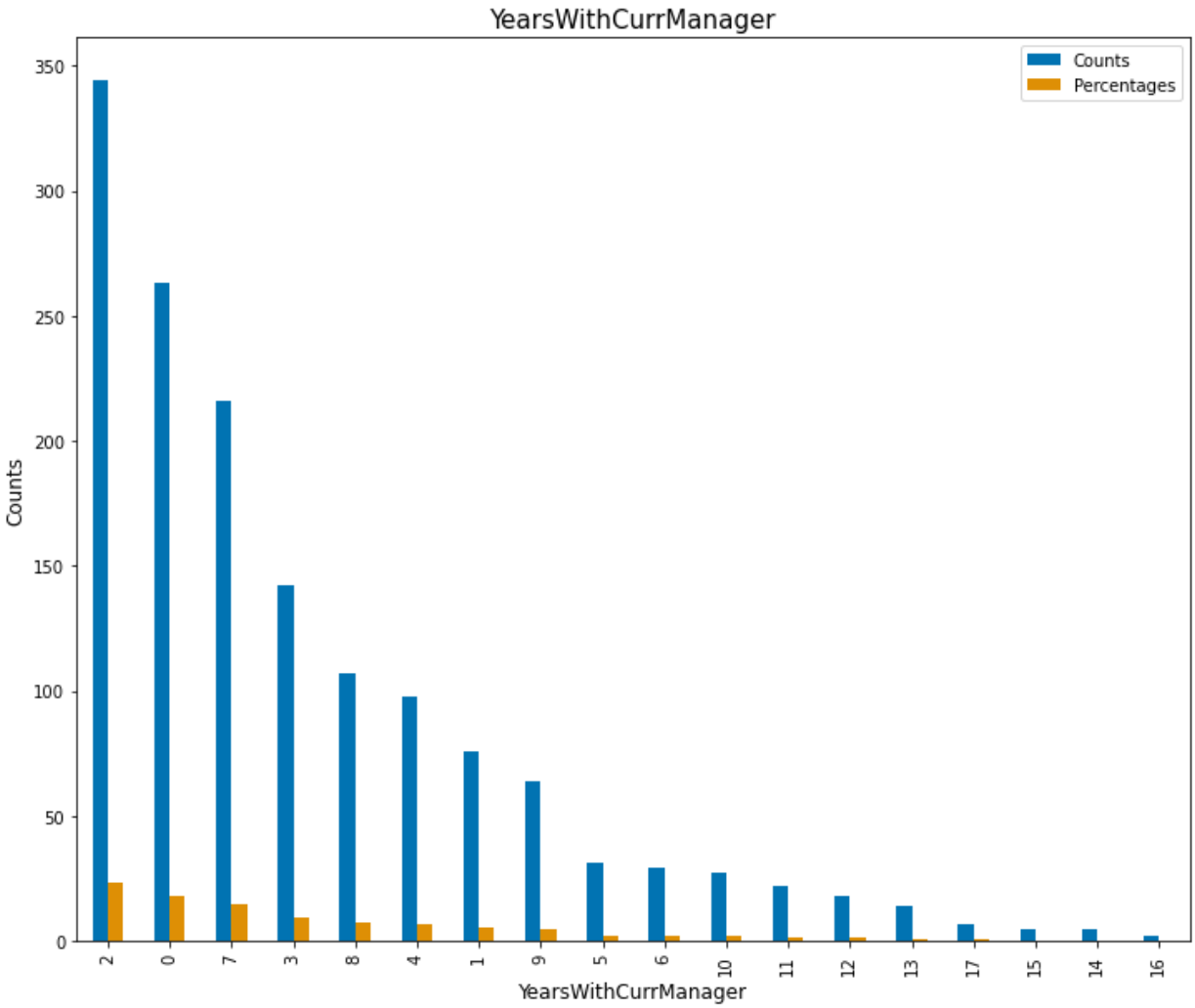
In [183…
```
# creating a new DataFrame with the YearsWithCurrManager counts and percentages
YearsWithCurrManager_data = pd.DataFrame({'Counts': df['YearsWithCurrManager'].value_c
print(YearsWithCurrManager_data.to_string(formatters={'Percentages': '{:.2f}%'.format}
```

```
    Counts Percentages
2      344      23.40%
0      263      17.89%
7      216      14.69%
3      142       9.66%
8      107       7.28%
4       98       6.67%
1       76       5.17%
9       64       4.35%
5       31       2.11%
6       29       1.97%
10      27       1.84%
11      22       1.50%
12      18       1.22%
13      14       0.95%
17       7       0.48%
15       5       0.34%
14       5       0.34%
16       2       0.14%
```

In [181…
```
df.YearsWithCurrManager.describe()
```

Out[181]:
```
count    1470.000000
mean        4.123129
std         3.568136
min         0.000000
25%         2.000000
50%         3.000000
75%         7.000000
max        17.000000
Name: YearsWithCurrManager, dtype: float64
```

In [191…
```
YearsWithCurrManager_data.plot(kind='bar', figsize=(12,10),color=colors)
# setting the chart title and axis labels
plt.title('YearsWithCurrManager', fontsize=15)
plt.xlabel('YearsWithCurrManager', fontsize=12)
plt.ylabel('Counts',fontsize=12)

# show the chart
plt.show()
```



7.5.1 Observation of YearsWithCurrManager

The column "YearsWithCurrManager" refers to the number of years employees have been working with their current manager. The mean number of years is 4.12, with a standard deviation of 3.57. The minimum number of years is 0, which indicates that some employees have had multiple managers within the same year. The 25th percentile of employees have worked with their current manager for 2 years or less, while the 50th percentile (median) is 3 years, and the 75th percentile is 7 years.

