EMPLOYEE ATTRITION

Importing necessary libraries

In [ ]:

```
# import libraries for data manipulation
import numpy as np
import pandas as pd

# import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
```

Loading the dataset

In [ ]:

```
# read the data
df = pd.read_csv('/content/Attrition data.csv')
```

Displaying the first few rows of the dataset

In [ ]:

```
# Returns the first 5 rows
df.head()
```

Out[ ]:

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

5 rows × 29 columns

Dispalying the last few rows of the dataset

In [ ]:

```
# Returns the last 5 rows
df.tail()
```

Out[ ]:

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCou
4405	4406	42	No	Travel_Rarely	Research &	5	4	Medical	

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCou
4406	4407	29	No	Travel_Rarely	Research & Development	2	4	Medical	
4407	4408	25	No	Travel_Rarely	Research & Development	25	2	Life Sciences	
4408	4409	42	No	Travel_Rarely	Sales	18	2	Medical	
4409	4410	40	No	Travel_Rarely	Research & Development	28	3	Medical	

5 rows × 29 columns

Checking the shape of the dataset

In [ ]:

```
# The shape of the attribute gives the number of rows and columns in the dataset
print("Number of rows in the dataset:",df.shape[0])
print("Number of columns in the dataset:",df.shape[1])
```

```
Number of rows in the dataset: 4410
Number of columns in the dataset: 29
```

Observations:

The Employee Attrition analysis dataset has 4410 rows and 29 columns.

Checking the datatypes of columns in the dataset

In [ ]:

```
# info() function helps in identifying the datatypes of the columns in the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4410 entries, 0 to 4409
Data columns (total 29 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   EmployeeID               4410 non-null   int64  
 1   Age                      4410 non-null   int64  
 2   Attrition                4410 non-null   object 
 3   BusinessTravel           4410 non-null   object 
 4   Department               4410 non-null   object 
 5   DistanceFromHome         4410 non-null   int64  
 6   Education                4410 non-null   int64  
 7   EducationField           4410 non-null   object 
 8   EmployeeCount            4410 non-null   int64  
 9   Gender                   4410 non-null   object 
 10  JobLevel                 4410 non-null   int64  
 11  JobRole                  4410 non-null   object 
 12  MaritalStatus            4410 non-null   object 
 13  MonthlyIncome            4410 non-null   int64  
 14  NumCompaniesWorked       4391 non-null   float64
 15  Over18                   4410 non-null   object 
 16  PercentSalaryHike        4410 non-null   int64  
 17  StandardHours            4410 non-null   int64  
 18  StockOptionLevel         4410 non-null   int64  
 19  TotalWorkingYears        4401 non-null   float64
 20  TrainingTimesLastYear    4410 non-null   int64  
 21  YearsAtCompany           4410 non-null   int64  
 22  YearsSinceLastPromotion  4410 non-null   int64  
 23  YearsWithCurrManager     4410 non-null   int64  
 24  EnvironmentSatisfaction  4385 non-null   float64
 25  JobSatisfaction          4390 non-null   float64
 26  WorkLifeBalance          4372 non-null   float64
```

```
 27  JobInvolvement           4410 non-null   int64
 28  PerformanceRating        4410 non-null   int64
dtypes: float64(5), int64(16), object(8)
memory usage: 999.3+ KB
```

Observations:

- ***We could understand that there are missing values in some columns like NumCompaniesWorked, TotalWorkingYears, EnvironmentSatisfaction, JobSatisfaction, WorkLifeBalance.***
- ***The columns EmployeeID, Age, DistanceFromHome, Education, EmployeeCount, JobLevel, MonthlyIncome, PercentSalaryHike, StandardHours, StockOptionLevel, TrainingTimesLastYear, YearsAtCompany, YearsSinceLastPromotion, YearsWithCurrManager, JobInvolvement, PerformanceRating is of integer datatype whereas NumCompaniesWorked, TotalWorkingYears, EnvironmentSatisfaction, JobSatisfaction, WorkLifeBalance is of float datatype.***
- ***The Attrition, BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, Over18, columns are of object type.***

Checking the statistical summary of the numerical variables in the dataset

In [ ]:

```
# describe() function is used to get the statistical summary of the dataset
print("The statistical summary of numerical variables")
print("-----------------------------------------------")
df.describe().T
```

```
The statistical summary of numerical variables
----------------------------------------------
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
EmployeeID	4410.0	2205.500000	1273.201673	1.0	1103.25	2205.5	3307.75	4410.0
Age	4410.0	36.923810	9.133301	18.0	30.00	36.0	43.00	60.0
DistanceFromHome	4410.0	9.192517	8.105026	1.0	2.00	7.0	14.00	29.0
Education	4410.0	2.912925	1.023933	1.0	2.00	3.0	4.00	5.0
EmployeeCount	4410.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0
JobLevel	4410.0	2.063946	1.106689	1.0	1.00	2.0	3.00	5.0
MonthlyIncome	4410.0	65029.312925	47068.888559	10090.0	29110.00	49190.0	83800.00	199990.0
NumCompaniesWorked	4391.0	2.694830	2.498887	0.0	1.00	2.0	4.00	9.0
PercentSalaryHike	4410.0	15.209524	3.659108	11.0	12.00	14.0	18.00	25.0
StandardHours	4410.0	8.000000	0.000000	8.0	8.00	8.0	8.00	8.0
StockOptionLevel	4410.0	0.793878	0.851883	0.0	0.00	1.0	1.00	3.0
TotalWorkingYears	4401.0	11.279936	7.782222	0.0	6.00	10.0	15.00	40.0
TrainingTimesLastYear	4410.0	2.799320	1.288978	0.0	2.00	3.0	3.00	6.0
YearsAtCompany	4410.0	7.008163	6.125135	0.0	3.00	5.0	9.00	40.0
YearsSinceLastPromotion	4410.0	2.187755	3.221699	0.0	0.00	1.0	3.00	15.0
YearsWithCurrManager	4410.0	4.123129	3.567327	0.0	2.00	3.0	7.00	17.0
EnvironmentSatisfaction	4385.0	2.723603	1.092756	1.0	2.00	3.0	4.00	4.0
JobSatisfaction	4390.0	2.728246	1.101253	1.0	2.00	3.0	4.00	4.0
WorkLifeBalance	4372.0	2.761436	0.706245	1.0	2.00	3.0	3.00	4.0
JobInvolvement	4410.0	2.729932	0.711400	1.0	2.00	3.0	3.00	4.0
PerformanceRating	4410.0	3.153741	0.360742	3.0	3.00	3.0	3.00	4.0

Observations:

- ***Majority of the employees are in their 30s and 40s.***
- ***A significant portion of employees live within a 14km radius***
- ***Most employees have completed at least a bachelor's degree.***

Checking the statistical summary of the categorical variables in the dataset

In [ ]:

```
# describe() function is used to get the statistical summary of the dataset
print("The statistical summary of categorical variables")
print("------------------------------------------------")
df.describe(include = "object").T
```

```
The statistical summary of categorical variables
------------------------------------------------
```

Out[ ]:

	count	unique	top	freq
Attrition	4410	2	No	3699
BusinessTravel	4410	3	Travel_Rarely	3129
Department	4410	3	Research & Development	2883
EducationField	4410	6	Life Sciences	1818
Gender	4410	2	Male	2646
JobRole	4410	9	Sales Executive	978
MaritalStatus	4410	3	Married	2019
Over18	4410	1	Y	4410

Observations:

The "No" response under the "Attrition" variable indicates that a sizable majority of the employees (3699 out of 4410) did not quit the company. This implies a low rate of attrition.

Checking for missing values in the dataset

In [ ]:

```
# isnull() function is used to identify the null values in a dataset
df.isnull().sum()
```

Out[ ]:

	0
EmployeeID	0
Age	0
Attrition	0
BusinessTravel	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EmployeeCount	0

Gender	0
JobLevel	0
JobRole	0
MaritalStatus	0
MonthlyIncome	0
NumCompaniesWorked	19
Over18	0
PercentSalaryHike	0
StandardHours	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

Observations:

There are missing values in NumCompaniesWorked, EnvironmentSatisfaction, JobSatisfaction and WorkLifeBalance.

Checking for duplicate values in the dataset

In [ ]:

```
# duplicated() function is used to identify the duplicate values in a dataset
print("The duplicate values in the dataset is",df.duplicated().sum())
```

```
The duplicate values in the dataset is 0
```

Observations:

There are no duplicate values in the dataset taken for analysis.

Exploratory Data Analysis (EDA)

Univariate Analysis

In [ ]:

```
# function to plot a boxplot and a histogram along the same scale.

def histogram_boxplot(data, feature, figsize=(15,10), kde=False, bins=None):
  """
  Boxplot and histogram combined
```

```
    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (15,10))
    kde: whether to show the density curve (default False)
    bins: number of bins for histogam (default None)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows = 2, # Number of rows of the subplot grid= 2
        sharex=True, # x-axis will be shared among all subplots
        gridspec_kw={"height_ratios": (0.25, 0.75)},
        figsize=figsize,
    ) # creating the 2 subplots
    sns.boxplot(
        data=data, x=feature, ax=ax_box2, showmeans=True, color="violet"
    )  # boxplot will be created and a triangle will indicate the mean value of the column
    sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2, bins=bins
    ) if bins else sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2
    )  # For histogram
    ax_hist2.axvline(
        data[feature].mean(), color="green", linestyle="--"
    ) # Add mean to the histogram
    ax_hist2.axvline(
        data[feature].median(), color="black", linestyle="-"
    ) # Add median to the histogram
```

In [ ]:

```
# function to create labeled barplots


def labeled_barplot(data, feature, perc=False, n=None):
    """
    Barplot with percentage at the top

    data: dataframe
    feature: dataframe column
    perc: whether to display percentages instead of count (default is False)
    n: displays the top n category levels (default is None, i.e., display all levels)
    """

    total = len(data[feature])  # length of the column
    count = data[feature].nunique()
    if n is None:
        plt.figure(figsize=(count + 2, 6))
    else:
        plt.figure(figsize=(n + 2, 6))

    plt.xticks(rotation=90, fontsize=15)
    ax = sns.countplot(
        data=data,
        x=feature,
        palette="Paired",
        order=data[feature].value_counts().index[:n],
    )

    for p in ax.patches:
        if perc == True:
            label = "{:.1f}%".format(
                100 * p.get_height() / total
            )  # percentage of each class of the category
        else:
            label = p.get_height()  # count of each level of the category

        x = p.get_x() + p.get_width() / 2  # width of the plot
        y = p.get_height()  # height of the plot

        ax.annotate(
            label,
```

```
            (x, y),
            ha="center",
            va="center",
            size=12,
            xytext=(0, 5),
            textcoords="offset points",
        )  # annotate the percentage

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

Age

In [ ]:

```
# Finding unique ages
print("Total number of unique ages:",df['Age'].nunique())
```

```
Total number of unique ages: 43
```

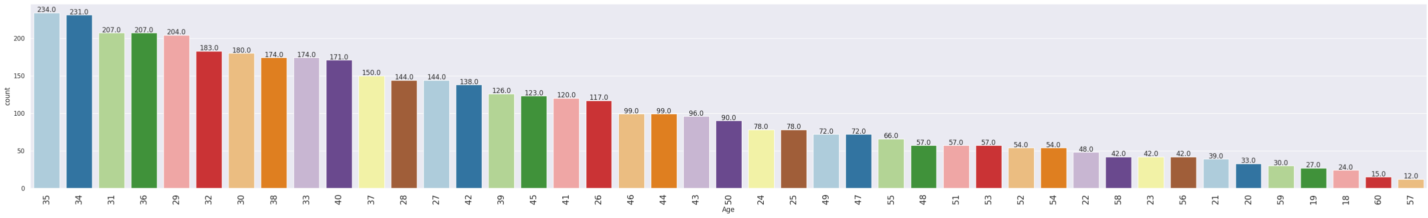
In [ ]:

```
labeled_barplot(df,'Age')
```

```
<ipython-input-12-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



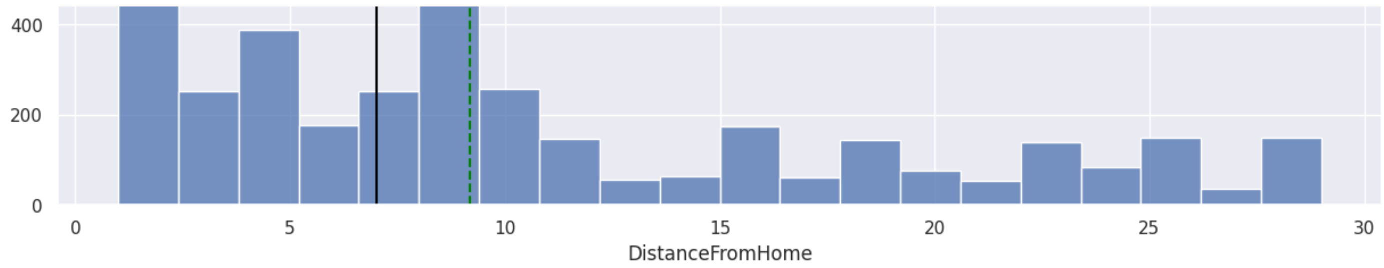
Observations

There are 43 unique ages in total with a majority grouped under the age of 35

DistanceFromeHome

In [ ]:

```
histogram_boxplot(df,'DistanceFromHome')
```



Observations:

- ***The distribution is right-skewed, indicating that most employees live closer to their workplace.***
- ***A smaller number of employees live farther away.***

EducationField

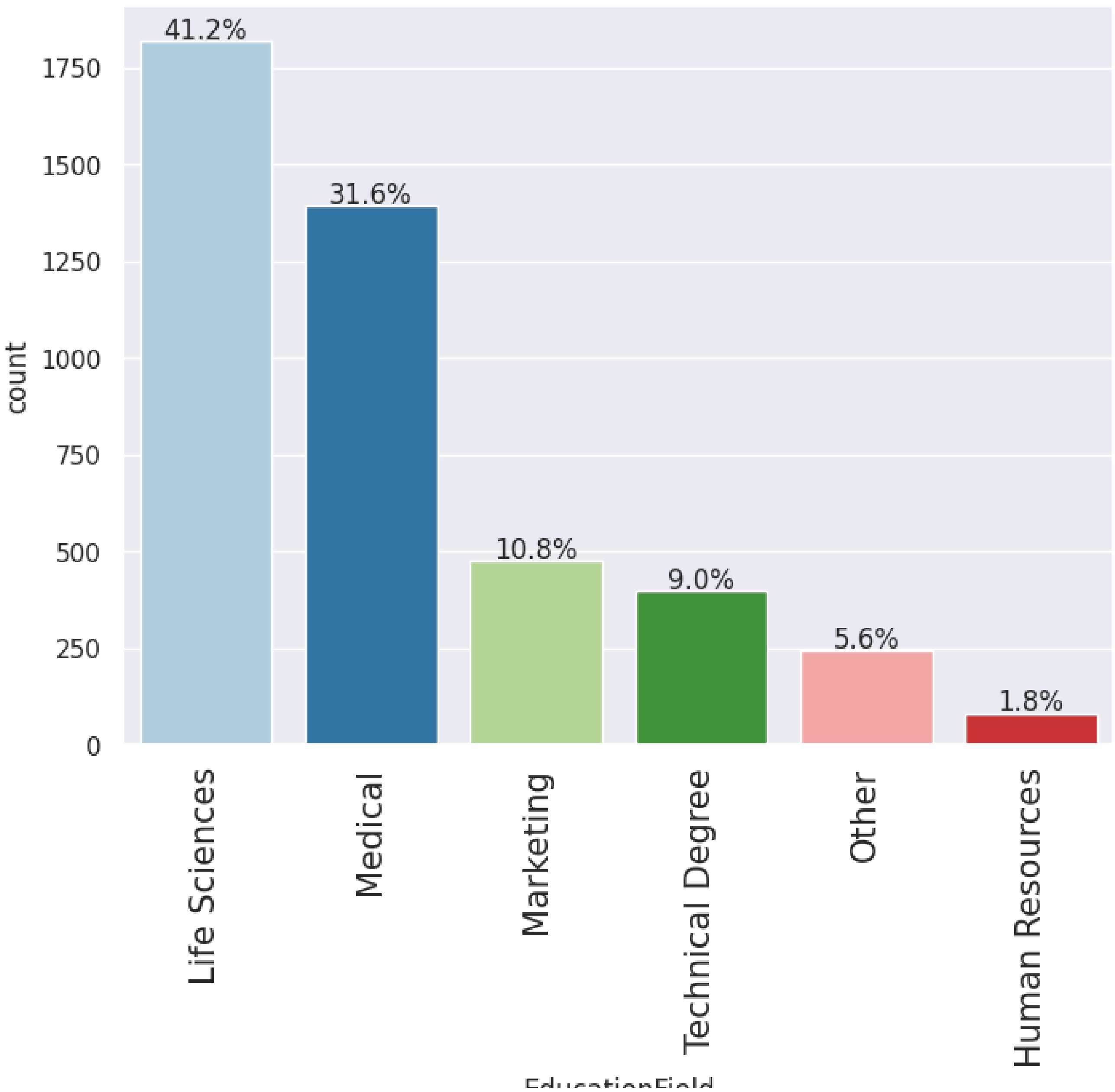
In [ ]:

