Data Analysis Project: HR Data Analysis

- **Name:** Giselle Halim
- **Email:** gisellehalim27@gmail.com
- **Dicoding ID:** gisellehalim

Defining Business Questions

- Is there a significant difference in attrition rates between genders?
- Is there a correlation between age and attrition rates?
- Do certain departments experience higher attrition rates than others? Are there gender differences in attrition rates across departments?
- Is the distance an employee lives from the workplace correlated with attrition?
- Does the number of business trips an employee takes affect employee attrition?
- Does marital status affect attrition, and does this differ by gender?
- Are employees who have worked for more companies more likely to leave?
- Is there a relationship between educational background and employee retention?
- How does satisfaction with the work environment impact employee turnover?
- Does feeling engaged in their work affect employee retention?
- Do employees at higher job levels experience lower attrition rates?
- Do employees with specific job roles experience lower attrition rates?
- Is there a clear correlation between job satisfaction and employee turnover rates?
- Does a healthy work-life balance reduce the risk of employees leaving?
- Do higher paid employees have a lower attrition rate?
- Do salary increases impact employee attrition rates?
- Do high-performing employees tend to stay with the company longer?
- Does working long hours or exceeding standard working hours contribute to employee burnout and attrition?
- Do employees with longer tenure in the company show lower turnover rates?

- Do employees with longer tenure with their current manager show lower turnover rates?
- Does staying in the same role for a long time affect employee retention?
- Does investment in employee training have a positive impact on retention?
- Do relationships between coworkers have an effect on employee attrition?

Import Libraries

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

Data Gathering and Cleaning

```
df =
pd.read_csv('https://raw.githubusercontent.com/dicodingacademy/dicodin
g_dataset/main/employee/employee_data.csv')
df.head()

{"type":"dataframe","variable_name":"df"}
```

Assessing & Cleaning Data

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   EmployeeId                1470 non-null   int64
 1   Age                       1470 non-null   int64
 2   Attrition                 1058 non-null   float64
 3   BusinessTravel            1470 non-null   object
 4   DailyRate                 1470 non-null   int64
 5   Department                1470 non-null   object
 6   DistanceFromHome          1470 non-null   int64
 7   Education                 1470 non-null   int64
 8   EducationField            1470 non-null   object
 9   EmployeeCount             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: float64(1), int64(26), object(8)
memory usage: 402.1+ KB
```

```
#Duplicates
print("Jumlah duplikasi: ", df.duplicated().sum())
```

```
Jumlah duplikasi:  0
```

```
df.isna().sum()
```

```
EmployeeId                    0
Age                           0
Attrition                   412
BusinessTravel                0
DailyRate                     0
Department                    0
DistanceFromHome              0
Education                     0
EducationField                0
EmployeeCount                 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
```

```
# Menghapus data null
df = df.dropna()
```

```
df.columns
```

```
Index(['EmployeeId', 'Age', 'Attrition', 'BusinessTravel',
'DailyRate',
       'Department', 'DistanceFromHome', 'Education',
'EducationField',
       'EmployeeCount', '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')
```

Value in Each Columns

```
#Looking at values in other columns
def value_list(df):
  for col in df.select_dtypes(include=['object']).columns:
    print(f"Value counts for column '{col}':")
    print(df[col].value_counts())
    print('\n')

value_list(df)
```

```
Value counts for column 'BusinessTravel':
BusinessTravel
Travel_Rarely        746
Travel_Frequently    205
Non-Travel           107
Name: count, dtype: int64


Value counts for column 'Department':
Department
Research & Development    701
Sales                     319
Human Resources            38
Name: count, dtype: int64


Value counts for column 'EducationField':
EducationField
Life Sciences       436
Medical             330
Marketing           122
Technical Degree     96
Other                59
Human Resources      15
Name: count, dtype: int64


Value counts for column 'Gender':
Gender
Male      620
Female    438
Name: count, dtype: int64


Value counts for column 'JobRole':
JobRole
Sales Executive              232
Research Scientist           214
Laboratory Technician        188
Manufacturing Director       107
Healthcare Representative     88
Manager                       79
Research Director             62
Sales Representative          58
```

```
Human Resources              30
Name: count, dtype: int64


Value counts for column 'MaritalStatus':
MaritalStatus
Married     464
Single      352
Divorced    242
Name: count, dtype: int64


Value counts for column 'Over18':
Over18
Y    1058
Name: count, dtype: int64


Value counts for column 'OverTime':
OverTime
No     751
Yes    307
Name: count, dtype: int64
```

Exploratory Data Analysis (EDA)

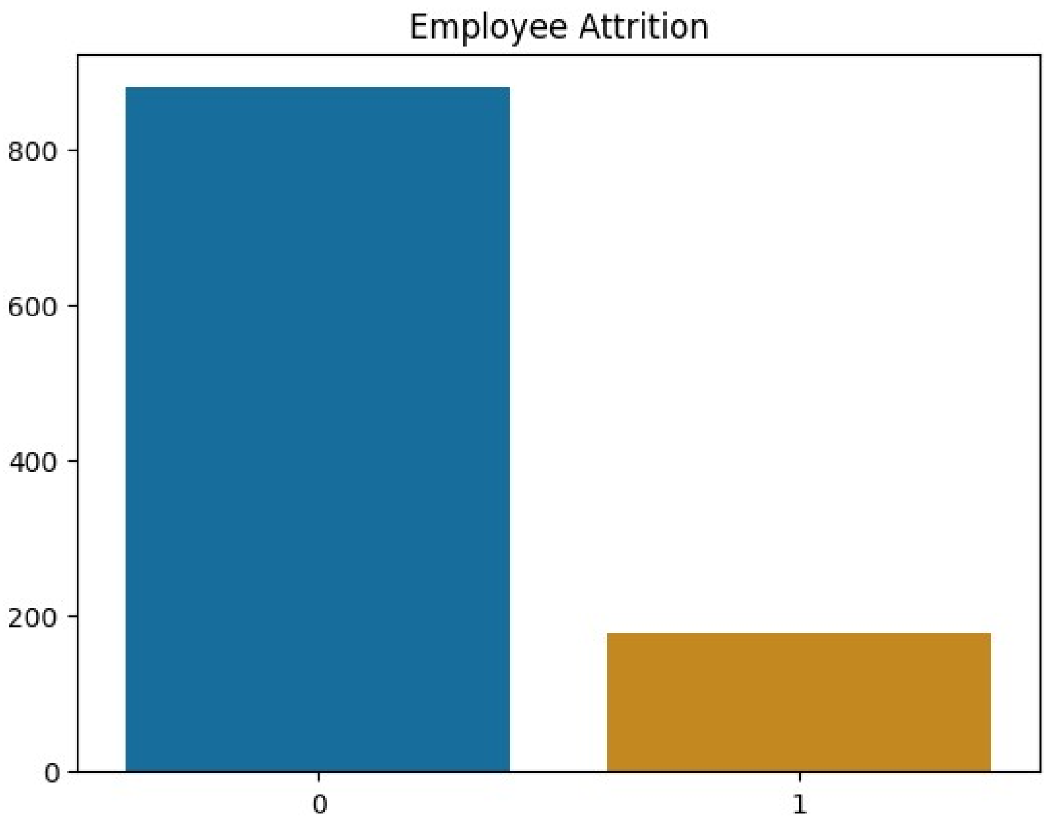
Data Exploration

Describing the Data

```
pd.options.display.max_columns = None
df.select_dtypes(include=['int']).describe()

{"type":"dataframe"}

df['Attrition'] = df['Attrition'].astype(int)
```

Is there a significant difference in attrition rates between genders?

```
#Univariate analysis
data_plot  = df['Attrition'].value_counts().to_list()
label_plot = df['Attrition'].value_counts().index.to_list()

title = 'Employee Attrition'

plot       = sns.barplot(x = label_plot, y = data_plot, palette =
'colorblind')
plot_title = plt.title(title)

plt.show()
```



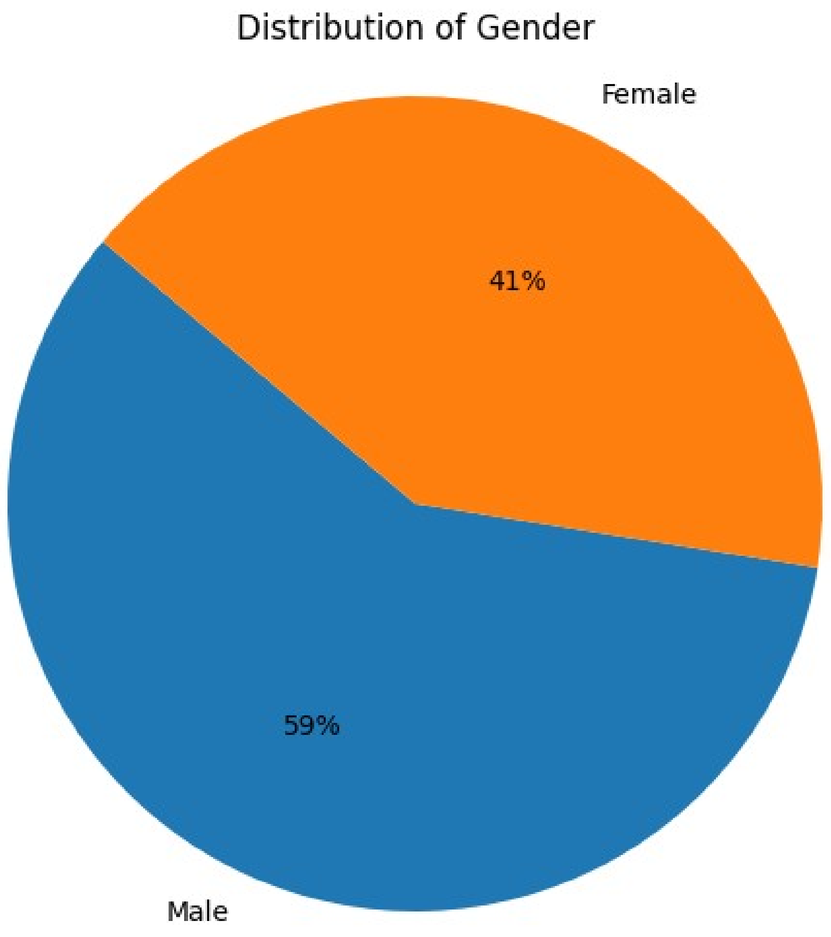
```
# Attrition
att = df['Attrition'].value_counts()
print(att)
```

```
Attrition
0    879
1    179
Name: count, dtype: int64
```

```
# Get total number of employees
total_count = len(df)

# Calculate attrition rate (percentage of those who left)
attrition_rate = (att[1] / total_count) * 100

print(f"Attrition Rate: {attrition_rate:.1f}%")
```

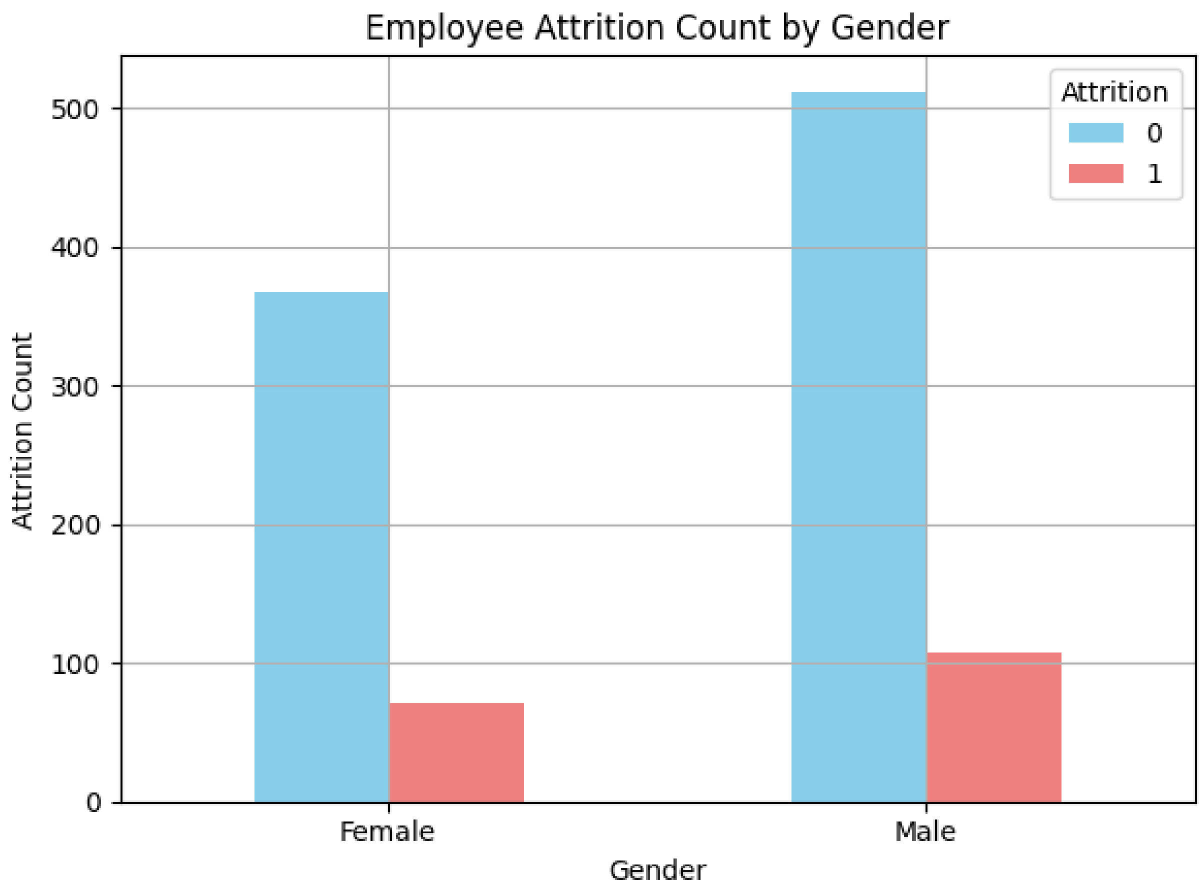
```
Attrition Rate: 16.9%
```

```
# Gender percentage
order_stats_counts =
df['Gender'].value_counts().sort_values(ascending=False)

plt.figure(figsize=(8, 6))
plot = plt.pie(order_stats_counts, labels=order_stats_counts.index,
autopct='%.0f%%', startangle=140)

plt.title('Distribution of Gender')
plt.axis('equal')

plt.show()
```



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

Gender
Male      620
Female    438
Name: count, dtype: int64
```

```
gender_attrition_counts = df.groupby('Gender')
['Attrition'].value_counts().unstack()

plt.figure(figsize=(8, 6))
gender_attrition_counts.plot(kind='bar', color=['skyblue',
'lightcoral'])
plt.xlabel('Gender')
plt.ylabel('Attrition Count')
plt.title('Employee Attrition Count by Gender')
plt.xticks(rotation=0)
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



```
att_count = df.groupby('Gender')['Attrition'].value_counts()
print("Total Attrition per Gender\n")
att_count
```

```
Total Attrition per Gender


Gender  Attrition
Female  0            367
        1             71
Male    0            512
        1            108
Name: count, dtype: int64
```

```
# Calculate total employee by gender
gender_count = df['Gender'].value_counts()

# Calculate attrition rate (percentage)
attrition_pct = np.round((att_count / gender_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage per Gender\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage per Gender

Gender  Attrition
Female  0            83.8%
        1            16.2%
Male    0            82.6%
        1            17.4%
```

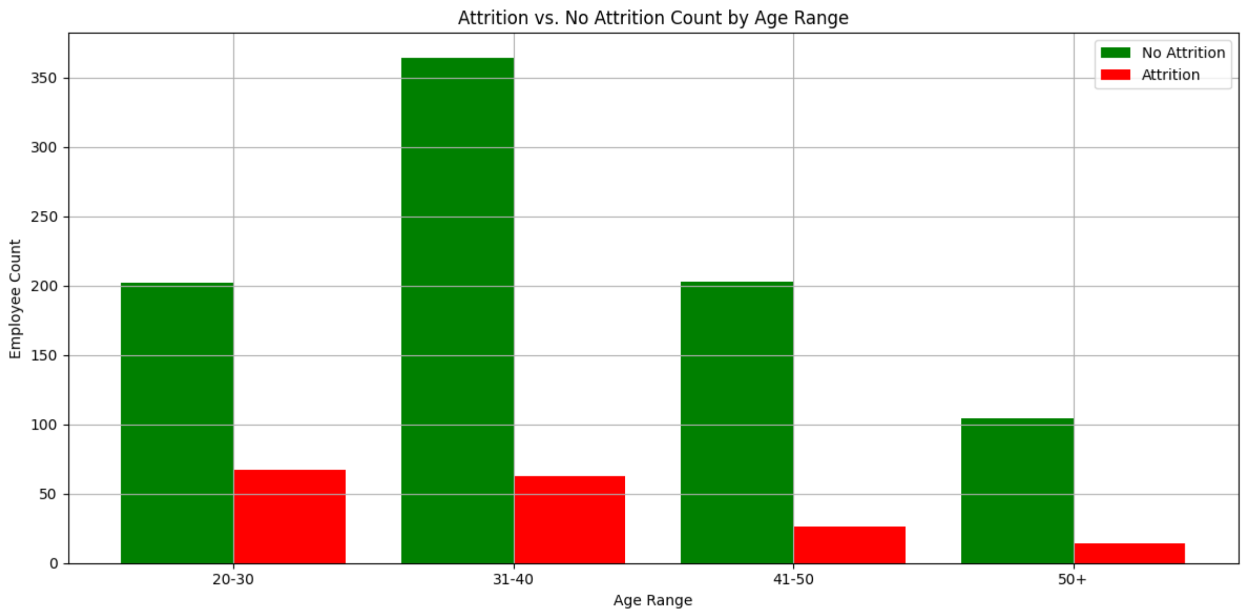
Insight

- The number of attrition experienced by the company reached 179 employees or 16.9% of the total number of employees. This number is quite high because it can affect the company's performance if not addressed (20% is the upper limit).
- The number of male and female employees is quite balanced with a percentage of 59% and 41% for each. Regarding the number of attrition, male employees tend to leave the company more. But this number is not much different from female employees so gender does not affect the overall exit of employees.

Is there a correlation between age and attrition rates?