It appears that almost 500 people, which is more than 1/3 of the total sample, have had a shift to different managers after working for 2.5 years. This suggests that some employees may be seeking new management opportunities or may be assigned to different managers for organizational reasons.

Additionally, almost 300 people have shown a shift to different managers within the first year of working with their current manager. This could be due to various reasons, such as dissatisfaction with the manager or a change in job responsibilities.

Overall, the data shows that there is a considerable amount of variation in the number of years employees have worked with their current manager, with a significant number of employees changing managers relatively quickly.This information could be useful for understanding

employee turnover rates and job satisfaction. It suggests that many employees may not be satisfied with their current manager or may be seeking new opportunities for career growth.

8. Salary related analysis

8.1 Statistical Analysis of HourlyRate, MonthlyIncome, MonthlyRate and PercentSalaryHike

In [139…
```
salary_cols=['HourlyRate','MonthlyIncome','MonthlyRate','PercentSalaryHike']
```

In [140…
```
df[salary_cols].describe()
```

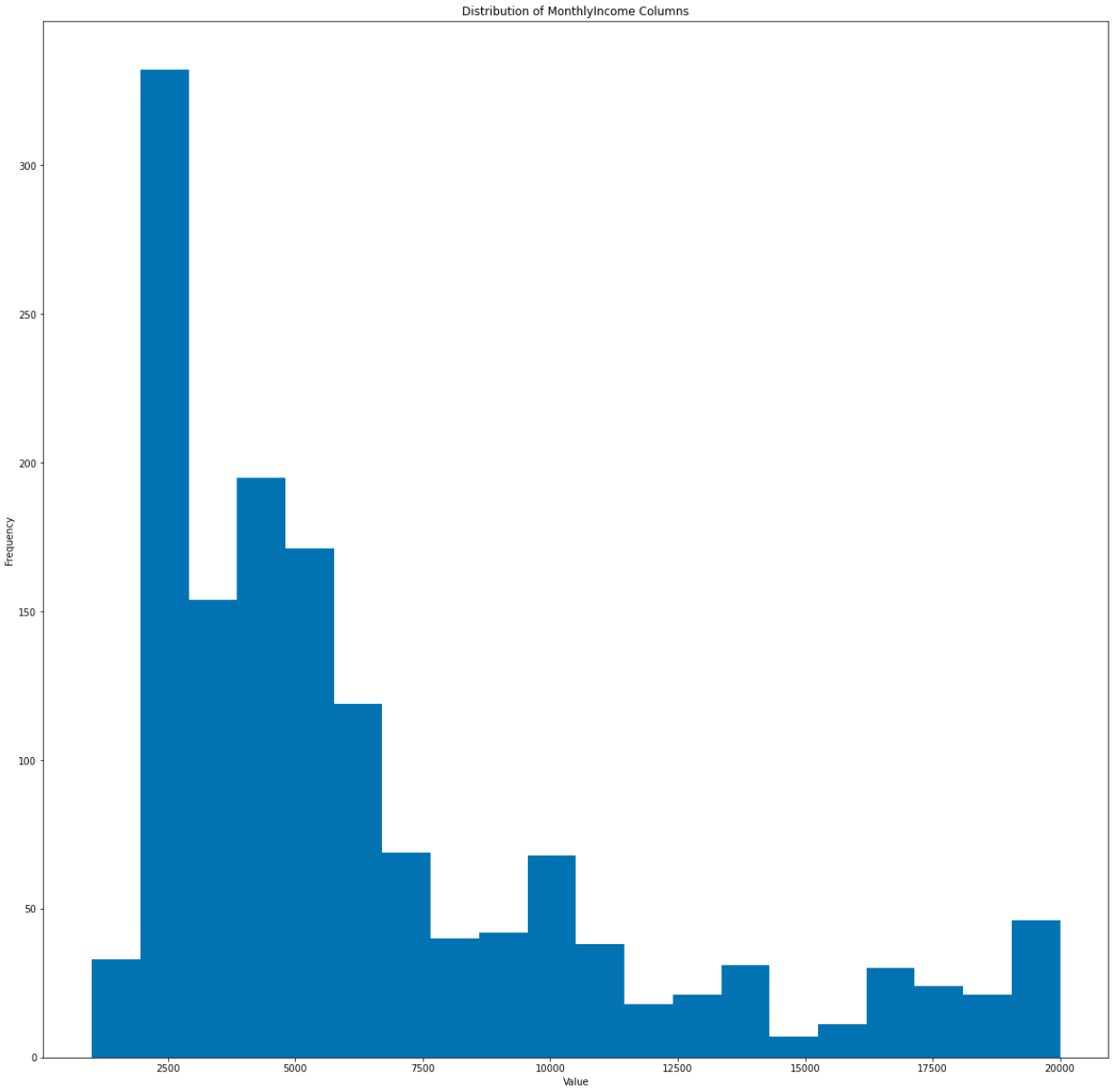
Out[140]:

	HourlyRate	MonthlyIncome	MonthlyRate	PercentSalaryHike
count	1470.000000	1470.000000	1470.000000	1470.000000
mean	65.891156	6502.931293	14313.103401	15.209524
std	20.329428	4707.956783	7117.786044	3.659938
min	30.000000	1009.000000	2094.000000	11.000000
25%	48.000000	2911.000000	8047.000000	12.000000
50%	66.000000	4919.000000	14235.500000	14.000000
75%	83.750000	8379.000000	20461.500000	18.000000
max	100.000000	19999.000000	26999.000000	25.000000