```
labeled_barplot(df,'EducationField',perc=True)
```

```
<ipython-input-12-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



EducationField

Observations:

41.2% employee's education field is life sciences and about 1.8% employee's education field is human resources

JobRole

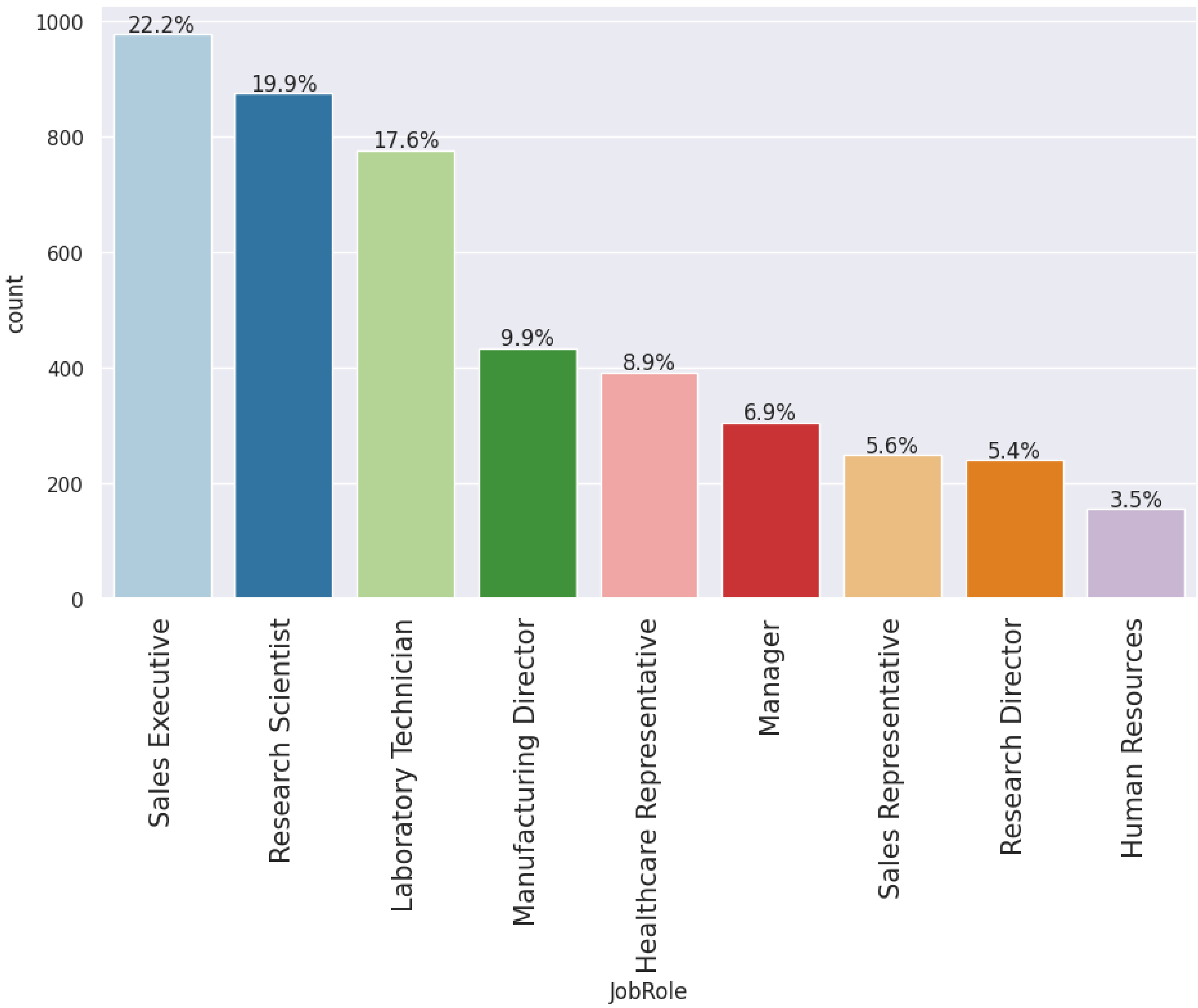
In [ ]:

```
labeled_barplot(df,'JobRole', perc=True)
```

```
<ipython-input-12-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



Observations:

22.2% employees are Sales Executive and 3.5% employees are Human Resources.

Attrition

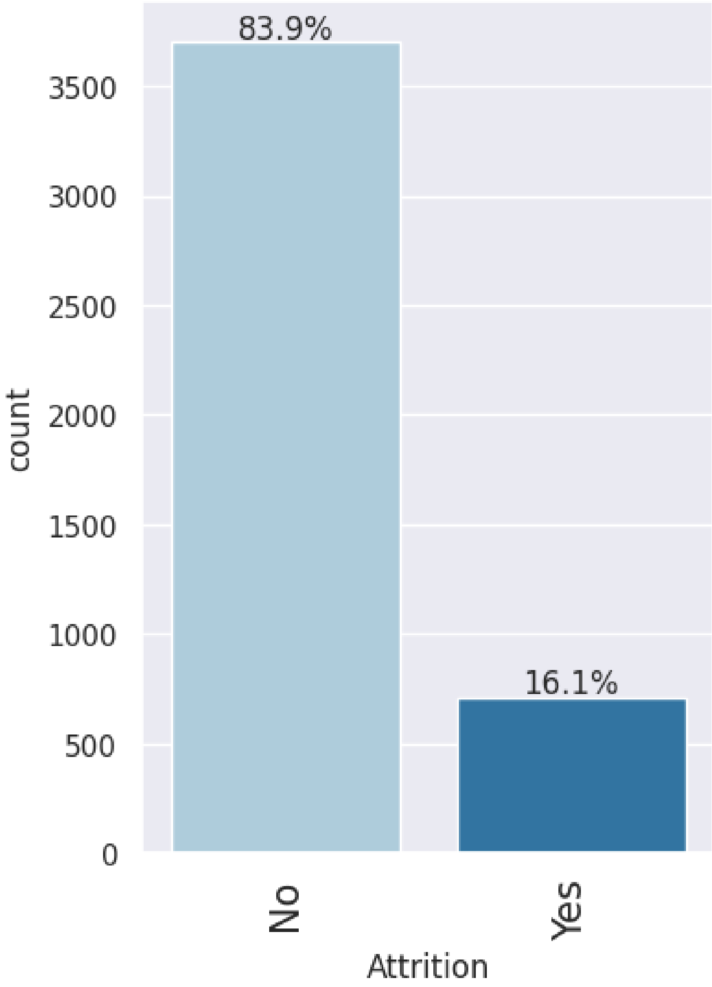
In [ ]:

```
labeled_barplot(df,'Attrition', perc= True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



Observations

There is an attrition rate of 16.1% among the employees.

BusinessTravel

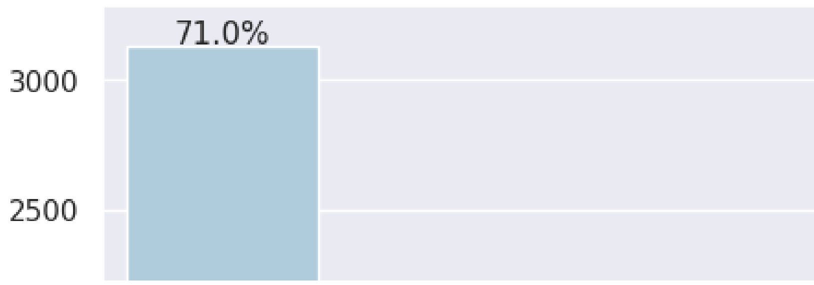
In [ ]:

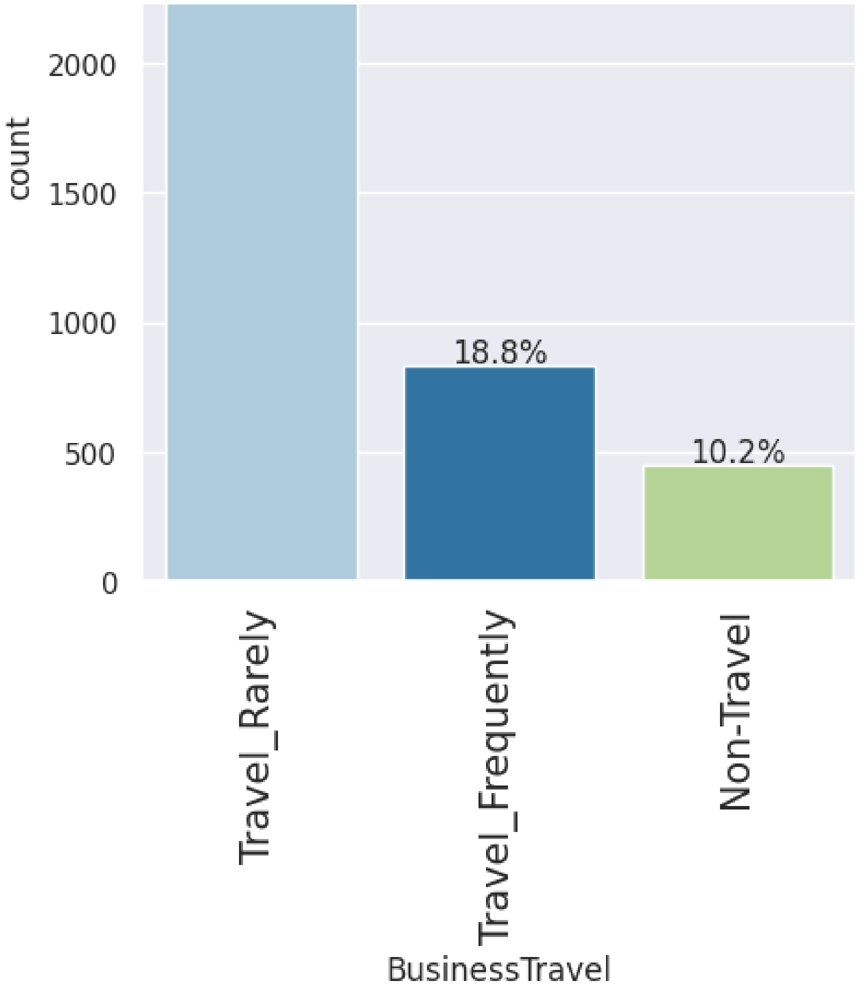
```
labeled_barplot(df,'BusinessTravel',perc = True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```





Observations:

- ***The distribution is heavily skewed towards 'Travel_Rarely'.***
- ***Around 18.8% of employees travel frequently and only 10.2% of employees do not travel at all.***

Department

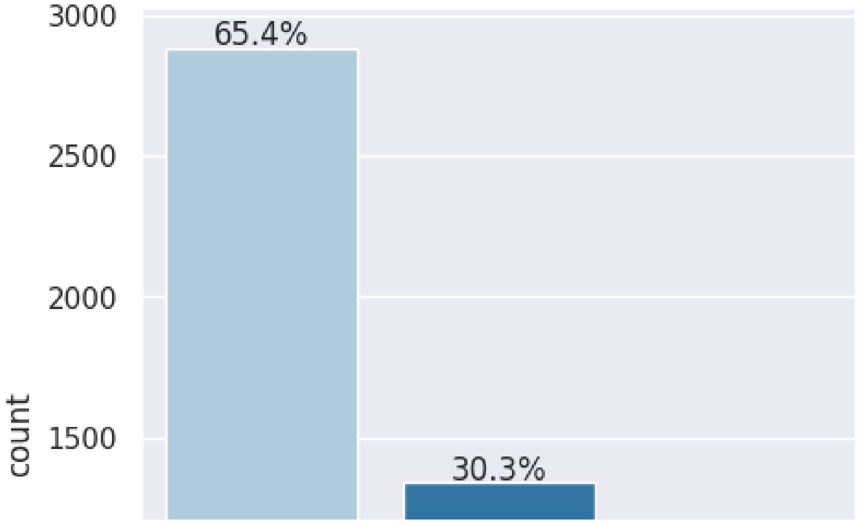
In [ ]:

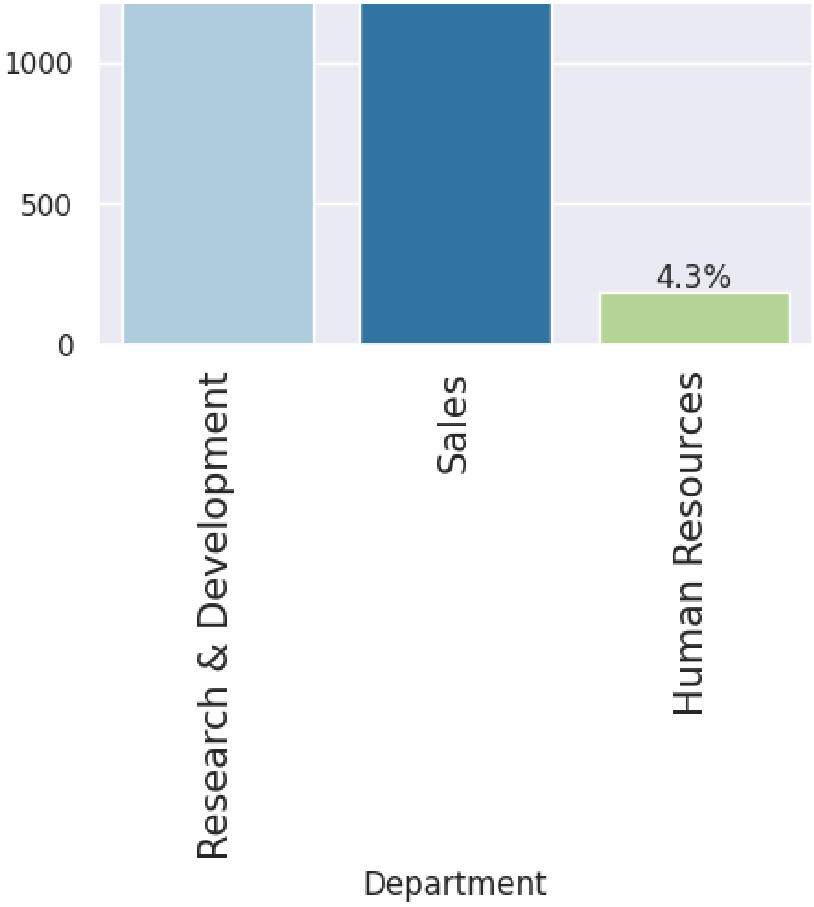
```
labeled_barplot(df,'Department', perc = True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```





Observations:

65.4% of the employees are in the department of Research and Development, 30.3% in Sales and 4.3% in Human Resources.

