In [1]:

```python
# Open the image
from IPython.display import Image
Image(filename="C:/Users/Rajesh/Downloads/employee-attrition-rate.jpg")
```

Out[1]:

What is employee attrition?

Employee attrition is defined as employees leaving their organizations for unpredictable or uncontrollable reasons. Many terms make up attrition, the most common being termination, resignation, planned or voluntary retirement, structural changes, long-term illness, layoffs.

Employee attrition is the gradual reduction in employee numbers. Employee attrition happens when the size of your workforce diminishesover time. This means that employees are leavining faster than they are hired.

Columns in dataset

EmployeeID - Numerical Categorical Data

Age - Numerical Discrete Data

Attrition - Text Categorical Data

BusinessTravel - Text Categorical Data

Department - Text Categorical Data

DistanceFromHome - Numerical Discrete Data

Education - Numerical Categorical Data

```
1 : Below College
2 : College
3 : Bachelor
4 : Master
5 : Doctor
```

EducationField - Text Categorical Data

EmployeeCount - Numerical Discrete Data

Gender - Text Categorical Data

JobLevel - Numerical Discrete Data

JobRole - Text Categorical Data

MaritalStatus - Text Categorical Data

MonthlyIncome - Numerical Discrete Data

NumCompaniesWorked - Numerical Discrete Data

Over18 - Text Categorical Data

PercentSalaryHike - Numerical Discrete Data

StandardHours - Numerical Discrete Data

StockOptionLevel - Numerical Categorical Data

TotalWorkingYears - Numerical Discrete Data

TrainingTimesLastYear - Numerical Discrete Data

YearsAtCompany - Numerical Discete Data

YearsSinceLastPromotion - Numerical Discete Data

YearsWithCurrManager - Numerical Discete Data

EnvironmentSatisfaction - Numerical Categorical Data

```
1 : Low
2 : Medium
3 : High
4 : Very High
```

JobSatisfaction - Numerical Categorical Data

```
1 : Low
2 : Medium
3 : High
4 : Very High
```

WorkLifeBalance - Numerical Categorical Data

```
1 : Bad
2 : Good
3 : Better
4 : Best
```

JobInvolvement - Numerical Categorical Data

```
1 : Low
2 : Medium
3 : High
4 : Very High
```

PerformanceRating - Numerical Categorical Data

```
1 : Low
2 : Good
3 : Excellent
4 : Outstanding
```

Import Library

Importing Libraries and Datasets The libraries used are :

Pandas: This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go. Seaborn/Matplotlib: For data visualization. Numpy: Numpy arrays are very fast and can perform large computations in a very short time.

In [2]:

```
# Import Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
# Read Attrition data .csv file and print first 5 records
df = pd.read_csv("Attrition data.csv")
df.head()
```

Out[3]:

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

5 rows × 29 columns

Check total number of columns

In [4]:
```
print(f"Total colums-",len(df.columns))
print(f"Total entries-",df.size)
```

```
Total colums- 29
Total entries- 127890
```

Total colums- 29 and total entries are 127890

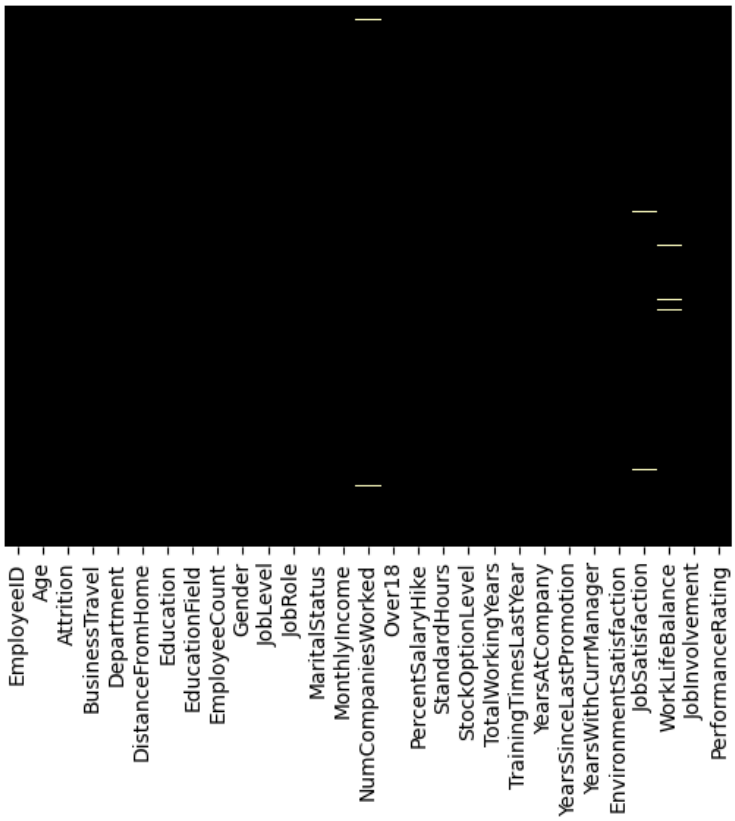
In [5]:
```
# df. shape method provides information about the number of rows and columns in a DataFrame quickly and easi
df.shape
```

Out[5]: (4410, 29)

Checking for missing values

In [6]:
```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='magma')
```

Out[6]: <Axes: >



Find null values

In [7]:

```
# Now, let’s have a look at whether this dataset has any null values or not
print(df.isnull().sum())
print()
print()
print(df.isna().sum())
```

```
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


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
```

In [8]:

```
print(df.isnull().sum().sum())
print()
print(df.isna().sum().sum())
```

```
111

111
```

Null values are very few, we can drop them without affecting data set

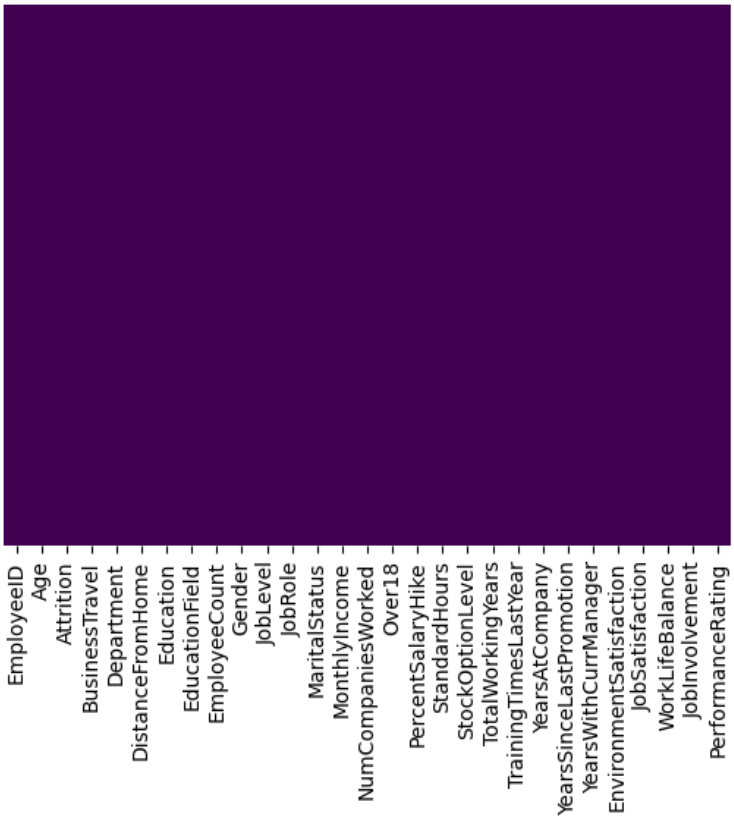
In [9]:
```
#Drop null values
df=df.dropna(axis=0)
```

In [10]:
```
# df. shape method provides information about the number of rows and columns in a DataFrame quickly and easi
df.shape
```

Out[10]: (4300, 29)

In [11]:
```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

Out[11]: <Axes: >



To print the information of the data we can use data.info() command.

In [12]:

```
# view the data types and missing values in each column
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4300 entries, 0 to 4408
Data columns (total 29 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   EmployeeID                4300 non-null   int64
 1   Age                       4300 non-null   int64
 2   Attrition                 4300 non-null   object
 3   BusinessTravel            4300 non-null   object
 4   Department                4300 non-null   object
 5   DistanceFromHome          4300 non-null   int64
 6   Education                 4300 non-null   int64
 7   EducationField            4300 non-null   object
 8   EmployeeCount             4300 non-null   int64
 9   Gender                    4300 non-null   object
 10  JobLevel                  4300 non-null   int64
 11  JobRole                   4300 non-null   object
 12  MaritalStatus             4300 non-null   object
 13  MonthlyIncome             4300 non-null   int64
 14  NumCompaniesWorked        4300 non-null   float64
 15  Over18                    4300 non-null   object
 16  PercentSalaryHike         4300 non-null   int64
 17  StandardHours             4300 non-null   int64
 18  StockOptionLevel          4300 non-null   int64
 19  TotalWorkingYears         4300 non-null   float64
 20  TrainingTimesLastYear     4300 non-null   int64
 21  YearsAtCompany            4300 non-null   int64
 22  YearsSinceLastPromotion   4300 non-null   int64
 23  YearsWithCurrManager      4300 non-null   int64
 24  EnvironmentSatisfaction   4300 non-null   float64
 25  JobSatisfaction           4300 non-null   float64
 26  WorkLifeBalance           4300 non-null   float64
 27  JobInvolvement            4300 non-null   int64
 28  PerformanceRating         4300 non-null   int64
dtypes: float64(5), int64(16), object(8)
memory usage: 1007.8+ KB
None
```

Let’s see the mean, count , minimum and maximum values of the data

In [13]:

```
#The describe() method returns description of the data in the DataFrame.
# view the summary statistics for each column
df.describe().style.background_gradient(cmap='copper')
```

Out[13]:

	EmployeeID	Age	DistanceFromHome	Education	EmployeeCount	JobLevel	MonthlyIncome	NumCompaniesWorke
count	4300.000000	4300.000000	4300.000000	4300.000000	4300.000000	4300.000000	4300.000000	4300.0000
mean	2211.695116	36.926977	9.197907	2.913256	1.000000	2.066977	65059.844186	2.6900
std	1272.117692	9.146517	8.097059	1.024774	0.000000	1.106633	47045.398914	2.4957
min	1.000000	18.000000	1.000000	1.000000	1.000000	1.000000	10090.000000	0.0000
25%	1110.750000	30.000000	2.000000	2.000000	1.000000	1.000000	29260.000000	1.0000
50%	2215.500000	36.000000	7.000000	3.000000	1.000000	2.000000	49360.000000	2.0000
75%	3314.250000	43.000000	14.000000	4.000000	1.000000	3.000000	83802.500000	4.0000
max	4409.000000	60.000000	29.000000	5.000000	1.000000	5.000000	199990.000000	9.0000

In [14]:

```python
# Now, let’s have a look at whether this dataset has any null values or not
print(df.isnull().sum())
print()
print()
print(df.isna().sum())
```

```
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         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