Visualizing the statistical distribution

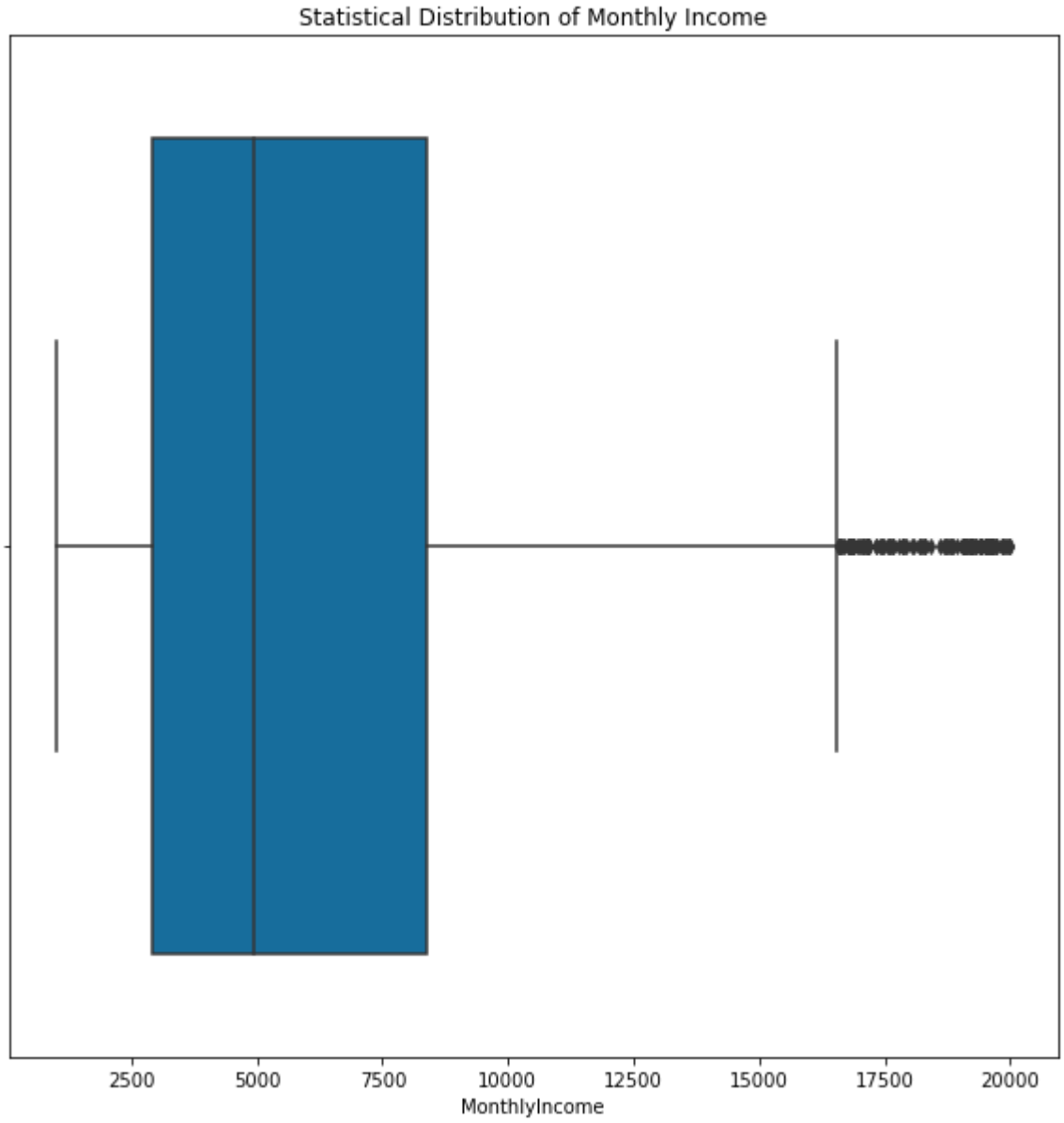
8.2 MonthlyIncome

In [165…
```
# Set the figure size and colorblind palette
sns.set_palette("colorblind")
plt.figure(figsize=(20, 20))
plt.title('Distribution of MonthlyIncome Columns')
# Add labels and titles
plt.xlabel('Value')
plt.ylabel('Frequency')
df['MonthlyIncome'].hist(bins=20,grid=False)

# Show the plot
plt.show()
```



In [166…

```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(x=df["MonthlyIncome"],ax=ax)
plt.title("Statistical Distribution of Monthly Income")
plt.show()
```