```
# Define age ranges
age_ranges = {
    '20-30': "Age >= 20 & Age <= 30",
    '31-40': "Age > 30 & Age <= 40",
    '41-50': "Age > 40 & Age <= 50",
    '50+': "Age > 50",
}

attrition_counts = {}
```

```python
no_attrition_counts = {}

for age_range, condition in age_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[age_range] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[age_range] = len(filtered_data)

plt.figure(figsize=(12, 6))
age_labels = list(age_ranges.keys())
x = np.arange(len(age_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Age Range")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, age_labels)
plt.title("Attrition vs. No Attrition Count by Age Range")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
# Group and count attrition occurrences within each Age range
attrition_by_age = {}
for range_label, condition in age_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_age[range_label] = attrition_counts.fillna(0)

print("Attrition by Age Range\n")
for range_label, counts in attrition_by_age.items():
  print(f"{range_label}: {counts}\n")
```

```
Attrition by Age Range

20-30: Attrition
0    202
1     67
Name: count, dtype: int64

31-40: Attrition
0    364
1     63
Name: count, dtype: int64

41-50: Attrition
0    203
1     26
Name: count, dtype: int64

50+: Attrition
0    104
```

```
1     14
Name: count, dtype: int64
```

```
# Group and calculate percentage of attrition within each Age range
attrition_by_age = {}
for range_label, condition in age_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_age[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    attrition_by_age[range_label][attn_value] = round(percentage, 1)

print("Attrition Percentage by Age\n")
for range_label, attrition_percentages in attrition_by_age.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")
```

```
Attrition Percentage by Age

20-30:
	0: 75.1%
	1: 24.9%
31-40:
	0: 85.2%
	1: 14.8%
41-50:
	0: 88.6%
	1: 11.4%
50+:
	0: 88.1%
	1: 11.9%
```

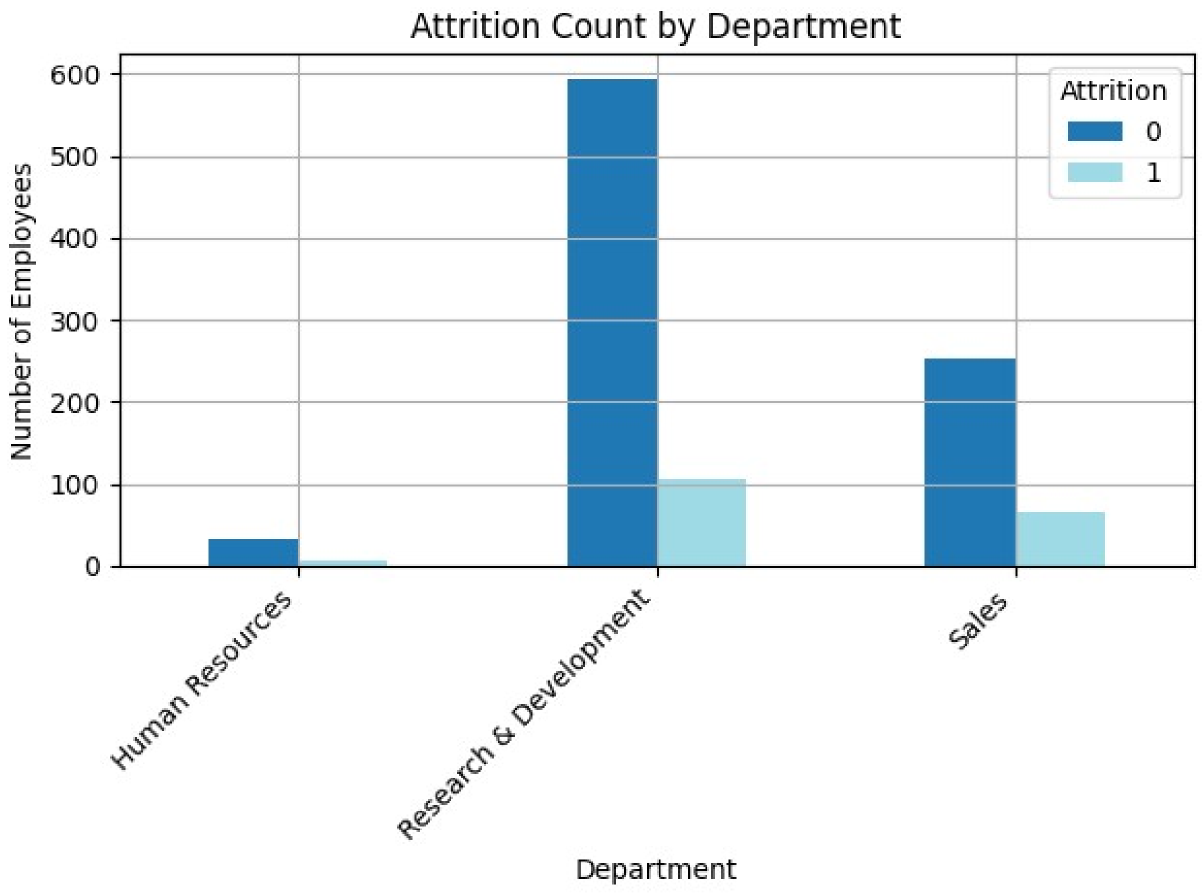
Insight

- The highest number of attrition comes from employees with an age range of 20 - 30 years. This could be because at that age, employees tend to focus on career development rather than stability. This can be seen from the decreasing attrition trend the older the age.

Do certain departments experience higher attrition rates than others? Are there gender differences in attrition rates across departments?

```
dept_counts = (
    df.groupby('Department')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
dept_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Department")
plt.xlabel("Department")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)

plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



```
dept_att = df.groupby('Department')['Attrition'].value_counts()
print("Attrition per Department\n")
print(dept_att)
```

```
Attrition per Department
```

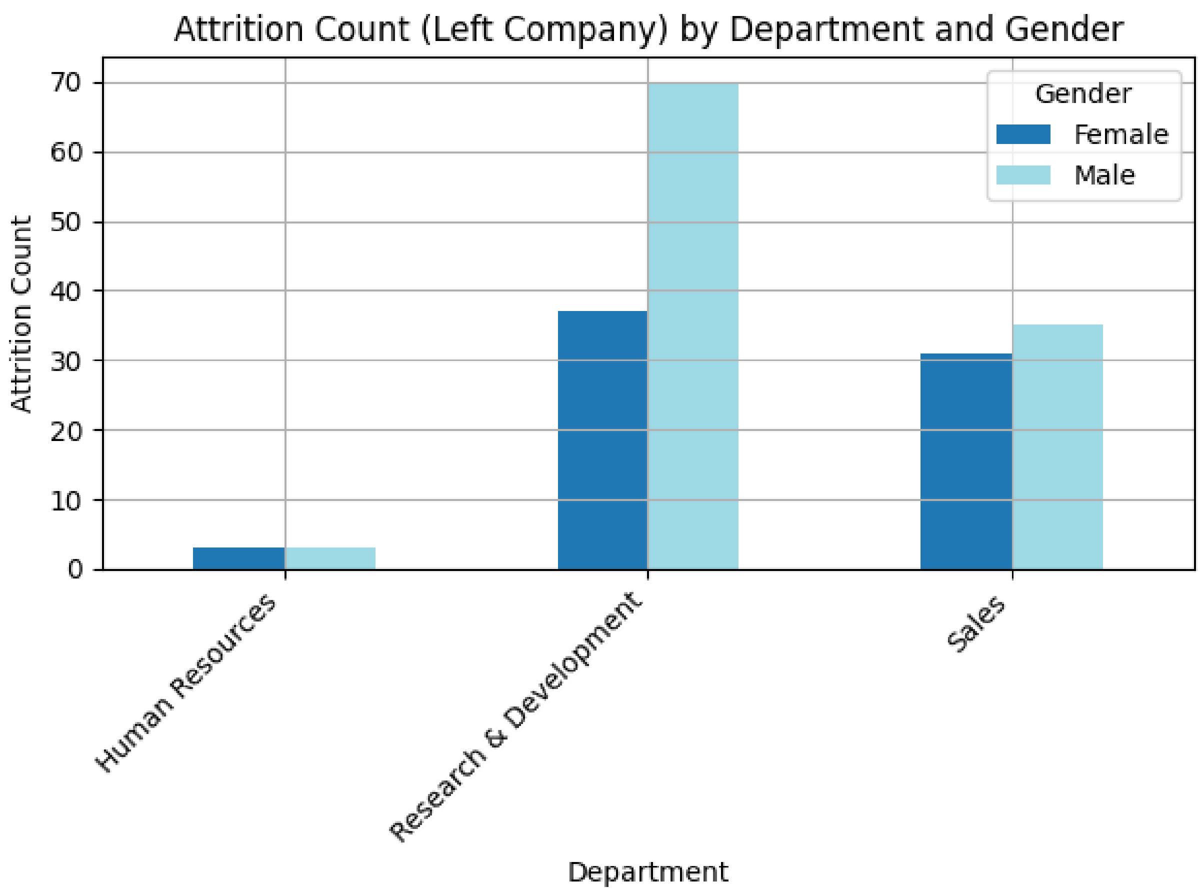
```
Department              Attrition
Human Resources         0              32
                        1               6
Research & Development  0             594
                        1             107
Sales                   0             253
                        1              66
Name: count, dtype: int64
```

```python
dept_count = df['Department'].value_counts()

# Calculate attrition rate (percentage)
attrition_pct = np.round((dept_att / dept_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Department\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Department

Department              Attrition
Human Resources         0            84.2%
                        1            15.8%
Research & Development  0            84.7%
                        1            15.3%
Sales                   0            79.3%
                        1            20.7%
```

```python
# Filter data for attrition (left the company)
attrition_data = df[df['Attrition'] == 1]

# Group data by department and gender (count attrition)
attrition_by_dept_gender = (
    attrition_data.groupby(['Department', 'Gender'])
    .size()
    .unstack(fill_value=0)
    .fillna(0)
)

plt.figure(figsize=(12, 6))
attrition_by_dept_gender.plot(kind='bar', colormap='tab20')

# Customize the plot for better readability
plt.title("Attrition Count (Left Company) by Department and Gender")
plt.xlabel("Department")
plt.ylabel("Attrition Count")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Gender')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



```
dept_emp = df.groupby('Gender')['Department'].value_counts()
print("Total Employee per Department\n")
print(dept_emp)

Total Employee per Department

Gender  Department
Female  Research & Development    287
        Sales                     139
        Human Resources            12
Male    Research & Development    414
        Sales                     180
        Human Resources            26
Name: count, dtype: int64

# Calculate total employee count by department and gender
dept_emp_count = df.groupby(['Department',
'Gender']).size().unstack(fill_value=0)

# Calculate attrition count by department and gender (already filtered
in many cases)
attrition_count = df[df['Attrition'] == 1].groupby(['Department',
'Gender']).size().unstack(fill_value=0)

# Calculate attrition rate (percentage)
attrition_pct = np.round((attrition_count / dept_emp_count) * 100,
1).applymap(lambda x: f"{x:.1f}%")

print("Attrition Percentage per Department\n")
print(attrition_pct.to_string())
```

```
Attrition Percentage per Department

Gender                  Female   Male
Department
Human Resources          25.0%  11.5%
Research & Development   12.9%  16.9%
Sales                    22.3%  19.4%
```

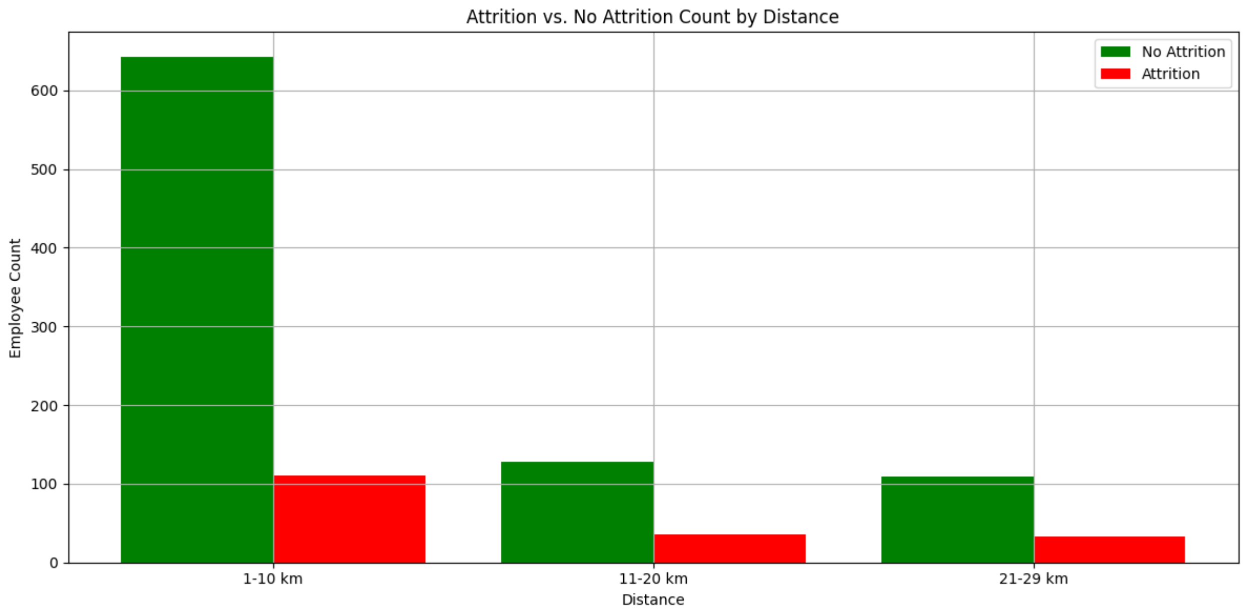
Insight

- Attrition is most prevalent in the sales department, reaching 20.7% with a difference of about 5% when compared to other departments. This could be due to the heavy workload of the sales department.
- When compared per gender, female employees in the human resources and sales departments leave the company at a higher rate. While male employees in the RnD department leave the company the most.

Is the distance an employee lives from the workplace correlated with attrition?

```
# Define distance ranges
distance_ranges = {
    '1-10 km': "DistanceFromHome >= 1 & DistanceFromHome <= 10",
    '11-20 km': "DistanceFromHome > 10 & DistanceFromHome <= 20",
    '21-29 km': "DistanceFromHome > 20 & DistanceFromHome <= 29",
}

attrition_counts = {}
no_attrition_counts = {}

for range_label, condition in distance_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[range_label] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[range_label] = len(filtered_data)
```

```
plt.figure(figsize=(12, 6))
distance_labels = list(distance_ranges.keys())
x = np.arange(len(distance_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Distance")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, distance_labels)
plt.title("Attrition vs. No Attrition Count by Distance")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



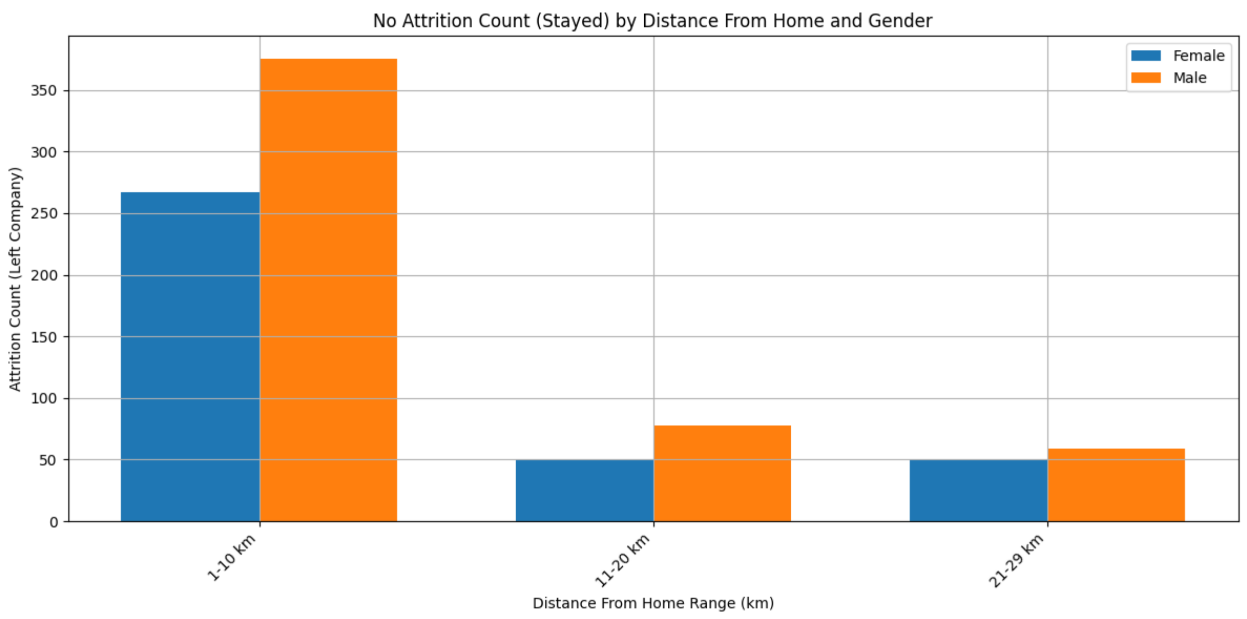
```
attrition_by_distance = {}
for range_label, condition in distance_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_distance[range_label] = attrition_counts.fillna(0)
```

```
print("Attrition by Distance From Home\n")
for range_label, counts in attrition_by_distance.items():
  print(f"{range_label}: {counts}\n")
```

```
Attrition by Distance From Home

1-10 km: Attrition
0    642
1    110
Name: count, dtype: int64

11-20 km: Attrition
0    128
1     36
Name: count, dtype: int64

21-29 km: Attrition
0    109
1     33
Name: count, dtype: int64
```

```
attrition_by_distance = {}
for range_label, condition in distance_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_distance[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    attrition_by_distance[range_label][attn_value] = round(percentage,
1)

print("Attrition Percentage by Distance From Home\n")
for range_label, attrition_percentages in
attrition_by_distance.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")
```

```
Attrition Percentage by Distance From Home

1-10 km:
	0: 85.4%
	1: 14.6%
11-20 km:
	0: 78.0%
	1: 22.0%
21-29 km:
      0: 76.8%
      1: 23.2%

# Filter data for attrition (left the company)
attrition_data = df[df['Attrition'] == 0]

attrition_by_distance_gender = {}
for gender in df['Gender'].unique():
  attrition_by_distance_gender[gender] = []
  for distance_range, condition in distance_ranges.items():
    filtered_data = attrition_data.query(condition)
    count = len(filtered_data[filtered_data['Gender'] == gender])
    attrition_by_distance_gender[gender].append(count)

plt.figure(figsize=(12, 6))
distance_labels = list(distance_ranges.keys())
x = np.arange(len(distance_labels))
width = 0.35

for i, gender in enumerate(attrition_by_distance_gender.keys()):
  plt.bar(x - width/2 + i * width,
attrition_by_distance_gender[gender], width, label=gender)

plt.xlabel("Distance From Home Range (km)")
plt.ylabel("Attrition Count (Left Company)")
plt.xticks(x, distance_labels, rotation=45, ha='right')
plt.title("No Attrition Count (Stayed) by Distance From Home and
Gender")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```python
# Group and calculate attrition percentage by gender within each
distance range
attrition_by_distance = {}
for range_label, condition in distance_ranges.items():
  filtered_data = df.query(condition)
  attrition_by_gender = filtered_data[filtered_data['Attrition'] ==
1].groupby('Gender').size()
  total_count = len(filtered_data)
  attrition_by_distance[range_label] = {}
  for gender in attrition_by_gender.index:
    attrition_count = attrition_by_gender[gender]
    percentage = (attrition_count / total_count) * 100 if total_count
> 0 else 0
    attrition_by_distance[range_label][gender] = round(percentage, 1)

print("Attrition Percentage by Distance From Home and Gender (Left
Company)\n")
for range_label, attrition_per_gender in
attrition_by_distance.items():
  print(f"{range_label}:")
  for gender, percentage in attrition_per_gender.items():
    print(f"\t- {gender}: {percentage}%")
  print()
```