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         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
```

Data Visualization

In this section, we will try to understand and compare all columns.

Let’s count the columns with different datatypes like Category, Integer, Float.

In [15]:
```
df.dtypes
```

Out[15]:
```
EmployeeID                  int64
Age                         int64
Attrition                  object
BusinessTravel             object
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeCount               int64
Gender                     object
JobLevel                    int64
JobRole                    object
MaritalStatus              object
MonthlyIncome               int64
NumCompaniesWorked        float64
Over18                     object
PercentSalaryHike           int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears         float64
TrainingTimesLastYear       int64
YearsAtCompany              int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
EnvironmentSatisfaction   float64
JobSatisfaction           float64
WorkLifeBalance           float64
JobInvolvement              int64
PerformanceRating           int64
dtype: object
```

In [16]:
```
print(f"Number of categorical columns:", len(df.select_dtypes(include='object').columns))
print(f"Number of integer columns:", len(df.select_dtypes(include='int').columns))
print(f"Number of float columns:", len(df.select_dtypes(include='float').columns))
```

```
Number of categorical columns: 8
Number of integer columns: 16
Number of float columns: 5
```

In [17]:
```
# Exploring Department type
df.Department.value_counts()
```

Out[17]:
```
Research & Development    2807
Sales                     1307
Human Resources            186
Name: Department, dtype: int64
```

In [18]:
```
df.Gender.value_counts()
```

Out[18]:
```
Male      2571
Female    1729
Name: Gender, dtype: int64
```

Overall attrition rate

In [19]:
```
attrition_rate=((df[df['Attrition'] =='Yes'].count()[0])/df.shape[0])*100
attrition_rate
```

Out[19]:
```
16.162790697674417
```

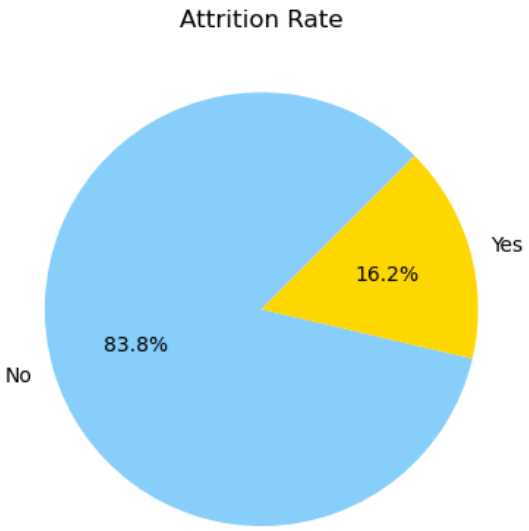
In [20]:
```
attrition_counts = df['Attrition'].value_counts()
attrition_counts
```

Out[20]:
```
No     3605
Yes     695
Name: Attrition, dtype: int64
```

Now we start The Visualization Part

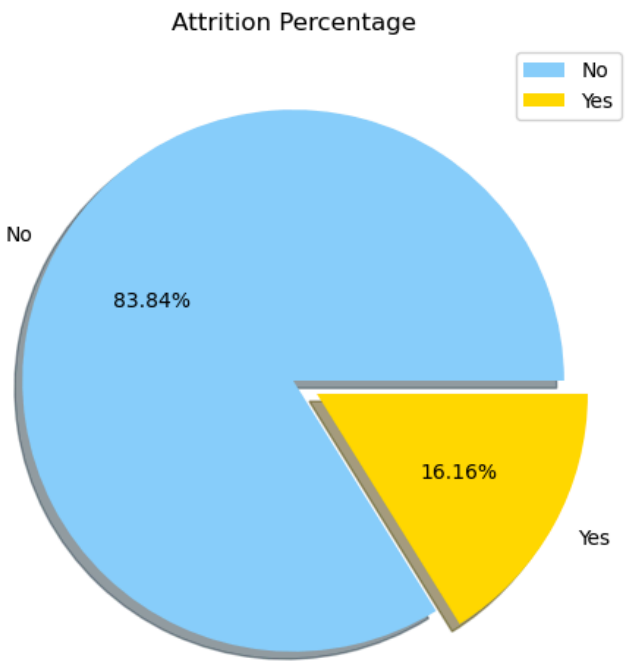
we find out wich factors affecting attrition

In [21]:
```
plt.pie(attrition_counts, labels=attrition_counts.index,autopct='%1.1f%%', startangle=45, colors=['lightskyb
plt.title('Attrition Rate')
plt.show()
```



In [22]:
```
labels = df['Attrition'].value_counts().index
size = df['Attrition'].value_counts().values
plt.figure(figsize = (6,6))
plt.pie(size,colors = ['lightskyblue', 'gold'],explode = (0, 0.1), labels = labels,shadow = True,autopct = "
plt.title('Attrition Percentage')
plt.axis('off')
plt.legend()
```

Out[22]: <matplotlib.legend.Legend at 0x250754f8280>

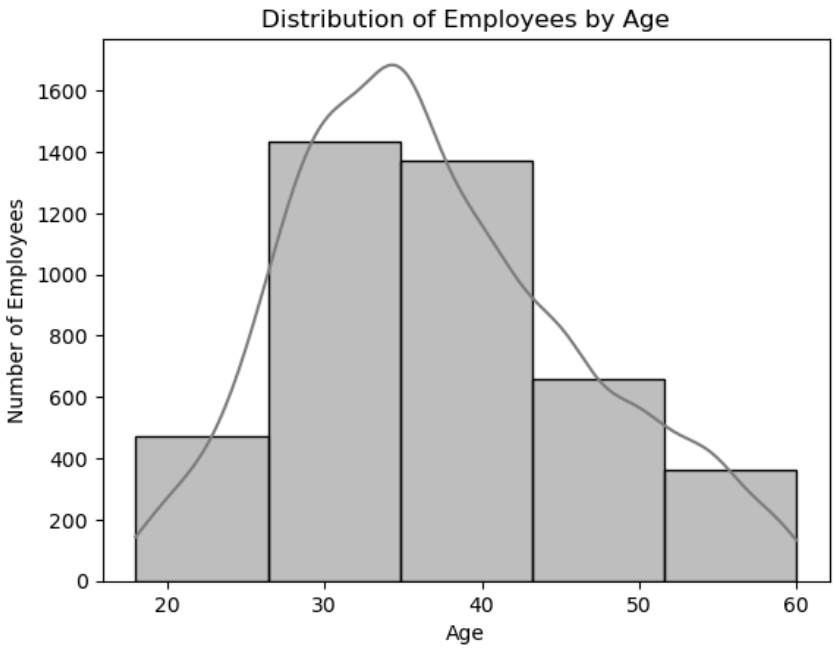


Positive class accounts for about 16.16% of data.So we can say that our dataset is imbalanced.

Age Factor

Age Diversity

In [23]:
```
sns.histplot(data= df, x= 'Age', bins= 5, color= 'grey', kde=True)
plt.ylabel('Number of Employees')
plt.title('Distribution of Employees by Age')
plt.show()
```



Impact of age on Attrition

In [24]:
```
bins= [10,30,50,np.inf]
values= ['Young', 'Adult', 'Senile']
df['Age_category']= pd.cut(df['Age'], bins=bins,labels=values)
```

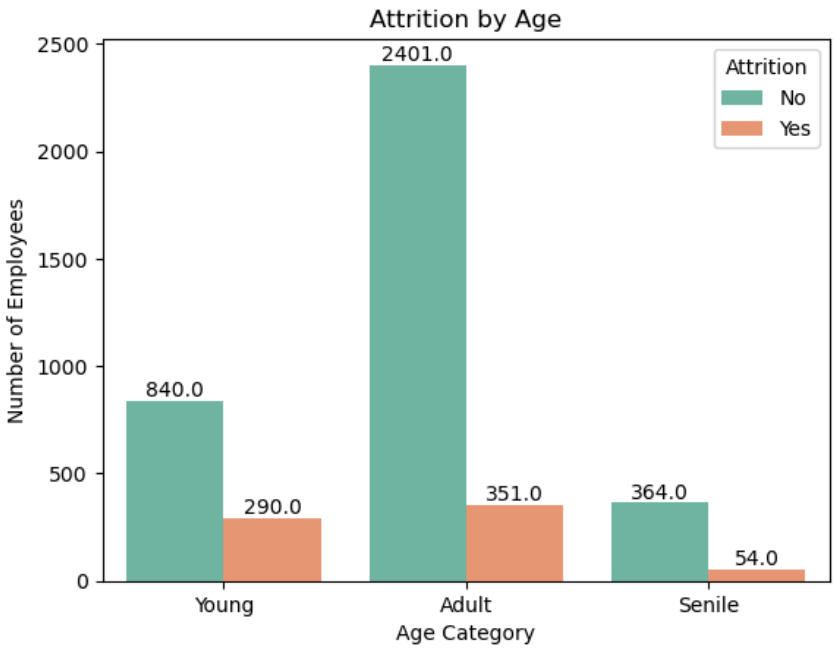
In [25]:
```
attrition_age=df.pivot_table(index='Age_category',columns='Attrition',values='EmployeeCount',aggfunc='count'
attrition_age
```

Out[25]:

Attrition	No	Yes
Age_category		
Young	840	290
Adult	2401	351
Senile	364	54

In [26]:

```
m=sns.countplot(data= df,x= 'Age_category', hue='Attrition',palette='Set2')
plt.xticks(rotation=0)
plt.xlabel('Age Category')
plt.ylabel('Number of Employees')
plt.title('Attrition by Age')

for p in m.patches:
    height = p.get_height()
    m.annotate(f'{height}', (p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom', fontsize=10)
plt.show()
```



Employee Distribution by department

In [27]:

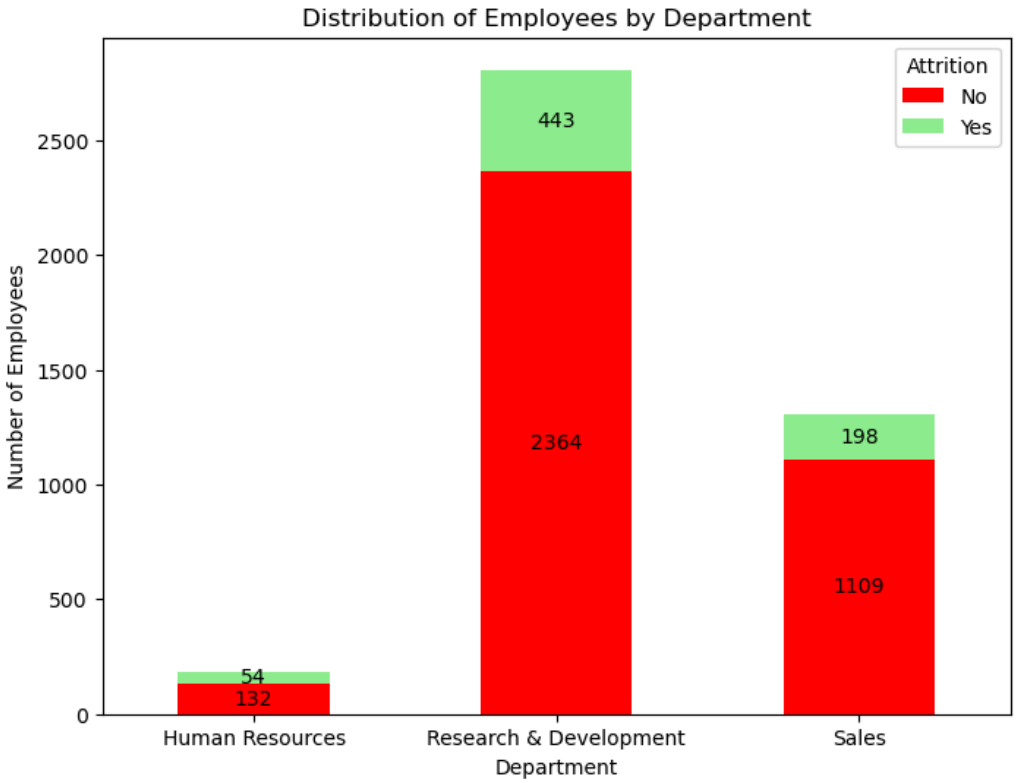
```
df['Department'].value_counts()
```

Out[27]:

```
Research & Development    2807
Sales                     1307
Human Resources            186
Name: Department, dtype: int64
```

In [28]:

```
attrition_by_department = df.groupby(['Department', 'Attrition']).size().unstack()

# Create a stacked bar chart
ax = attrition_by_department.plot(kind='bar', stacked=True, figsize=(8, 6),color=['red','lightgreen'])
plt.legend(title='Attrition', loc='upper right')
plt.xticks(rotation=0)
plt.xlabel('Department')
plt.ylabel('Number of Employees')
plt.title('Distribution of Employees by Department')
m.bar_label(m.containers[0], fontsize=10)


for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{int(height)}', (x + width/2, y + height/2), ha='center', va='center')

plt.show()
```



In [29]:

```
#Percentage attrition departmentwise
department_groups = df.groupby('Department')
total_employees_by_department = department_groups.size().reset_index(name='TotalEmployees')
attrition_count_by_department = department_groups['Attrition'].apply(lambda x: (x == 'Yes').sum()).reset_ind
attrition_percentage_by_department = pd.merge(total_employees_by_department, attrition_count_by_department,
attrition_percentage_by_department['AttritionPercentage'] = (attrition_percentage_by_department['AttritionCo
print(attrition_percentage_by_department)
```

```
               Department  TotalEmployees  AttritionCount  AttritionPercentage
0         Human Resources             186              54            29.032258
1  Research & Development            2807             443            15.781974
2                   Sales            1307             198            15.149197
```

Conclusion:

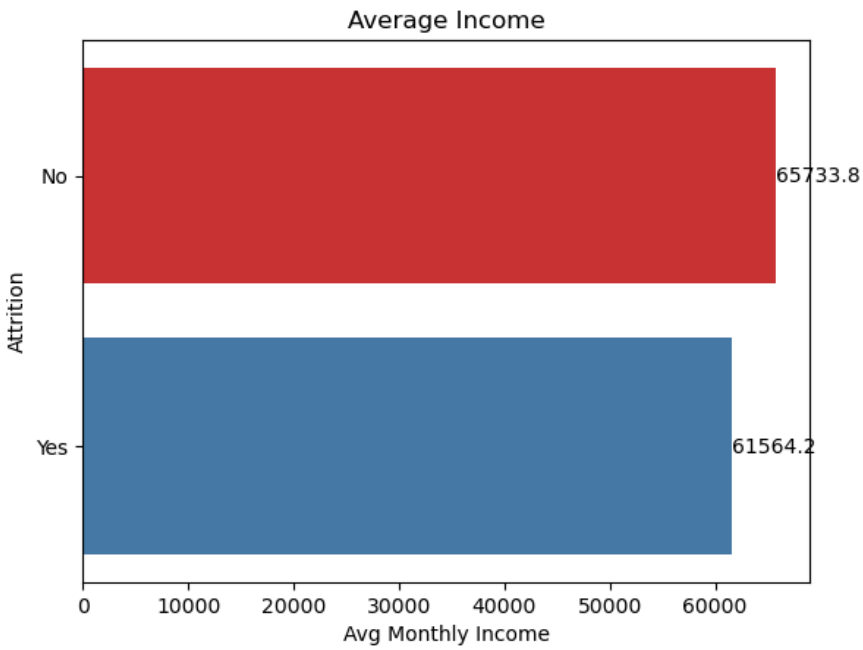
There are varying levels of attrition across departments, with Sales and Human Resources experiencing higher attrition rates compared to Research & Development. The Sales department experiences a higher attrition rate, with approximately 20.63% of employees leaving.The Research & Development department has a relatively lower attrition rate, with approximately 13.84% of employees leaving. This department seems to have better employee retention compared to Human Resources and Sales.

Effect of income on attrition

In [30]:
```
avg_income=df.groupby('Attrition')['MonthlyIncome'].mean()
avg_income
```

Out[30]:
```
Attrition
No     65733.758669
Yes    61564.215827
Name: MonthlyIncome, dtype: float64
```

In [31]:
```
p=sns.barplot(data=df, y= avg_income.index, x=avg_income.values,errorbar=None,palette="Set1")
p.bar_label(p.containers[0], fontsize=10)
plt.title('Average Income')
plt.xlabel('Avg Monthly Income')
plt.show()
```



Explore data for Education Field distribution

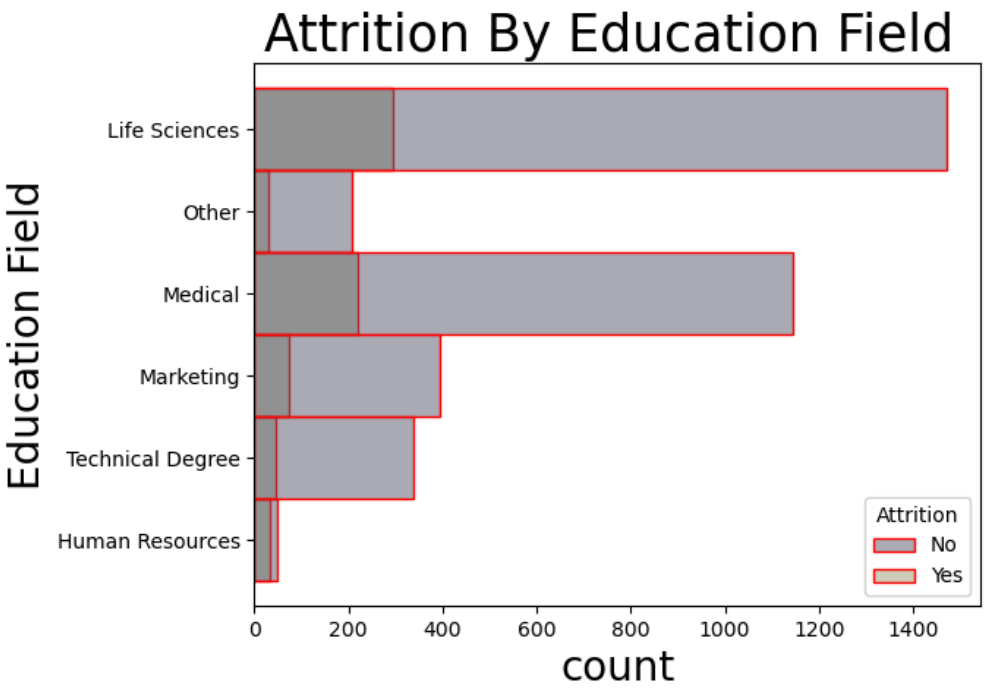
Education levels of employees

In [32]:
```
edu_dist=df['EducationField'].value_counts()
edu_dist
```

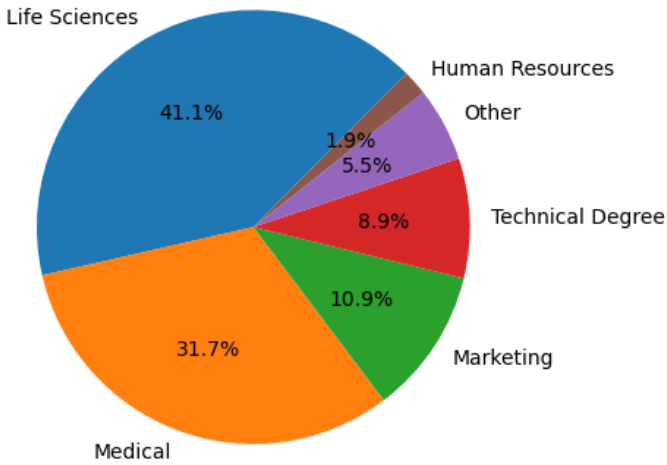
Out[32]:
```
Life Sciences       1766
Medical             1364
Marketing            469
Technical Degree     384
Other                237
Human Resources       80