Gender

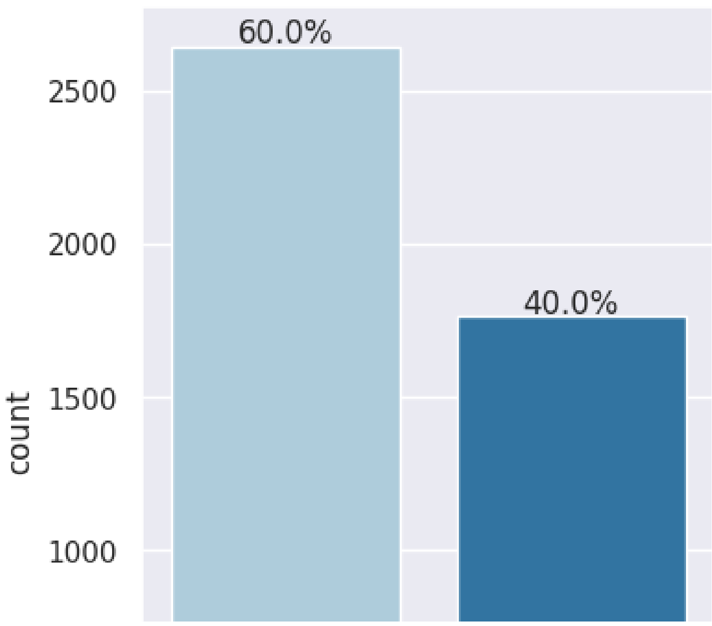
In [ ]:

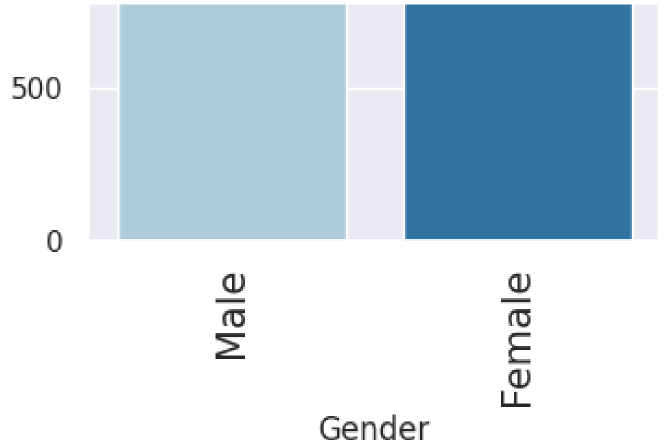
```
labeled_barplot(df,'Gender', perc = True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```





Observations:

60.0% employees are Male and 40.0% employees are Female.

Education

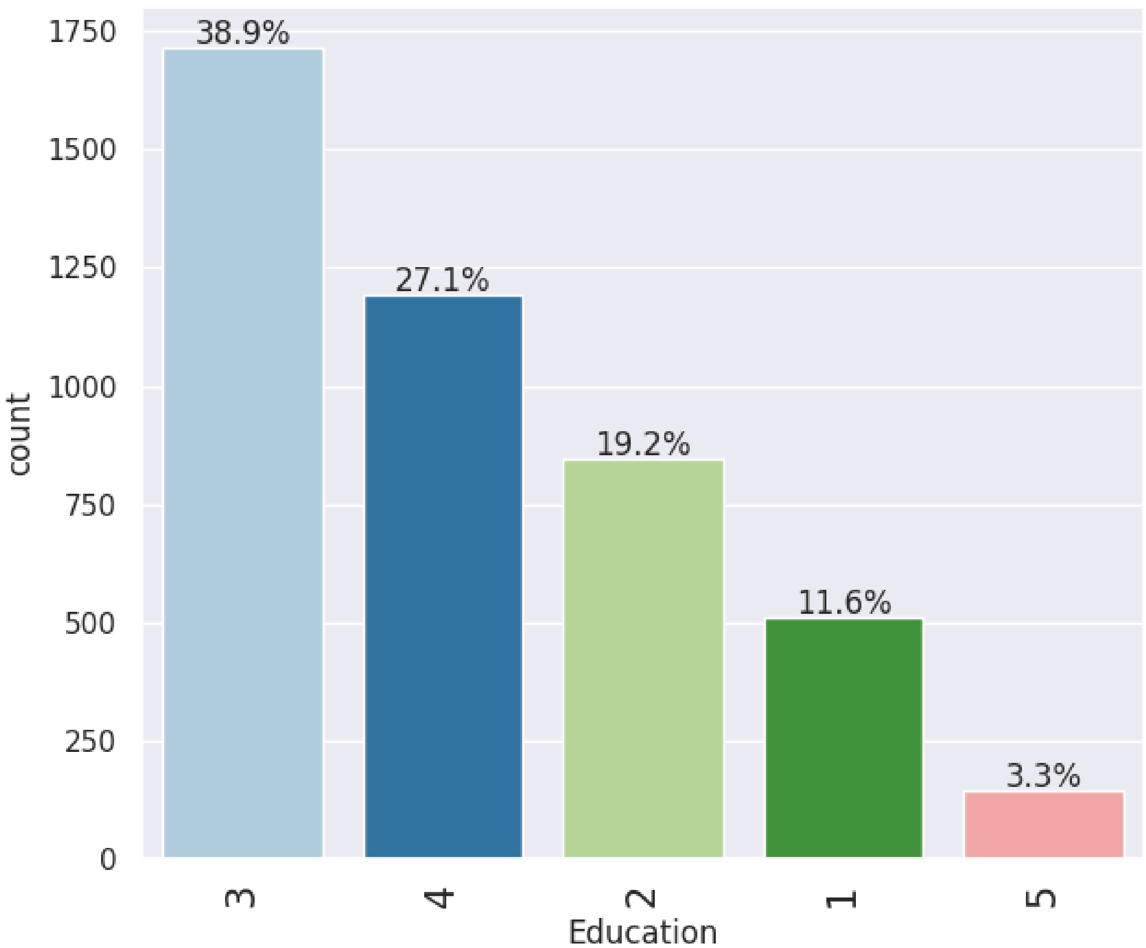
In [ ]:

```
labeled_barplot(df,'Education', perc = True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



Observations

- ***As the education level increases, the frequency of responses decreases.***
- ***About 3.3% of the employees have highest education level (5).***

JobLevel

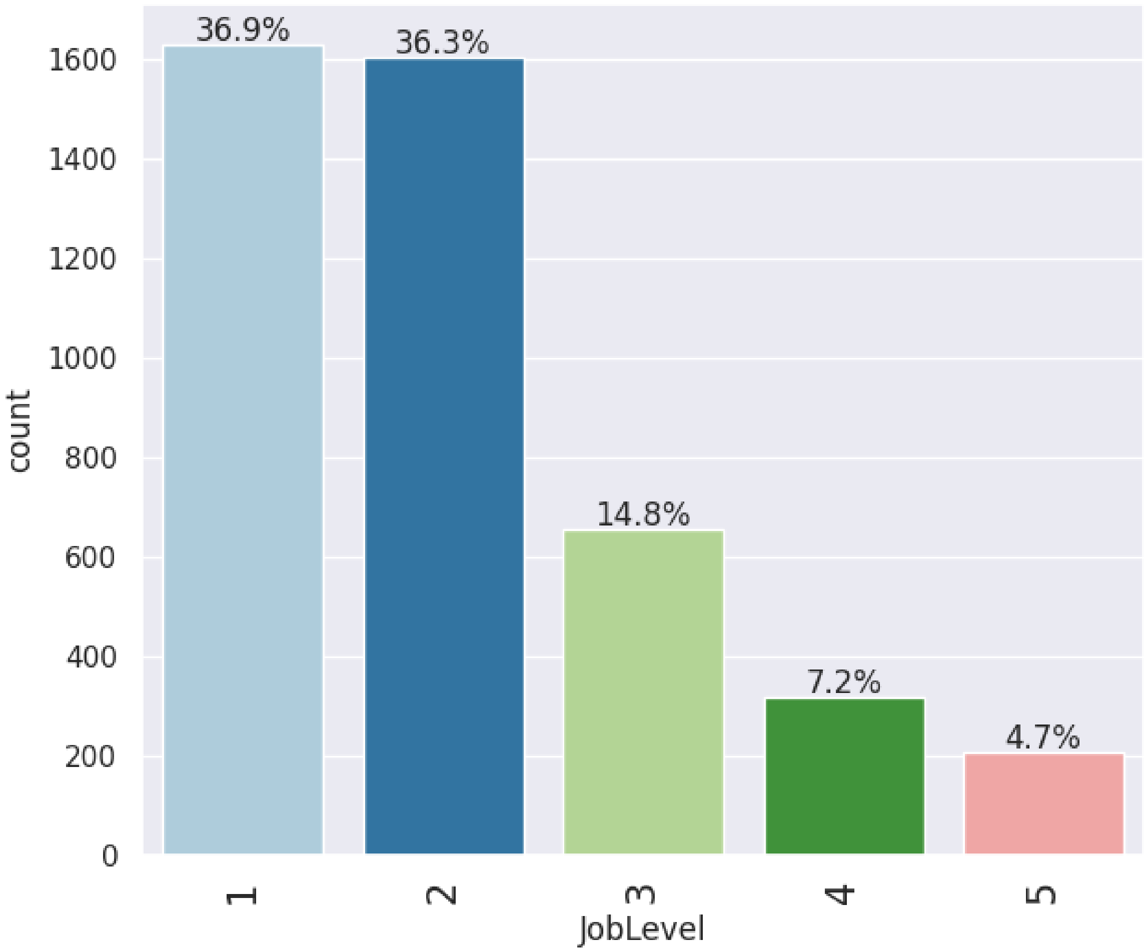
In [ ]:

```
labeled_barplot(df,'JobLevel', perc = True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



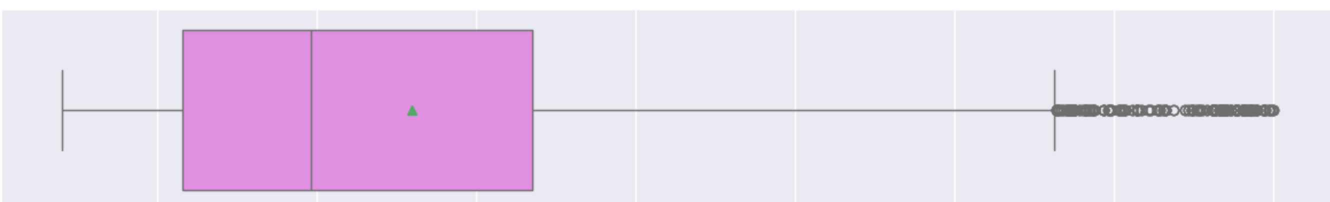
Observations:

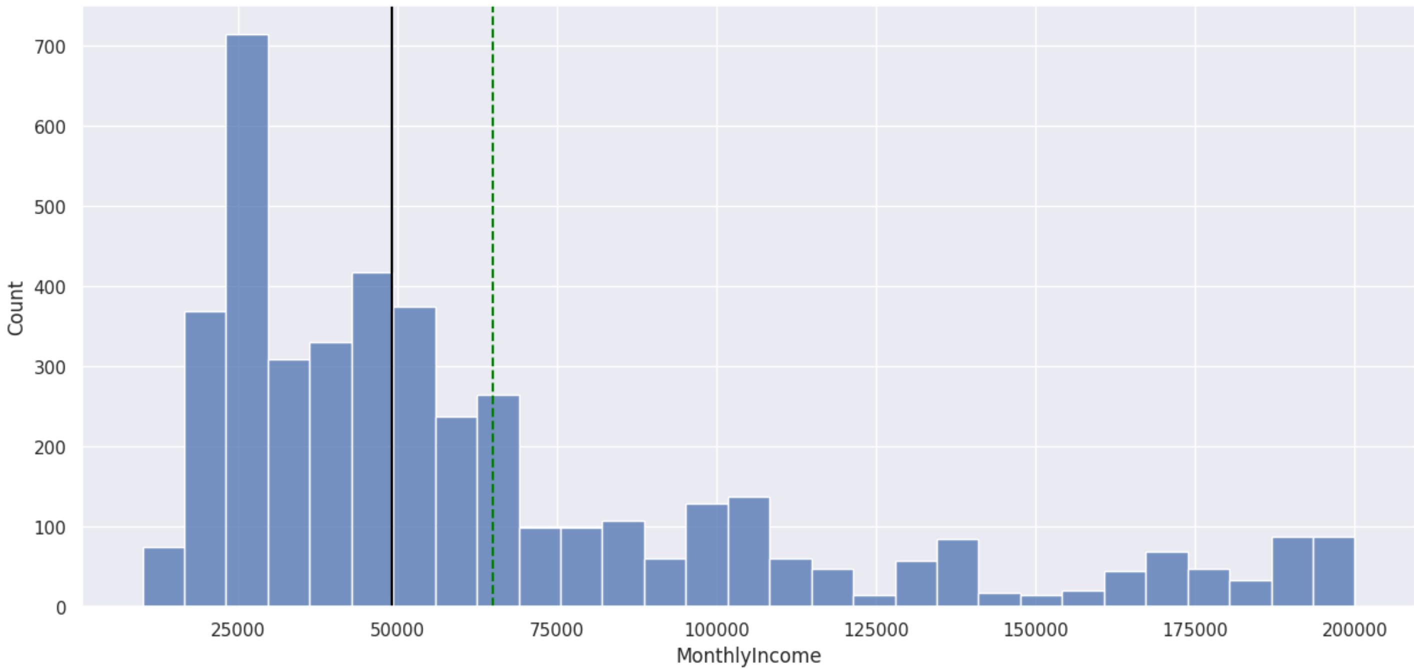
- ***As the job level increases, the frequency of employees generally decreases.***
- ***The highest job level (5) has the lowest representation of about 4.7% of the employees.***

MonthlyIncome

In [ ]:

```
histogram_boxplot(df,'MonthlyIncome')
```





Observations:

- ***The distribution is right-skewed, indicating that most employees have lower monthly incomes.***
- ***The highest frequency is around the 25,000 mark, with about 700 counts.***
- ***There is a significant concentration of employees earning less than 75,000.***

NumCompaniesWorked

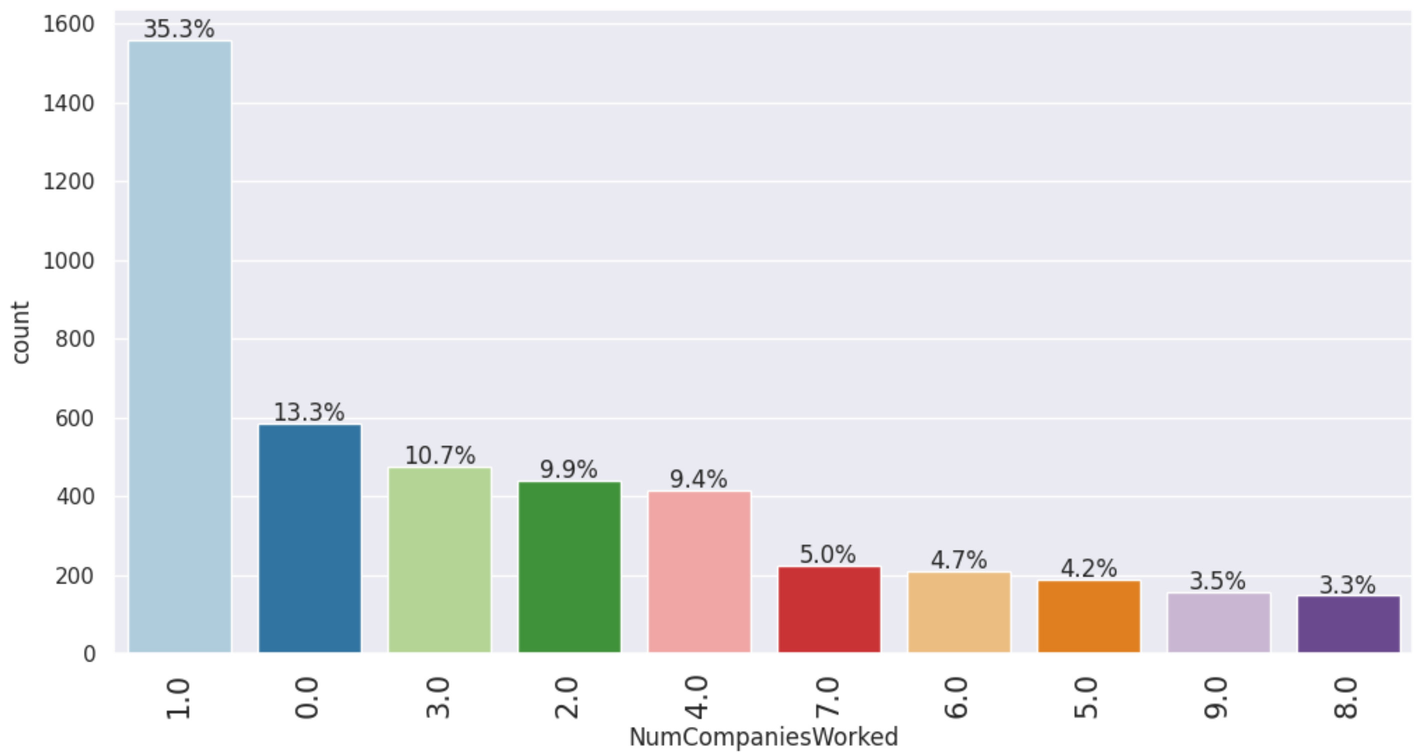
In [ ]:

```
labeled_barplot(df,'NumCompaniesWorked', perc = True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



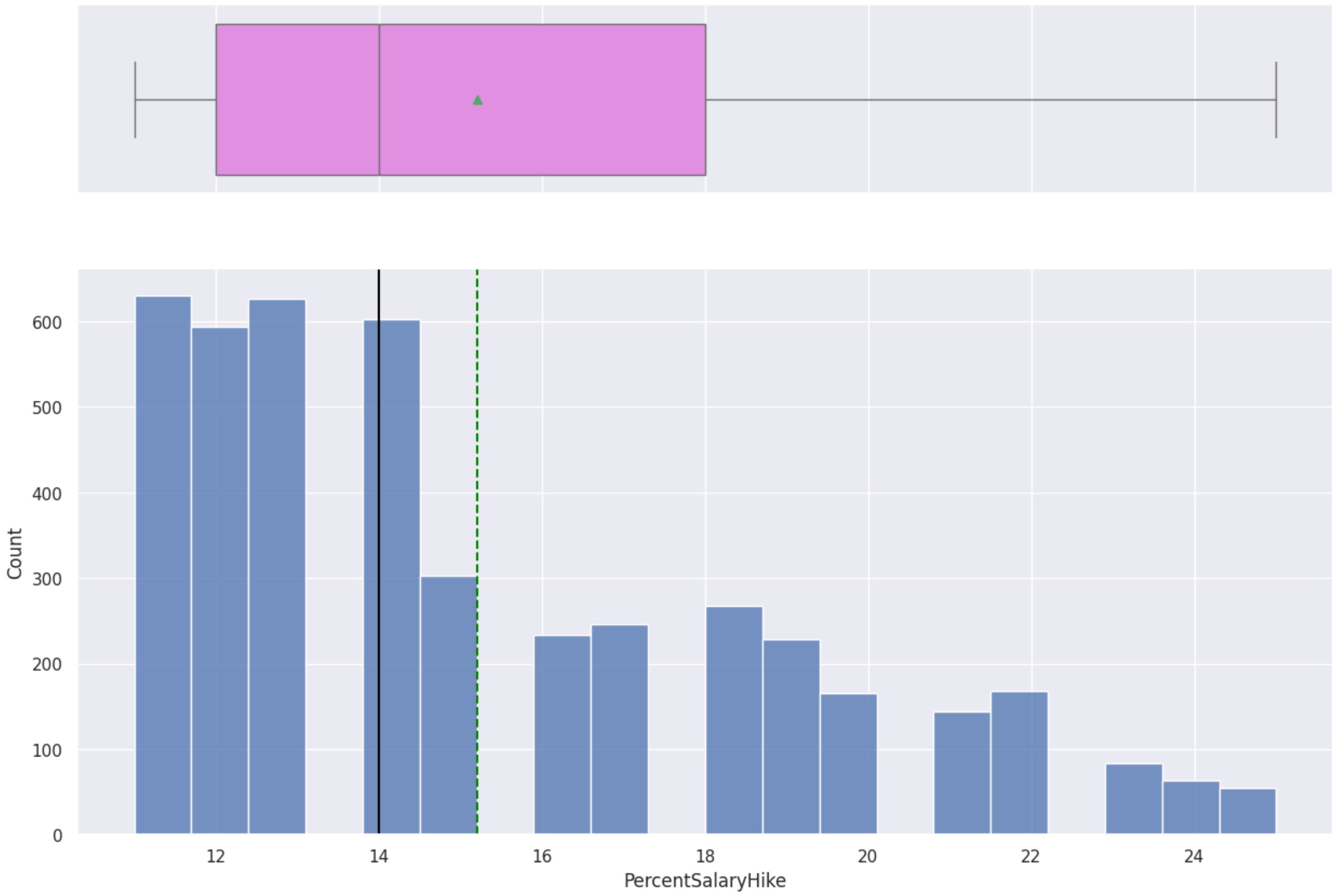
Observations:

- ***35.3% employees worked in 1 comapny and 3.3% employees worked in 8 companies.***
- ***13.3% employees are freshers.***

PercentSalaryHike

In [ ]:

```
histogram_boxplot(df,'PercentSalaryHike')
```



Observations:

The frequency of salary hikes decreases as the percentage increases. This means that fewer employees received higher salary hikes compared to those in the lower and middle ranges.

StockOptionLevel

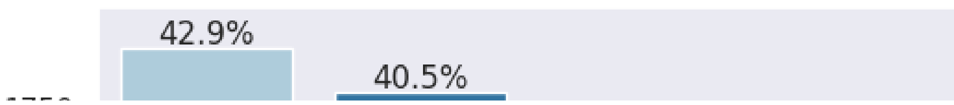
In [ ]:

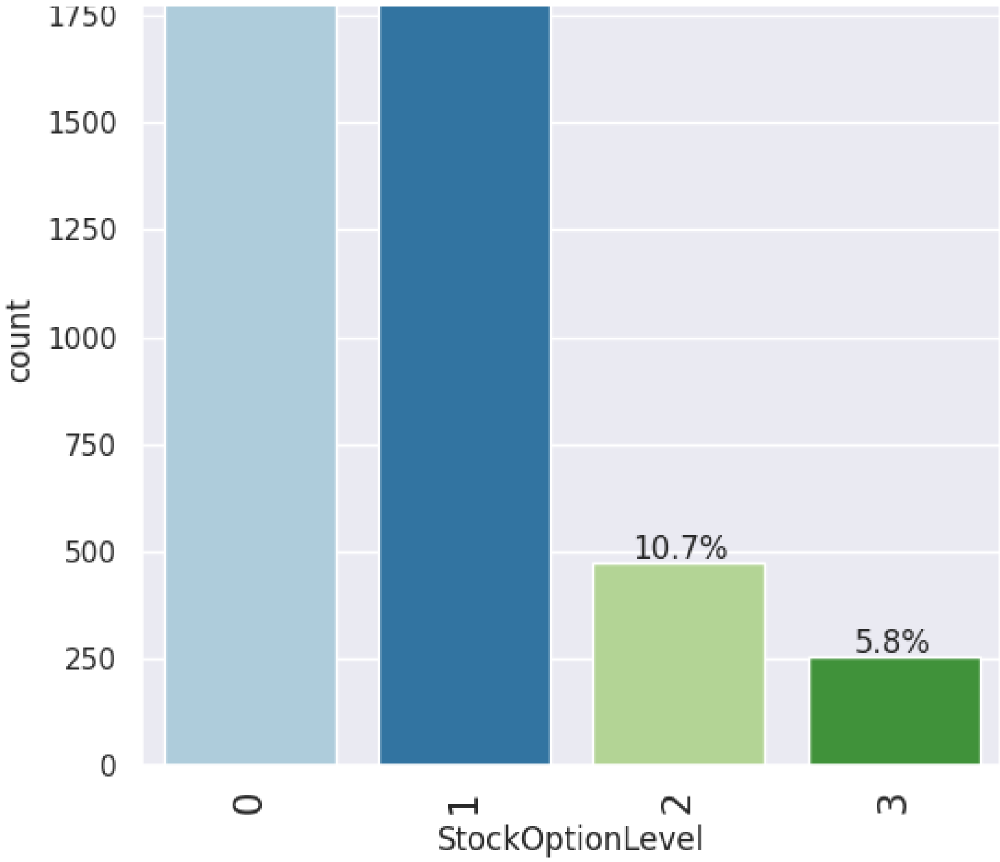
```
labeled_barplot(df,'StockOptionLevel',perc=True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```





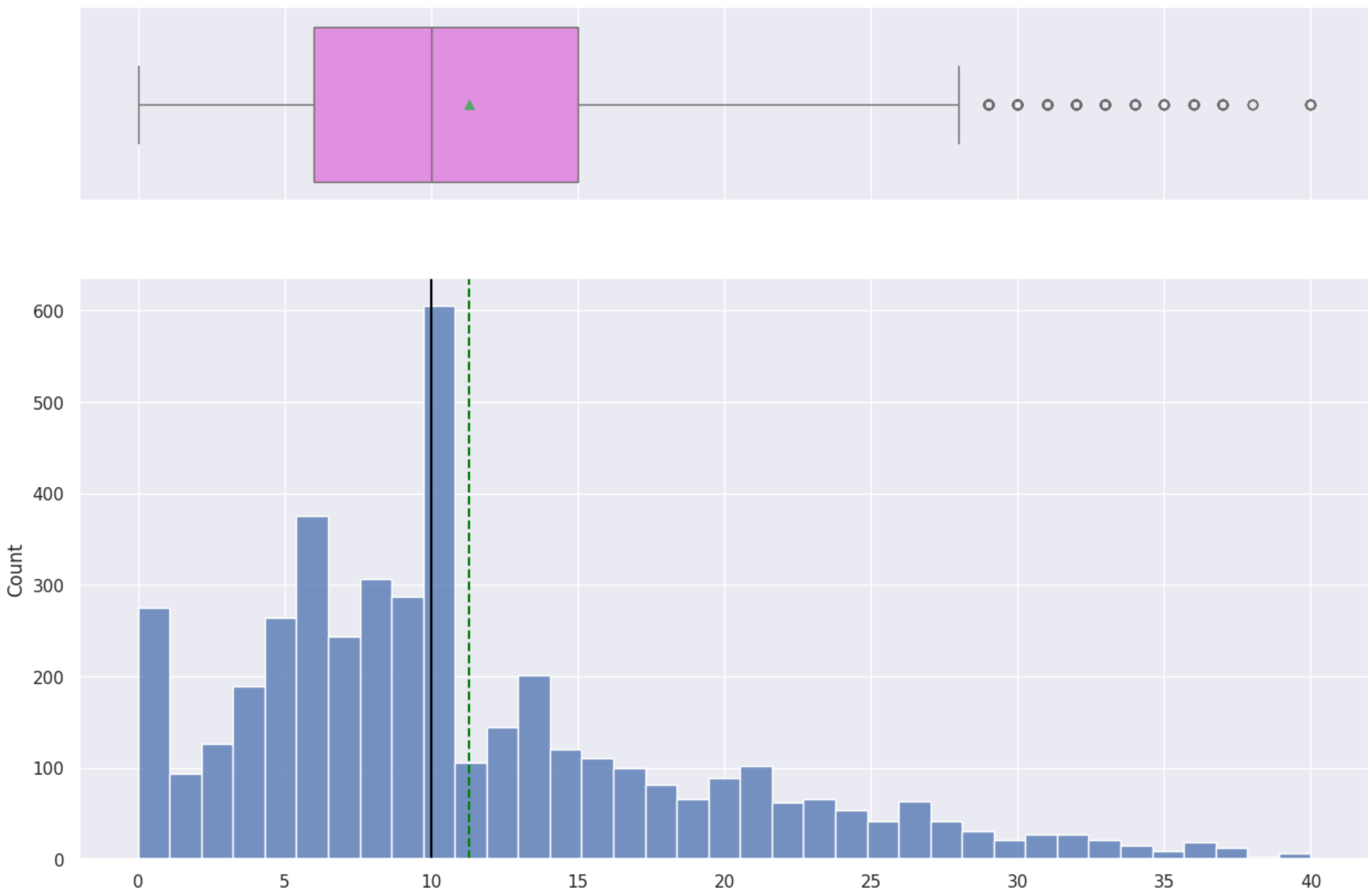
Observations:

Only a small percentage of employees have stock option levels 2 or 3.

TotalWorkingYears

In [ ]:

```
histogram_boxplot(df,'TotalWorkingYears')
```



TotalWorkingYears

Observations:

- ***This graph indicates that most employees have a relatively short to moderate working experience (5-15 years).***
- ***A small group of employees has significantly more experience than the majority.***

TrainingTimesLastYear

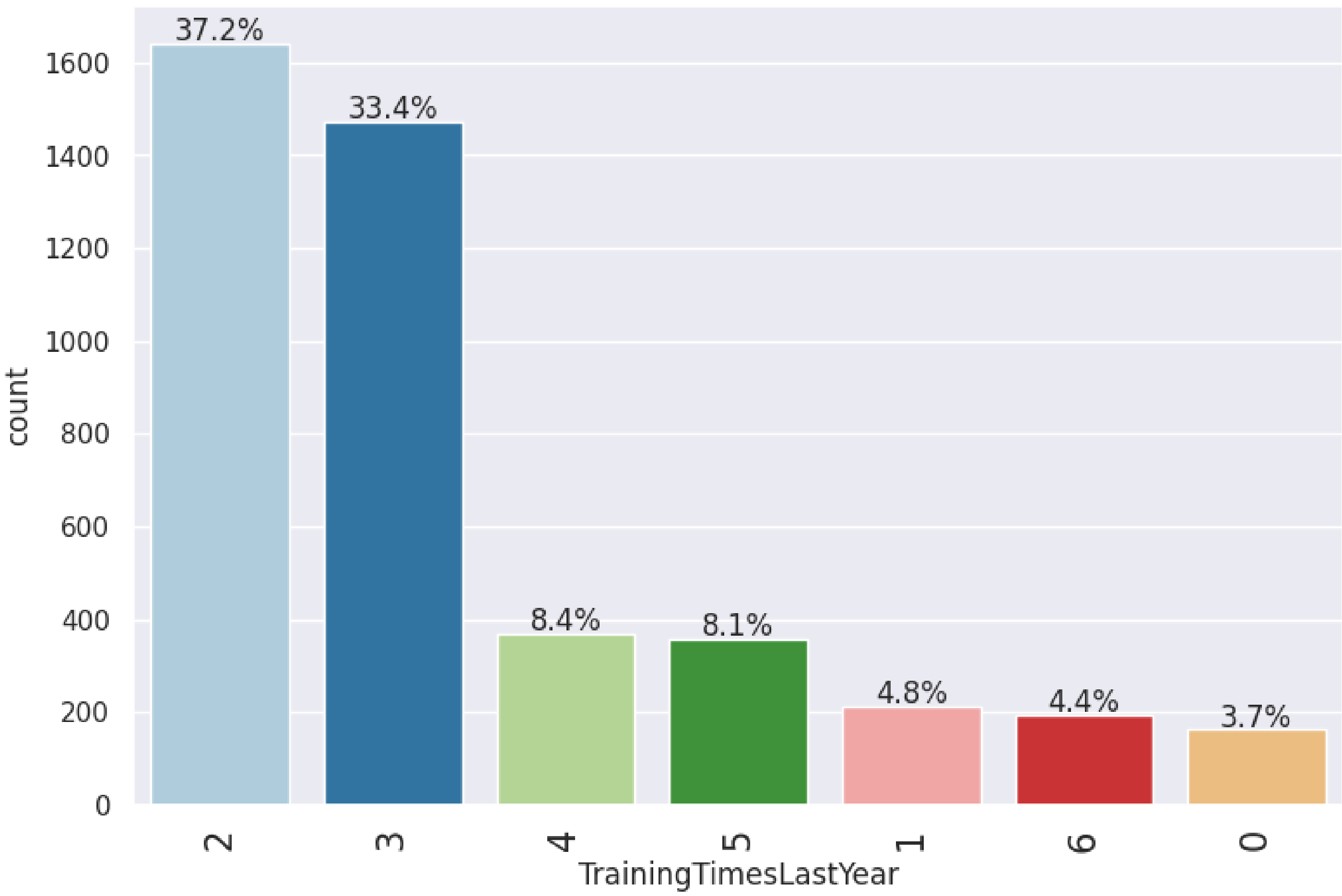
In [ ]:

```
labeled_barplot(df,'TrainingTimesLastYear',perc=True)
```

```
<ipython-input-14-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



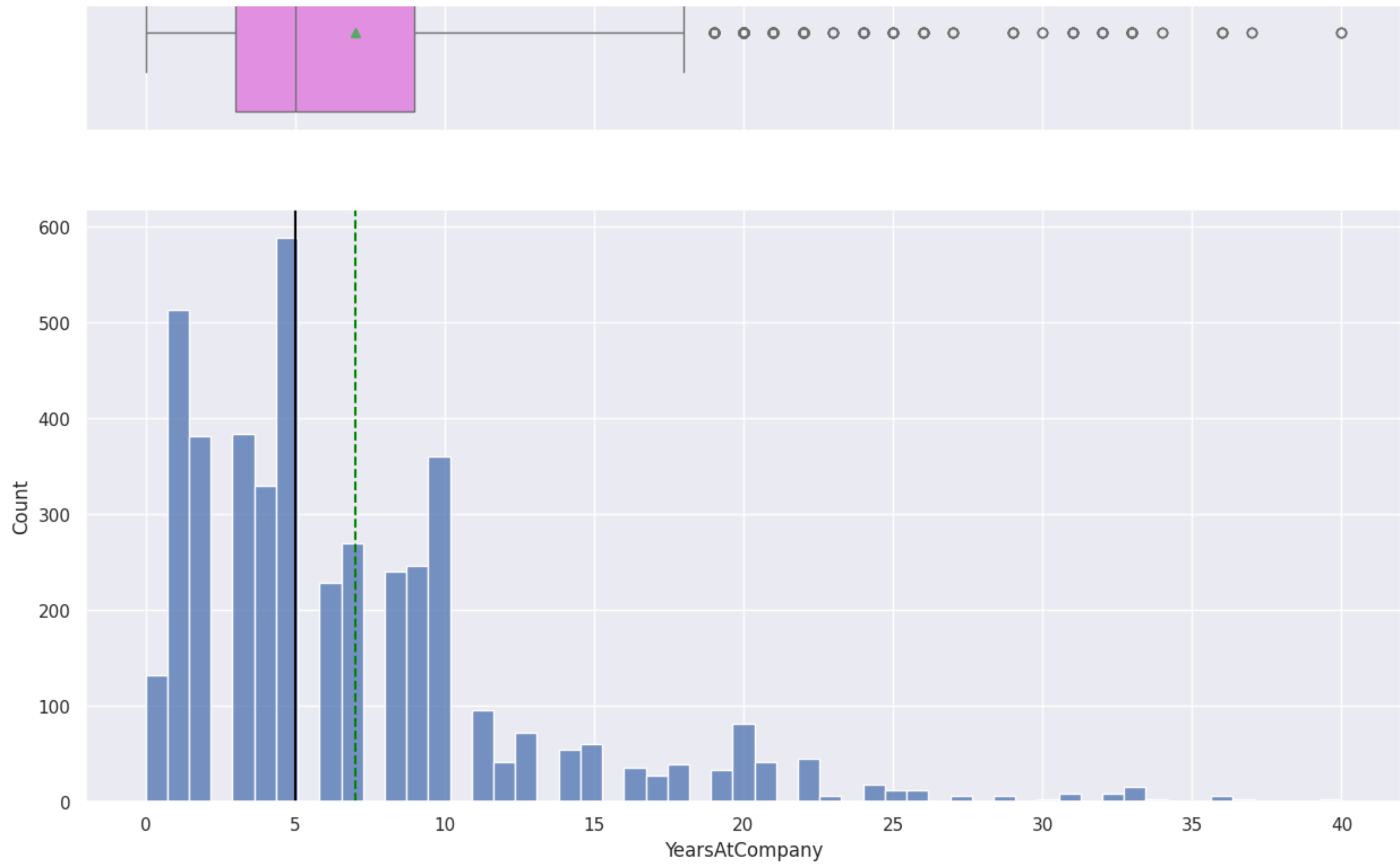
Observations:

The distribution of training times last year is heavily skewed towards right.

YearsAtCompany

In [ ]:

```
histogram_boxplot(df,'YearsAtCompany')
```



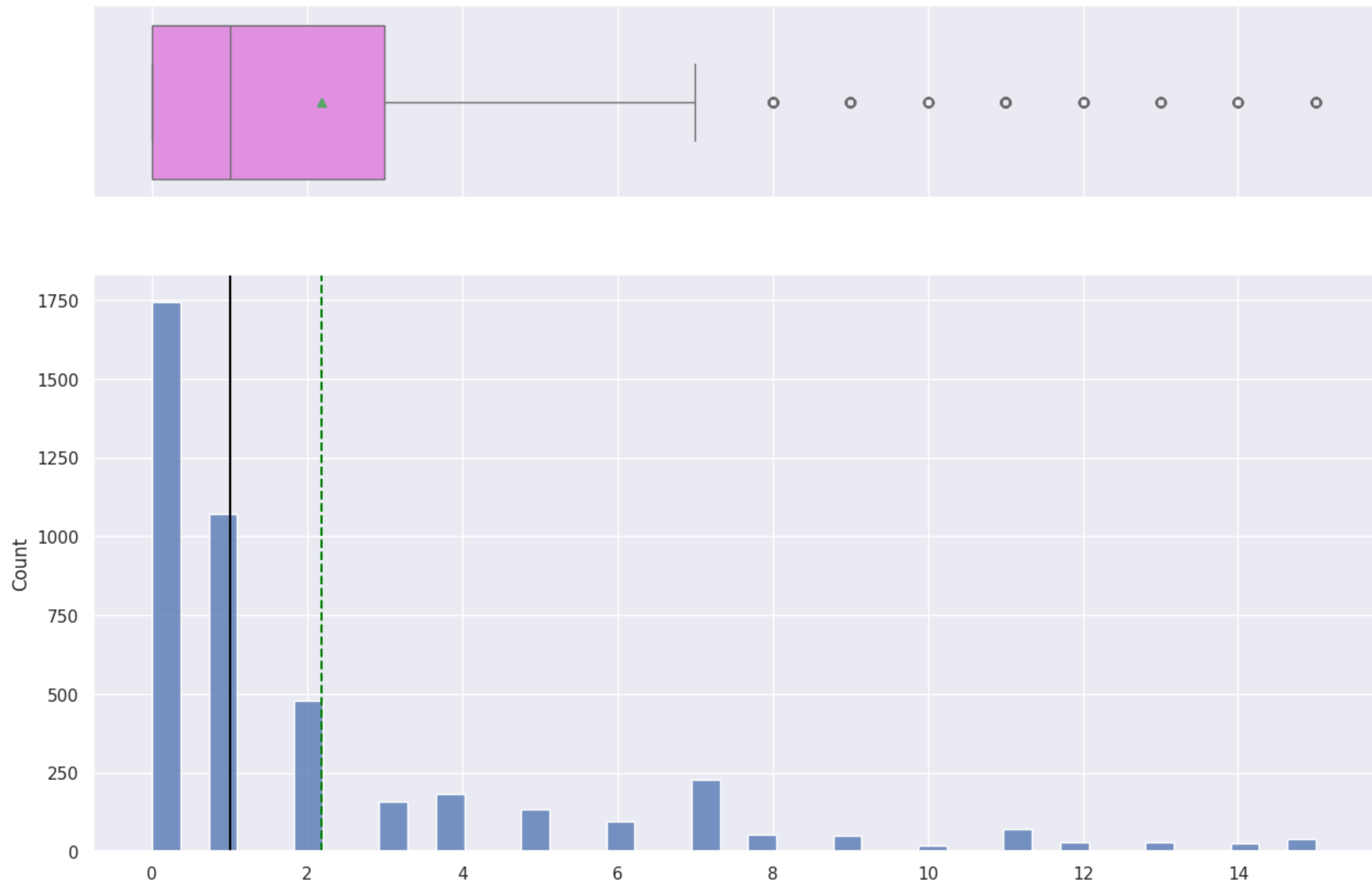
Observations:

A small group of employees has significantly more experience with the company than the majority.

YearsSinceLastPromotion

In [ ]:

```
histogram_boxplot(df,'YearsSinceLastPromotion')
```



YearsSinceLastPromotion

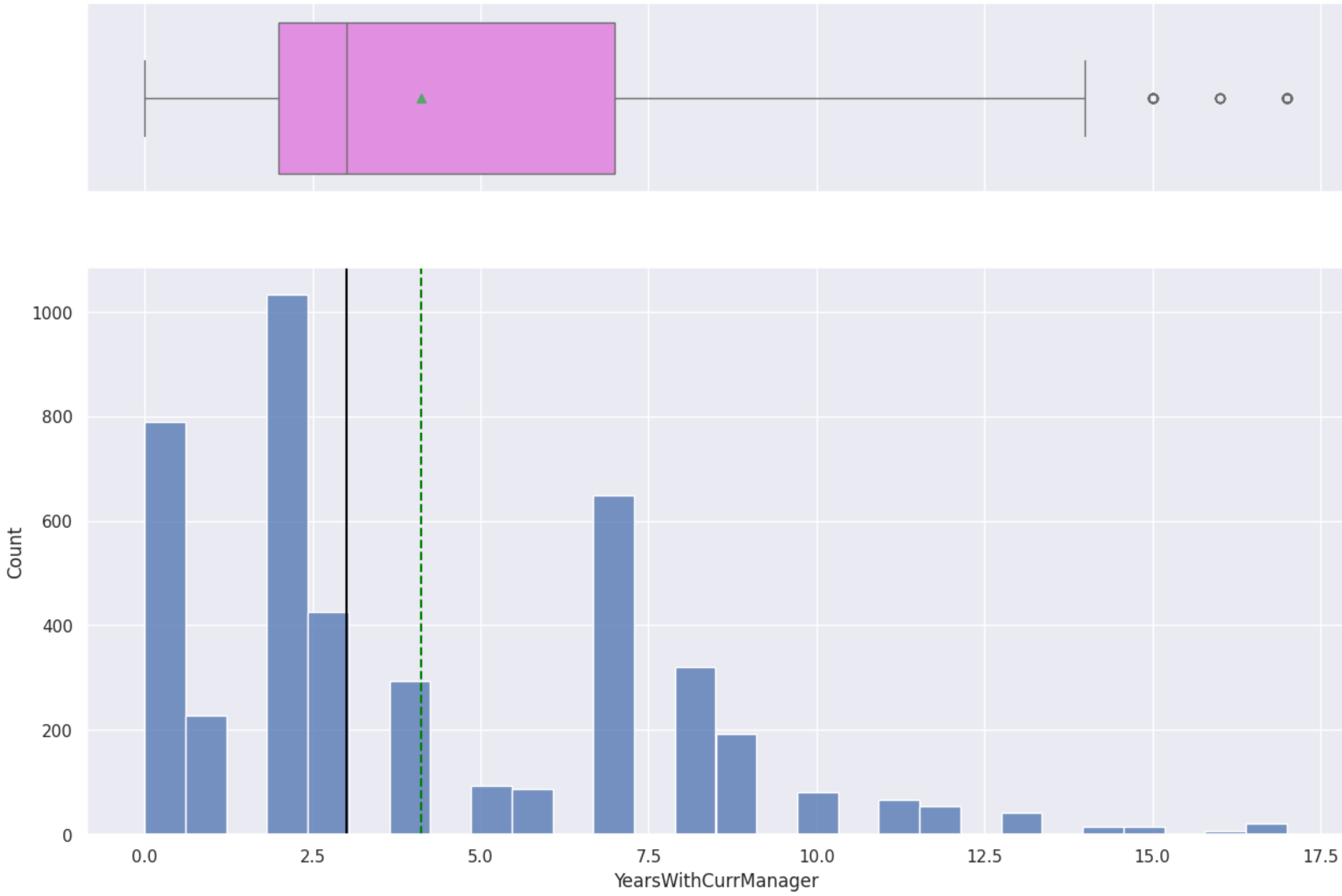
Observations:

- ***A signifiacnt number of employees haven't been promoted.***
- ***A large group of employees has been without a promotion for an extended period.***

YearsWithCurrManager

In [ ]:

```
histogram_boxplot(df,'YearsWithCurrManager')
```



Observations:

Most of the employees have worked with their current manager for a relatively short period (0-4 years).