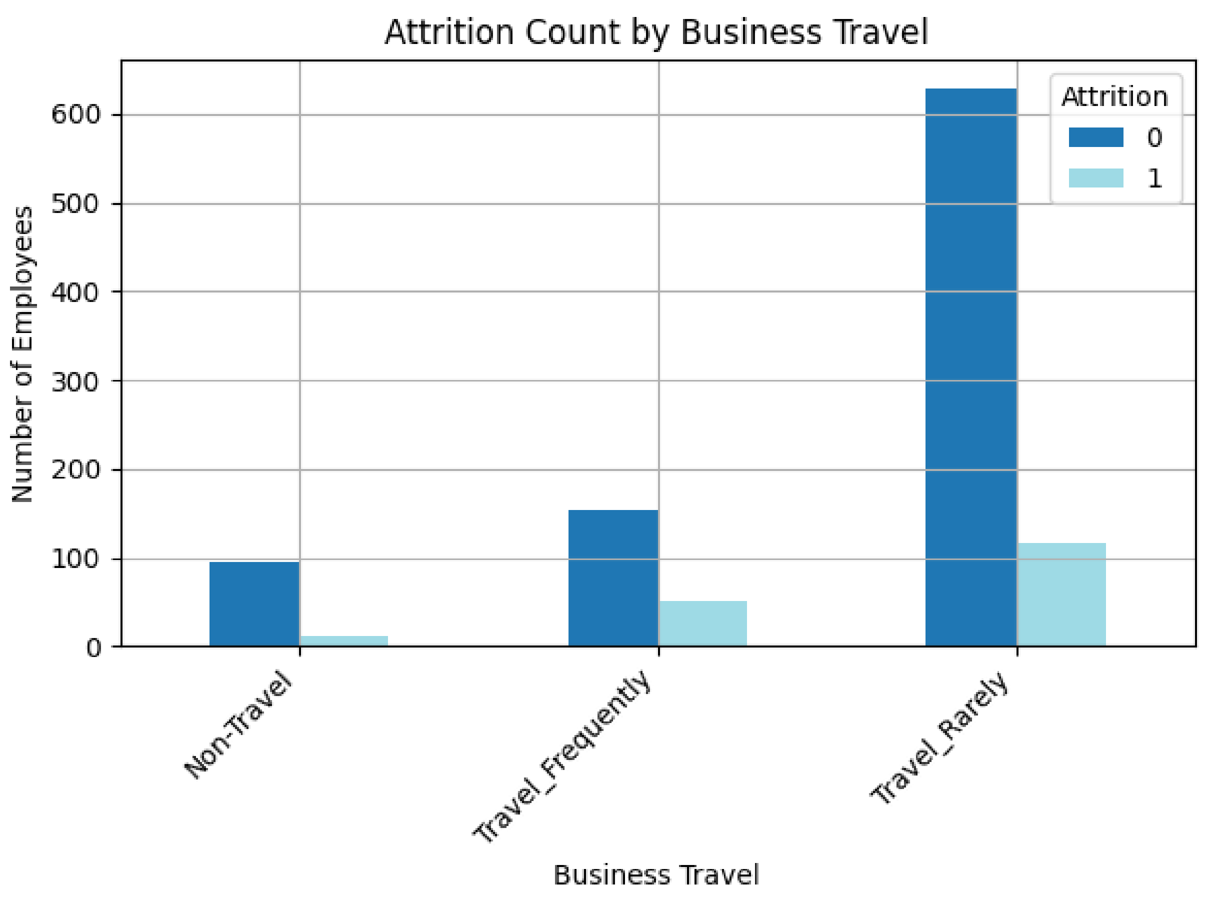
```
Attrition Percentage by Distance From Home and Gender (Left Company)

1-10 km:
      - Female: 5.2%
      - Male: 9.4%

11-20 km:
      - Female: 12.2%
      - Male: 9.8%

21-29 km:
      - Female: 8.5%
      - Male: 14.8%
```

Insight

- The farther the office is from home, the more likely employees are to leave the company. This can be caused by high transportation costs and other factors.
- When viewed by gender, both men and women will be more likely to leave if the office is far from their residence, so there is no relationship with gender.

Does the number of business trips an employee takes affect employee attrition?

```
trip_counts = (
    df.groupby('BusinessTravel')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
trip_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Business Travel")
plt.xlabel("Business Travel")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)

plt.show()

<Figure size 800x600 with 0 Axes>
```



```python
trip_att = df.groupby('BusinessTravel')['Attrition'].value_counts()
print("Attrition by Business Travel\n")
print(trip_att)
```

```
Attrition by Business Travel

BusinessTravel     Attrition
Non-Travel         0              96
                   1              11
Travel_Frequently  0             154
                   1              51
Travel_Rarely      0             629
                   1             117
Name: count, dtype: int64
```

```python
trip_count = df['BusinessTravel'].value_counts()

attrition_pct = np.round((trip_att / trip_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Business Travel\n")
print(attrition_pct.to_string())
```

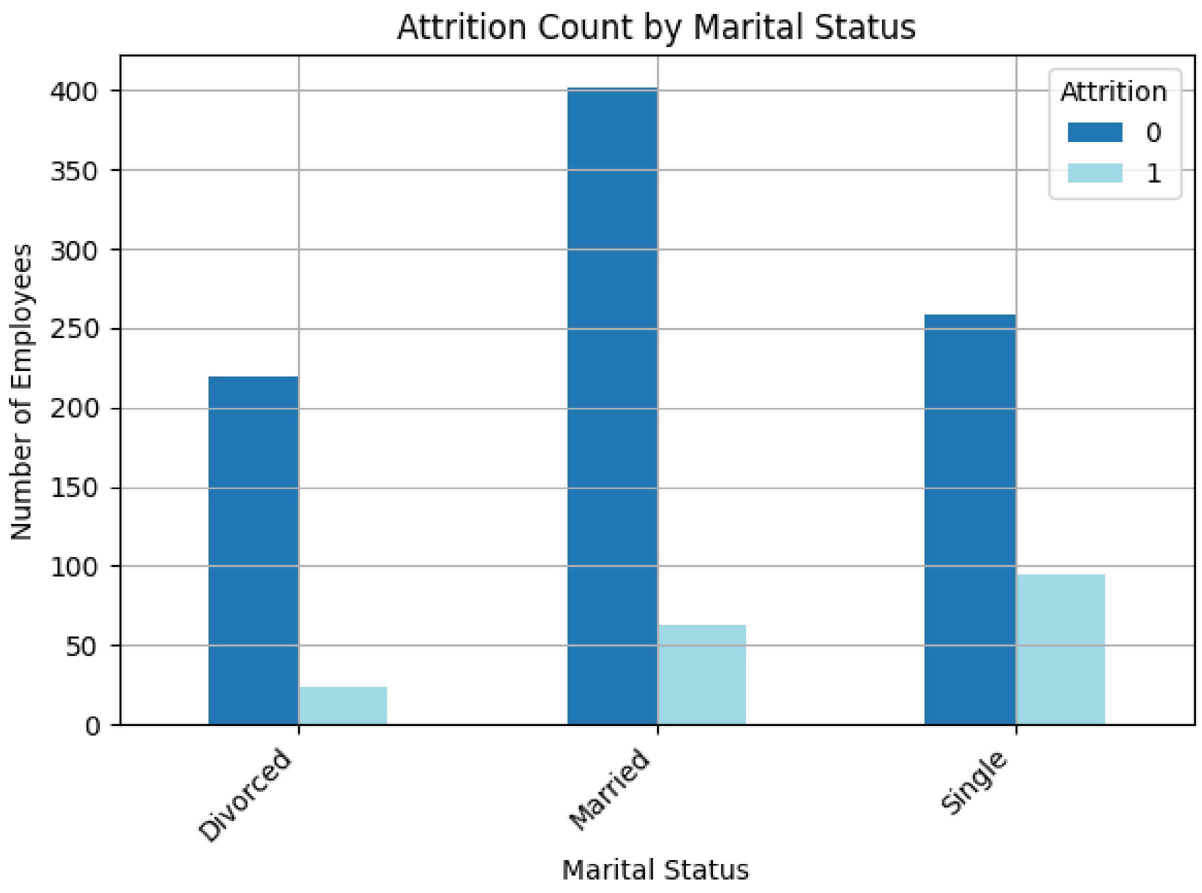
```
Total Attrition Percentage According to Business Travel

BusinessTravel     Attrition
Non-Travel         0            89.7%
                   1            10.3%
Travel_Frequently  0            75.1%
                   1            24.9%
Travel_Rarely      0            84.3%
                   1            15.7%
```

Insight

- The more business trips that are taken, the greater the potential for employees to leave the company. This can be caused by travel fatigue.

Does marital status affect attrition, and does this differ by gender?

```
marriage_attrition_counts = (
    df.groupby('MaritalStatus')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
marriage_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Marital Status")
plt.xlabel("Marital Status")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```

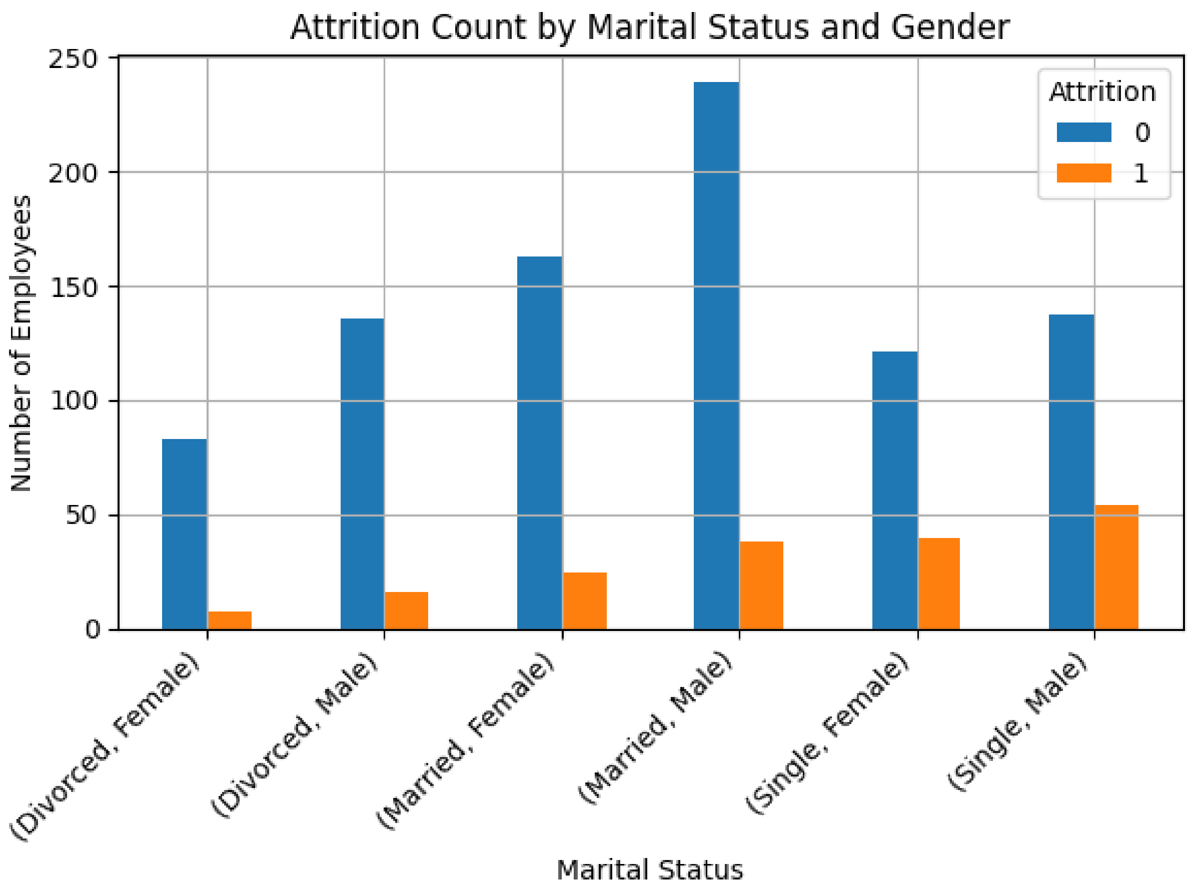


```
marital_sts_att = df.groupby('MaritalStatus')
['Attrition'].value_counts()
print("Attrition According to Marital Status\n")
print(marital_sts_att)

Attrition According to Marital Status

MaritalStatus  Attrition
Divorced       0            219
               1             23
Married        0            402
               1             62
Single         0            258
               1             94
Name: count, dtype: int64

marital_sts_count = df['MaritalStatus'].value_counts()

attrition_pct = np.round((marital_sts_att / marital_sts_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Marital Status\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Marital Status

MaritalStatus  Attrition
Divorced       0            90.5%
               1             9.5%
Married        0            86.6%
               1            13.4%
Single         0            73.3%
               1            26.7%
```

```
attrition_by_marriage_gender = (
    df.groupby(['MaritalStatus', 'Gender'])['Attrition']
    .value_counts()
    .unstack(fill_value=0)
    .fillna(0)
)

plt.figure(figsize=(12, 6))
attrition_by_marriage_gender.plot(kind='bar', stacked=False)
plt.title("Attrition Count by Marital Status and Gender")
plt.xlabel("Marital Status")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



```
marriage_emp = df.groupby('Gender')['MaritalStatus'].value_counts()
print("Total Employee by Marital Status\n")
print(marriage_emp)

Total Employee by Marital Status

Gender  MaritalStatus
Female  Married          187
        Single           161
        Divorced          90
Male    Married          277
        Single           191
        Divorced         152
Name: count, dtype: int64

marry_emp_count = df.groupby(['MaritalStatus',
'Gender']).size().unstack(fill_value=0)

attrition_count = df[df['Attrition'] == 1].groupby(['MaritalStatus',
'Gender']).size().unstack(fill_value=0)

attrition_pct = np.round((attrition_count / marry_emp_count) * 100,
1).applymap(lambda x: f"{x:.1f}%")
```

```
print("Attrition Percentage by Marital Status\n")
print(attrition_pct.to_string())
```

```
Attrition Percentage by Marital Status

Gender         Female   Male
MaritalStatus
Divorced         7.8%  10.5%
Married         12.8%  13.7%
Single          24.8%  28.3%
```

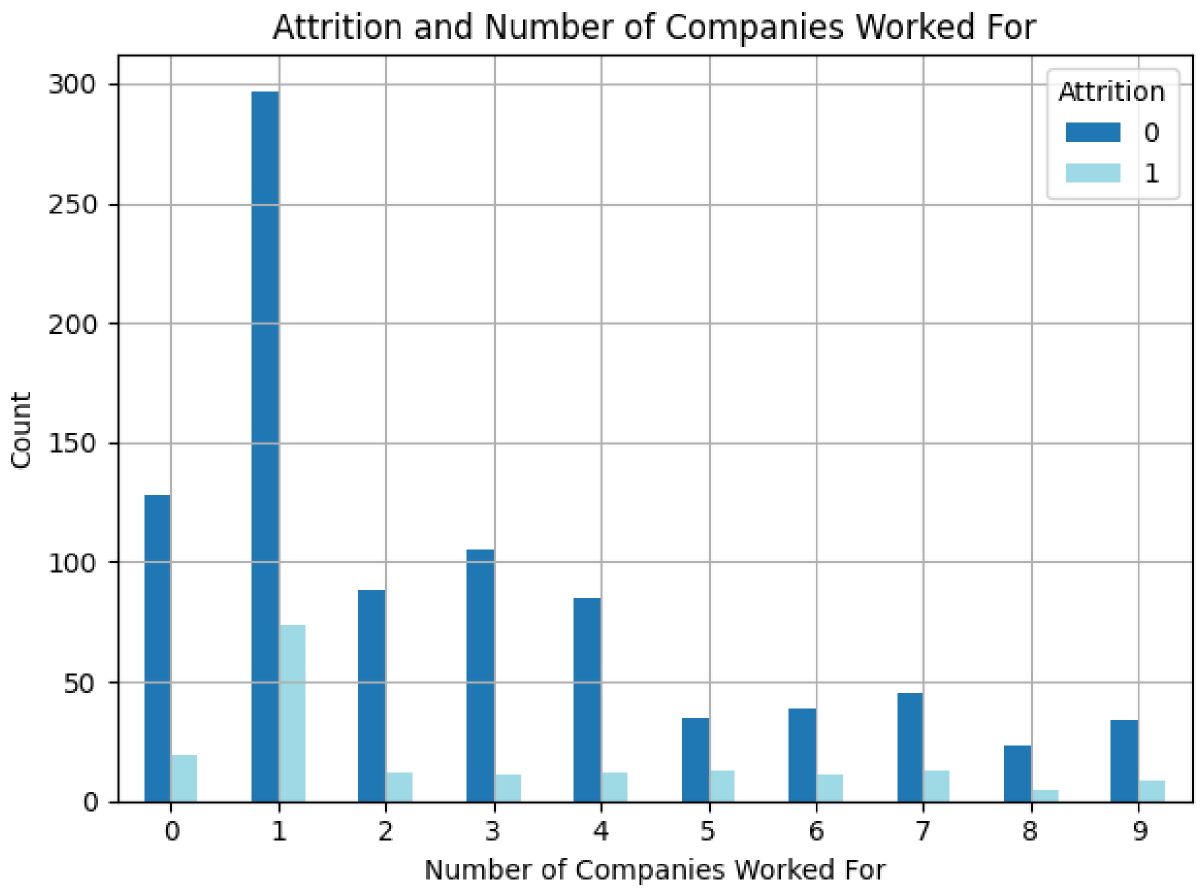
Insight

- Marital status is one of the factors that influence employee departure from the company, single employees leave the most while divorced employees leave less frequently. This could be due to the tendency of single employees not having dependents and therefore being free to look for other opportunities, whereas married or divorced employees usually have dependents (e.g. spouse or children).
- This is not influenced by gender as for both men and women, the same trend can be seen.