Name: EducationField, dtype: int64
```

In [33]:
```
sns.histplot(hue='Attrition',y='EducationField',data=df,edgecolor='red',palette="cividis")
plt.ylabel('Education Field',fontsize=20)
plt.xlabel('count',fontsize=20)
plt.title('Attrition By Education Field ',fontsize=25)
plt.show
```

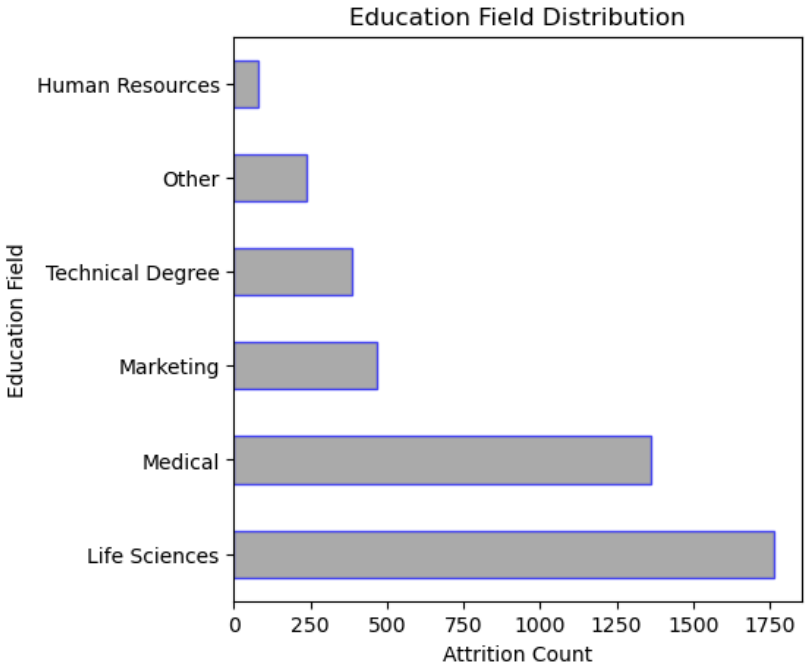
Out[33]: <function matplotlib.pyplot.show(close=None, block=None)>



In [34]:
```
plt.pie(edu_dist, labels=edu_dist.index,autopct='%1.1f%%', startangle=45)
plt.show()
```



In [35]:
```
plt.figure(figsize=(5,5))
df.EducationField.value_counts().plot(kind='barh',edgecolor='blue',color='grey',alpha=.65)
plt.title("Education Field Distribution")
plt.ylabel("Education Field")
plt.xlabel("Attrition Count")
plt.show()
```



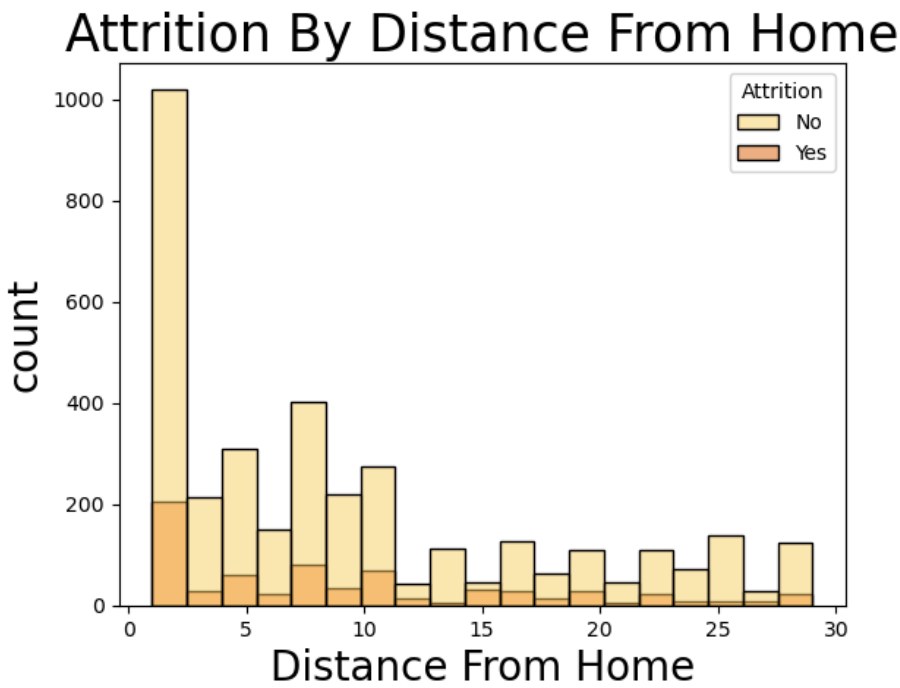
Location Proximity

In [36]:
```
avg_distance=df.groupby('Attrition')['DistanceFromHome'].mean()
avg_distance
```

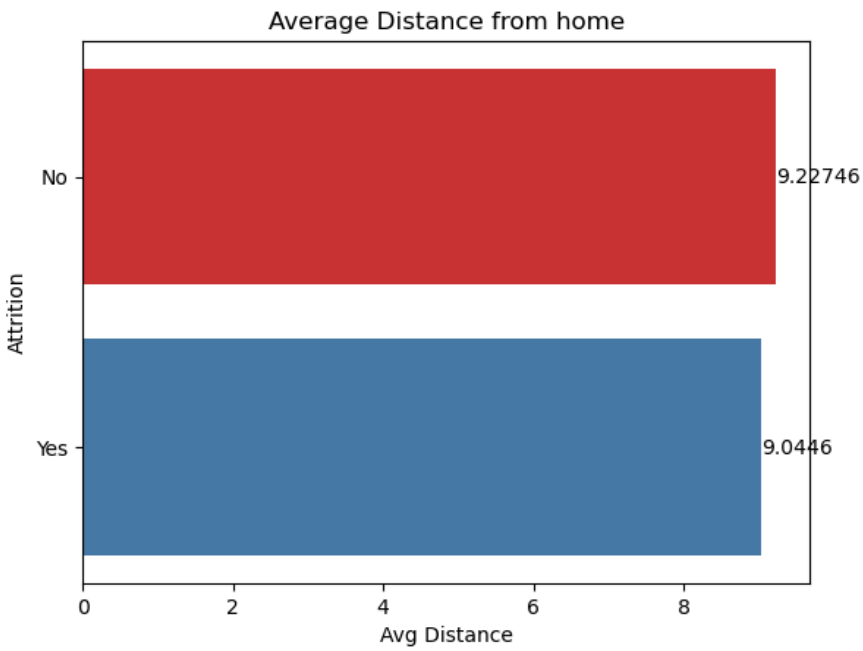
Out[36]:
```
Attrition
No     9.227462
Yes    9.044604
Name: DistanceFromHome, dtype: float64
```

In [37]:
```
sns.histplot(hue='Attrition', data=df,x='DistanceFromHome',palette="YlOrBr")
plt.xlabel('Distance From Home', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.title('Attrition By Distance From Home',fontsize=25)
plt.show
```

Out[37]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [38]:
```
p=sns.barplot(data=df, y= avg_distance.index, x=avg_distance.values,errorbar=None,palette="Set1")
p.bar_label(p.containers[0], fontsize=10)
plt.title('Average Distance from home')
plt.xlabel('Avg Distance')
plt.show()
```

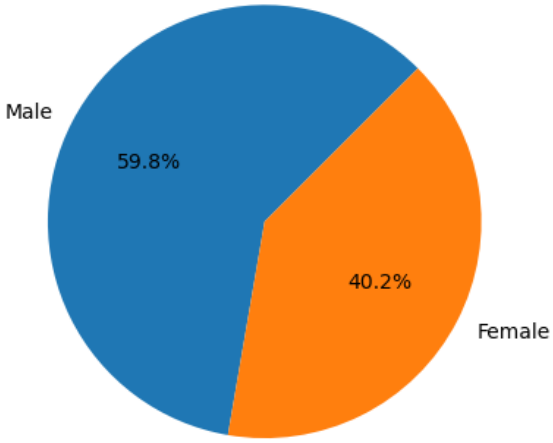


Gender Diversity and its impact on attrition

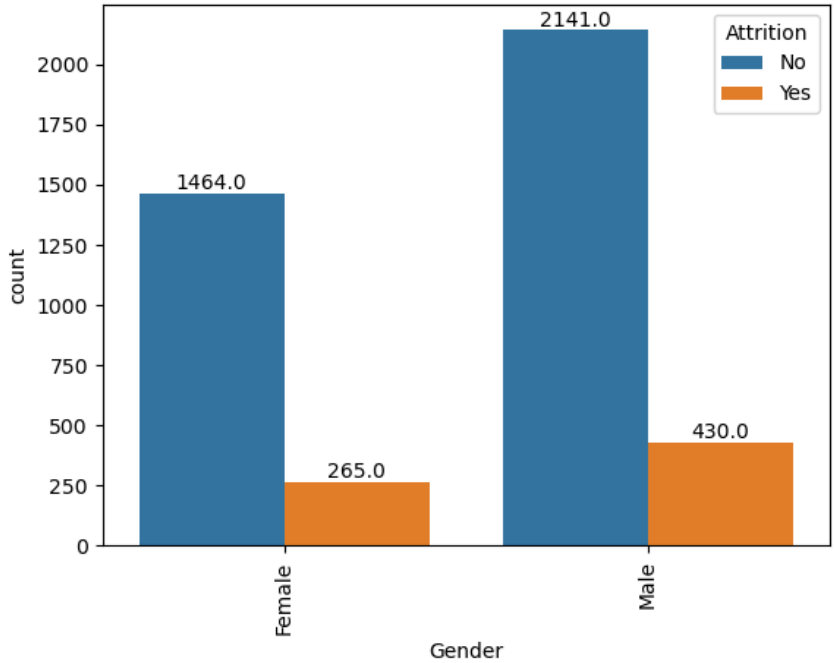
In [39]:
```
gend_dist=df['Gender'].value_counts()
gend_dist
```

Out[39]:
```
Male      2571
Female    1729
Name: Gender, dtype: int64
```

In [40]:
```
plt.pie(gend_dist, labels=gend_dist.index,autopct='%1.1f%%', startangle=45)
plt.show()
```



In [41]:
```
m=sns.countplot(data=df, x='Gender', hue='Attrition')
plt.xticks(rotation=90)
for p in m.patches:
    height = p.get_height()
    m.annotate(f'{height}', (p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom', fontsize=10)
plt.show()
```



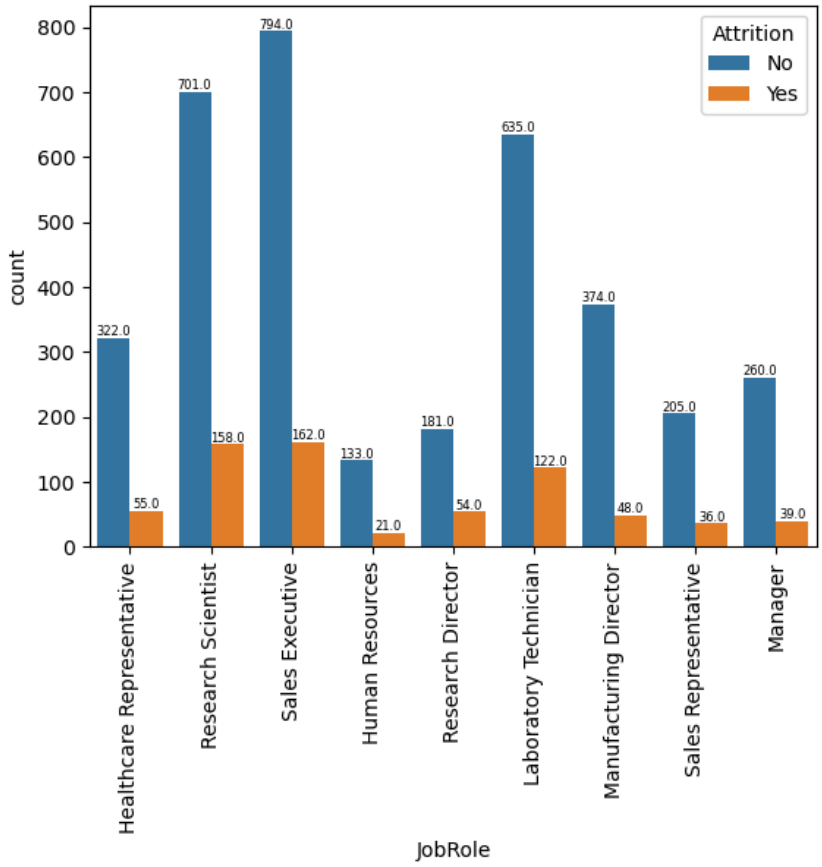
In [42]:
```
#Percentage attrition by gender

gender_groups = df.groupby('Gender')
total_employees_by_gender = gender_groups.size().reset_index(name='TotalEmployees')
attrition_count_by_gender = gender_groups['Attrition'].apply(lambda x: (x == 'Yes').sum()).reset_index(name=
attrition_percentage_by_gender = pd.merge(total_employees_by_gender, attrition_count_by_gender, on='Gender')
attrition_percentage_by_gender['AttritionPercentage'] = (attrition_percentage_by_gender['AttritionCount'] /
print(attrition_percentage_by_gender)
```

```
   Gender  TotalEmployees  AttritionCount  AttritionPercentage
0  Female            1729             265            15.326778
1    Male            2571             430            16.725010
```

Job Profile and its impact

In [43]:

```
jp=sns.countplot(data=df, x='JobRole', hue='Attrition')
plt.xticks(rotation=90)
for p in jp.patches:
    height = p.get_height()
    jp.annotate(f'{height}', (p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom', fontsize=6)
plt.show()
```



In [44]:

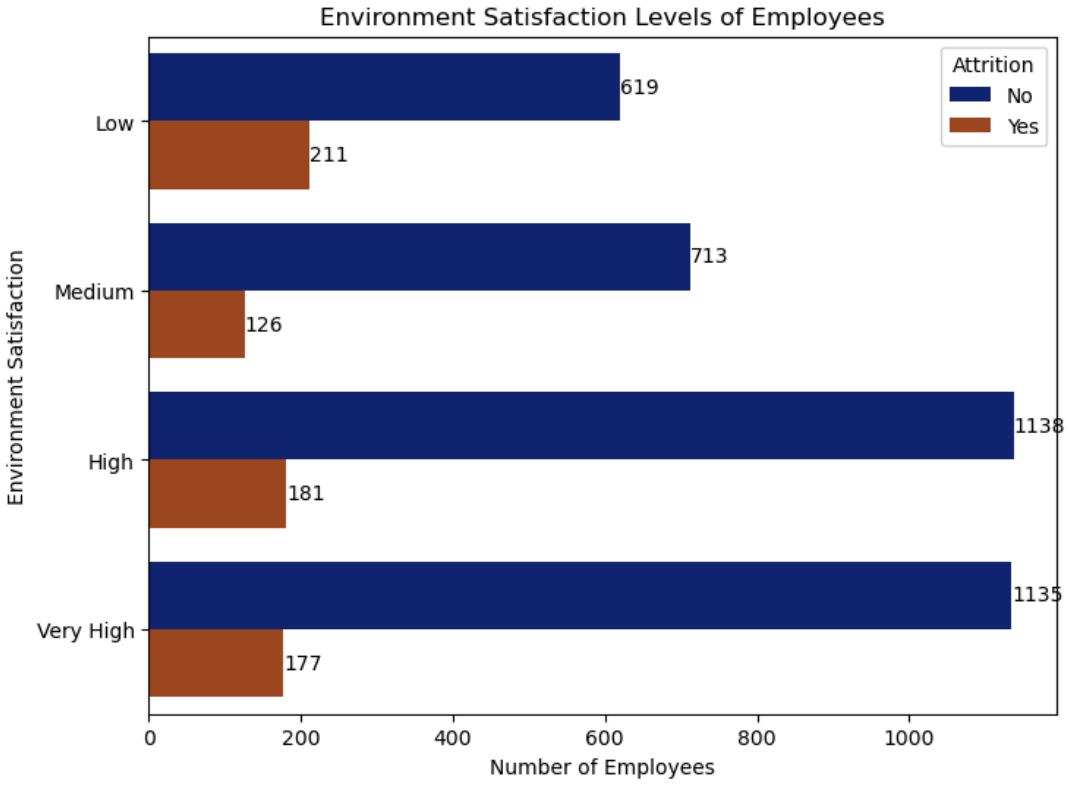
```
job_profile_groups = df.groupby('JobRole')
total_employees_by_jobrole = job_profile_groups.size().reset_index(name='TotalEmployees')
attrition_count_by_jobrole = job_profile_groups['Attrition'].apply(lambda x: (x == 'Yes').sum()).reset_index
attrition_percentage_by_jobrole = pd.merge(total_employees_by_jobrole, attrition_count_by_jobrole, on='JobRo
attrition_percentage_by_jobrole['AttritionPercentage'] = (attrition_percentage_by_jobrole['AttritionCount']
print(attrition_percentage_by_jobrole)
```

```
                     JobRole  TotalEmployees  AttritionCount  \
0  Healthcare Representative             377              55
1            Human Resources             154              21
2      Laboratory Technician             757             122
3                    Manager             299              39
4     Manufacturing Director             422              48
5          Research Director             235              54
6         Research Scientist             859             158
7            Sales Executive             956             162
8       Sales Representative             241              36

   AttritionPercentage
0            14.588859
1            13.636364
2            16.116248
3            13.043478
4            11.374408
5            22.978723
6            18.393481
7            16.945607
8            14.937759
```

Environment Satisfaction Rating

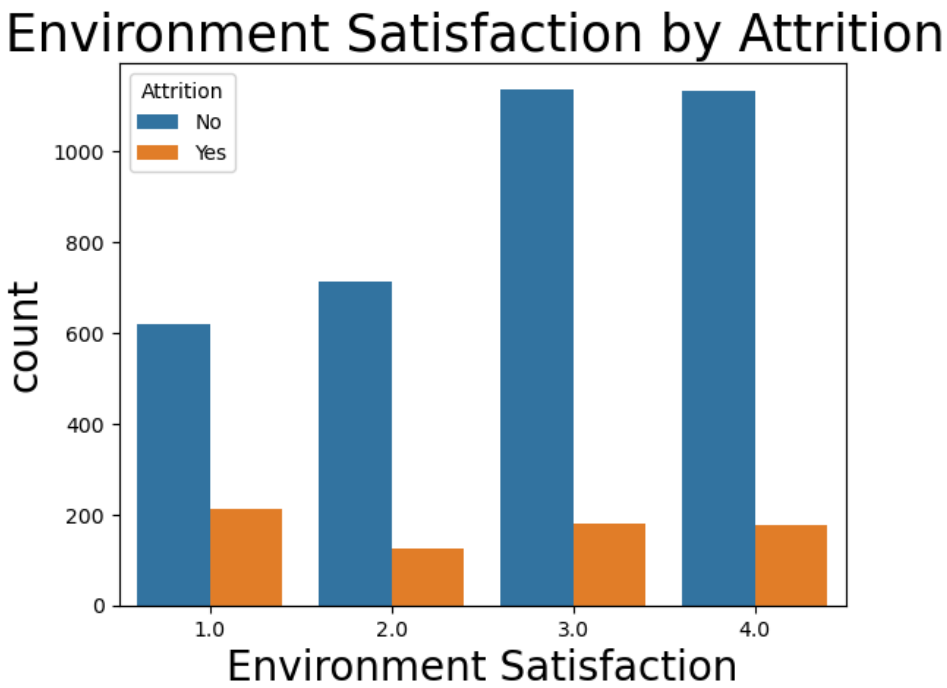
In [45]:

```
plt.figure(figsize=(8, 6))
env_sat = sns.countplot(data=df, y='EnvironmentSatisfaction', hue='Attrition',palette='dark')
plt.title('Environment Satisfaction Levels of Employees')
plt.xlabel('Number of Employees')
plt.ylabel('Environment Satisfaction')

for p in env_sat.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_x() + width, p.get_y() + height/2
    plt.annotate(f'{int(width)}', (x, y), ha='left', va='center', fontsize=10)

env_sat.set_yticklabels(['Low', 'Medium', 'High', 'Very High'])

plt.show()
```



In [46]:
```
sns.countplot(hue='Attrition', x='EnvironmentSatisfaction', data=df)
plt.xlabel('Environment Satisfaction', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.title( 'Environment Satisfaction by Attrition ',fontsize=25)
plt.show
```

Out[46]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



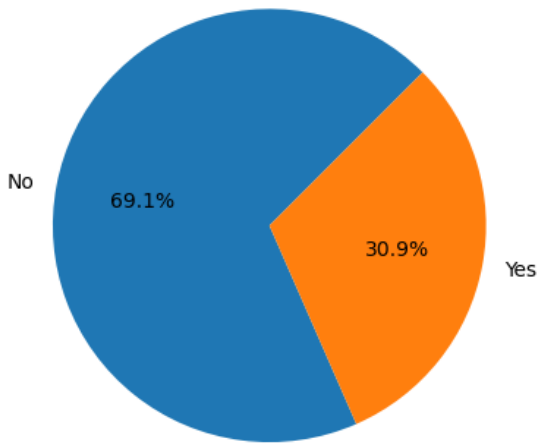
Work-Life Balance

In [47]:
```
df1= df[df['WorkLifeBalance']==1]
```

In [48]:
```
att_cnt=df1['Attrition'].value_counts()
att_cnt
```

Out[48]:
```
No     163
Yes     73
Name: Attrition, dtype: int64
```

In [49]:
```
plt.pie(att_cnt, labels=att_cnt.index,autopct='%1.1f%%', startangle=45)
plt.show()
```



In [50]:

```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='WorkLifeBalance', palette='dark')
plt.title('Distribution of Work-Life Balance')
plt.xlabel('Work-Life Balance')
plt.ylabel('Count')
plt.show()
```

Business Travel

In [51]:

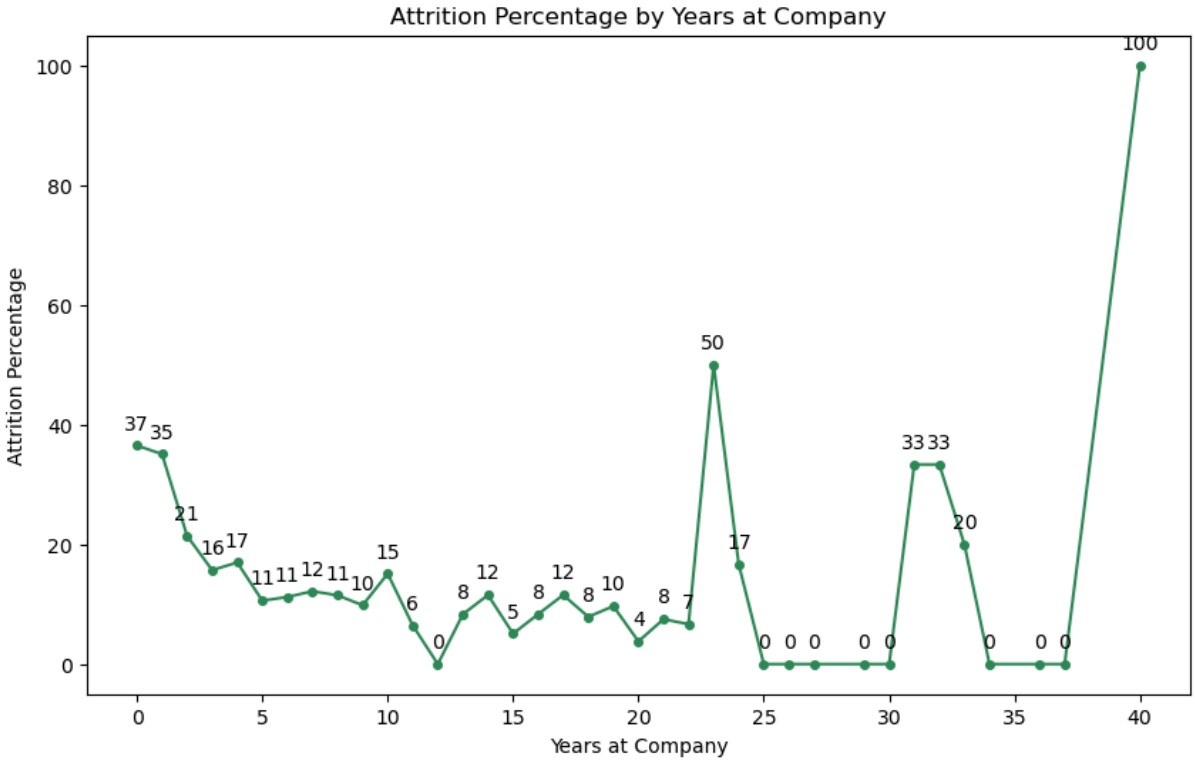
```
pivot_table = pd.pivot_table(df, values='Attrition', index='BusinessTravel', aggfunc=['count', lambda x: (x
pivot_table.columns = ['TotalEmployees', 'AttritionCount']
pivot_table['AttritionPercentage'] = (pivot_table['AttritionCount'] / pivot_table['TotalEmployees']) * 100
pivot_table.reset_index(inplace=True)