Observation of MonthlyIncome

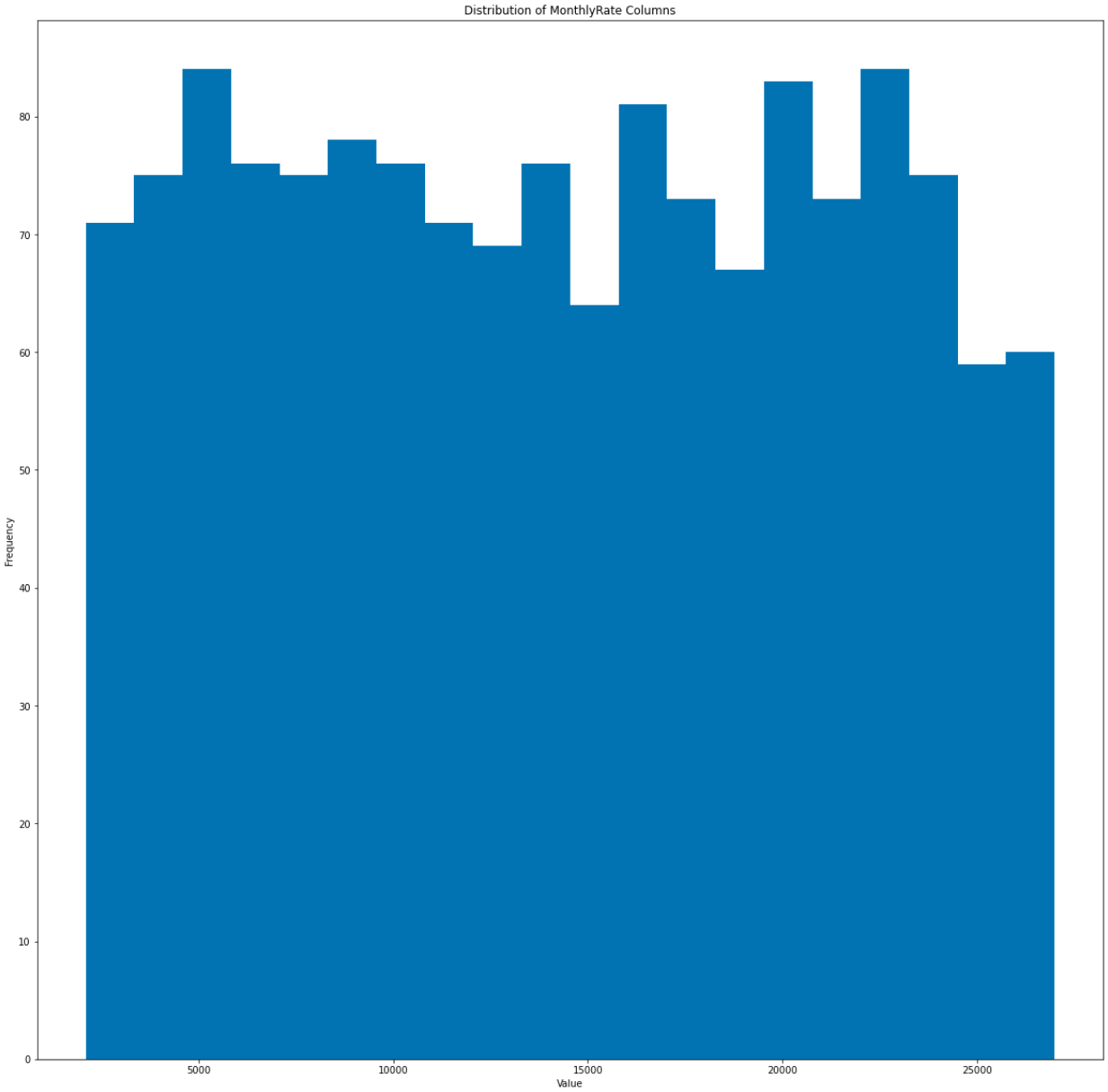
The fact that the Monthly Income data is right-skewed means that there are fewer employees with higher incomes, and the majority of employees fall within the lower income ranges. This is indicated by the fact that the mean (6502.93) is higher than the median (4919), which suggests that there are a few high-income employees who are pulling the average up, but the majority of employees have lower incomes.

The gap of 1500 between the mean and median is significant and indicates that the distribution is not symmetrical. This is further supported by the fact that the distribution has a long right tail. The majority of the sample falls below the upper quartile, which means that the top 25% of employees have higher incomes than the rest of the sample. This could be due to factors such as seniority, education, or job position that are associated with higher salaries.