Are employees who have worked for more companies more likely to leave?

```
# Group data by NumCompaniesWorked and calculate attrition count
attrition_by_worked_companies = (
    df.groupby('NumCompaniesWorked')['Attrition']
    .value_counts()
    .unstack(fill_value=0)
    .fillna(0)
)

plt.figure(figsize=(10, 6))
attrition_by_worked_companies.plot(kind='bar', colormap='tab20')

plt.xlabel("Number of Companies Worked For")
plt.ylabel("Count")
plt.xticks(rotation=0)
plt.title("Attrition and Number of Companies Worked For")
plt.legend(title='Attrition')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



```
num_comp_att = df.groupby('NumCompaniesWorked')
['Attrition'].value_counts()
print("Attrition According to Number of Companies Worked For\n")
print(num_comp_att)

Attrition According to Number of Companies Worked For

NumCompaniesWorked  Attrition
0                   0            128
                    1             19
1                   0            297
                    1             74
2                   0             88
                    1             12
3                   0            105
                    1             11
4                   0             85
                    1             12
5                   0             35
                    1             13
6                   0             39
                    1             11
```

```
7                    0              45
                     1              13
8                    0              23
                     1               5
9                    0              34
                     1               9
Name: count, dtype: int64
```

```python
num_comp_count = df['NumCompaniesWorked'].value_counts()

attrition_pct = np.round((num_comp_att / num_comp_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Attrition According to
Number of Companies Worked For\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Attrition According to Number
of Companies Worked For

NumCompaniesWorked  Attrition
0                   0             87.1%
                    1             12.9%
1                   0             80.1%
                    1             19.9%
2                   0             88.0%
                    1             12.0%
3                   0             90.5%
                    1              9.5%
4                   0             87.6%
                    1             12.4%
5                   0             72.9%
                    1             27.1%
6                   0             78.0%
                    1             22.0%
7                   0             77.6%
                    1             22.4%
8                   0             82.1%
                    1             17.9%
9                   0             79.1%
                    1             20.9%
```

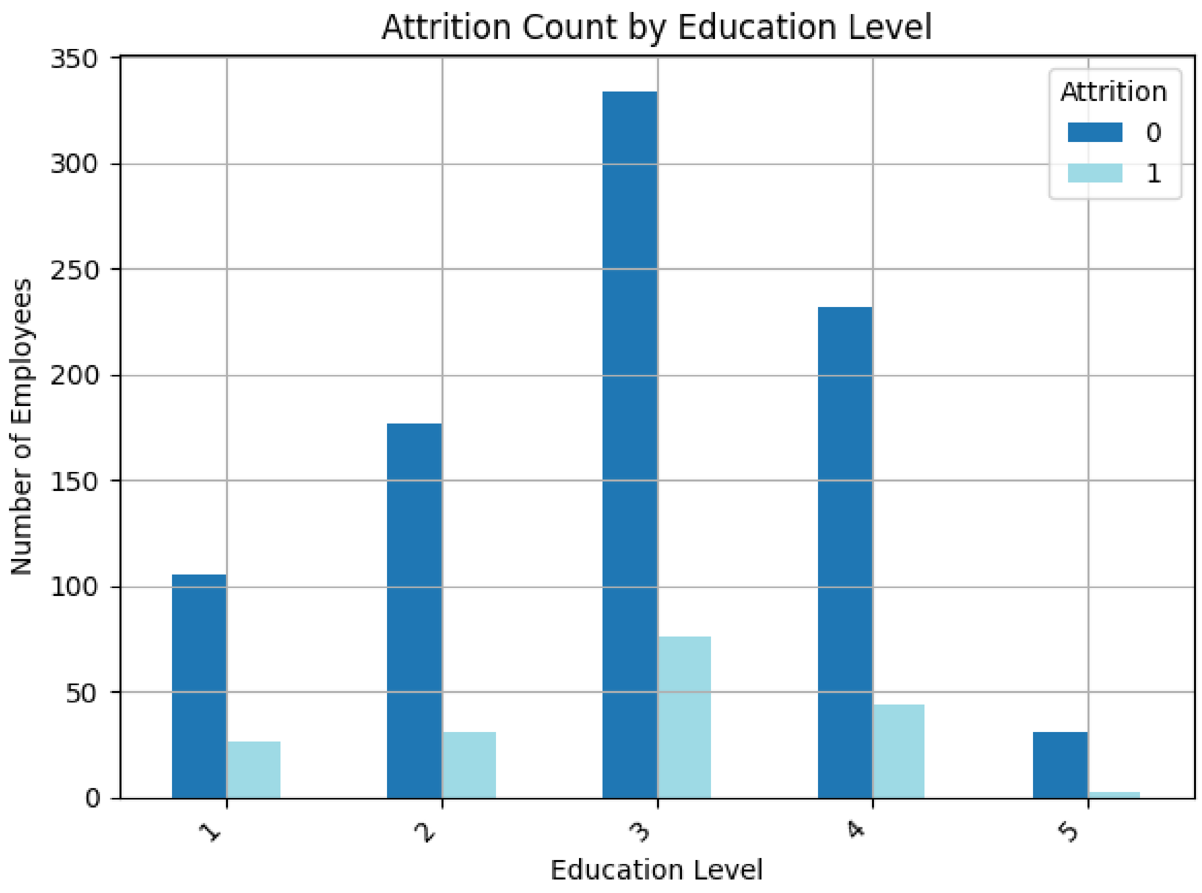
Insight

- Employees who have had more work experience are more likely to leave the company, but the difference is not much with employees with less experience.

Is there a relationship between educational background and employee retention?

```
education_attrition_counts = (
    df.groupby('Education')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
education_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



```
edu_att = df.groupby('Education')['Attrition'].value_counts()
print("Attrition According to Education Level\n")
print(edu_att)
```

```
Attrition According to Education Level

Education  Attrition
1          0            105
           1             26
2          0            177
           1             31
3          0            334
           1             76
4          0            232
           1             44
5          0             31
           1              2
Name: count, dtype: int64
```

```
edu_count = df['Education'].value_counts()

attrition_pct = np.round((edu_att / edu_count) * 100, 1).apply(lambda
x: f"{x:.1f}%")

print("Total Attrition Percentage According to Education Level\n")
print(attrition_pct.to_string())
```

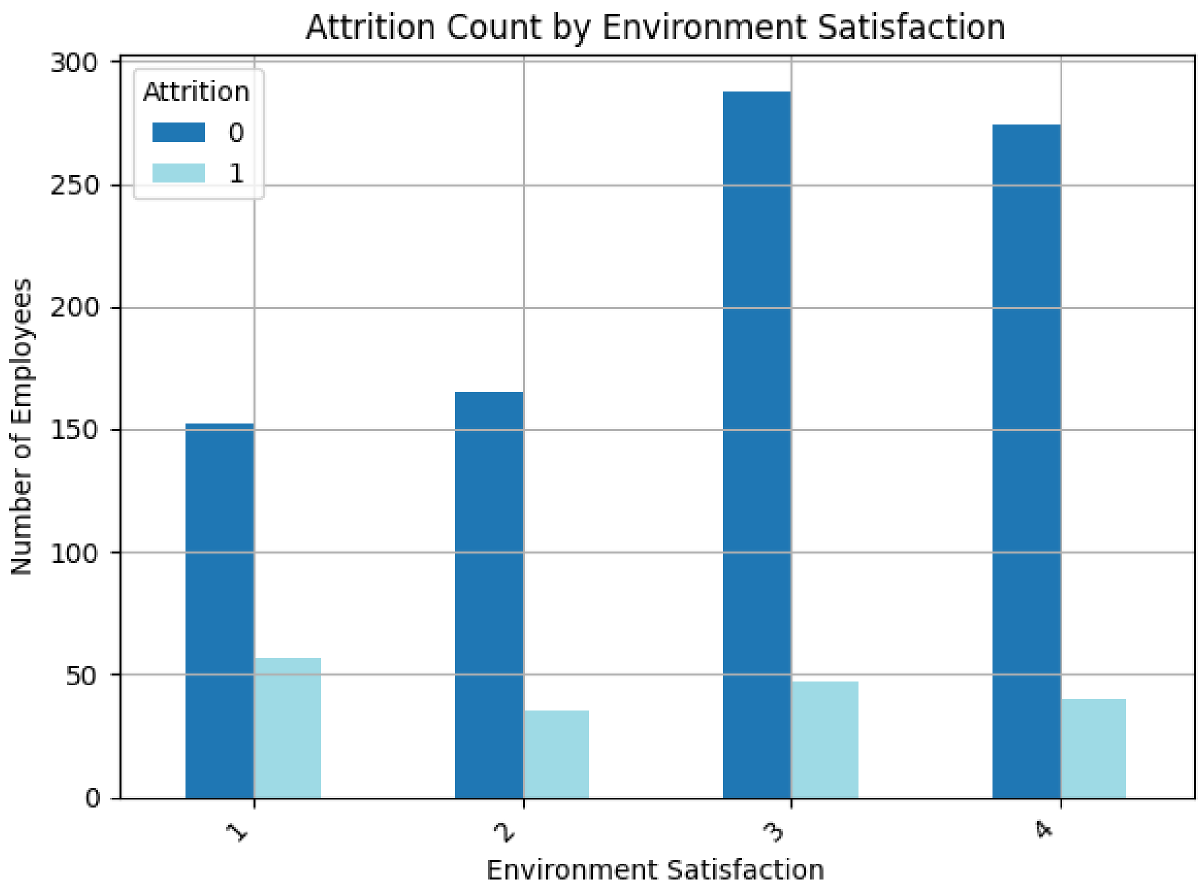
```
Total Attrition Percentage According to Education Level

Education  Attrition
1          0            80.2%
           1            19.8%
2          0            85.1%
           1            14.9%
3          0            81.5%
           1            18.5%
4          0            84.1%
           1            15.9%
5          0            93.9%
           1             6.1%
```

Insight

- The education level of employees does not really affect their exit from the company. Employees with category 5 education levels do have a lower attrition rate. There is a general trend of increasing attrition from category 5 (lowest attrition) to category 1 (highest attrition). This trend is visible except in category 2.

How does satisfaction with the work environment impact employee turnover?

```
env_attrition_counts = (
    df.groupby('EnvironmentSatisfaction')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
env_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Environment Satisfaction")
plt.xlabel("Environment Satisfaction")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



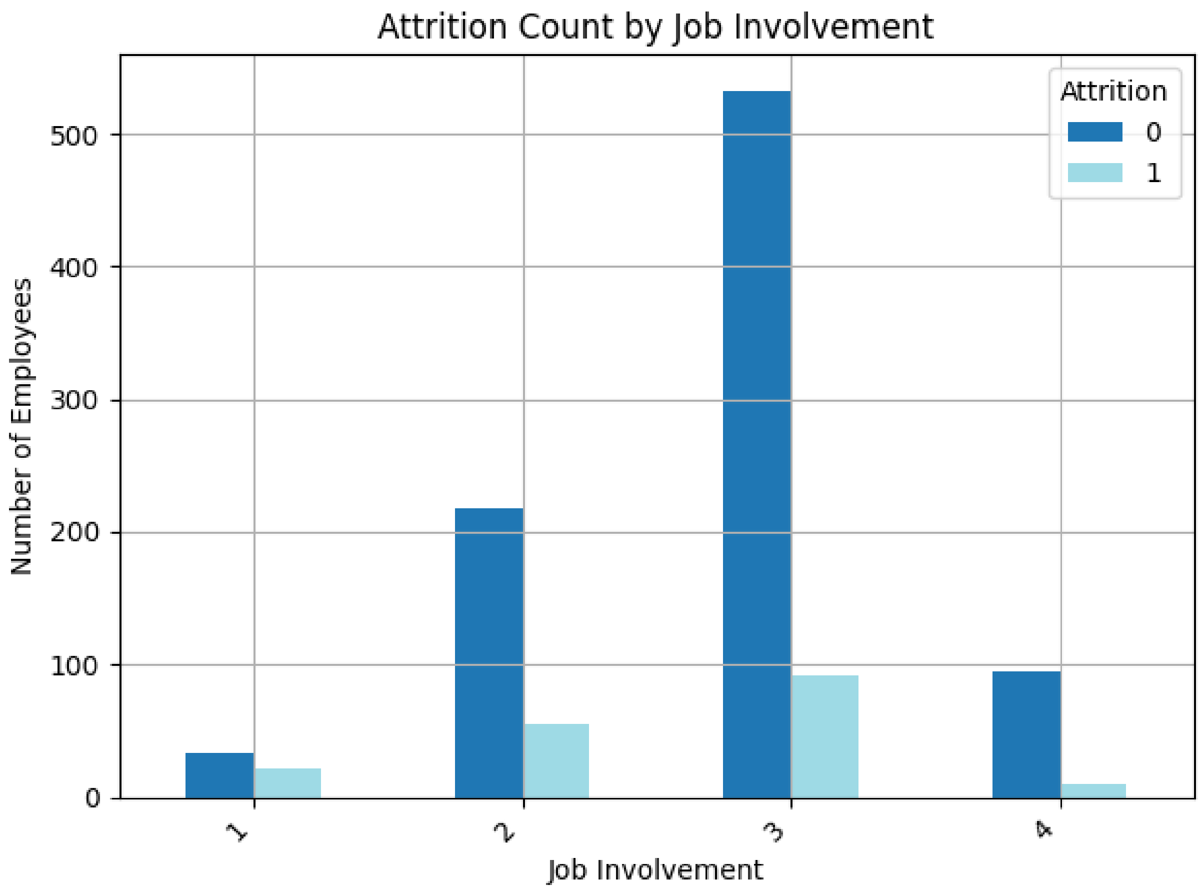
```
env_sts_att = df.groupby('EnvironmentSatisfaction')
['Attrition'].value_counts()
print("Attrition According to Environment Satisfaction\n")
print(env_sts_att)

Attrition According to Environment Satisfaction

EnvironmentSatisfaction  Attrition
1                        0            152
                         1             57
2                        0            165
                         1             35
3                        0            288
                         1             47
4                        0            274
                         1             40
Name: count, dtype: int64

env_sts_count = df['EnvironmentSatisfaction'].value_counts()

attrition_pct = np.round((env_sts_att / env_sts_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Attrition According to
Environment Satisfaction\n")
print(attrition_pct.to_string())

Total Attrition Percentage According to Attrition According to
Environment Satisfaction

EnvironmentSatisfaction  Attrition
1                        0            72.7%
                         1            27.3%
2                        0            82.5%
                         1            17.5%
3                        0            86.0%
                         1            14.0%
4                        0            87.3%
                         1            12.7%
```

Insight

- Employees' satisfaction with their work environment is one of the factors they leave the company. The less satisfied they are with their work environment, the greater the potential for them to leave the company.

Does feeling engaged in their work affect employee retention?

```
job_inv_attrition_counts = (
    df.groupby('JobInvolvement')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
job_inv_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Job Involvement")
plt.xlabel("Job Involvement")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



```
att_count = df.groupby('JobInvolvement')['Attrition'].value_counts()

# Calculate total employee count by job involvement
job_inv_count = df['JobInvolvement'].value_counts()

# Calculate attrition rate (percentage)
attrition_pct = np.round((att_count / job_inv_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Job Involvement\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Job Involvement

JobInvolvement  Attrition
1               0            60.0%
                1            40.0%
2               0            79.9%
                1            20.1%
3               0            85.3%
                1            14.7%
4               0            90.5%
                1             9.5%
```

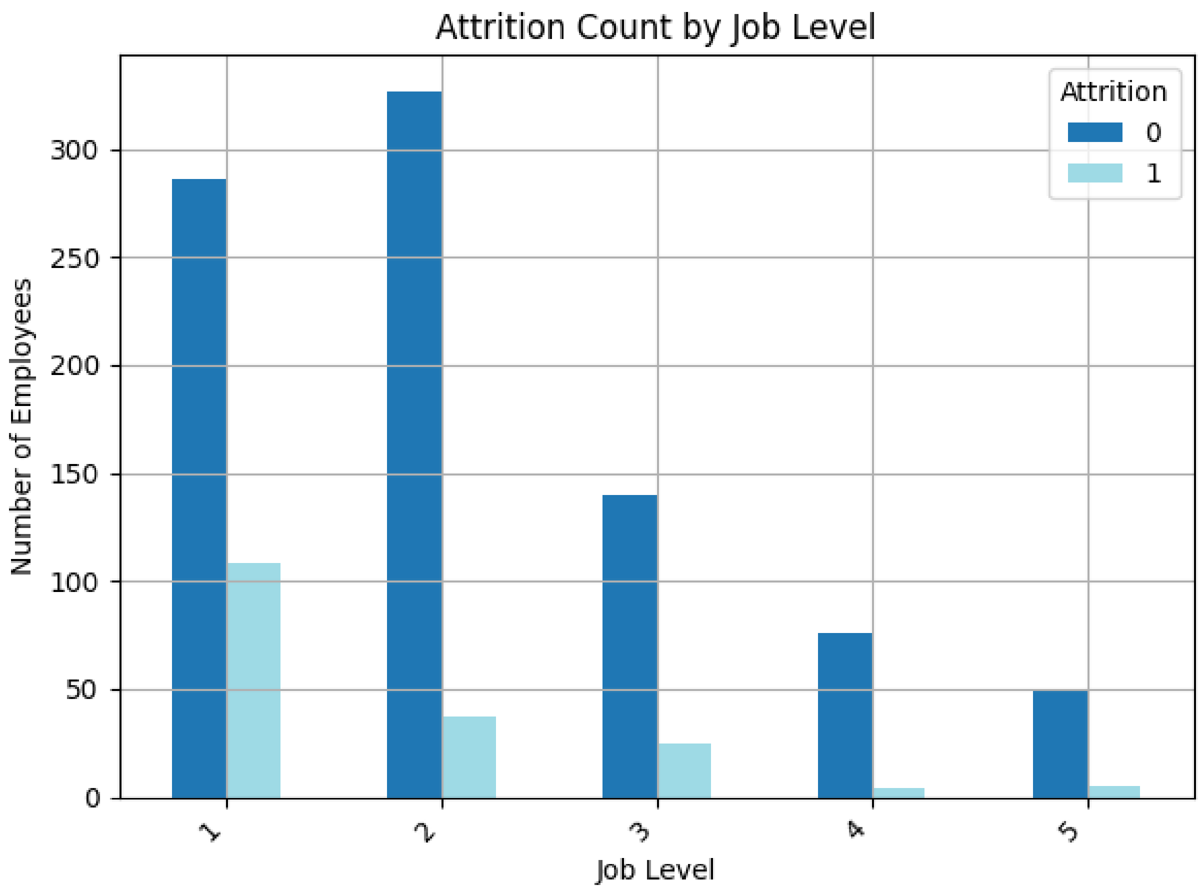
Insight

- Feeling engaged with their work is one of the factors they stay with the company. The more they contribute, the less likely they are to leave the company. This can be caused by employees' desire to grow and contribute to their respective job desks.