print(pivot_table)
```

```
      BusinessTravel  TotalEmployees  AttritionCount  AttritionPercentage
0         Non-Travel             440              36             8.181818
1  Travel_Frequently             809             199            24.598269
2      Travel_Rarely            3051             460            15.077024
```

Number of years spent in the company

In [52]:

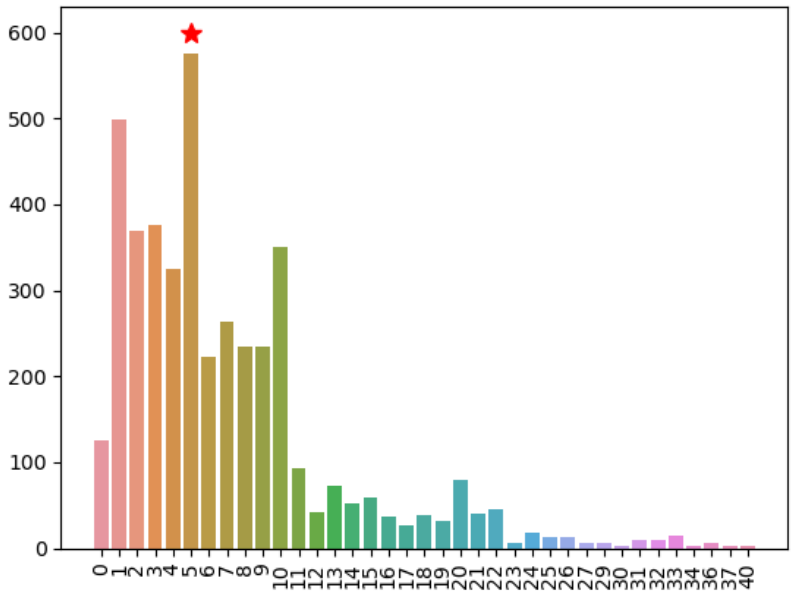
```
attrition_by_years = df.groupby('YearsAtCompany')['Attrition'].apply(lambda x: (x == 'Yes').mean() * 100)

# Create a line chart
plt.figure(figsize=(10, 6))
plt.plot(attrition_by_years.index, attrition_by_years.values, marker='o', linestyle='-',markersize=4,color='
plt.title('Attrition Percentage by Years at Company')
plt.xlabel('Years at Company')
plt.ylabel('Attrition Percentage')

label_offset = 2  # Adjust this value to control label height
for x, y in zip(attrition_by_years.index, attrition_by_years.values):
    label = f'{int(round(y))}'  # Round the percentage and convert it to an integer
    plt.text(x, y + label_offset, label, ha='center', va='bottom', fontsize=10)

plt.show()
```



In [53]:
```
x = df['YearsAtCompany'].value_counts()
x
```

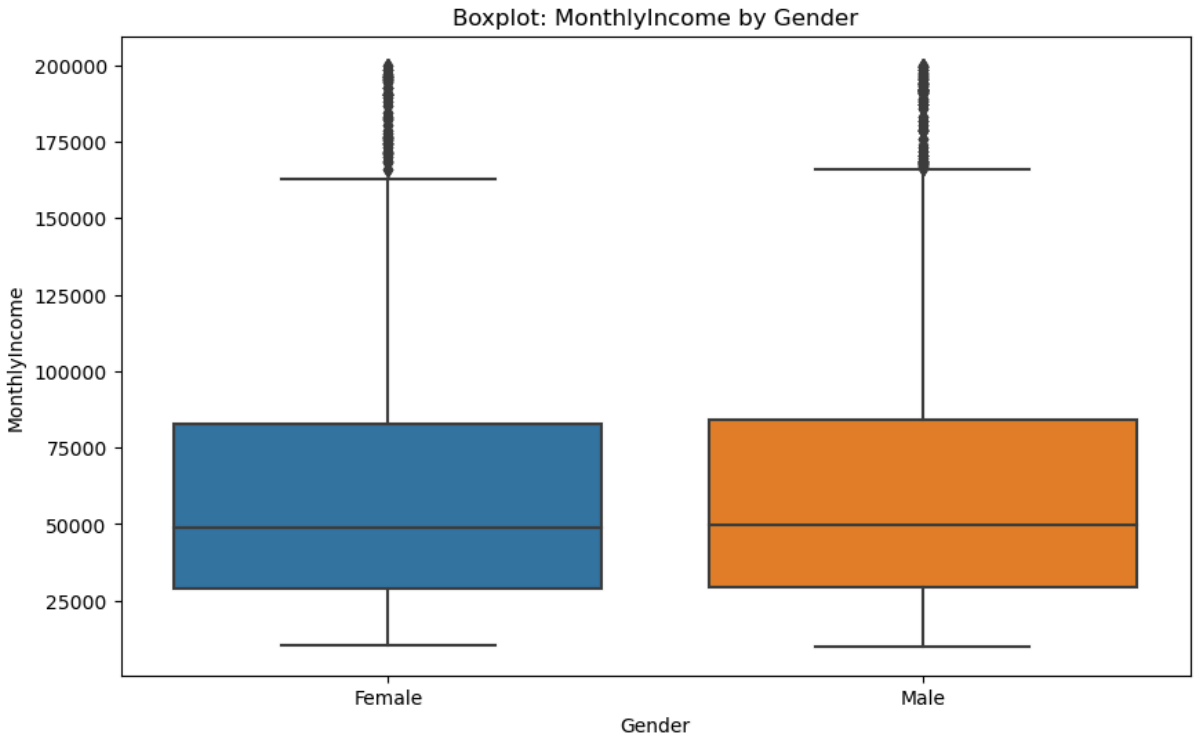
Out[53]:
```
5     576
1     499
3     376
2     369
10    351
4     324
7     263
8     235
9     234
6     223
0     126
11     93
20     79
13     72
15     59
14     52
22     45
12     42
21     40
18     38
16     36
19     31
17     26
24     18
33     15
26     12
25     12
31      9
32      9
27      6
29      6
36      6
23      6
34      3
30      3
37      3
40      3
Name: YearsAtCompany, dtype: int64
```

In [54]:
```
ax = sns.barplot(
     x=x.index, y=x.values,
       errorbar=None,
)
plt.xticks(rotation=90)

ax.plot(5, 600, "*", markersize=10, color="r")
plt.show()
```



Gender vs Monthly Income

In [55]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='MonthlyIncome', data=df)
plt.title('Boxplot: MonthlyIncome by Gender')
plt.show()
```



In [56]:
```
df.pivot_table(index='MaritalStatus',columns='JobInvolvement',values='EmployeeCount',aggfunc='count')
```

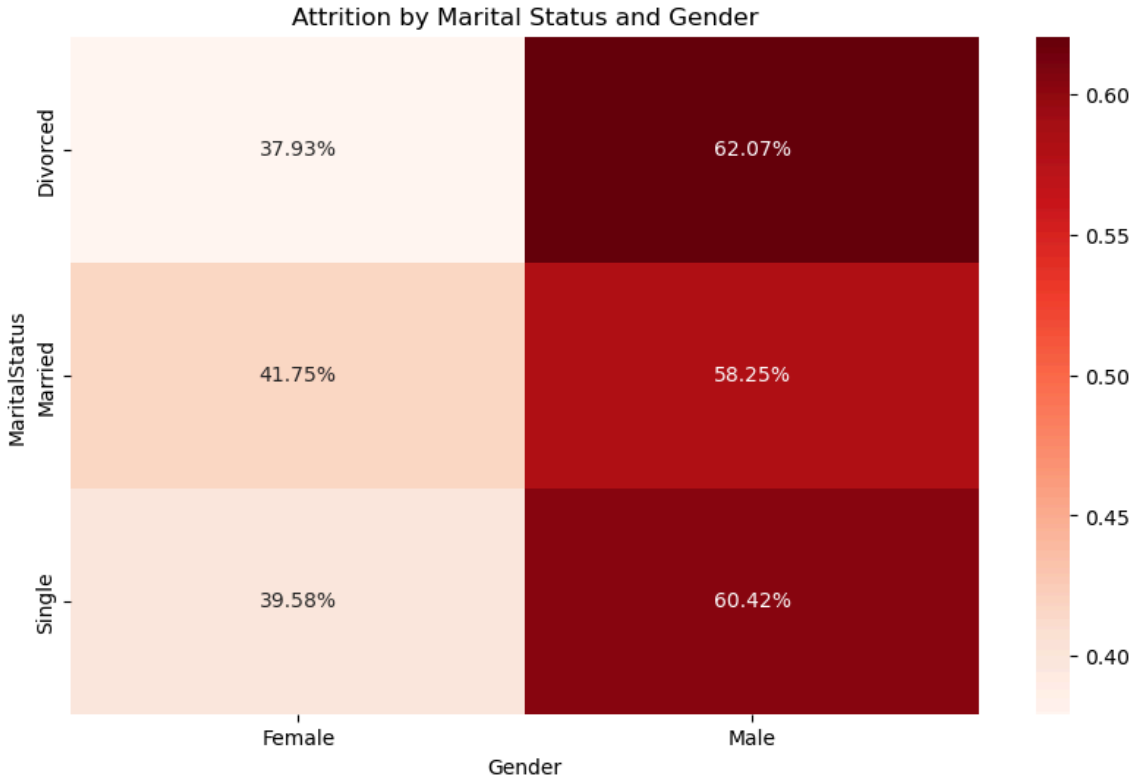
Out[56]:

JobInvolvement	1	2	3	4
MaritalStatus				
Divorced	36	258	564	91
Married	103	535	1148	183
Single	102	311	823	146

Attrition by Marital Status and Gender

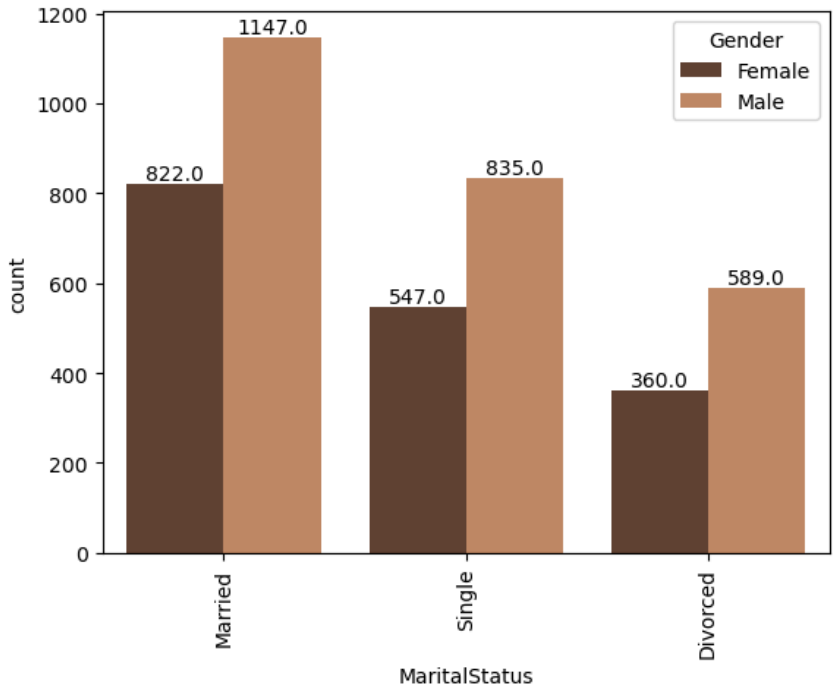
In [57]:

```
cross_tab = pd.crosstab(df['MaritalStatus'], df['Gender'], values=df['Attrition'], aggfunc='count', normaliz
plt.figure(figsize=(10, 6))
sns.heatmap(cross_tab, annot=True, cmap='Reds', fmt=".2%", cbar=True)
plt.title('Attrition by Marital Status and Gender')
plt.show()
```