8.3 MonthlyRate

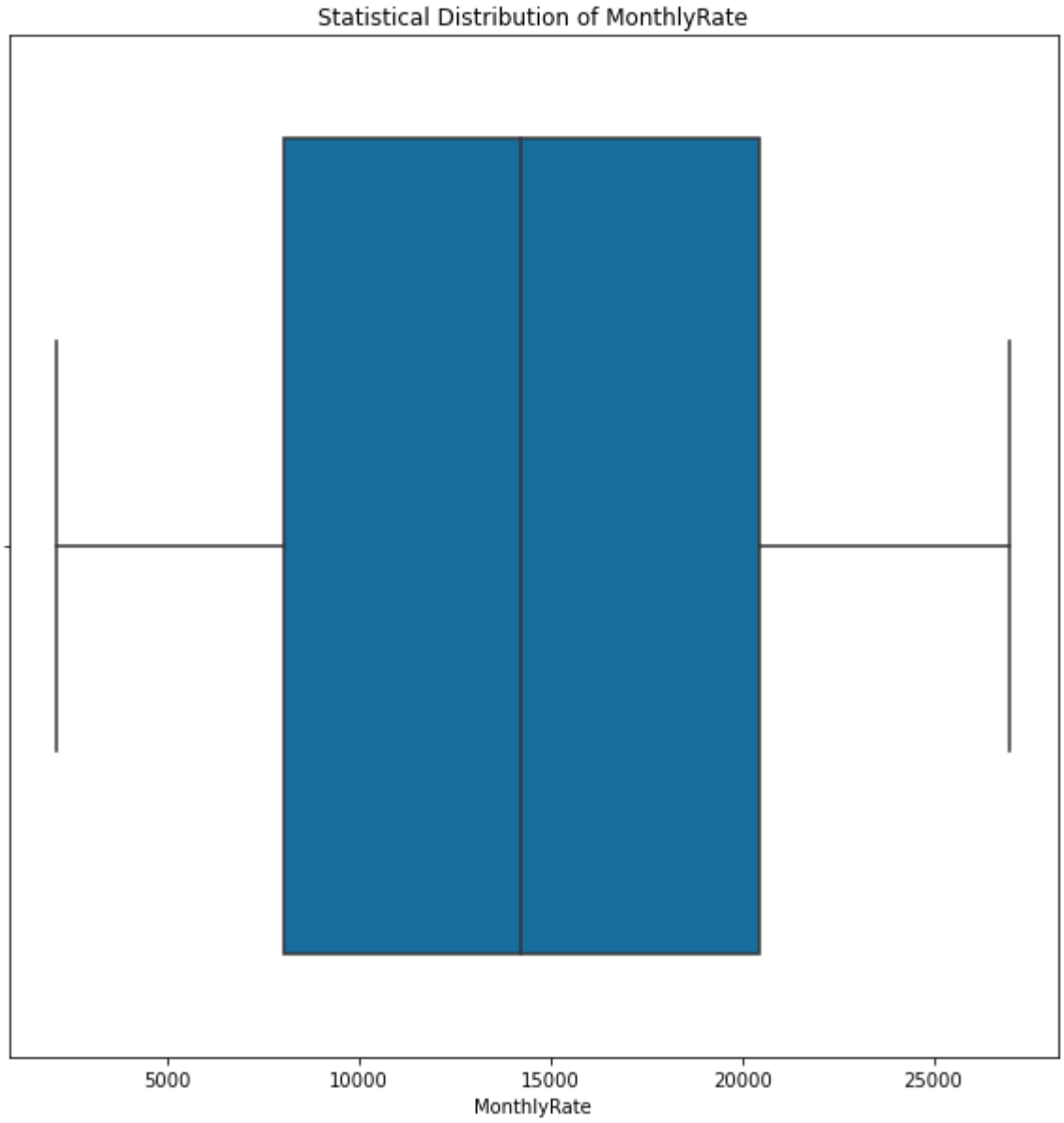
In [167…

```
# Set the figure size and colorblind palette
sns.set_palette("colorblind")
plt.figure(figsize=(20, 20))
plt.title('Distribution of MonthlyRate Columns')
# Add labels and titles
plt.xlabel('Value')
plt.ylabel('Frequency')
df['MonthlyRate'].hist(bins=20,grid=False)

# Show the plot
plt.show()
```



In [168…

```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(x=df["MonthlyRate"],ax=ax)
plt.title("Statistical Distribution of MonthlyRate")
plt.show()
```



Observation of MonthlyRate

The statistical distribution of the MonthlyRate variable indicates that the average monthly rate is 14313.10 with a standard deviation of 7117.79. The minimum monthly rate is 2094, and the maximum is 26999. The 25th percentile, which represents the lower quartile, is 8047, while the 50th percentile, also known as the median, is 14235.50. The upper quartile, which is the 75th percentile, is 20461.50.

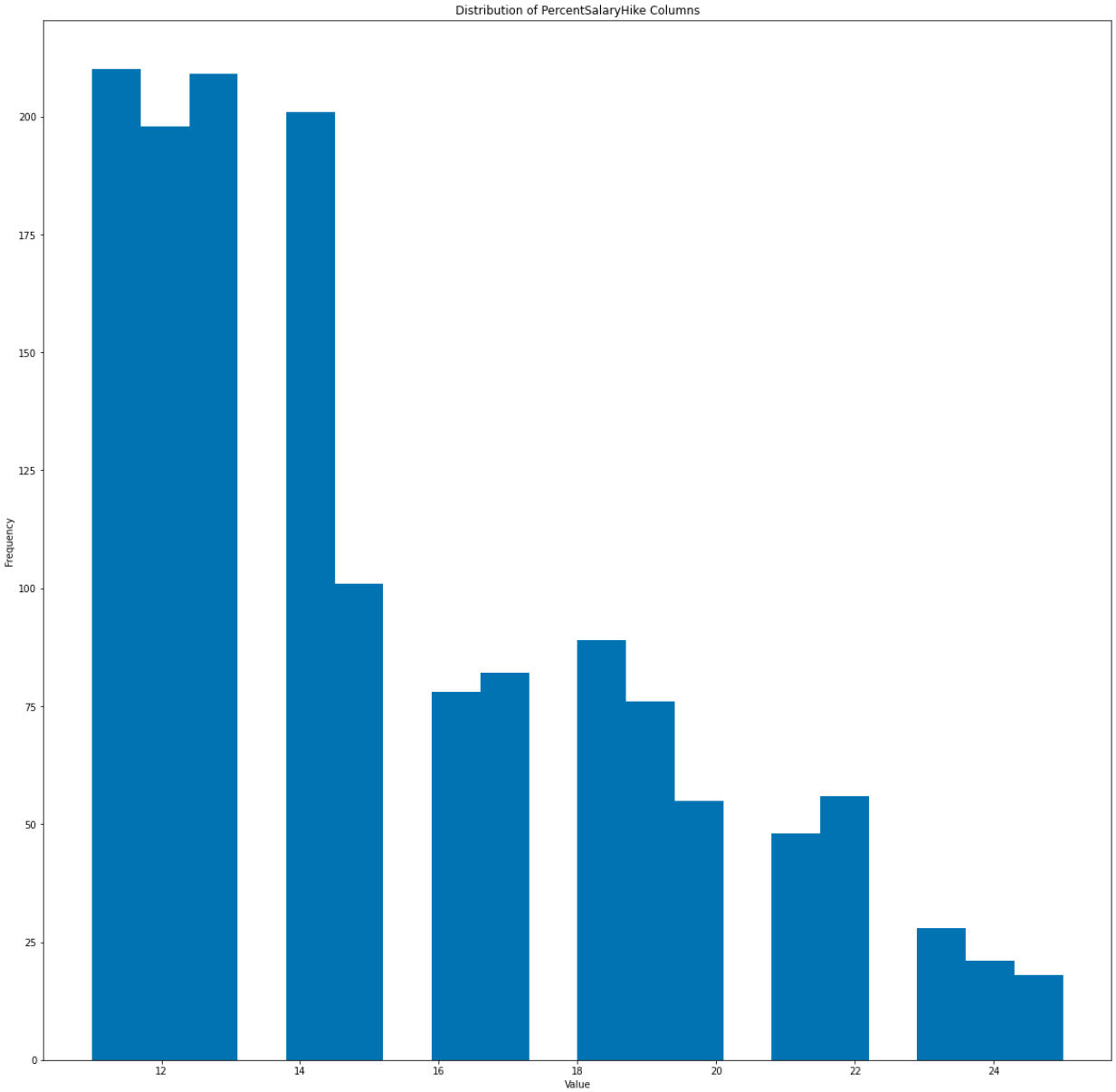
Based on the given statistical values, the MonthlyRate variable appears to have a normal distribution, which means that the data points are evenly distributed around the mean value. The distribution is symmetric and bell-shaped, with most data points concentrated around the mean value. However, there could be some outliers on the higher side of the distribution, as evidenced by the maximum value of 26999.