Do employees at higher job levels experience lower attrition rates?

```
job_level_counts = (
    df.groupby('JobLevel')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
job_level_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Job Level")
plt.xlabel("Job Level")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



```
job_level_att = df.groupby('JobLevel')['Attrition'].value_counts()
print("Attrition per Job Level\n")
print(job_level_att)
```

```
Attrition per Job Level

JobLevel  Attrition
1         0            286
          1            108
2         0            327
          1             37
3         0            140
          1             25
4         0             76
          1              4
5         0             50
          1              5
Name: count, dtype: int64
```

```
job_level_count = df['JobLevel'].value_counts()

# Calculate attrition rate (percentage)
attrition_pct = np.round((job_level_att / job_level_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Job Level\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Job Level

JobLevel  Attrition
1         0            72.6%
          1            27.4%
2         0            89.8%
          1            10.2%
3         0            84.8%
          1            15.2%
4         0            95.0%
          1             5.0%
5         0            90.9%
          1             9.1%
```

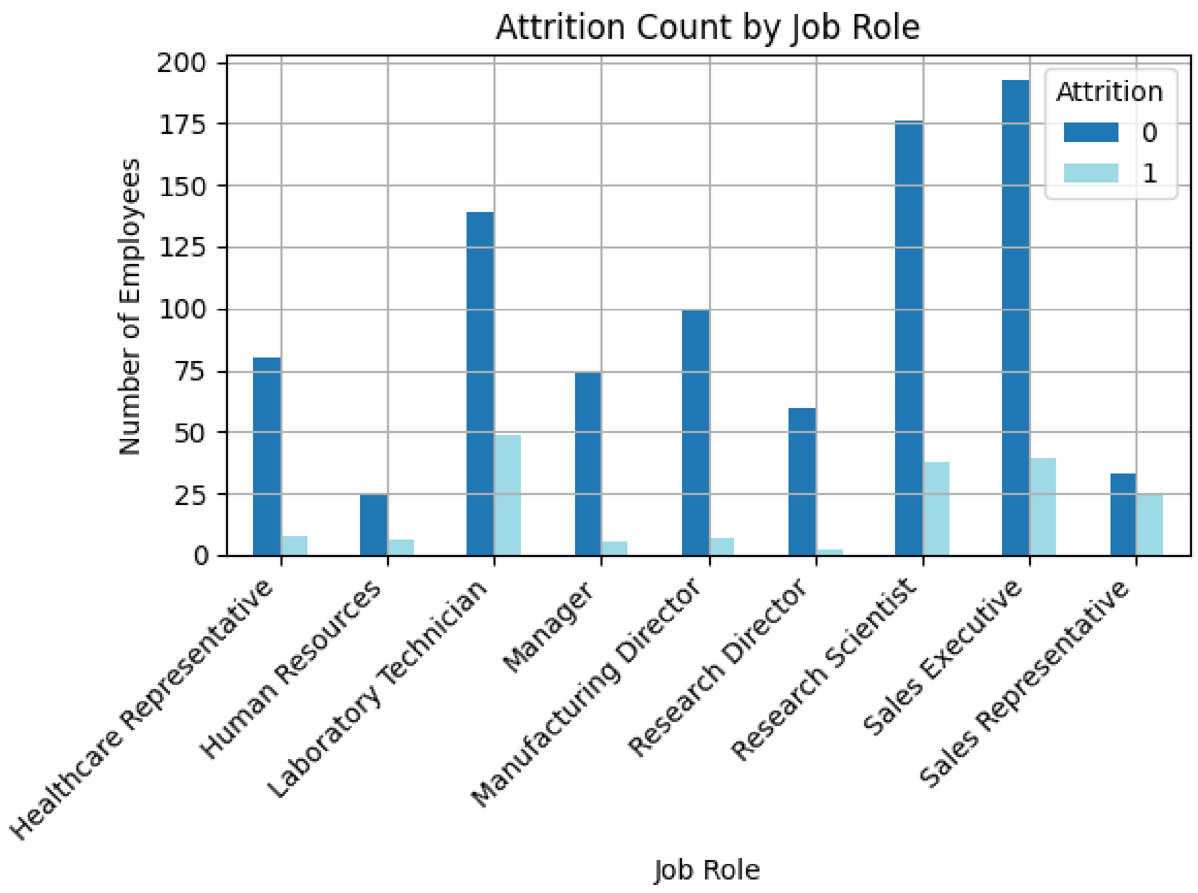
Insight

- Although the trend goes up and down, there is a tendency for higher-ranking employees to stay with the company.

Do employees with specific job roles experience lower attrition rates?

```
education_attrition_counts = (
    df.groupby('JobRole')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
education_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Job Role")
plt.xlabel("Job Role")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



```
job_role_att = df.groupby('JobRole')['Attrition'].value_counts()
print("Attrition per Job Role\n")
print(job_role_att)
```

```
Attrition per Job Role

JobRole                    Attrition
Healthcare Representative  0             80
                           1              8
Human Resources            0             24
                           1              6
Laboratory Technician      0            139
                           1             49
Manager                    0             74
                           1              5
Manufacturing Director     0            100
                           1              7
Research Director          0             60
                           1              2
Research Scientist         0            176
                           1             38
Sales Executive            0            193
```

```
                           1             39
Sales Representative       0             33
                           1             25
Name: count, dtype: int64
```

```
job_role_count = df['JobRole'].value_counts()

attrition_pct = np.round((job_role_att / job_role_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Job Role\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Job Role

JobRole                    Attrition
Healthcare Representative  0            90.9%
                           1             9.1%
Human Resources            0            80.0%
                           1            20.0%
Laboratory Technician      0            73.9%
                           1            26.1%
Manager                    0            93.7%
                           1             6.3%
Manufacturing Director     0            93.5%
                           1             6.5%
Research Director          0            96.8%
                           1             3.2%
Research Scientist         0            82.2%
                           1            17.8%
Sales Executive            0            83.2%
                           1            16.8%
Sales Representative       0            56.9%
                           1            43.1%
```

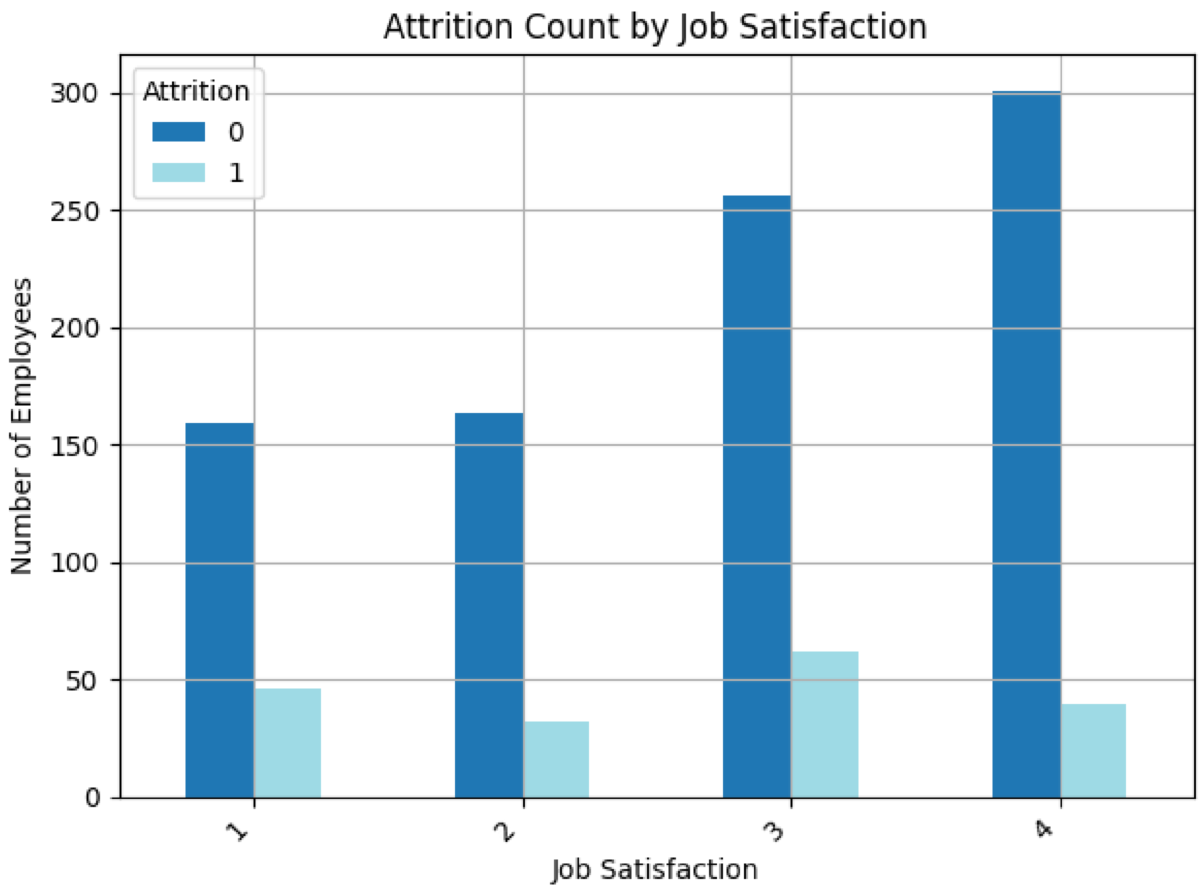
Insight

- The positions of sales representative, sales executive, human resources, laboratory technician, and research scientist experience considerable attrition compared to other positions. The highest attrition occurs for employees in the sales representative position which is likely due to high pressure.

Is there a clear correlation between job satisfaction and employee turnover rates?

```
job_sts_attrition_counts = (
    df.groupby('JobSatisfaction')
['Attrition'].value_counts().unstack(fill_value=0)
)
```

```
plt.figure(figsize=(8, 6))
job_sts_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Job Satisfaction")
plt.xlabel("Job Satisfaction")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



```
job_sts_att = df.groupby('JobSatisfaction')
['Attrition'].value_counts()
print("Attrition per Job Satisfaction\n")
print(job_sts_att)

Attrition per Job Satisfaction

JobSatisfaction  Attrition
1                0            159
```

```
                     1               46
2                    0              163
                     1               32
3                    0              256
                     1               62
4                    0              301
                     1               39
Name: count, dtype: int64
```

```
job_sts_count = df['JobSatisfaction'].value_counts()

attrition_pct = np.round((job_sts_att / job_sts_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Job Satisfaction\n")
print(attrition_pct.to_string())
```

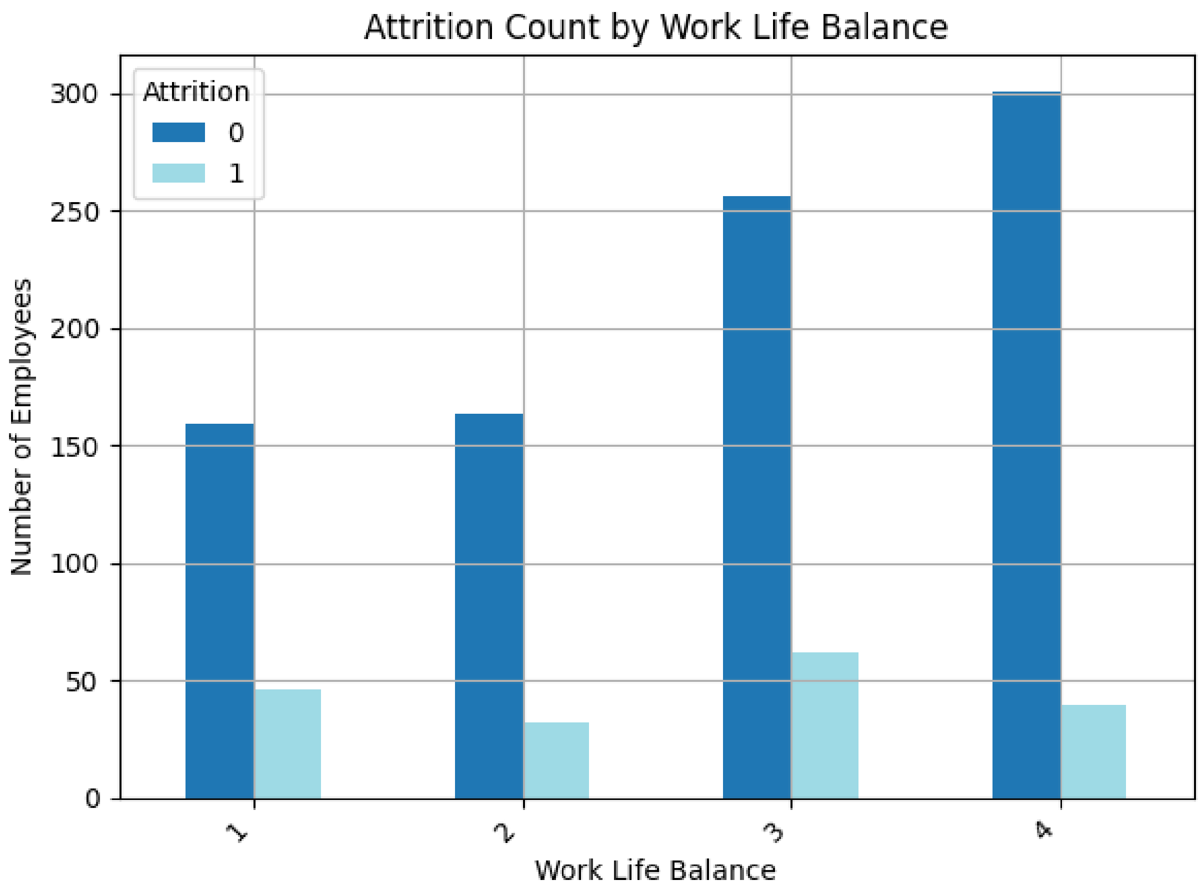
```
Total Attrition Percentage According to Job Satisfaction

JobSatisfaction  Attrition
1                0              77.6%
                 1              22.4%
2                0              83.6%
                 1              16.4%
3                0              80.5%
                 1              19.5%
4                0              88.5%
                 1              11.5%
```

Insight

- Employees' satisfaction with their jobs has a role to play in their chances of leaving the company. The more satisfied they are with their jobs, the more likely employees are to stay.

Does a healthy work-life balance reduce the risk of employees leaving?

```
wlb_attrition_counts = (
    df.groupby('WorkLifeBalance')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
job_sts_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Work Life Balance")
plt.xlabel("Work Life Balance")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



```
wlb_att = df.groupby('WorkLifeBalance')['Attrition'].value_counts()
print("Attrition According to Work Life Balance\n")
print(wlb_att)

Attrition According to Work Life Balance

WorkLifeBalance  Attrition
1                0             38
                 1             18
2                0            206
                 1             45
3                0            544
                 1             94
4                0             91
                 1             22
Name: count, dtype: int64
```

```
wlb_count = df['WorkLifeBalance'].value_counts()

attrition_pct = np.round((wlb_att / wlb_count) * 100, 1).apply(lambda
x: f"{x:.1f}%")

print("Total Attrition Percentage According to Work Life Balance\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Work Life Balance

WorkLifeBalance  Attrition
1                0            67.9%
                 1            32.1%
2                0            82.1%
                 1            17.9%
3                0            85.3%
                 1            14.7%
4                0            80.5%
                 1            19.5%
```

Insight

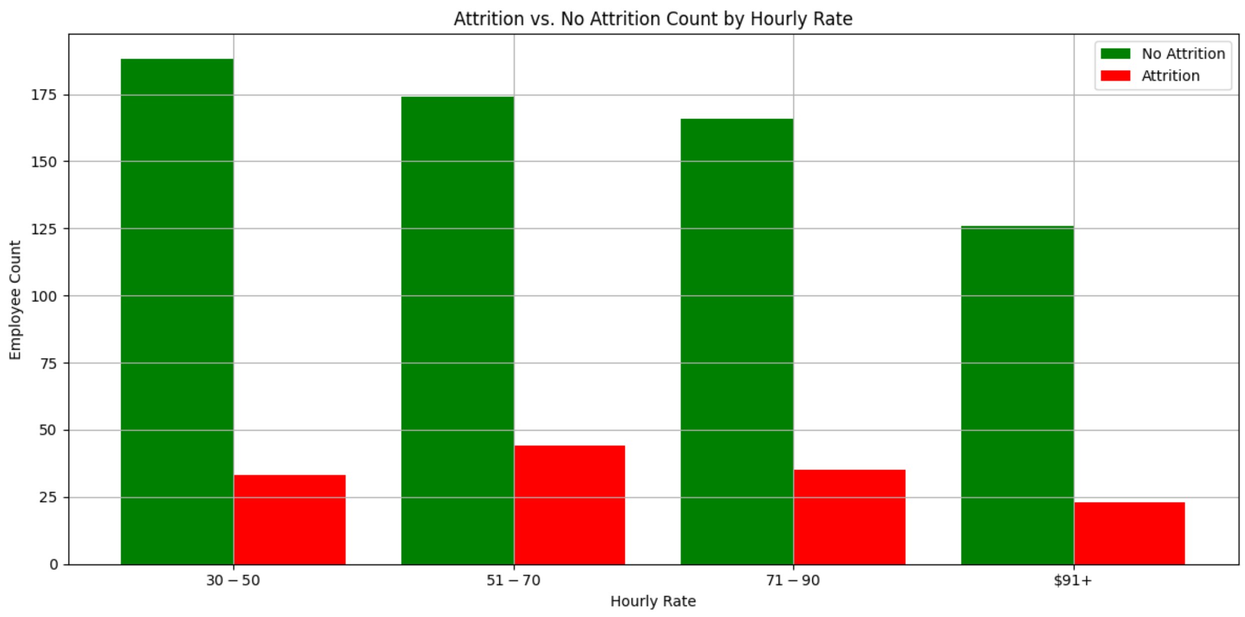
- Work life balance moderately influences an employee's decision to resign. The better the work-life balance, the less likely employees are to leave the company although attrition increases for employees who score 4.

Do higher paid employees have a lower attrition rate?