In [58]:

```
m=sns.countplot(data=df, x='MaritalStatus', hue='Gender',palette='copper')
plt.xticks(rotation=90)
for p in m.patches:
    height = p.get_height()
    m.annotate(f'{height}', (p.get_x() + p.get_width() / 2, height),
               ha='center', va='bottom', fontsize=10)
plt.show()
```



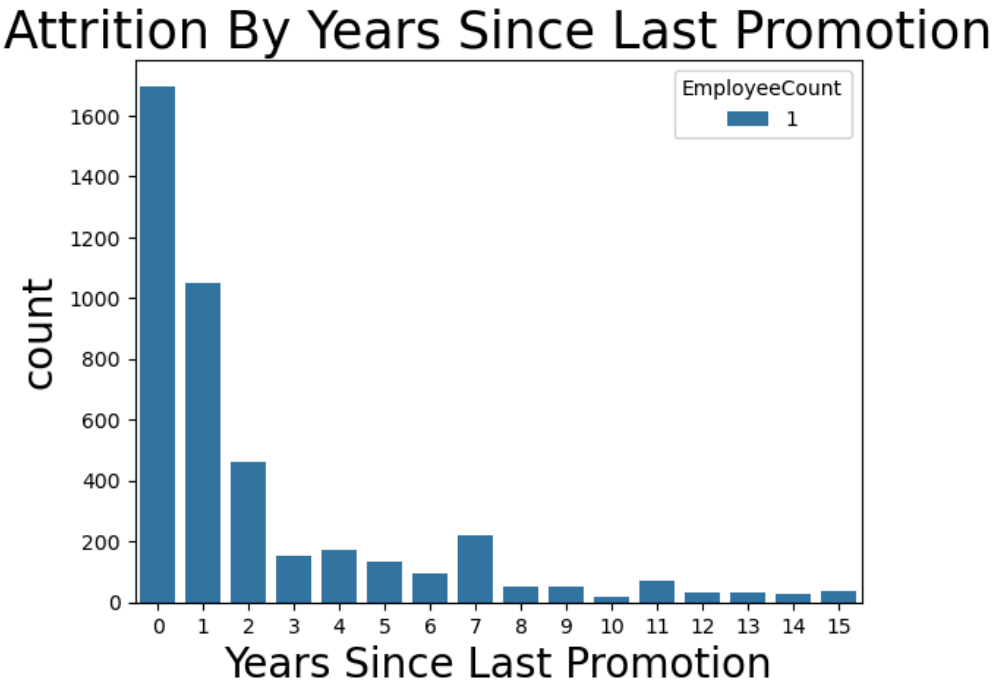
How many employees have never received a promotion?

In [59]:
```
df[df['YearsSinceLastPromotion']==0]['EmployeeCount'].count()
```

Out[59]: 1697

In [60]:
```
sns.countplot(hue='EmployeeCount', x='YearsSinceLastPromotion', data=df)
plt.xlabel('Years Since Last Promotion', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.title('Attrition By Years Since Last Promotion', fontsize=25)
plt.show
```

Out[60]: <function matplotlib.pyplot.show(close=None, block=None)>



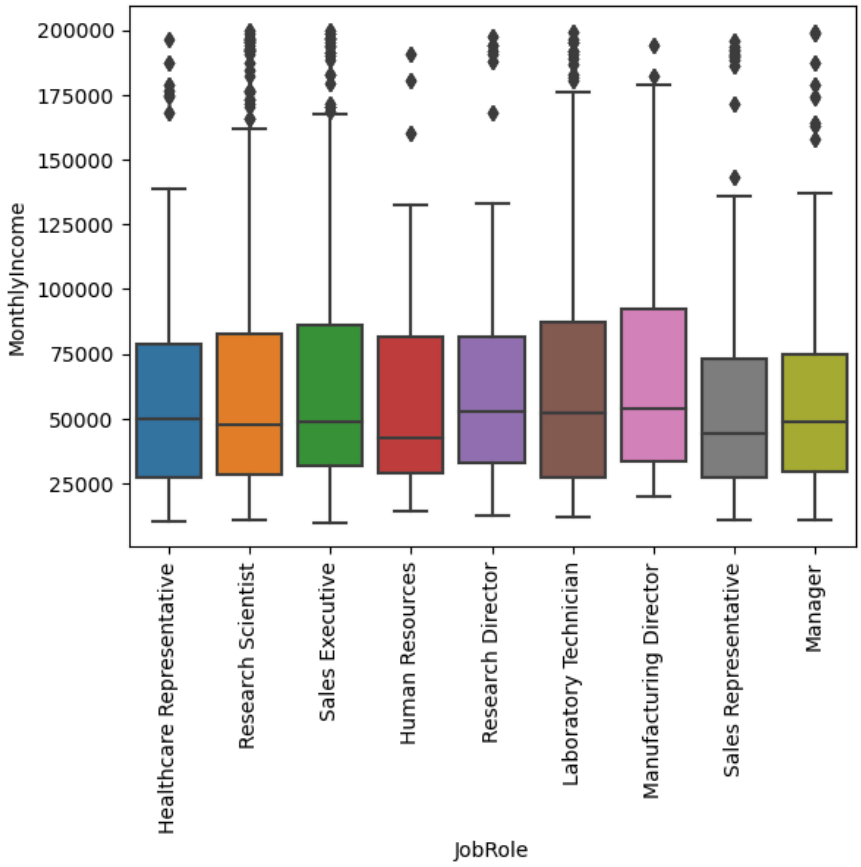
Is there a correlation between performance rating and percent salary hike?

In [61]:
```
df['PerformanceRating'].corr(df['PercentSalaryHike'])
```

Out[61]: 0.7739021713777523

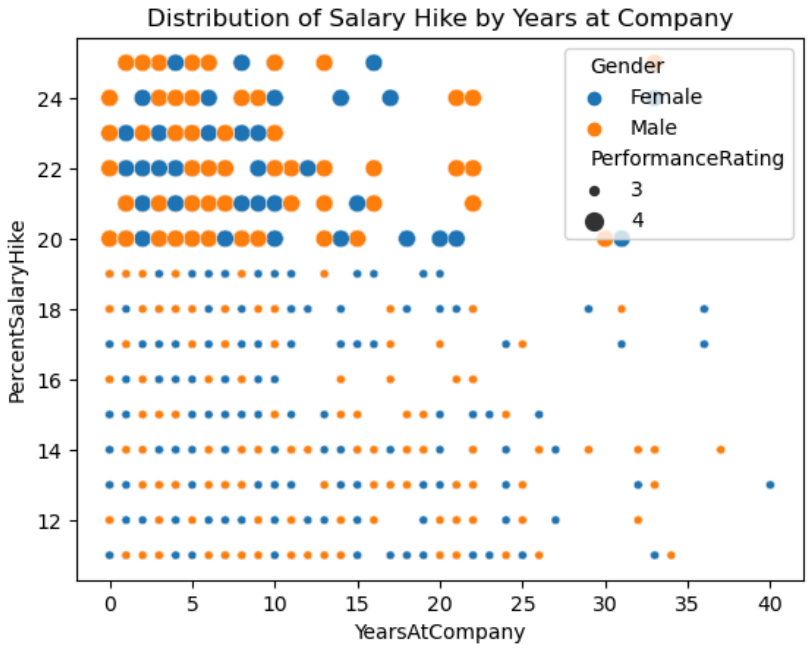
Distribution of Monthly income amongst various job roles

In [62]:
```
sns.boxplot(data=df, x='JobRole', y='MonthlyIncome')
plt.xticks(rotation=90)
plt.show()
```



Salary hike by Performance Rating and Years spent at company

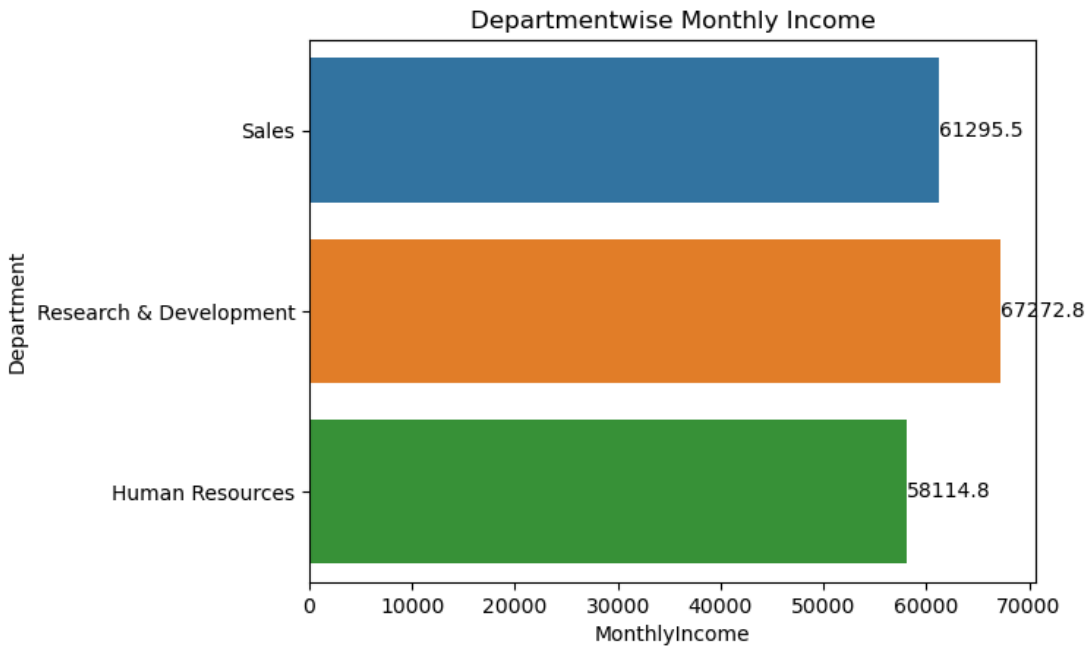
In [63]:
```
sns.scatterplot(data=df, x='YearsAtCompany', y='PercentSalaryHike', hue='Gender', size='PerformanceRating')
plt.title('Distribution of Salary Hike by Years at Company')
plt.show()
```