EnvironmentSatisfaction

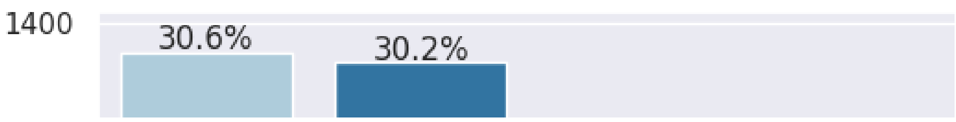
In [ ]:

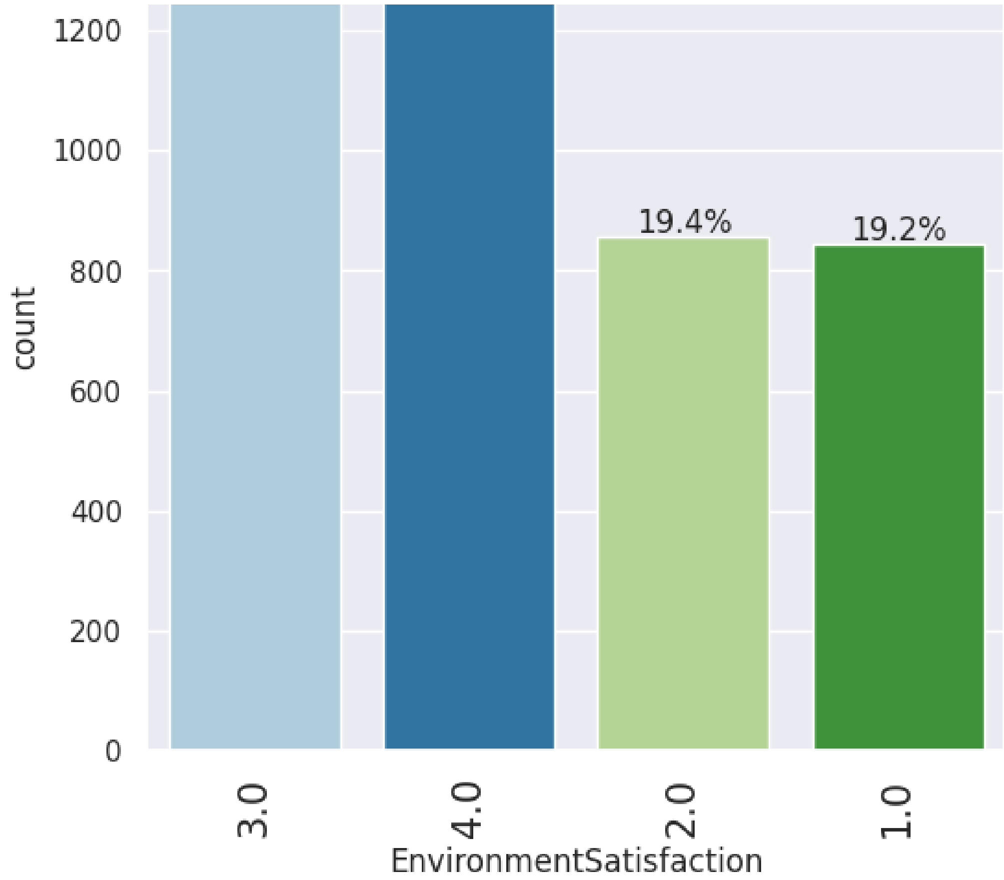
```
labeled_barplot(df,'EnvironmentSatisfaction',perc = True)
```

```
<ipython-input-3-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```





Observations:

- ***The distribution is relatively balanced across the four levels.***
- ***The company has a mix of employees with varying levels of environment satisfaction.***

JobSatisfaction

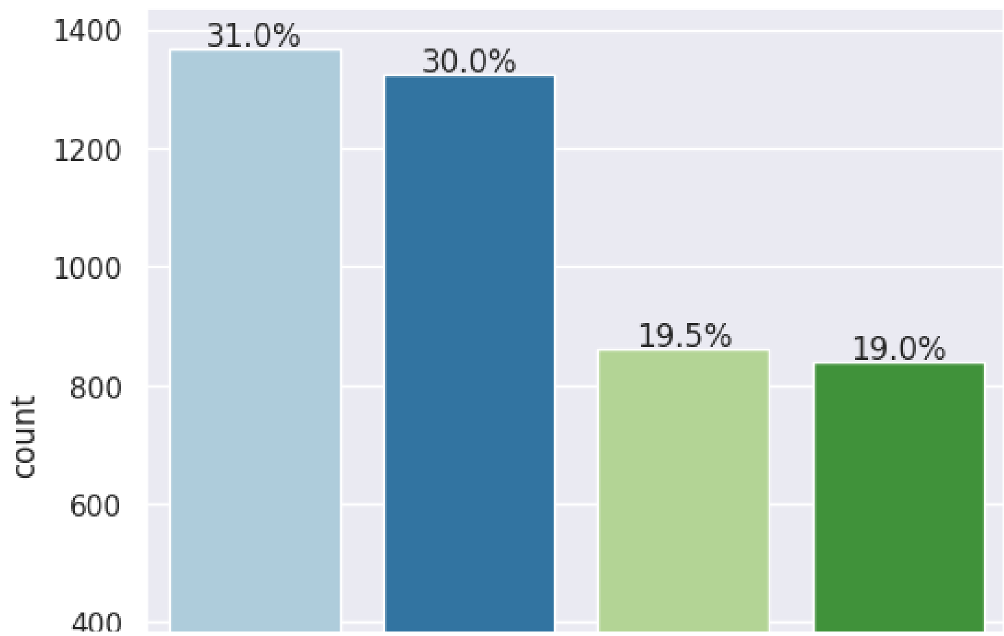
In [ ]:

```
labeled_barplot(df,'JobSatisfaction', perc = True)
```

```
<ipython-input-3-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



Observations:

- ***The distribution of job satisfaction is relatively balanced across four levels.***
- ***While there is a significant portion of employees satisfied with their jobs (levels 3.0 and 4.0), there is also a considerable group experencing lower levels of satisfaction.***

WorkLifeBalance

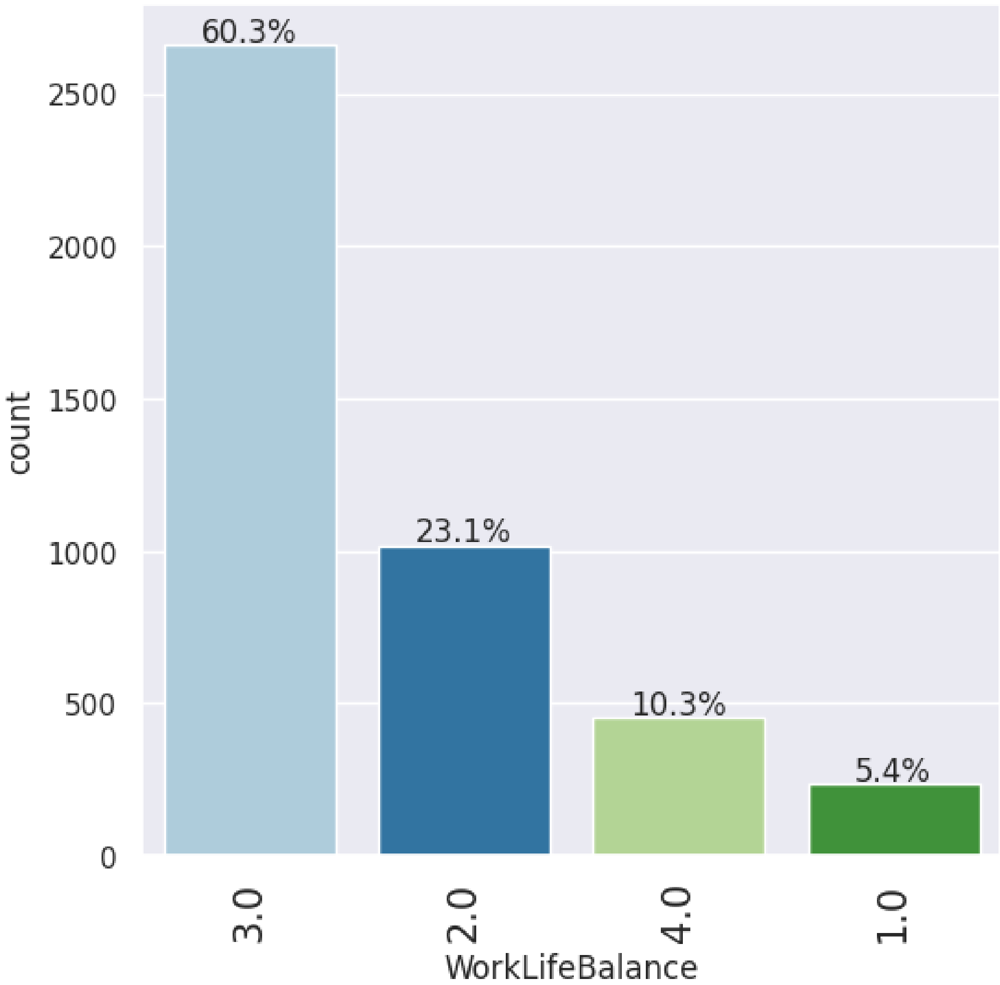
In [ ]:

```
labeled_barplot(df,'WorkLifeBalance', perc=True)
```

```
<ipython-input-3-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



Observations:

The high concentration of employees at work-life balance level3 suggests that there might be opportunities to enhance work-life balance for a significant portion of the workforce.

JobInvolvement

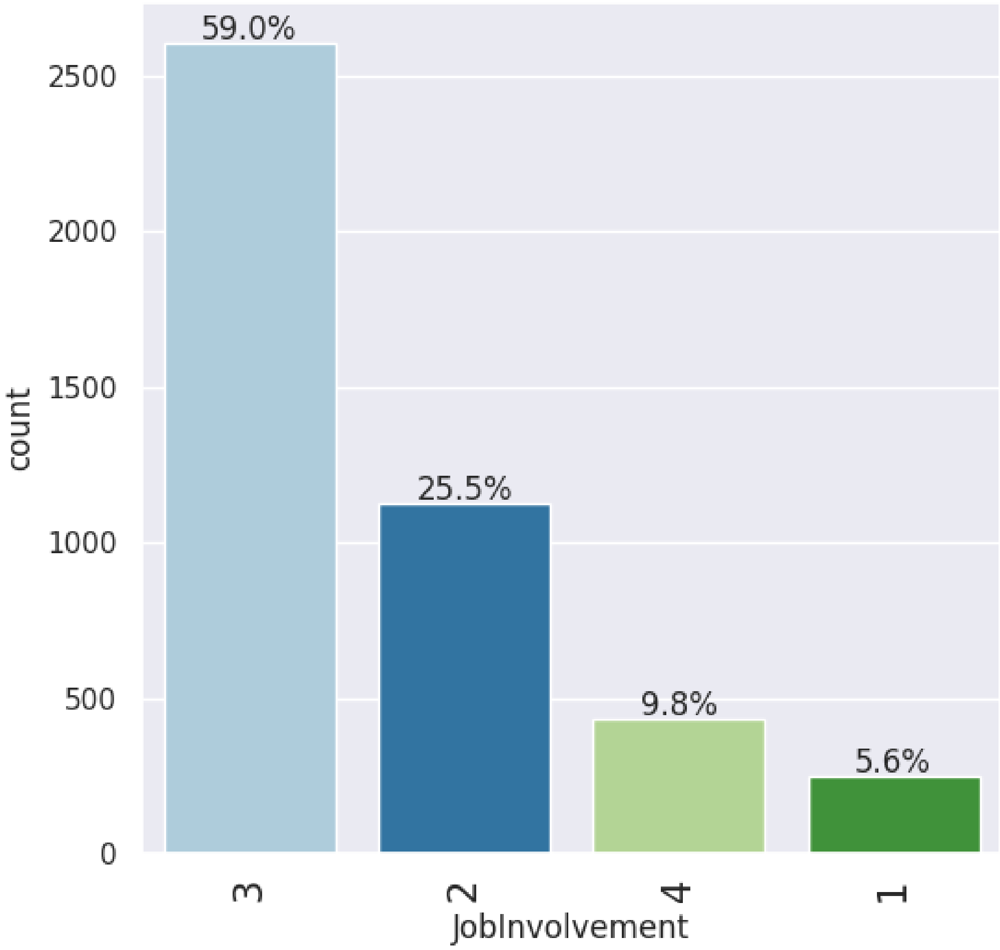
In [ ]:

```
labeled_barplot(df,'JobInvolvement',perc=True)
```

```
<ipython-input-3-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



Observations:

The high concentration of employees at job involvement level 3 suggests that there might be opportunities to enhance job engagement for a significant portion of the workforce.

PerformanceRating

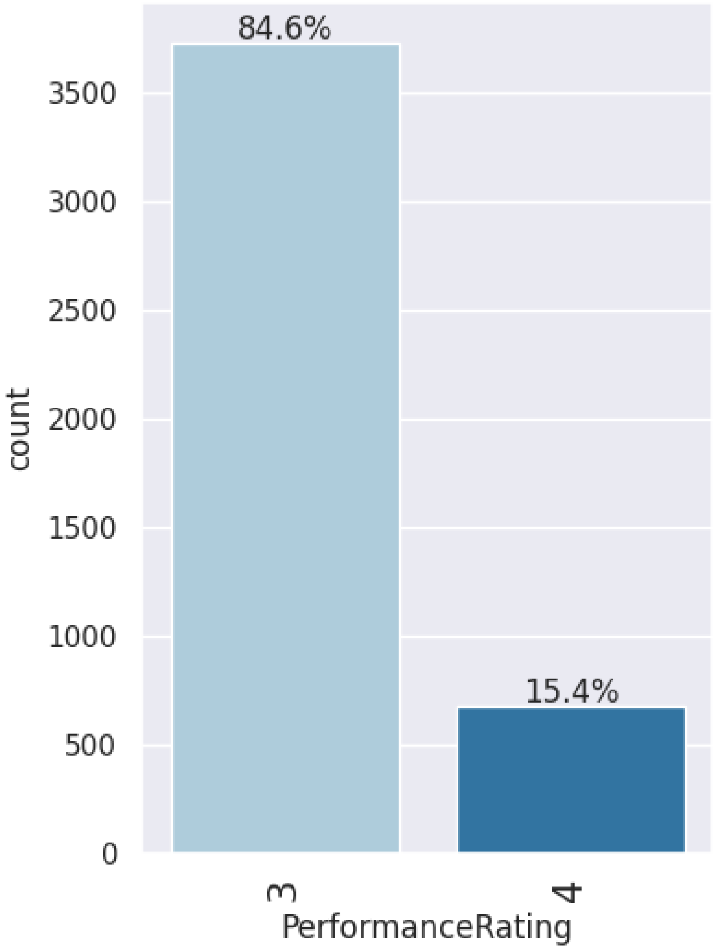
In [ ]:

```
labeled_barplot(df,'PerformanceRating',perc=True)
```

```
<ipython-input-3-0aaf8dec4340>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(
```



Observations:

The distribution is heavily skewed towards a Performance rating of 3.