Overall, the distribution suggests that the monthly rates are spread out, with a few employees having rates much higher than the average, while most employees have rates around the median value.

8.4 PercentSalaryHike

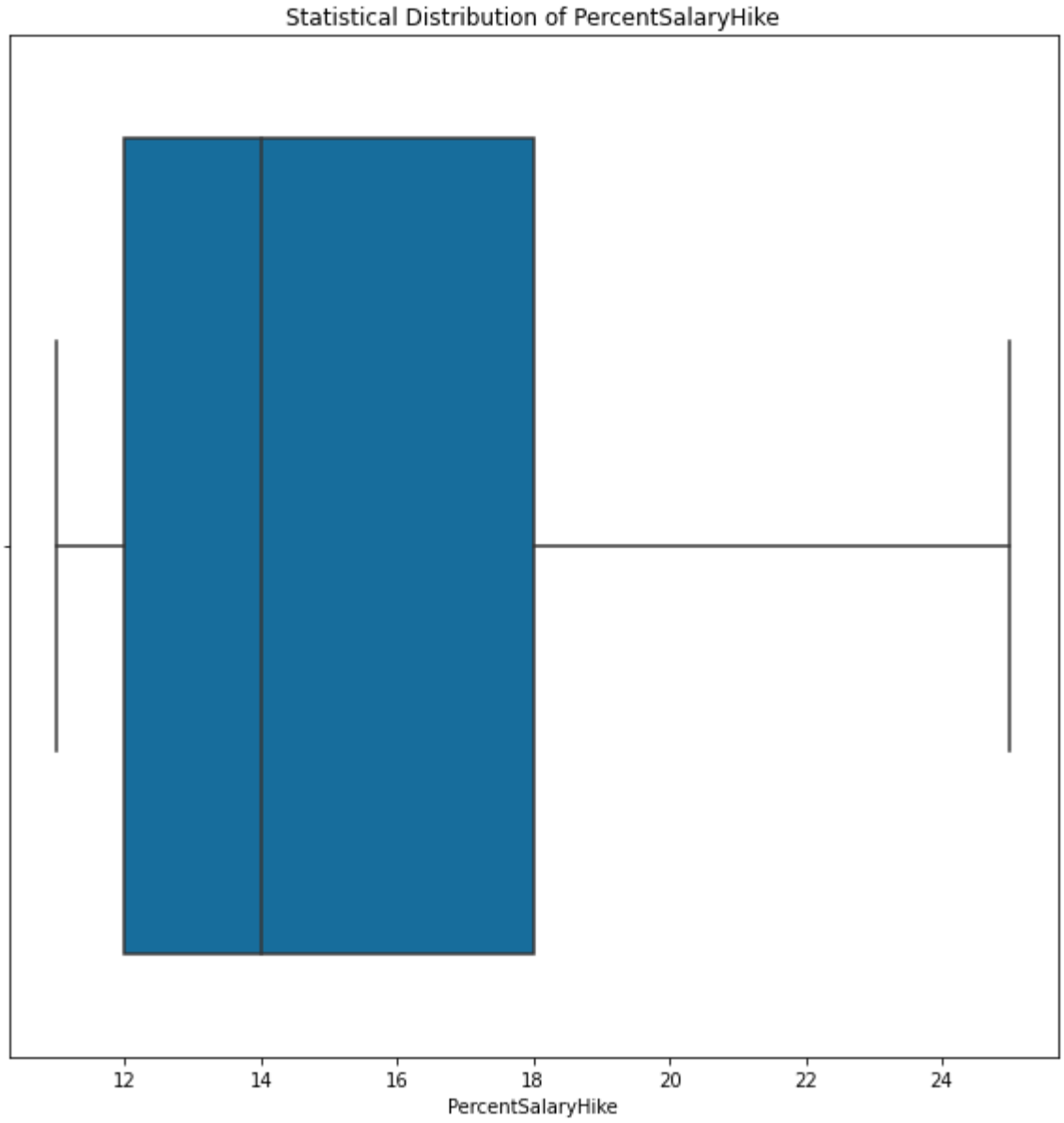
In [170…

```
# Set the figure size and colorblind palette
sns.set_palette("colorblind")
plt.figure(figsize=(20, 20))
plt.title('Distribution of PercentSalaryHike Columns')
# Add labels and titles
plt.xlabel('Value')
plt.ylabel('Frequency')
df['PercentSalaryHike'].hist(bins=20,grid=False)

# Show the plot
plt.show()
```