Departmentwise Monthly Income

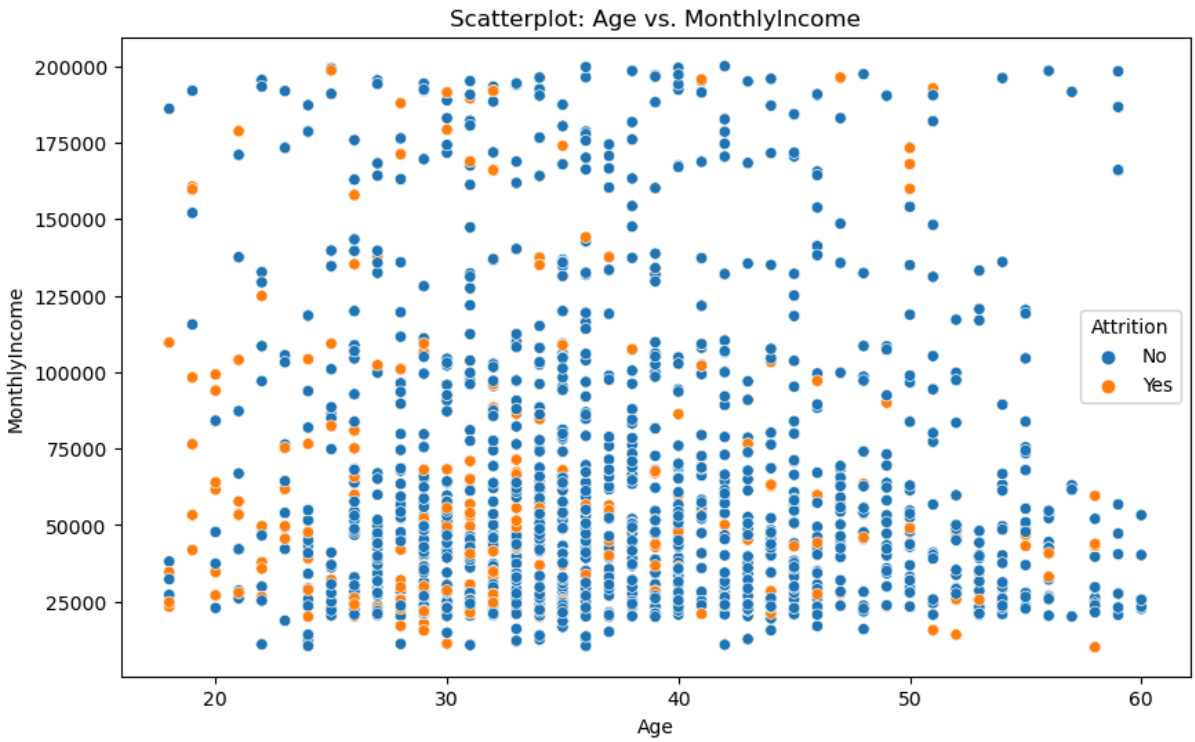
In [64]:
```
p=sns.barplot(data=df, y='Department', x='MonthlyIncome',errorbar=None)
p.bar_label(p.containers[0], fontsize=10)
plt.title('Departmentwise Monthly Income')
```

Out[64]: Text(0.5, 1.0, 'Departmentwise Monthly Income')



Age vs. Monthly Income

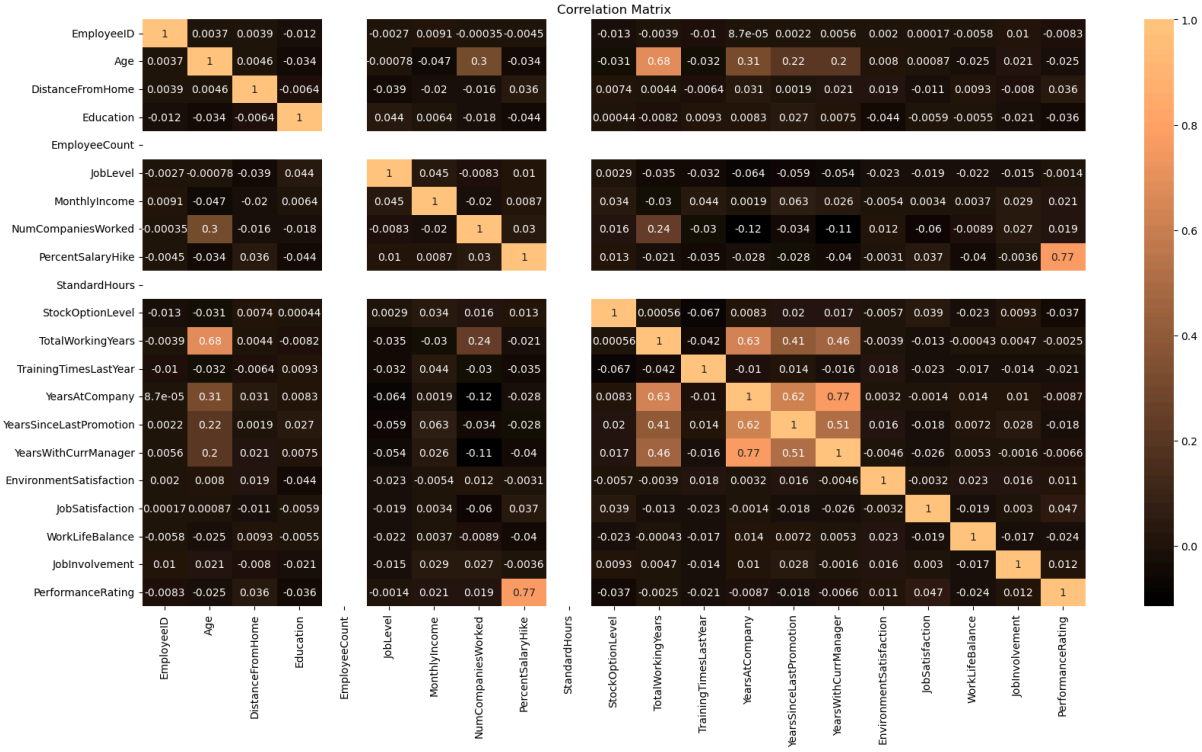
In [65]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='MonthlyIncome', data=df, hue='Attrition')
plt.title('Scatterplot: Age vs. MonthlyIncome')
plt.show()
```



Correlation Matrix

In [66]:

```
correlation_matrix = df.corr(numeric_only=True)
plt.figure(figsize=(20, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='copper')
plt.title('Correlation Matrix')
plt.show()
```



TotalWorkingYears

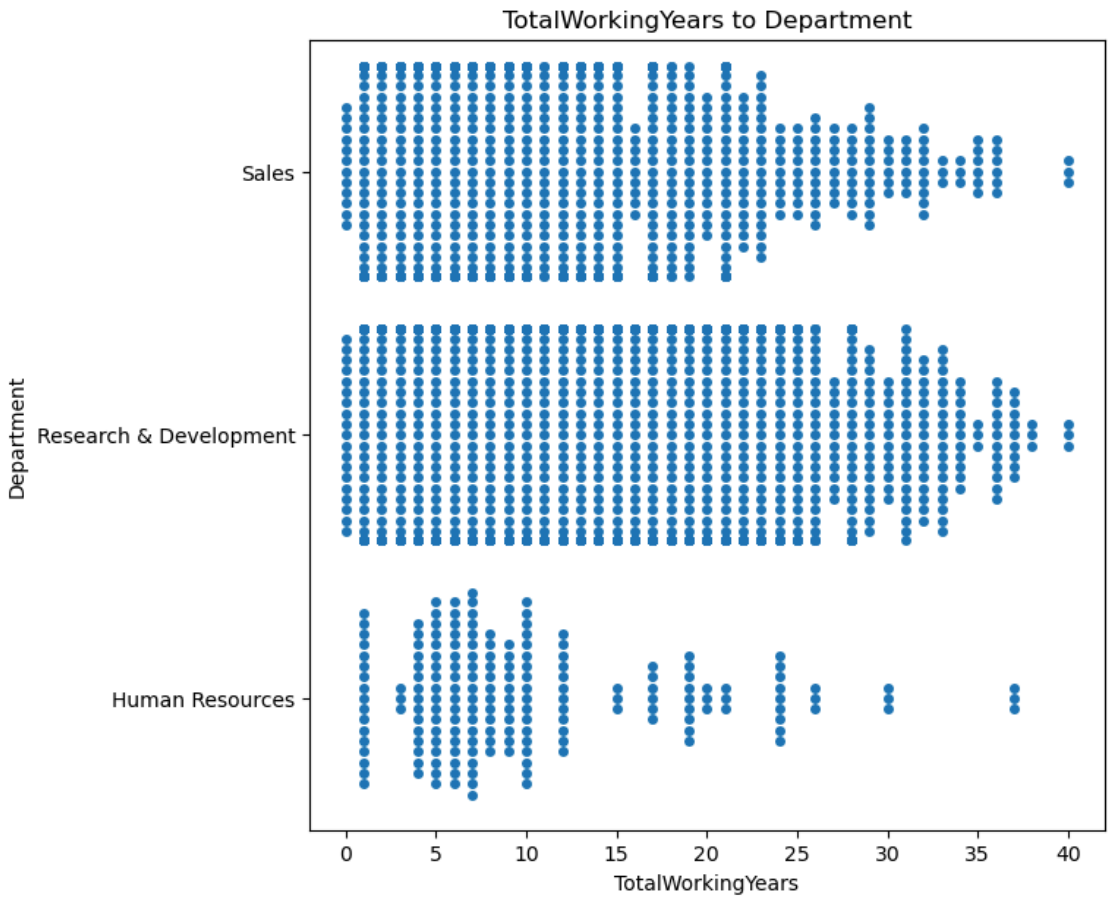
In [67]:

```
sns.boxplot(hue='Attrition',y='TotalWorkingYears',data=df,width=0.6,x="Department",palette="viridis")
plt.ylabel("Total Working Years",fontsize=20)
plt.xlabel('Department',fontsize=20)
plt.title('Attrition By Total Working Years',fontsize=25)
plt.show
plt.grid
```

Out[67]: <function matplotlib.pyplot.grid(visible=None, which='major', axis='both', **kwargs)>

In [68]:
```
ax = plt.subplots(figsize=(7,7))
sns.swarmplot(x= df["TotalWorkingYears"], y= df["Department"]).set( title= 'TotalWorkingYears to Department'
```

Out[68]: [Text(0.5, 1.0, 'TotalWorkingYears to Department')]



Seeing the total working years for employees and the rate of attrition, we can see that the more working years the employees have, the less likely they are to leave the job. Also, employees who have worked for 1 and 10 years have the greater likelihood to leave the job.

How can managers reduce employee attrition?

There's no arguing that managers help employees feel connected to and valued by the organisation (which is essential to retaining talent). But how exactly do they do it…? Well it's simple, great managers prioritise people over productivity.

Great managers build genuine relationships and create a culture of psychological safety (link to other article) which are the catalysts to growing and retaining highly effective teams.

How to reduce attrition of employees

Regularly connect with your team on both an individual and group basis. This will allow managers to identify and resolve potential issues quickly.

Lead with empathy , ask questions and listen to problems. Active listening and maintaining open lines of communication shows your team you're here for them and have their backs.

Advocate for your people. Managers may not always have the power to implement new policies but advocating for and finding creative solutions for your team will demonstrate the organisation's human-centric culture.

Provide regular feedback. Employees who receive daily feedback from their manager are 3x more likely to be engaged than those who receive feedback once a year or less. This doesn't have to be a formal review process. It can be a quick Zoom catch up or email to check in with each other. Remember feedback is a two way street.

Give recognition and encouragement daily. It can be something as simple as saying "thank you for all your hard work today" but the result is happier, more engaged and more connected employees.

End project

By Rajesh Hinduja

THANK YOU!!