Hourly

```
# Define hourly rate ranges
hourly_rate_ranges = {
    '$30 - $50': "HourlyRate >= 30 & HourlyRate <= 45",
    '$51 - $70': "HourlyRate > 46 & HourlyRate <= 60",
    '$71 - $90': "HourlyRate > 61 & HourlyRate <= 75",
    '$91+': "HourlyRate > 91"
}

attrition_counts = {}
no_attrition_counts = {}

for range_label, condition in hourly_rate_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[range_label] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[range_label] = len(filtered_data)

plt.figure(figsize=(12, 6))
```

```
hourly_rate_labels = list(hourly_rate_ranges.keys())
x = np.arange(len(hourly_rate_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Hourly Rate")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, hourly_rate_labels)
plt.title("Attrition vs. No Attrition Count by Hourly Rate")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



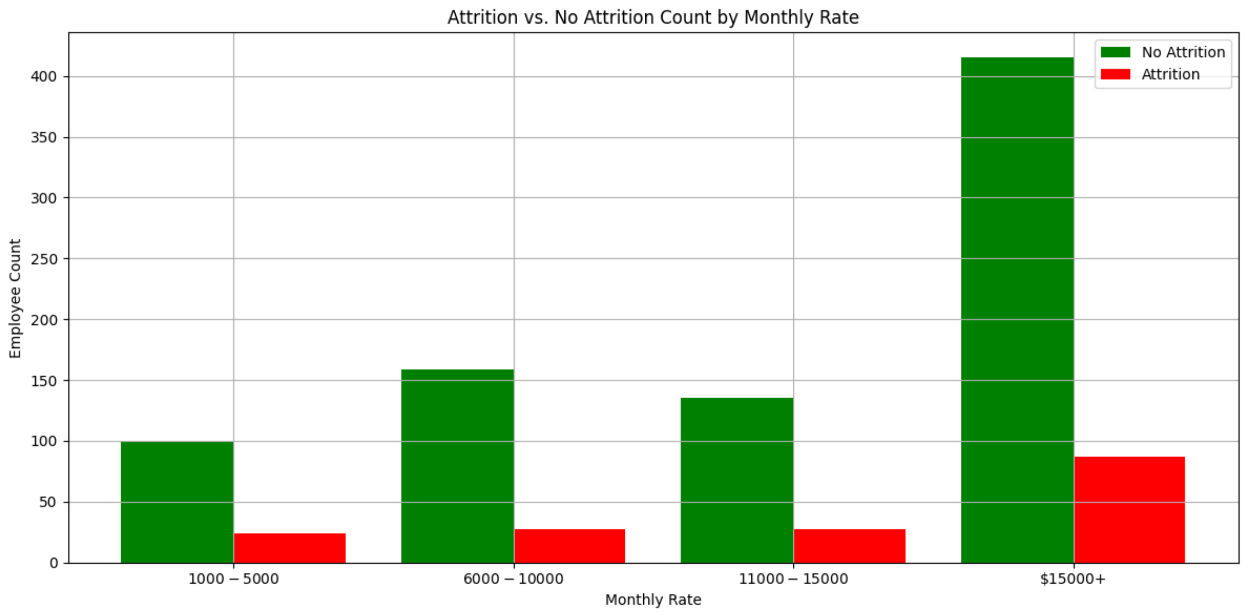
```
attrition_by_hourly_rate = {}
for range_label, condition in hourly_rate_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_hourly_rate[range_label] = attrition_counts.fillna(0)

print("Attrition by Hourly Rate\n")
```

```
for range_label, counts in attrition_by_hourly_rate.items():
  print(f"{range_label}: {counts}\n")

Attrition by Hourly Rate

$30 - $50: Attrition
0    188
1     33
Name: count, dtype: int64

$51 - $70: Attrition
0    174
1     44
Name: count, dtype: int64

$71 - $90: Attrition
0    166
1     35
Name: count, dtype: int64

$91+: Attrition
0    126
1     23
Name: count, dtype: int64


attrition_by_hourly_rate = {}
for range_label, condition in hourly_rate_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_hourly_rate[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    attrition_by_hourly_rate[range_label][attn_value] =
round(percentage, 1)

print("Attrition Percentage by Hourly Rate\n")
for range_label, attrition_percentages in
attrition_by_hourly_rate.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")
  print()

Attrition Percentage by Hourly Rate

$30 - $50:
	0: 85.1%
	1: 14.9%
```

```
$51 - $70:
      0: 79.8%
      1: 20.2%

$71 - $90:
      0: 82.6%
      1: 17.4%

$91+:
      0: 84.6%
      1: 15.4%
```

Monthly

```
# Define monthly rate ranges
monthly_rate_ranges = {
    '$1000 - $5000': "MonthlyRate >= 1000 & MonthlyRate <= 5000",
    '$6000 - $10000': "MonthlyRate > 6000 & MonthlyRate <= 10000",
    '$11000 - $15000': "MonthlyRate > 11000 & MonthlyRate <= 15000",
    '$15000+': "MonthlyRate > 15000"
}

attrition_counts = {}
no_attrition_counts = {}

for range_label, condition in monthly_rate_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[range_label] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[range_label] = len(filtered_data)

plt.figure(figsize=(12, 6))
monthly_rate_labels = list(monthly_rate_ranges.keys())
x = np.arange(len(monthly_rate_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Monthly Rate")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, monthly_rate_labels)
plt.title("Attrition vs. No Attrition Count by Monthly Rate")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
attrition_by_monthly_rate = {}
for range_label, condition in monthly_rate_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_monthly_rate[range_label] = attrition_counts.fillna(0)

print("Attrition by Monthly Rate\n")
for range_label, counts in attrition_by_monthly_rate.items():
  print(f"{range_label}: {counts}\n")
```

```
Attrition by Monthly Rate

$1000 - $5000: Attrition
0    99
1    24
Name: count, dtype: int64

$6000 - $10000: Attrition
0    159
1     27
Name: count, dtype: int64
```

```
$11000 - $15000: Attrition
0    135
1     27
Name: count, dtype: int64

$15000+: Attrition
0    415
1     87
Name: count, dtype: int64
```

```
attrition_by_monthly_rate = {}
for range_label, condition in monthly_rate_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_monthly_rate[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    attrition_by_monthly_rate[range_label][attn_value] =
round(percentage, 1)

print("Attrition Percentage by Monthly Rate\n")
for range_label, attrition_percentages in
attrition_by_monthly_rate.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")
  print()
```

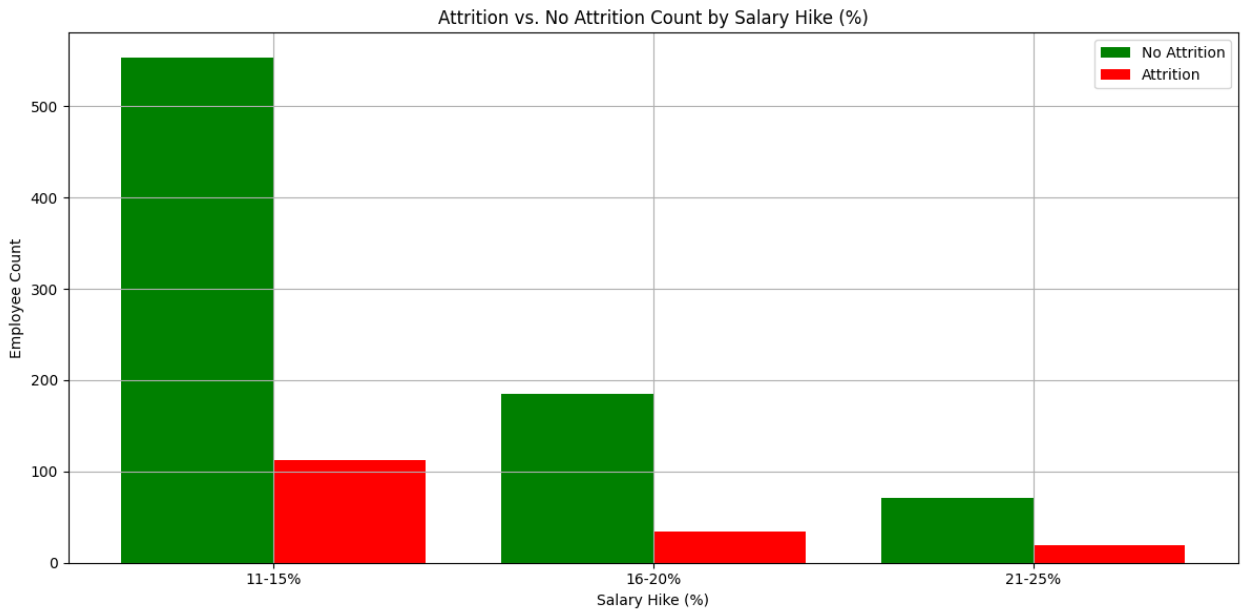
```
Attrition Percentage by Monthly Rate

$1000 - $5000:
	0: 80.5%
	1: 19.5%

$6000 - $10000:
	0: 85.5%
	1: 14.5%

$11000 - $15000:
	0: 83.3%
	1: 16.7%

$15000+:
	0: 82.7%
	1: 17.3%
```

Insight

- An employee's salary (hourly or monthly) does not greatly influence their decision to leave the company.

Do salary increases impact employee attrition rates?

```
# Define salary hike percentage ranges
salary_hike_ranges = {
    '11-15%': "PercentSalaryHike >= 11 & PercentSalaryHike <= 15",
    '16-20%': "PercentSalaryHike > 16 & PercentSalaryHike <= 20",
    '21-25%': "PercentSalaryHike > 21 & PercentSalaryHike <= 25"
}

attrition_counts = {}
no_attrition_counts = {}

for range_label, condition in salary_hike_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[range_label] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[range_label] = len(filtered_data)

plt.figure(figsize=(12, 6))
salary_hike_labels = list(salary_hike_ranges.keys())
x = np.arange(len(salary_hike_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Salary Hike (%)")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, salary_hike_labels)
plt.title("Attrition vs. No Attrition Count by Salary Hike (%)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
attrition_by_salary_hike = {}
for range_label, condition in salary_hike_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_salary_hike[range_label] = attrition_counts.fillna(0)

print("Attrition by Salary Hike (%)\n")
for range_label, counts in attrition_by_salary_hike.items():
  print(f"{range_label}: {counts}\n")
```

```
Attrition by Salary Hike (%)

11-15%: Attrition
0    553
1    112
Name: count, dtype: int64

16-20%: Attrition
0    185
1     34
Name: count, dtype: int64

21-25%: Attrition
0    71
1    19
Name: count, dtype: int64
```

```
attrition_by_salary_hike = {}
for range_label, condition in salary_hike_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_salary_hike[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    attrition_by_salary_hike[range_label][attn_value] =
round(percentage, 1)

print("Attrition Percentage by Salary Hike (%)\n")
for range_label, attrition_percentages in
attrition_by_salary_hike.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")
```

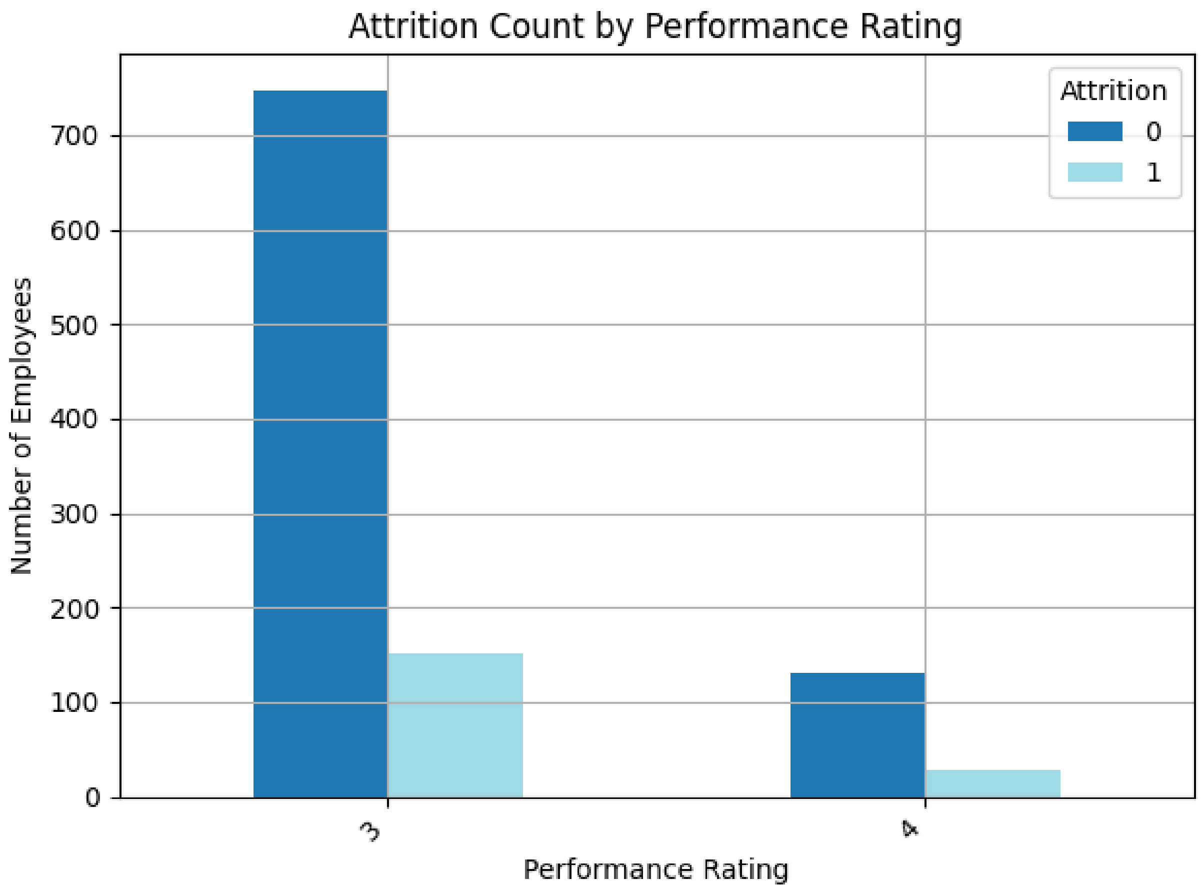
```
Attrition Percentage by Salary Hike (%)

11-15%:
	0: 83.2%
	1: 16.8%
16-20%:
	0: 84.5%
	1: 15.5%
21-25%:
	0: 78.9%
	1: 21.1%
```

Insight

- Employee salary increases do not greatly influence their decision to leave the company.

Do high-performing employees tend to stay with the company longer?

```
perf_attrition_counts = (
    df.groupby('PerformanceRating')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
perf_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Performance Rating")
plt.xlabel("Performance Rating")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



```
perf_att = df.groupby('PerformanceRating')['Attrition'].value_counts()
print("Attrition per Performance Rating\n")
print(perf_att)

Attrition per Performance Rating

PerformanceRating  Attrition
3                  0            748
                   1            151
4                  0            131
                   1             28
Name: count, dtype: int64

perf_count = df['PerformanceRating'].value_counts()

# Calculate attrition rate (percentage)
attrition_pct = np.round((perf_att / perf_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Performance Rating\n")
print(attrition_pct.to_string())

Total Attrition Percentage According to Performance Rating

PerformanceRating  Attrition
3                  0            83.2%
                   1            16.8%
4                  0            82.4%
                   1            17.6%
```

Insight

- An employee's performance rating does not greatly influence their decision to leave the company.

Does working long hours or exceeding standard working hours contribute to employee burnout and attrition?

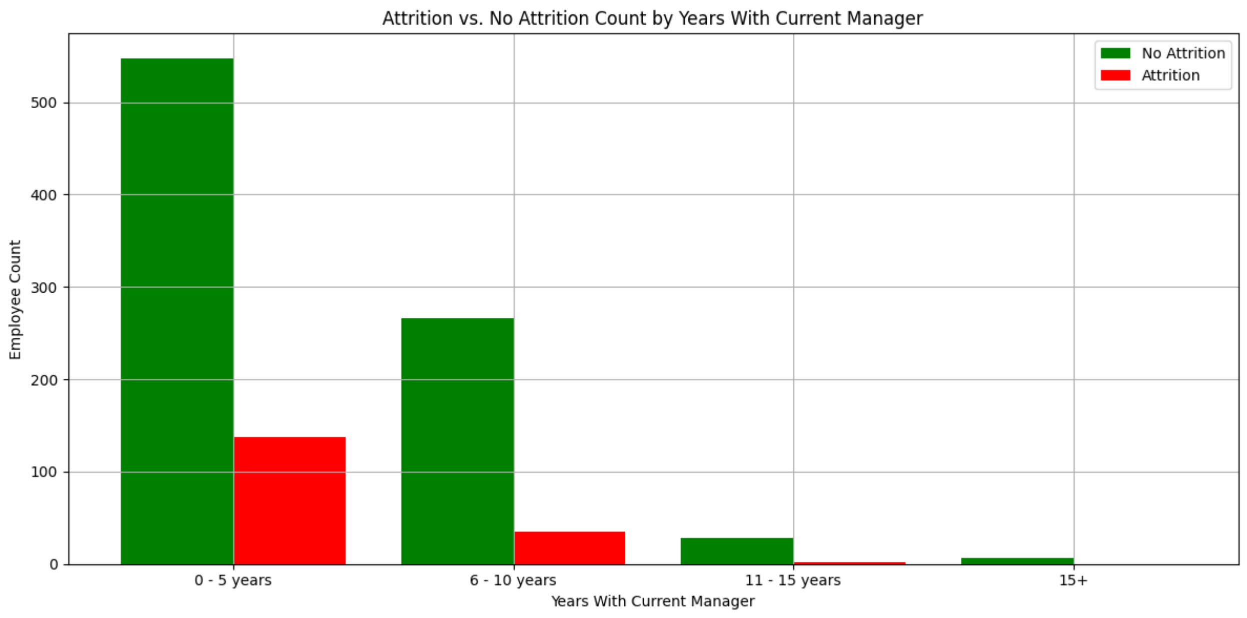
```
overtime_att = df.groupby('OverTime')['Attrition'].value_counts()
print("Attrition by Overtime\n")
print(overtime_att)

Attrition by Overtime

OverTime  Attrition
No        0            670
          1             81
Yes       0            209
          1             98
Name: count, dtype: int64

overtime_count = df['OverTime'].value_counts()

# Calculate attrition rate (percentage)
attrition_pct = np.round((overtime_att / overtime_count) * 100,
1).apply(lambda x: f"{x:.1f}%")

print("Total Attrition Percentage According to Overtime\n")
print(attrition_pct.to_string())

Total Attrition Percentage According to Overtime

OverTime  Attrition
No        0            89.2%
          1            10.8%
Yes       0            68.1%
          1            31.9%
```

Insight

- Employees who work overtime are much more likely to leave the company. This can be caused by job burnout.