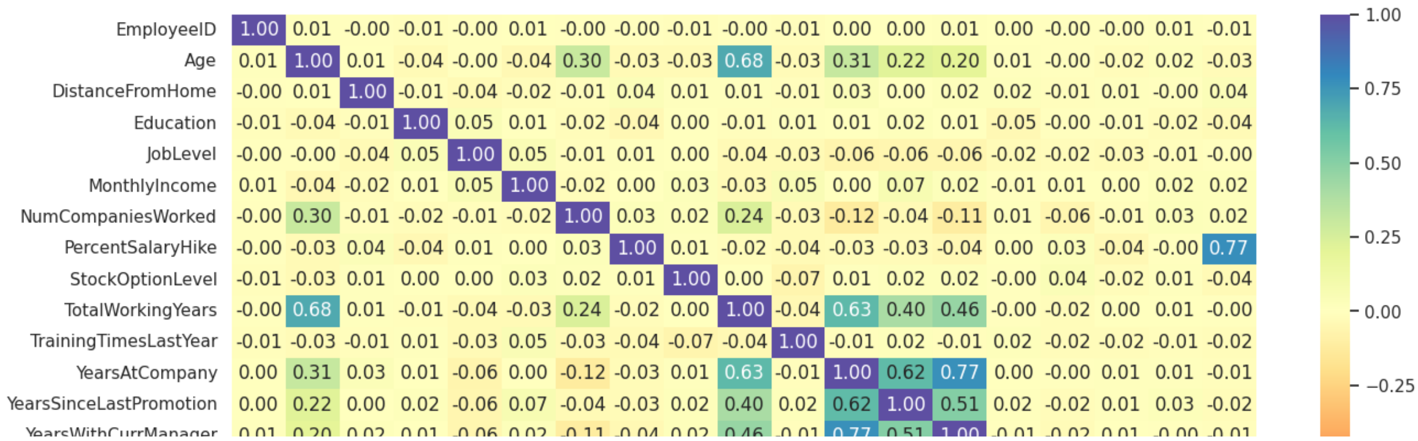
Bivariate Analysis

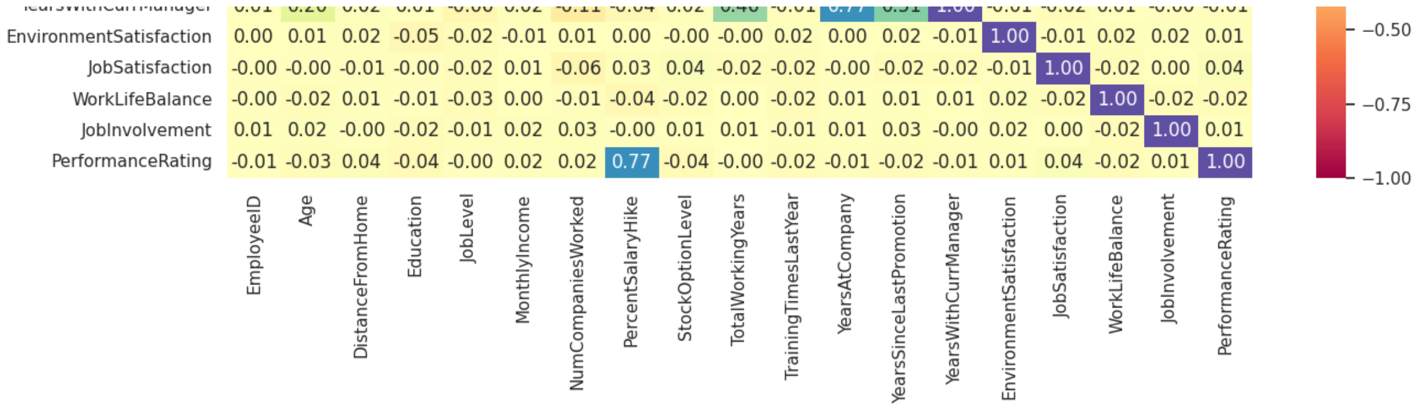
Checking correlation

In [ ]:

```
# Identifying numerical columns
num_cols=df.select_dtypes(include=np.number).columns.tolist()

# Dropping
num_cols.remove('EmployeeCount')
num_cols.remove('StandardHours')
# Plotting heatmap
plt.figure(figsize=(15, 7))
sns.heatmap(df[num_cols].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectral"
)
plt.show()
```





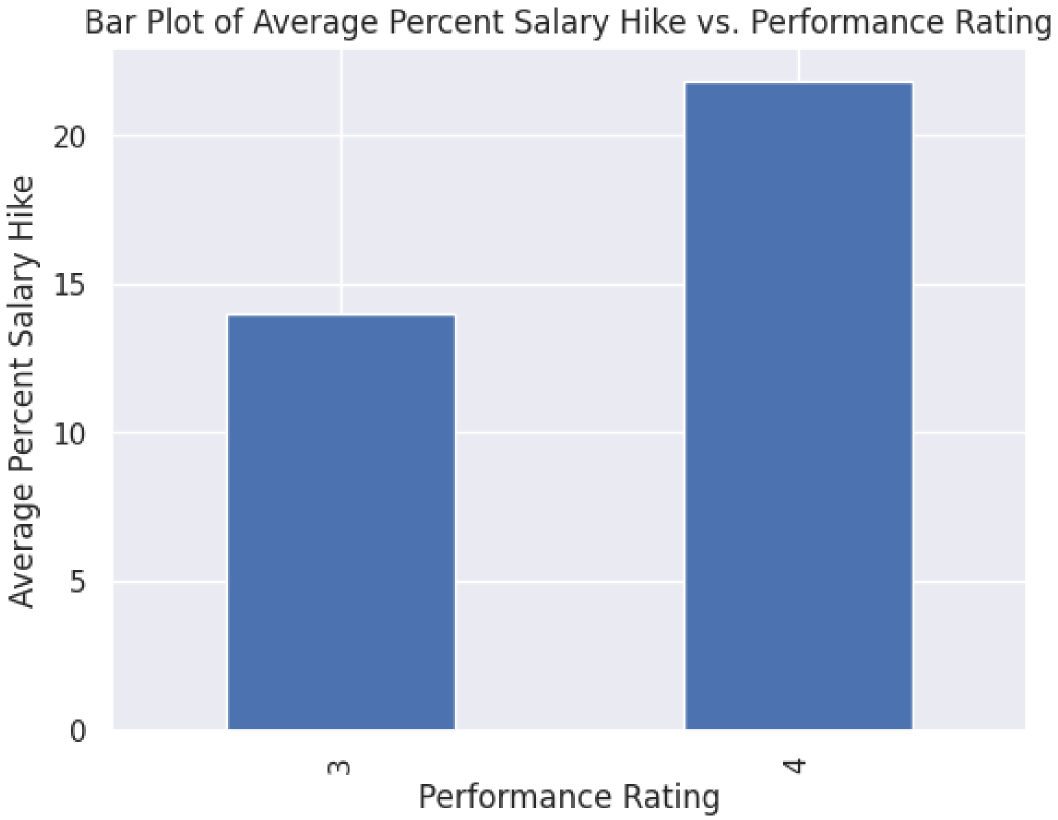
Observations:

PercentageSalaryHike Vs Performance Rating

In [ ]:

```
# Calculate the mean PercentSalaryHike for each PerformanceRating
mean_salary_hike = df.groupby('PerformanceRating')['PercentSalaryHike'].mean()

# Plot the bar plot
mean_salary_hike.plot(kind='bar')
plt.xlabel('Performance Rating')
plt.ylabel('Average Percent Salary Hike')
plt.title('Bar Plot of Average Percent Salary Hike vs. Performance Rating')
plt.show()
```



Observations

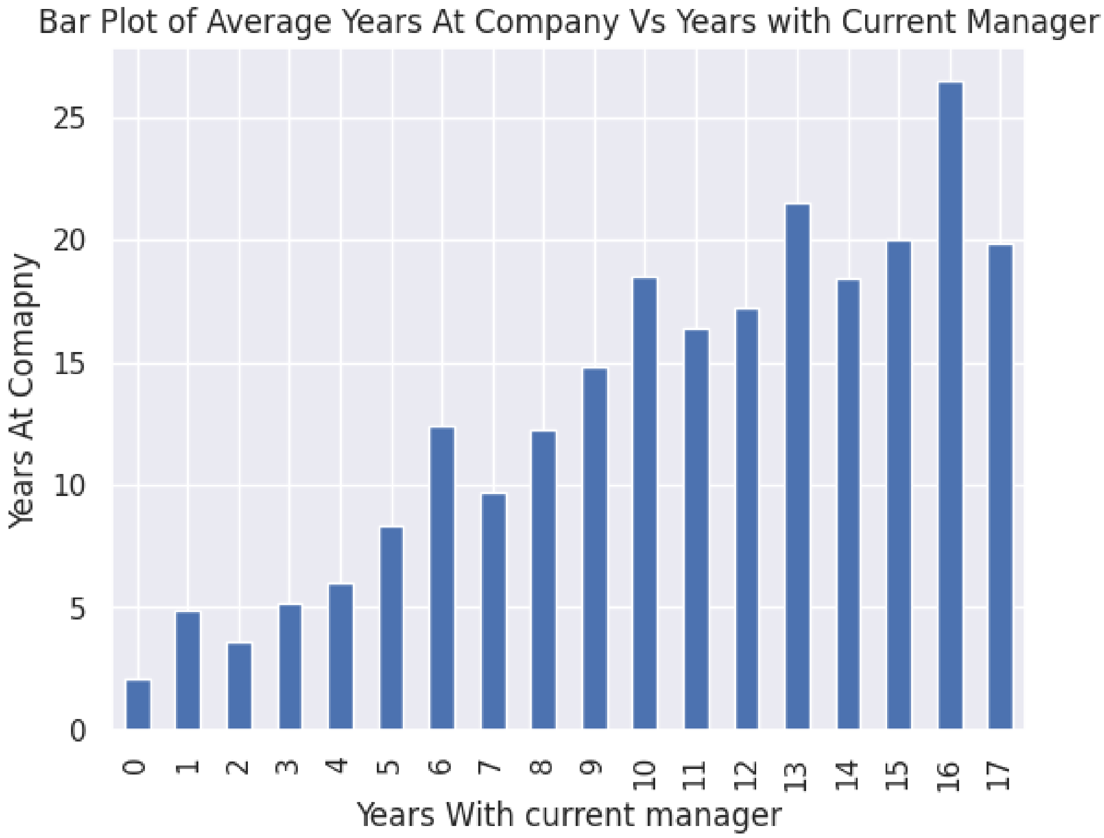
There is a positive correlation between performance rating and average percent salary hike. Employees with higher performance rating tends to receive larger salary increases.

YearsWithComapny Vs YearsWithCurrManager

In [ ]:

```
# Calculate the mean Years At Comapany for each Years with current manager
mean_years_comapny = df.groupby('YearsWithCurrManager')['YearsAtCompany'].mean()

# Plot the bar plot
mean_years_comapny.plot(kind='bar')
plt.xlabel('Years With current manager')
plt.ylabel('Years At Comapny')
plt.title('Bar Plot of Average Years At Company Vs Years with Current Manager')
plt.show()
```



Observations:

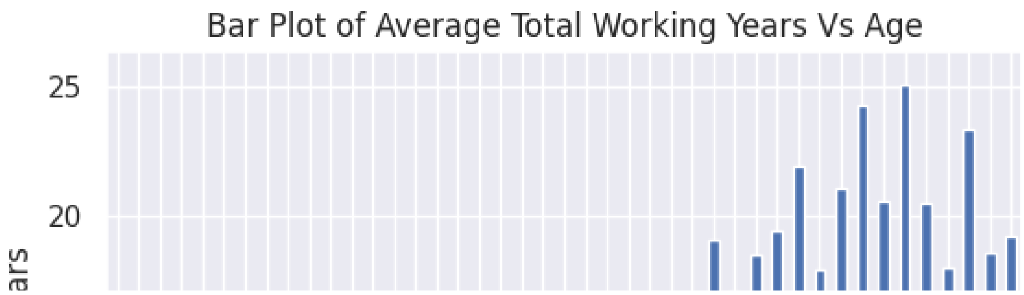
- ***There is a positive correlation between years with current manager and average years at the company. As the years with current manager increase, average years at company tends to increase***
- ***There is a increasing trend between the number of years of an employee has been with the current manager.***

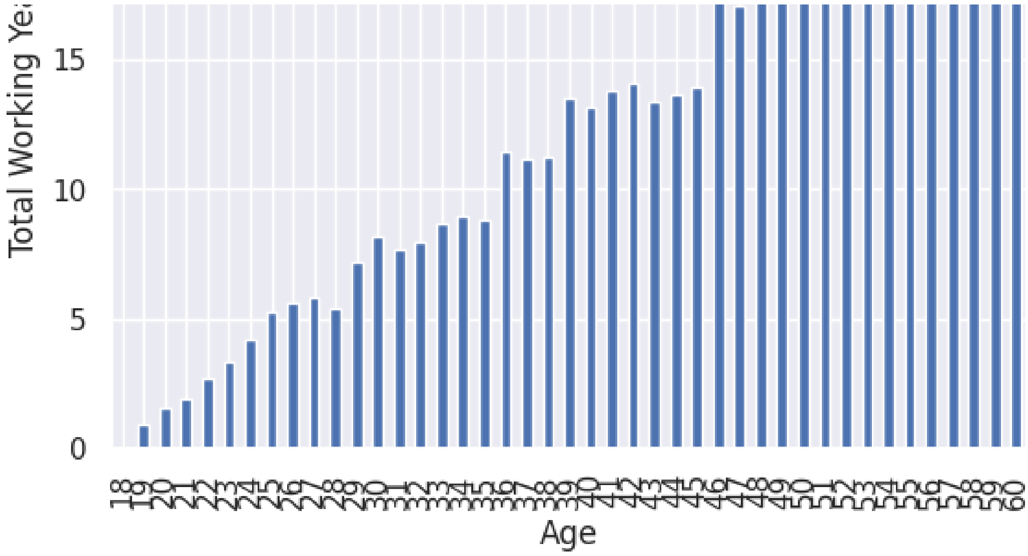
TotalWorkingHours Vs Age

In [ ]:

```
# Calculate the mean Total Working Years for each Age of employees
mean_years_comapny = df.groupby('Age')['TotalWorkingYears'].mean()

# Plot the bar plot
mean_years_comapny.plot(kind='bar')
plt.xlabel('Age')
plt.ylabel('Total Working Years')
plt.title('Bar Plot of Average Total Working Years Vs Age')
plt.show()
```





Observations:

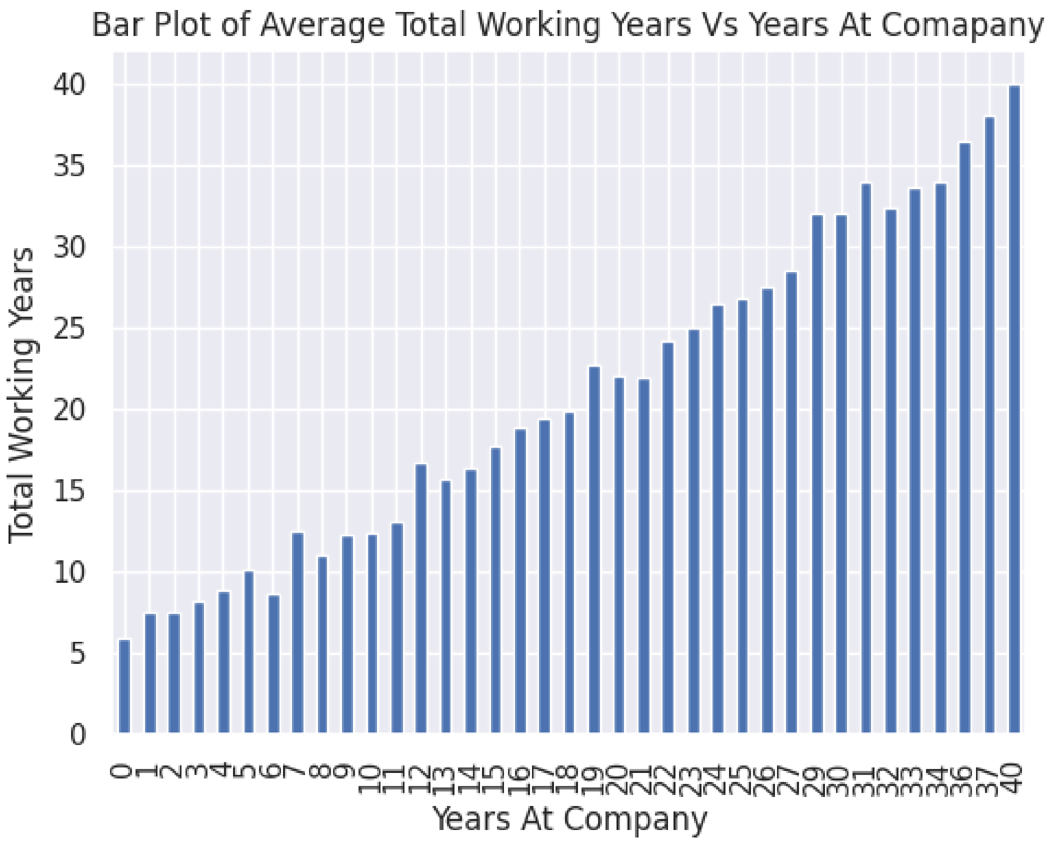
As the age increases, average total working years also increases which means older employees have more years of work experience.