In [171…

```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(x=df["PercentSalaryHike"],ax=ax)
plt.title("Statistical Distribution of PercentSalaryHike")
plt.show()
```



Observation of PercentSalaryHike

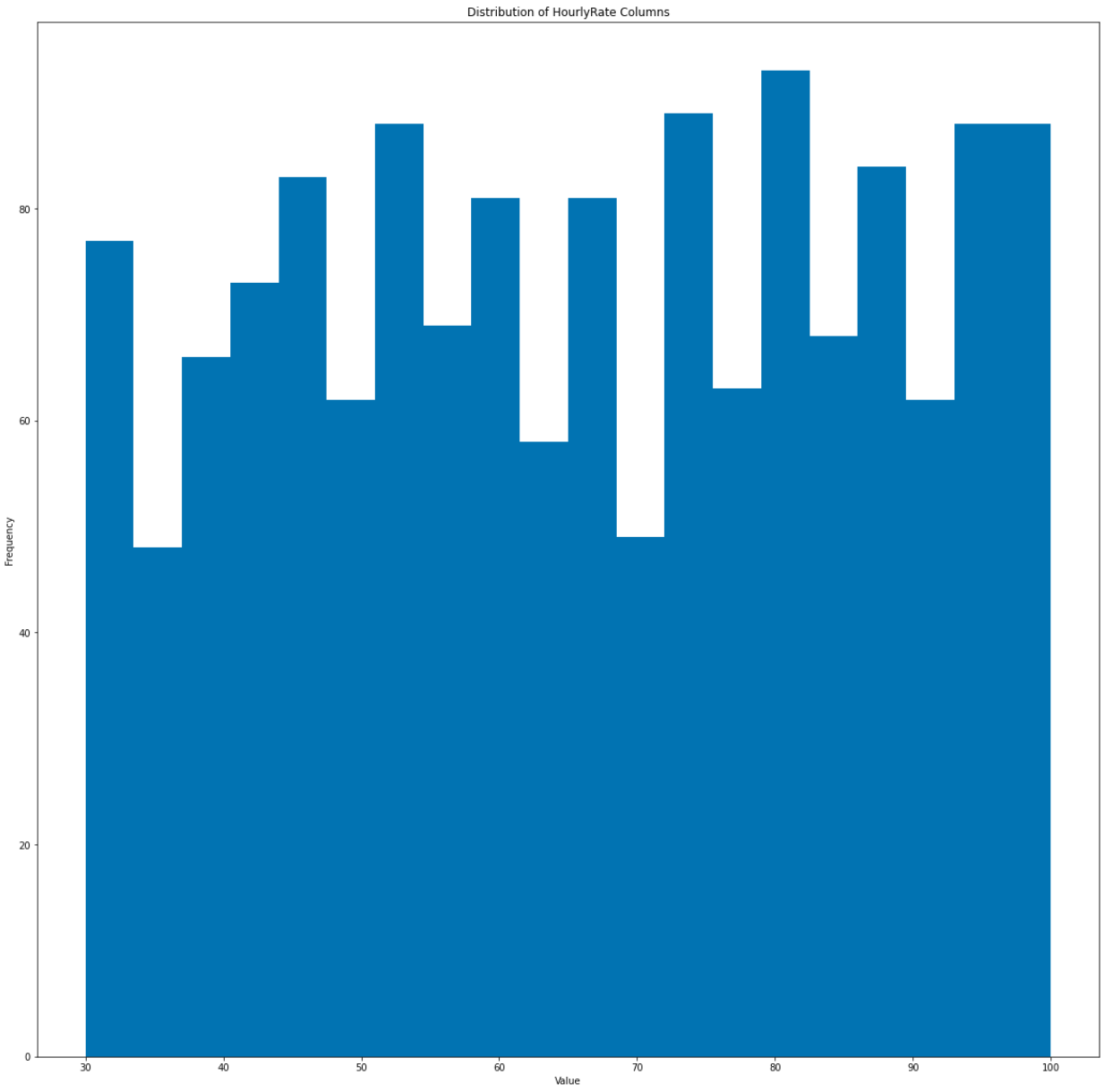
The statistical distribution of the PercentSalaryHike variable indicates that the average percent salary hike is 15.21 with a standard deviation of 3.66. The minimum percent salary hike is 11, and the maximum is 25. The 25th percentile, which represents the lower quartile, is 12, while the 50th percentile, also known as the median, is 14. The upper quartile, which is the 75th percentile, is 18.

Based on the given statistical values, the PercentSalaryHike variable appears to have a slightly right-skewed distribution, which means that there are more data points on the lower side of the distribution. The distribution is also quite narrow, with most data points clustered around the mean value. However, there are some employees who have received a higher than average salary hike, as evidenced by the maximum value of 25.