Do employees with longer tenure with their current manager show lower turnover rates?

```
curr_manager_ranges = {
    '0 - 5 years': "YearsWithCurrManager >= 0 & YearsWithCurrManager
<= 5",
    '6 - 10 years': "YearsWithCurrManager > 6 & YearsWithCurrManager
<= 10",
    '11 - 15 years': "YearsWithCurrManager > 11 & YearsWithCurrManager
<= 15",
    '15+': "YearsWithCurrManager > 15"
}

attrition_counts = {}
no_attrition_counts = {}

for range_label, condition in curr_manager_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[range_label] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[range_label] = len(filtered_data)

plt.figure(figsize=(12, 6))
curr_manager_labels = list(curr_manager_ranges.keys())
x = np.arange(len(curr_manager_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Years With Current Manager")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, curr_manager_labels)
plt.title("Attrition vs. No Attrition Count by Years With Current
Manager")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
attrition_by_curr_manager = {}
for range_label, condition in curr_manager_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_curr_manager[range_label] = attrition_counts.fillna(0)

print("Attrition by Years With Current Manager\n")
for range_label, counts in attrition_by_curr_manager.items():
  print(f"{range_label}: {counts}\n")
```

```
Attrition by Years With Current Manager

0 - 5 years: Attrition
0    547
1    137
Name: count, dtype: int64

6 - 10 years: Attrition
0    266
1     35
Name: count, dtype: int64

11 - 15 years: Attrition
0    28
1     2
Name: count, dtype: int64
```

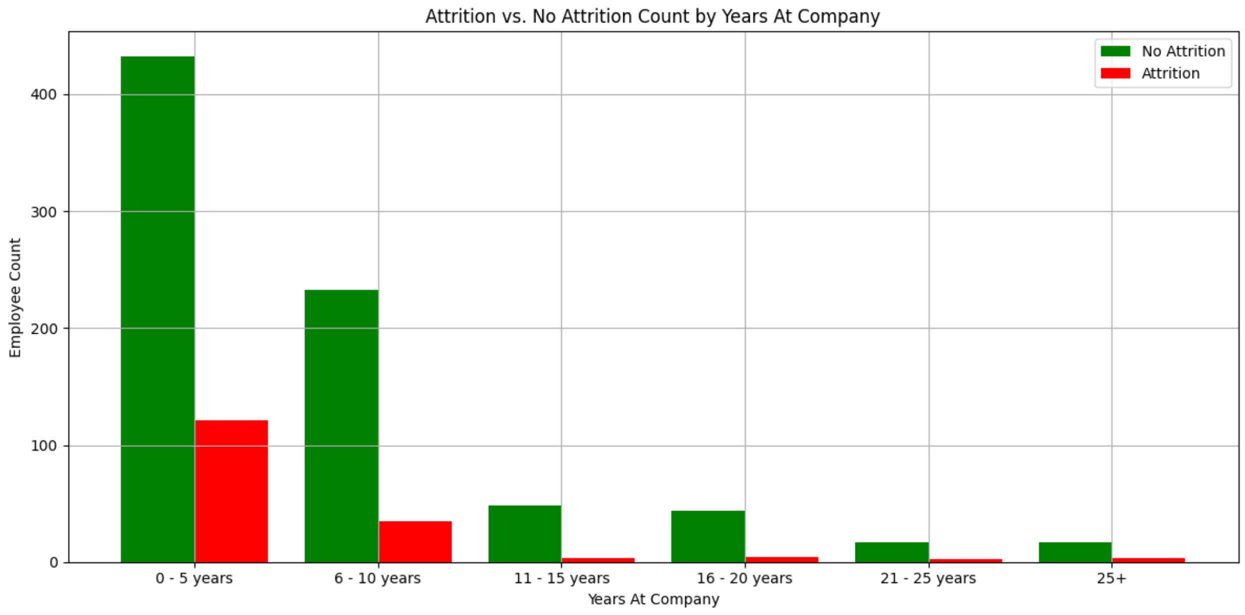
```
15+: Attrition
0    7
Name: count, dtype: int64


attrition_by_yrs_curr_mng = {}
for range_label, condition in curr_manager_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_yrs_curr_mng[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    attrition_by_yrs_curr_mng[range_label][attn_value] =
round(percentage, 1)

print("Attrition Percentage by Years Since Last Promotion\n")
for range_label, attrition_percentages in
attrition_by_yrs_curr_mng.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")

Attrition Percentage by Years Since Last Promotion

0 - 5 years:
	0: 80.0%
	1: 20.0%
6 - 10 years:
	0: 88.4%
	1: 11.6%
11 - 15 years:
	0: 93.3%
	1: 6.7%
15+:
	0: 100.0%
```

Insight

- Employees who spend less time working with their managers are more likely to leave the company. This could be due to a mismatch or lack of chemistry between the employee and manager.

Do employees with longer tenure in the company show lower turnover rates?

```
years_company_ranges = {
    '0 - 5 years': "YearsAtCompany >= 0 & YearsAtCompany <= 5",
    '6 - 10 years': "YearsAtCompany > 6 & YearsAtCompany <= 10",
    '11 - 15 years': "YearsAtCompany > 11 & YearsAtCompany <= 15",
    '16 - 20 years': "YearsAtCompany > 16 & YearsAtCompany <= 20",
    '21 - 25 years': "YearsAtCompany > 21 & YearsAtCompany <= 25",
    '25+': "YearsAtCompany > 25"
}

attrition_counts = {}
no_attrition_counts = {}

for range_label, condition in years_company_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[range_label] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[range_label] = len(filtered_data)

plt.figure(figsize=(12, 6))
yrs_company_labels = list(years_company_ranges.keys())
x = np.arange(len(years_company_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Years At Company")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, yrs_company_labels)
plt.title("Attrition vs. No Attrition Count by Years At Company")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
attrition_by_years_comp = {}
for range_label, condition in years_company_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_years_comp[range_label] = attrition_counts.fillna(0)

print("Attrition by Years At Company\n")
for range_label, counts in attrition_by_years_comp.items():
  print(f"{range_label}: {counts}\n")
```

```
Attrition by Years At Company

0 - 5 years: Attrition
0    432
1    121
Name: count, dtype: int64

6 - 10 years: Attrition
0    233
1     35
Name: count, dtype: int64

11 - 15 years: Attrition
0    48
1     3
Name: count, dtype: int64

16 - 20 years: Attrition
0    44
1     4
```

```
Name: count, dtype: int64

21 - 25 years: Attrition
0    17
1     2
Name: count, dtype: int64

25+: Attrition
0    17
1     3
Name: count, dtype: int64
```

```python
attrition_by_years_comp = {}
for range_label, condition in years_company_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  years_company_ranges[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    years_company_ranges[range_label][attn_value] = round(percentage,
1)

print("Attrition Percentage by Years At Company\n")
for range_label, attrition_percentages in
years_company_ranges.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")
```

```
Attrition Percentage by Years At Company

0 - 5 years:
	0: 78.1%
	1: 21.9%
6 - 10 years:
	0: 86.9%
	1: 13.1%
11 - 15 years:
	0: 94.1%
	1: 5.9%
16 - 20 years:
	0: 91.7%
	1: 8.3%
21 - 25 years:
	0: 89.5%
	1: 10.5%
25+:
```

```
0: 85.0%
1: 15.0%
```

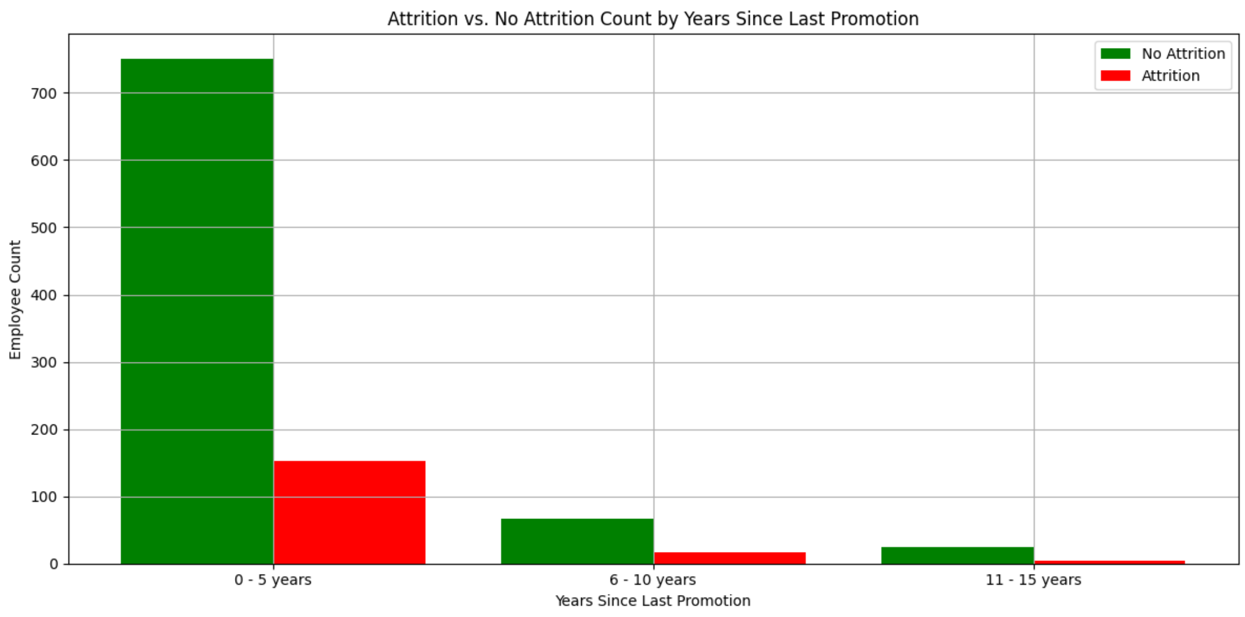
Insight

- An employee's length of time in the company does not have much influence on their chances of leaving the company.

Does staying in the same role for a long time affect employee retention?

```
years_last_promo_ranges = {
    '0 - 5 years': "YearsSinceLastPromotion >= 0 &
YearsSinceLastPromotion <= 5",
    '6 - 10 years': "YearsSinceLastPromotion > 6 &
YearsSinceLastPromotion <= 10",
    '11 - 15 years': "YearsSinceLastPromotion > 11 &
YearsSinceLastPromotion <= 15"
}

attrition_counts = {}
no_attrition_counts = {}

for range_label, condition in years_last_promo_ranges.items():
  attrition_data = df[df['Attrition'] == 1]
  filtered_data = attrition_data.query(condition)
  attrition_counts[range_label] = len(filtered_data)

  no_attrition_data = df[df['Attrition'] == 0]
  filtered_data = no_attrition_data.query(condition)
  no_attrition_counts[range_label] = len(filtered_data)

plt.figure(figsize=(12, 6))
promo_labels = list(years_last_promo_ranges.keys())
x = np.arange(len(years_last_promo_ranges))

width = 0.4
no_attrition_color = 'green'
attrition_color = 'red'

plt.bar(x, no_attrition_counts.values(), width, label='No Attrition',
color='green')
plt.bar(x + width, attrition_counts.values(), width,
label='Attrition', color='red')

plt.xlabel("Years Since Last Promotion")
plt.ylabel("Employee Count")
plt.xticks(x + width/2, promo_labels)
plt.title("Attrition vs. No Attrition Count by Years Since Last
Promotion")
```

```
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
attrition_by_last_promo = {}
for range_label, condition in years_last_promo_ranges.items():
  filtered_data = df.query(condition)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_last_promo[range_label] = attrition_counts.fillna(0)

print("Attrition by Years Since Last Promotion\n")
for range_label, counts in attrition_by_last_promo.items():
  print(f"{range_label}: {counts}\n")
```

```
Attrition by Years Since Last Promotion

0 - 5 years: Attrition
0    750
1    153
Name: count, dtype: int64

6 - 10 years: Attrition
0    67
1    16
Name: count, dtype: int64

11 - 15 years: Attrition
0    24
```

```
1      5
Name: count, dtype: int64
```

```
attrition_by_last_promo = {}
for range_label, condition in years_last_promo_ranges.items():
  filtered_data = df.query(condition)
  total_count = len(filtered_data)
  attrition_counts = filtered_data.groupby('Attrition')
['Attrition'].value_counts()
  attrition_by_last_promo[range_label] = {}
  for attn_value, count in attrition_counts.items():
    percentage = (count / total_count) * 100
    attrition_by_last_promo[range_label][attn_value] =
round(percentage, 1)

print("Attrition Percentage by Years Since Last Promotion\n")
for range_label, attrition_percentages in
attrition_by_last_promo.items():
  print(f"{range_label}:")
  for attn_value, percentage in attrition_percentages.items():
    print(f"\t{attn_value}: {percentage}%")
```

```
Attrition Percentage by Years Since Last Promotion

0 - 5 years:
     0: 83.1%
     1: 16.9%
6 - 10 years:
     0: 80.7%
     1: 19.3%
11 - 15 years:
     0: 82.8%
     1: 17.2%
```

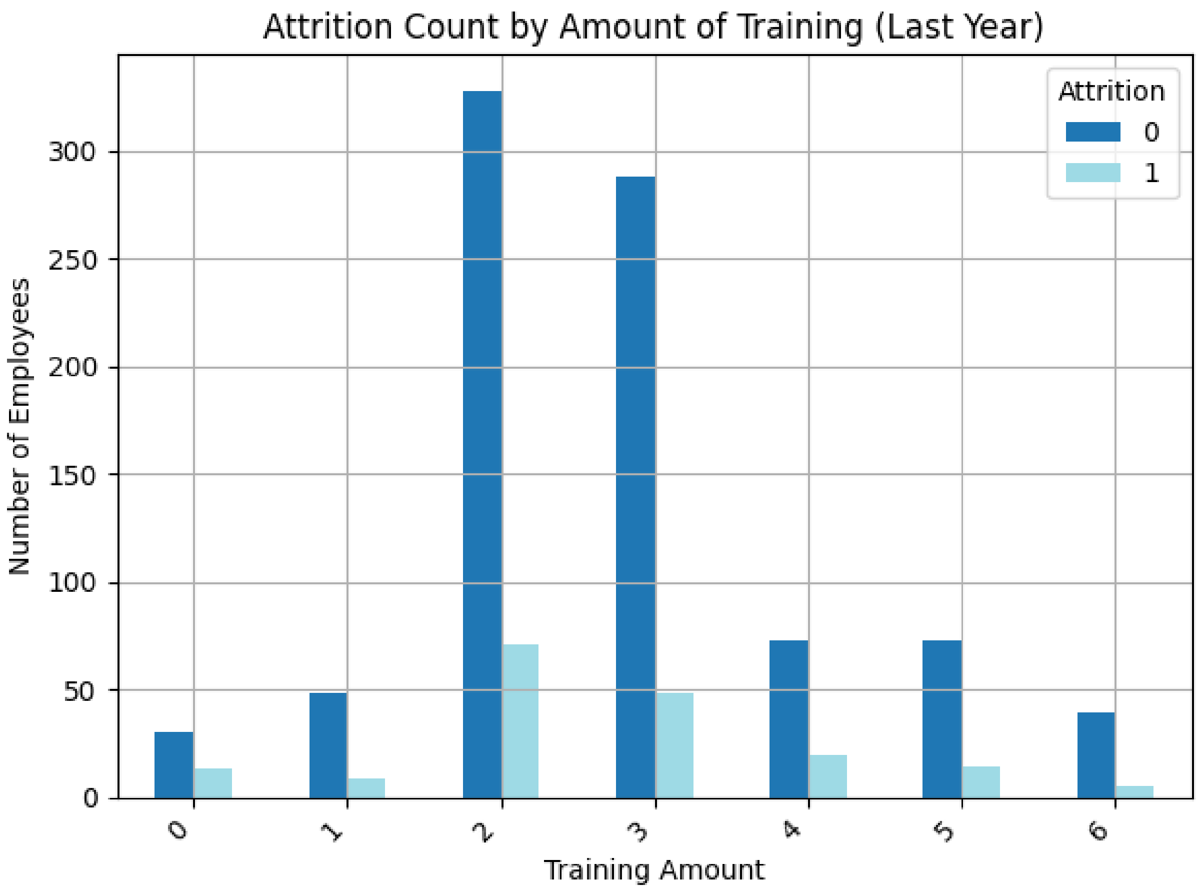
Insight

- Length of promotion does not have much influence on the chances of an employee leaving the company.

Does investment in employee training have a positive impact on retention?

```
education_attrition_counts = (
    df.groupby('TrainingTimesLastYear')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
education_attrition_counts.plot(kind='bar', colormap='tab20')
```

```
plt.title("Attrition Count by Amount of Training (Last Year)")
plt.xlabel("Training Amount")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



```
tly_att = df.groupby('TrainingTimesLastYear')
['Attrition'].value_counts()
print("Attrition According to Training Times Last Year\n")
print(tly_att)

Attrition According to Training Times Last Year

TrainingTimesLastYear  Attrition
0                      0             30
                       1             13
1                      0             48
```

```
                          1             9
2                         0           328
                          1            71
3                         0           288
                          1            48
4                         0            73
                          1            19
5                         0            73
                          1            14
6                         0            39
                          1             5
Name: count, dtype: int64
```

```
tly_count = df['TrainingTimesLastYear'].value_counts()

attrition_pct = np.round((tly_att / tly_count) * 100, 1).apply(lambda
x: f"{x:.1f}%")

print("Total Attrition Percentage According to Training Times Last
Year\n")
print(attrition_pct.to_string())
```

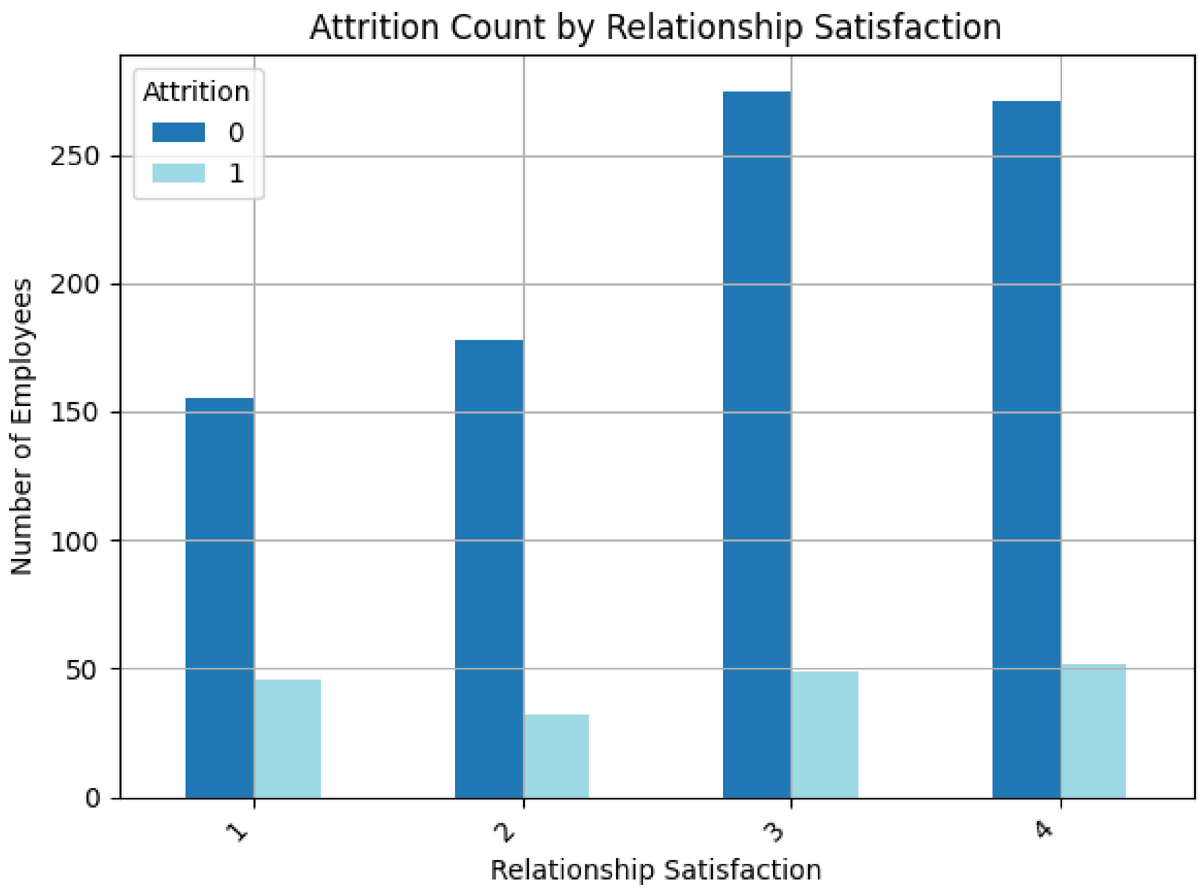
```
Total Attrition Percentage According to Training Times Last Year

TrainingTimesLastYear  Attrition
0                      0            69.8%
                       1            30.2%
1                      0            84.2%
                       1            15.8%
2                      0            82.2%
                       1            17.8%
3                      0            85.7%
                       1            14.3%
4                      0            79.3%
                       1            20.7%
5                      0            83.9%
                       1            16.1%
6                      0            88.6%
                       1            11.4%
```

Insight

- The amount of employee training in a year does not really affect the chances of employee departure. The difference is only seen in employees who are not given training at all. If employees are given training, they are less likely to leave.

Do relationships between coworkers have an effect on employee attrition?