Total Working Years Vs Years At Company

In [ ]:

```
# Calculate the mean Total Working Years for each Years At Company
mean_years_comapny = df.groupby('YearsAtCompany')['TotalWorkingYears'].mean()

# Plot the bar plot
mean_years_comapny.plot(kind='bar')
plt.xlabel('Years At Company')
plt.ylabel('Total Working Years')
plt.title('Bar Plot of Average Total Working Years Vs Years At Comapany')
plt.show()
```



Observations:

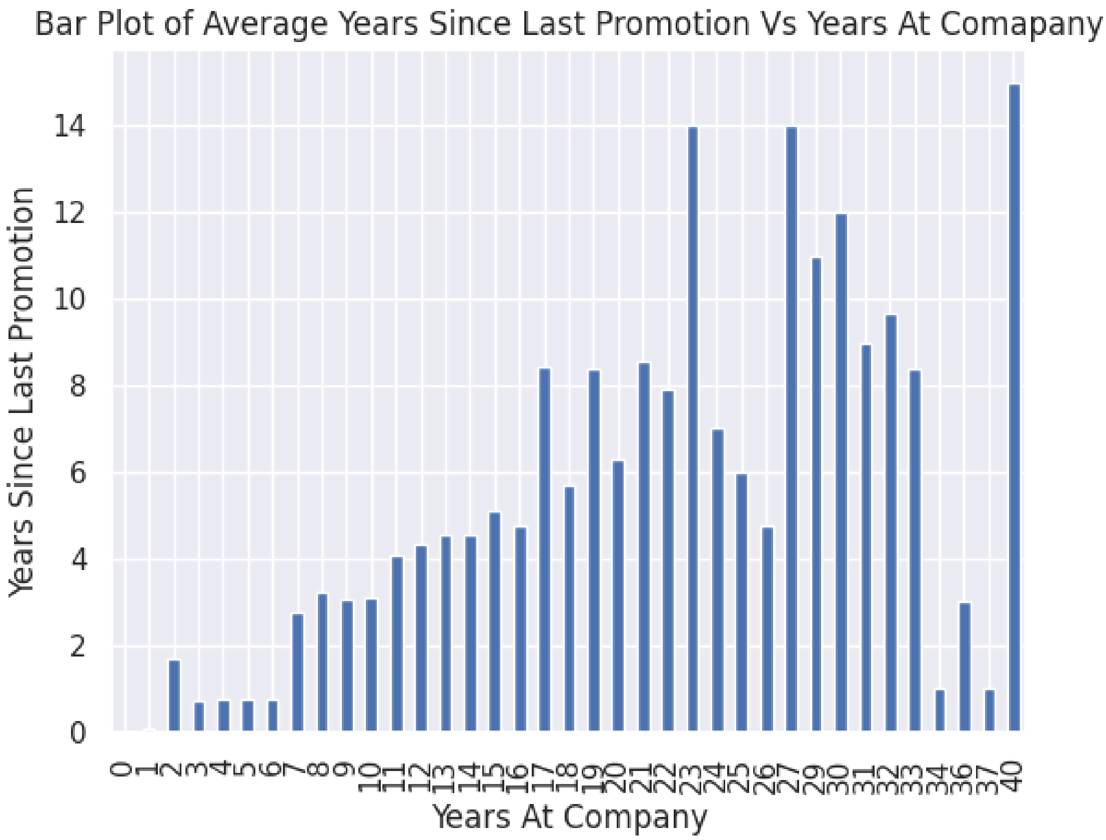
There is a positive correlation between the years of an employee spent at the company and their total working years which indicates as the years at the company increase, the total working years also increase.

YearsSinceLastPromotion Vs YearsAtCompany

In [ ]:

```
# Calculate the mean of Years since last promotion for each Years At Company
mean_years_comapny = df.groupby('YearsAtCompany')['YearsSinceLastPromotion'].mean()

# Plot the bar plot
mean_years_comapny.plot(kind='bar')
plt.xlabel('Years At Company')
plt.ylabel('Years Since Last Promotion')
plt.title('Bar Plot of Average Years Since Last Promotion Vs Years At Comapany')
plt.show()
```



Observations:

- ***The average years since the last promotion is relatively low for employees with less than 5 years at the company.***
- ***There are noticeable peaks and troughs in the average years since last promotion as the years at the comapany increase.***

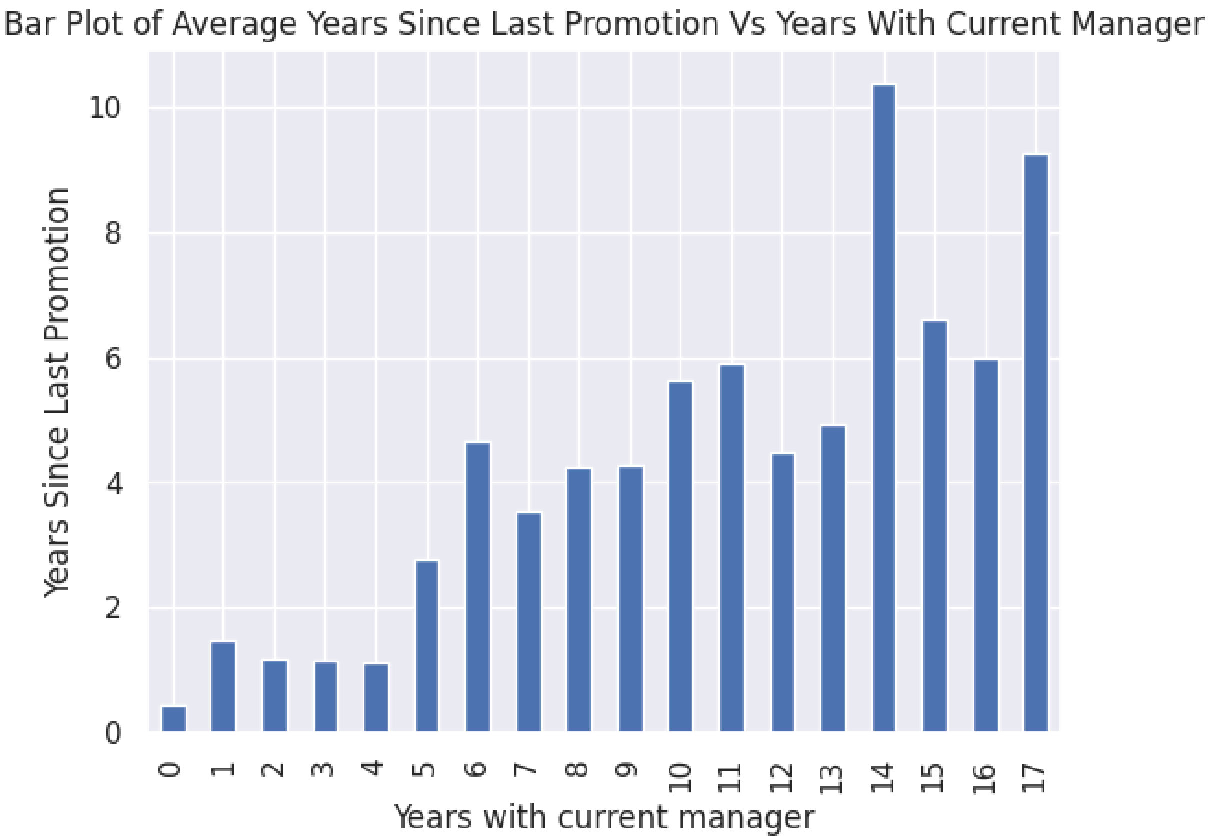
YearsSinceLastPromotion Vs YearsWithCurrManager

In [ ]:

```
# Calculate the mean of Years since last promotion for each Years with current manager
mean_years_comapny = df.groupby('YearsWithCurrManager')['YearsSinceLastPromotion'].mean(
)

# Plot the bar plot
mean_years_comapny.plot(kind='bar')
plt.xlabel('Years with current manager')
plt.ylabel('Years Since Last Promotion')
plt.title('Bar Plot of Average Years Since Last Promotion Vs Years With Current Manager')
```

```
plt.show()
```



Observations:

The average years since last promotion starts low for employees with a new manager but steadily inccreases as the number of years with the current manager grows.

Data Preprocessing

Missing Value Treatment

In [ ]:

```
# Creating a copy of the data to avoid changes to it
df1 = df.copy()
```

In [ ]:

```
df1.isnull().sum()
```

Out[ ]:

	0
EmployeeID	0
Age	0
Attrition	0
BusinessTravel	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EmployeeCount	0

	0
Gender	0
JobLevel	0
JobRole	0
MaritalStatus	0
MonthlyIncome	0
NumCompaniesWorked	19
Over18	0
PercentSalaryHike	0
StandardHours	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 [ ]:

```
# Filling missing values
df1['NumCompaniesWorked'].fillna(df1['NumCompaniesWorked'].median, inplace=True)
```

In [ ]:

```
df1['TotalWorkingYears'].fillna(df1['TotalWorkingYears'].median, inplace=True)
```

In [ ]:

```
df1['EnvironmentSatisfaction'].fillna(df1['EnvironmentSatisfaction'].mode()[0], inplace=
True)
```

In [ ]:

```
df1['JobSatisfaction'].fillna(df1['JobSatisfaction'].mode()[0], inplace=True)
```

In [ ]:

```
df1['WorkLifeBalance'].fillna(df1['WorkLifeBalance'].mode()[0], inplace=True)
```

In [ ]:

```
# Checking for missing values after filling values
df1.isnull().sum()
```

Out[ ]:

	0
EmployeeID	0
Age	0
Attrition	0
BusinessTravel	0

	0
BusinessTravel	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EmployeeCount	0
Gender	0
JobLevel	0
JobRole	0
MaritalStatus	0
MonthlyIncome	0
NumCompaniesWorked	0
Over18	0
PercentSalaryHike	0
StandardHours	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 [ ]:

```
# Encoding categorical variables
categorical_col = ['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gende
r', 'JobRole', 'MaritalStatus', 'Over18']

# Applying one-hot encoding on categorical variables
encoded_df1= pd.get_dummies(df1,columns=categorical_col)
print(encoded_df1.head())
```

```
   EmployeeID  Age  DistanceFromHome  Education  EmployeeCount  JobLevel  \
0           1   51                 6          2              1         1
1           2   31                10          1              1         1
2           3   32                17          4              1         4
3           4   38                 2          5              1         3
4           5   32                10          1              1         1

   MonthlyIncome NumCompaniesWorked  PercentSalaryHike  StandardHours  ...  \
0         131160                1.0                 11              8  ...
1          41890                0.0                 23              8  ...
2         193280                1.0                 15              8  ...
3          83210                3.0                 11              8  ...
4          23420                4.0                 12              8  ...

   JobRole_Manager JobRole_Manufacturing Director  JobRole_Research Director  \
0            False                          False                      False
1            False                          False                      False
2            False                          False                      False
```

```
3          False                        False                   False
4          False                        False                   False

   JobRole_Research Scientist  JobRole_Sales Executive  \
0                       False                    False   
1                        True                    False   
2                       False                     True   
3                       False                    False   
4                       False                     True   

   JobRole_Sales Representative  MaritalStatus_Divorced  \
0                         False                   False   
1                         False                   False   
2                         False                   False   
3                         False                   False   
4                         False                   False   

   MaritalStatus_Married  MaritalStatus_Single  Over18_Y  
0                   True                 False      True  
1                  False                  True      True  
2                   True                 False      True  
3                   True                 False      True  
4                  False                  True      True  

[5 rows x 49 columns]
```

In [ ]:

```
# Scaling numerical data
numerical_col = ['Age', 'DistanceFromHome', 'MonthlyIncome', 'PercentSalaryHike',
                 'TotalWorkingYears', 'YearsAtCompany', 'YearsSinceLastPromotion', '
YearsWithCurrManager']

# Ensure all numerical columns are of numeric type
encoded_df1[numerical_col] = encoded_df1[numerical_col].apply(pd.to_numeric, errors='coe
rce')
scaler = StandardScaler()
# Fit and transform the numerical columns
encoded_df1[numerical_col] = scaler.fit_transform(encoded_df1[numerical_col])
encoded_df1.head()
```

Out[ ]:

	EmployeeID	Age	DistanceFromHome	Education	EmployeeCount	JobLevel	MonthlyIncome	NumCompaniesWorked
0	1	1.541369	-0.393938	2	1	1	1.405136	1.0
1	2	-0.648668	0.099639	1	1	1	-0.491661	0.0
2	3	-0.539166	0.963398	4	1	4	2.725053	1.0
3	4	0.117845	-0.887515	5	1	3	0.386301	3.0
4	5	-0.539166	0.099639	1	1	1	-0.884109	4.0

5 rows × 49 columns

Reasons why employees are leaving the company and recommendations

Reasons

- ***The two biggest departments are sales and research and development (R&D). In these departments, department-specific problems like workload or management styles may be the cause of high attrition rates.***

- *A majority of workers are classified as 1 and 2 job levels. Increased attrition in these levels may indicate problems with entry-level jobs, like a lack of training or career advancement opportunities.*

- *Although many employees have worked for multiple companies, a significant percentage of the workforce (35.3%) has only worked for one. This suggests that workers may have a tendency to change jobs, which may be a factor in employee attrition.*
- *Most workers see an increase in salary of between 11% and 15%. Workers may become disappointed and attrition rates may rise if they feel these salaries are insufficient.*
- *A lot of employees have been with the company for just a 0-5 years. High attrition in this range may indicate problems with early career development, integration, or onboarding within the company.*
- *For 0 to 2 years, many employees have not received promotions. A lack of opportunities for career advancement may be a factor in employee dissatisfaction and attrition.*
- *The range of job satisfaction scores shows that a significant proportion of workers are not very satisfied (scores of 2 and 3). Attrition can be strongly influenced by lower job satisfaction.*
- *Most workers give their work-life balance a moderate rating (score of 3), with a smaller percentage giving it a low rating (score of 1 or 2). One known factor that can contribute to higher attrition rates is a poor work-life balance.*
- *A smaller percentage of workers have a performance rating of 4, while the majority have a rating of 3. Reduced performance ratings could be a sign of dissatisfaction with the evaluation procedure or a feeling of unacknowledgement, which would increase attrition.*

Recommendations

- *Provide employees with moderate education levels and particularly those in lower job levels to get clear career advancement paths and training opportunities.*
- *Based on performance and industry trends, raise salaries more frequently and by a greater percentage.Clearly explain the requirements for pay increases in order to control employee expectations.In addition to pay increases, offer other incentives and benefits.*
- *Evaluate employee satisfaction on a regular basis using feedback systems and surveys.Talk about common problems like workload, unclear duties, and job expectations.*
- *Provide remote work opportunities and flexible working hours. Offer tools and assistance for managing stress and affecting a balance between work and life.*
- *Make sure the system for evaluations of performance is unbiased, open, and acknowledges the efforts of the employees.Offer employees development plans and helpful feedback to help them perform better.To keep the performance evaluation criteria in accordance with business objectives, review and update them on a regular basis.*