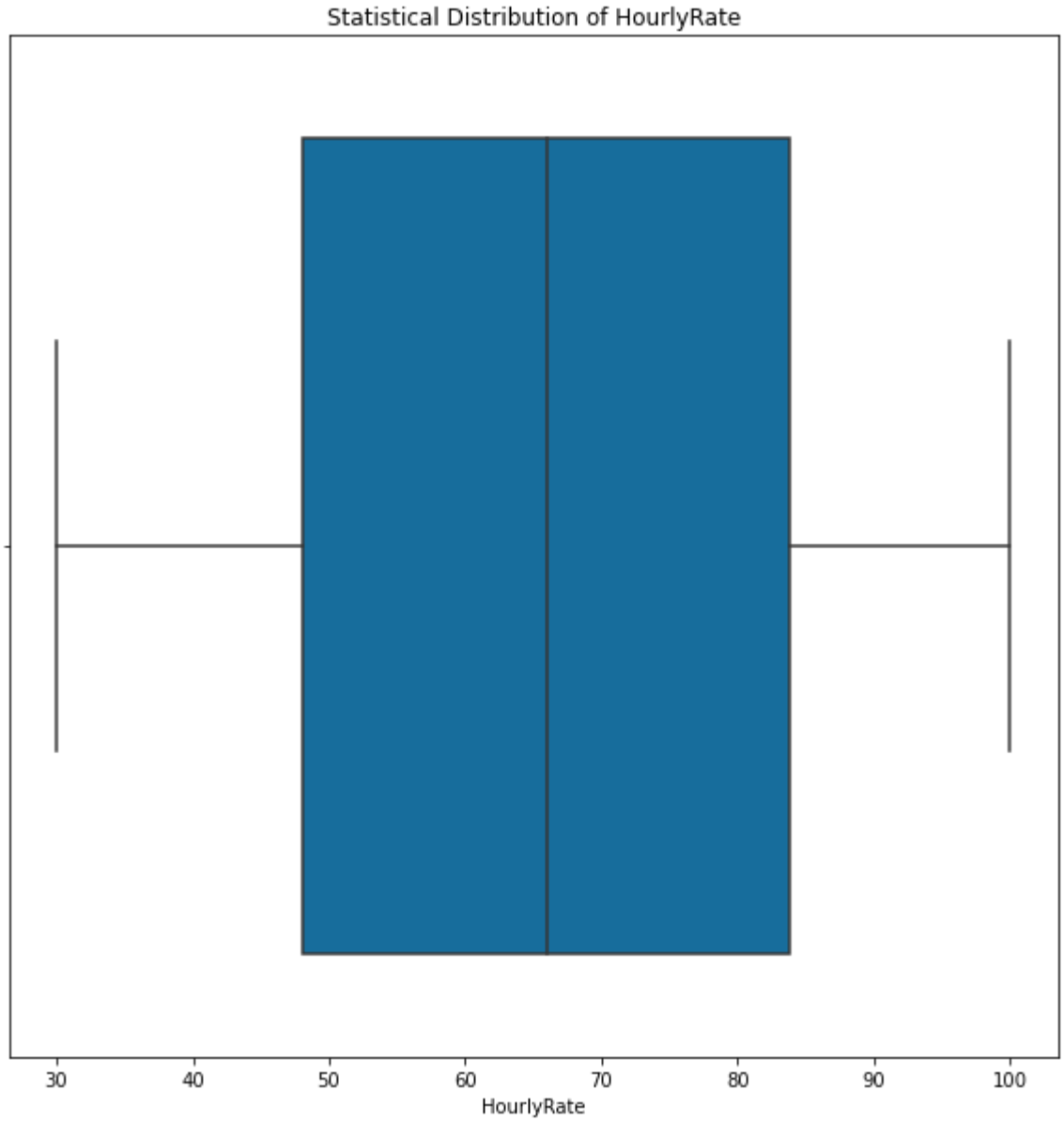
Overall, the distribution suggests that most employees received a salary hike within the range of 12 to 18 percent. However, some employees received a lower than average hike, while a few employees received a much higher than average hike.

8.5 HourlyRate

In [174…
```
# Set the figure size and colorblind palette
sns.set_palette("colorblind")
plt.figure(figsize=(20, 20))
plt.title('Distribution of HourlyRate Columns')
# Add labels and titles
plt.xlabel('Value')
plt.ylabel('Frequency')
df['HourlyRate'].hist(bins=20,grid=False)

# Show the plot
plt.show()
```



In [175…
```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(x=df["HourlyRate"],ax=ax)
plt.title("Statistical Distribution of HourlyRate")
plt.show()
```



Observation of HourlyRate

The average HourlyRate is 65.89, indicating the central tendency of the data.The standard deviation of the HourlyRate is 20.33, indicating the amount of variation in the data. The minimum HourlyRate is 30, and the maximum HourlyRate is 100, showing the range of HourlyRate values. The 25th percentile is 48, the median is 66, and the 75th percentile is 83.75, indicating the distribution of the data and how many HourlyRate values fall within certain ranges.

9. Conclusion

HR analytics involves using data analysis and statistical methods to gain insights and make data-driven decisions related to workforce management. Univariate analysis, which involves analyzing a single variable, is a statistical technique used in HR analytics to identify patterns and trends in HR data. Univariate analysis provides a foundation for more complex analyses and

helps organizations make data-driven decisions related to talent management, employee retention, recruitment, and other HR-related activities.