```
relationship_attrition_counts = (
    df.groupby('RelationshipSatisfaction')
['Attrition'].value_counts().unstack(fill_value=0)
)

plt.figure(figsize=(8, 6))
relationship_attrition_counts.plot(kind='bar', colormap='tab20')
plt.title("Attrition Count by Relationship Satisfaction")
plt.xlabel("Relationship Satisfaction")
plt.ylabel("Number of Employees")
plt.xticks(rotation=45, ha='right')
plt.legend(title='Attrition')
plt.tight_layout()
plt.grid(True)
plt.show()

<Figure size 800x600 with 0 Axes>
```



```
rls_att = df.groupby('RelationshipSatisfaction')
['Attrition'].value_counts()
print("Attrition According to Relationship Satisfaction\n")
print(rls_att)
```

```
Attrition According to Relationship Satisfaction

RelationshipSatisfaction  Attrition
1                         0            155
                          1             46
2                         0            178
                          1             32
3                         0            275
                          1             49
4                         0            271
                          1             52
Name: count, dtype: int64
```

```
rls_count = df['RelationshipSatisfaction'].value_counts()

attrition_pct = np.round((rls_att / rls_count) * 100, 1).apply(lambda
x: f"{x:.1f}%")

print("Total Attrition Percentage According to Relationship
Satisfaction\n")
print(attrition_pct.to_string())
```

```
Total Attrition Percentage According to Relationship Satisfaction

RelationshipSatisfaction  Attrition
1                         0            77.1%
                          1            22.9%
2                         0            84.8%
                          1            15.2%
3                         0            84.9%
                          1            15.1%
4                         0            83.9%
                          1            16.1%
```

Insight

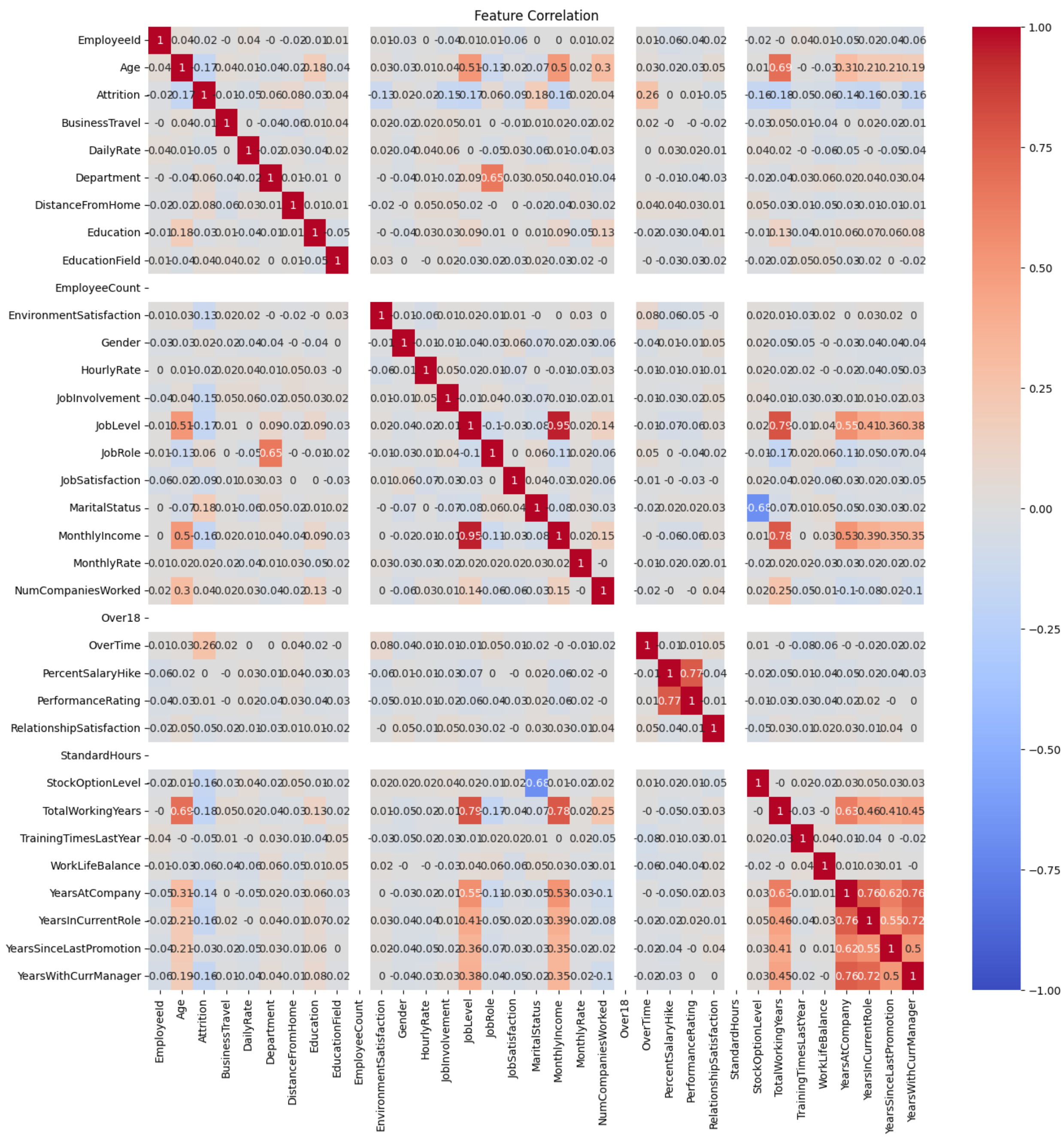
- Good relationships with coworkers have a modest effect on the likelihood of an employee leaving the company. However, the better an employee's relationship with their coworkers, the less likely they are to leave (although the trend increases slightly for employees who score 4).

Balancing Data + Splitting the dataset into the Training set and Test set

```python
from sklearn import preprocessing

category_col = df.select_dtypes(include=['object'])
labelEncoder = preprocessing.LabelEncoder()

mapping_dict = {}
for col in category_col:
    df[col] = labelEncoder.fit_transform(df[col])

    le_name_mapping = dict(zip(labelEncoder.classes_,

labelEncoder.transform(labelEncoder.classes_)))

    mapping_dict[col] = le_name_mapping
print(mapping_dict)
```

```
{'BusinessTravel': {'Non-Travel': 0, 'Travel_Frequently': 1,
'Travel_Rarely': 2}, 'Department': {'Human Resources': 0, 'Research &
Development': 1, 'Sales': 2}, 'EducationField': {'Human Resources': 0,
'Life Sciences': 1, 'Marketing': 2, 'Medical': 3, 'Other': 4,
'Technical Degree': 5}, 'Gender': {'Female': 0, 'Male': 1}, 'JobRole':
{'Healthcare Representative': 0, 'Human Resources': 1, 'Laboratory
Technician': 2, 'Manager': 3, 'Manufacturing Director': 4, 'Research
Director': 5, 'Research Scientist': 6, 'Sales Executive': 7, 'Sales
Representative': 8}, 'MaritalStatus': {'Divorced': 0, 'Married': 1,
'Single': 2}, 'Over18': {'Y': 0}, 'OverTime': {'No': 0, 'Yes': 1}}
```

```python
#Correlations between features
matrix = df.corr().round(2)
plt.figure(figsize=(16,16))
sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0,
cmap='coolwarm')
plt.title("Feature Correlation")
plt.show()
```



```python
target_column = 'Attrition'
correlation_threshold = 0.05

correlation_matrix = df.corr()

# Filter columns where the absolute correlation with the target column
is greater than the threshold
high_correlation_columns =
correlation_matrix.index[abs(correlation_matrix['Attrition']) >
correlation_threshold]
```

```
# Create a new DataFrame with only the columns that have high
correlation
filtered_df = df[high_correlation_columns]

# Display the filtered DataFrame
print(filtered_df)

      Age  Attrition  DailyRate  Department  DistanceFromHome  \
1      37          1       1141           1                11
2      51          1       1323           1                 4
3      42          0        555           2                26
6      40          0       1124           2                 1
7      55          1        725           1                 2
...   ...        ...        ...         ...               ...
1464   28          1       1366           1                24
1465   38          0        168           1                 1
1467   28          1       1485           1                12
1468   40          0        458           1                16
1469   19          1        602           2                 1

      EnvironmentSatisfaction  JobInvolvement  JobLevel  JobRole  \
1                           1               1         2        0
2                           1               3         1        6
3                           3               3         4        7
6                           2               1         2        7
7                           4               3         5        3
...                       ...             ...       ...      ...
1464                        2               2         3        0
1465                        3               3         3        4
1467                        3               3         1        2
1468                        3               3         1        6
1469                        3               1         1        8

      JobSatisfaction  MaritalStatus  MonthlyIncome  OverTime  \
1                   2              1           4777         0
2                   3              1           2461         1
3                   2              1          13525         0
6                   4              1           7457         1
7                   1              1          19859         1
...               ...            ...            ...       ...
1464                1              2           8722         0
1465                3              2           7861         1
1467                4              1           2515         1
1468                3              0           3544         0
1469                1              2           2325         0

      RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
1                            1                 0                 15
2                            3                 3                 18
```

```
3                                  4               1                    23
6                                  3               3                     6
7                                  4               1                    24
...                              ...             ...                   ...
1464                               1               0                    10
1465                               4               0                    10
1467                               4               0                     1
1468                               2               1                     6
1469                               1               0                     1

      WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
1                   1               1                   0
2                   4              10                   0
3                   4              20                   4
6                   2               4                   3
7                   3               5                   2
...               ...             ...                 ...
1464                2              10                   7
1465                4               1                   0
1467                2               1                   1
1468                3               4                   2
1469                4               0                   0

      YearsWithCurrManager
1                        0
2                        7
3                        8
6                        2
7                        4
...                    ...
1464                     9
1465                     0
1467                     0
1468                     0
1469                     0

[1058 rows x 20 columns]

# Defining x and y
x = filtered_df.drop(columns=['Attrition'])
y = filtered_df['Attrition']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 42)

from imblearn.over_sampling import SMOTE
# Define oversampling strategy
SMOTE = SMOTE()
```

```python
#fit and apply the transform
x_train, y_train = SMOTE.fit_resample(x_train, y_train)
```

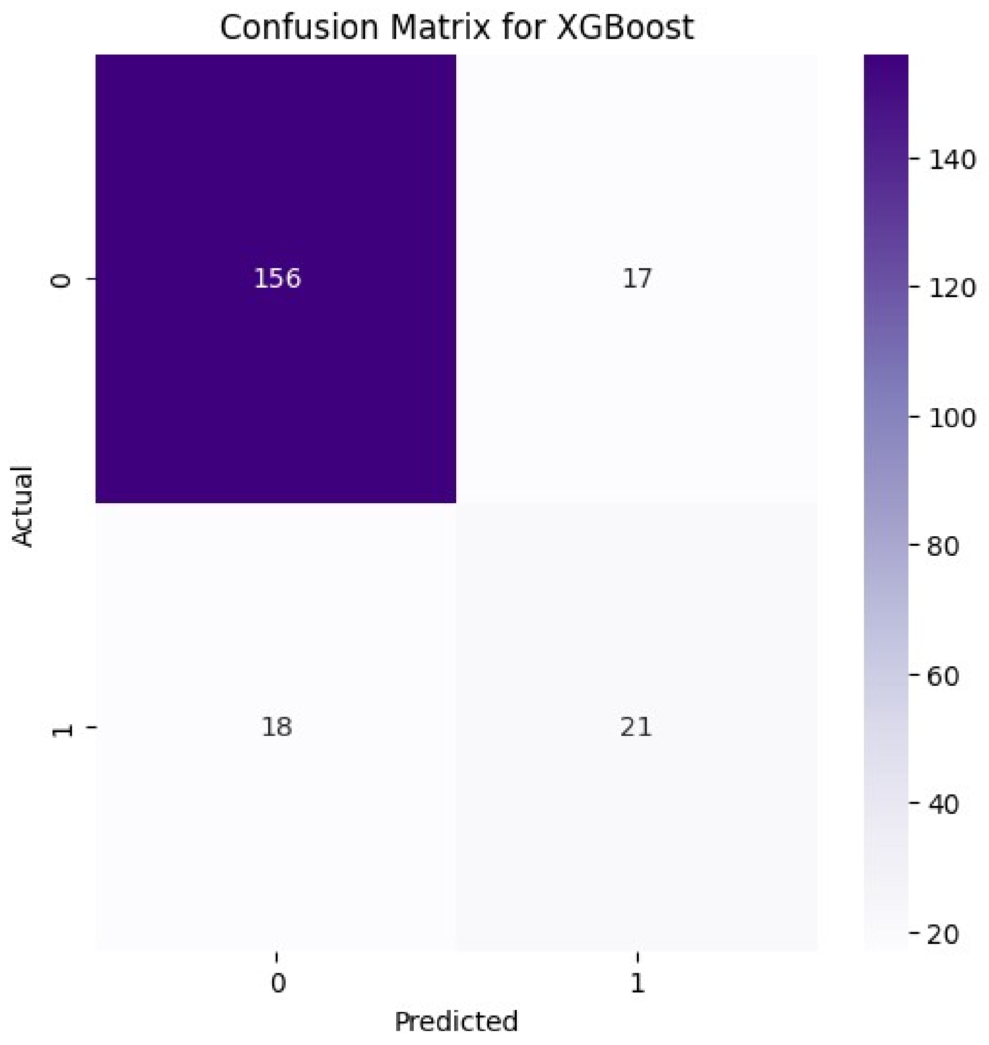
Machine Learning

XGBoost

```python
from xgboost import XGBClassifier

classifier_xgb = XGBClassifier()
classifier_xgb.fit(x_train, y_train)
y_pred_xgb = classifier_xgb.predict(x_test)

print('Training-set accuracy score:', classifier_xgb.score(x_train,
y_train))
print('Test-set accuracy score:', classifier_xgb.score(x_test,
y_test))
```

```
Training-set accuracy score: 1.0
Test-set accuracy score: 0.8349056603773585
```

```python
from sklearn.metrics import confusion_matrix, classification_report,
roc_auc_score

plt.figure(figsize=(6,6))
sns.heatmap(confusion_matrix(y_test,y_pred_xgb), annot=True, fmt='d',
cmap='Purples')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for XGBoost')
plt.show()
```



```
#Classification report
print(classification_report(y_test, y_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       173
           1       0.55      0.54      0.55        39

    accuracy                           0.83       212
   macro avg       0.72      0.72      0.72       212
weighted avg       0.83      0.83      0.83       212
```

```
np.round(roc_auc_score(y_test, y_pred_xgb), 3)
```

```
0.72
```

Export Model

```
import pickle

with open("model.pkl", "wb") as f:
    pickle.dump(classifier_xgb, f)

!pip freeze >> requirements.txt
```

Conclusion

From the insights that have been obtained from the data, it can be seen that the attrition rate in the company is quite high so action is needed to overcome the problem. Some of the suggestions below can be considered to improve company performance:

Tackle Overall Attrition Rate:

- Conduct periodic surveys to identify aspects that need improvement in the work environment. Create a positive and supportive work culture.
- Enhance employee development and training programs to help them grow and feel challenged in their work. Provide quality and relevant training and development programs.
- Clarify the company's vision, mission and values to increase employees' sense of engagement. Make employees feel that their contributions result in something good for the company and themselves. Utilize reward and recognition programs to reward employee contributions.
- Implement a flexible working hours policy.
- Support leave and other work-life balance programs.
- Limit overtime as much as possible. One way is to implement more effective and efficient workflows that minimize the need for overtime.
- Conduct market research to ensure salaries and benefits are competitive.
- Improve communication between managers and employees, and encourage cooperation and collaboration between employees. One way is to conduct team building activities for employees or provide leadership and team development training for leaders.
- Conduct a transparent and fair promotion process. Consider employee performance, experience, and skills in the promotion process. Provide opportunities for employees to grow and take a bigger role in the company.

Handling Employee Problems in the Sales Department:

- Reduce sales workload by improving sales process efficiency.

- Increase commissions and incentives for outstanding sales.

Help Employees Who Work Far Away:

- Provide transportation allowances or other compensation to help remote employees.
- Improve communication and collaboration technologies to help remote employees stay connected with their teams.
- Limit business travel to essentials to minimize employee fatigue and use of company resources.

Retain Highly Experienced Employees:

- Offer challenging and impactful career development programs for employees with a lot of work experience.
- Provide opportunities for employees with high work experience to take on leadership roles within the company.

Improve Employee Attrition Rates in Specific Positions:

- Conduct further analysis to identify specific reasons for attrition in certain positions, such as sales representative (most urgent), sales executive, human resources, laboratory technician, and research scientist.
- Develop targeted intervention programs to address these specific reasons.