

```
[13] import pandas as pd  
  
# Correct file path  
file_path = r'F:\data analysis\BQ\HR Data.xlsx'  
  
# Load the Excel file and print the sheet names  
excel_file = pd.ExcelFile(file_path)  
print("Available sheet names:", excel_file.sheet_names)
```

Python

```
... Available sheet names: ['Employee', 'PerformanceRating', 'EducationLevel', 'RatingLevel', 'SatisfiedLevel', 'Pivot Table']
```



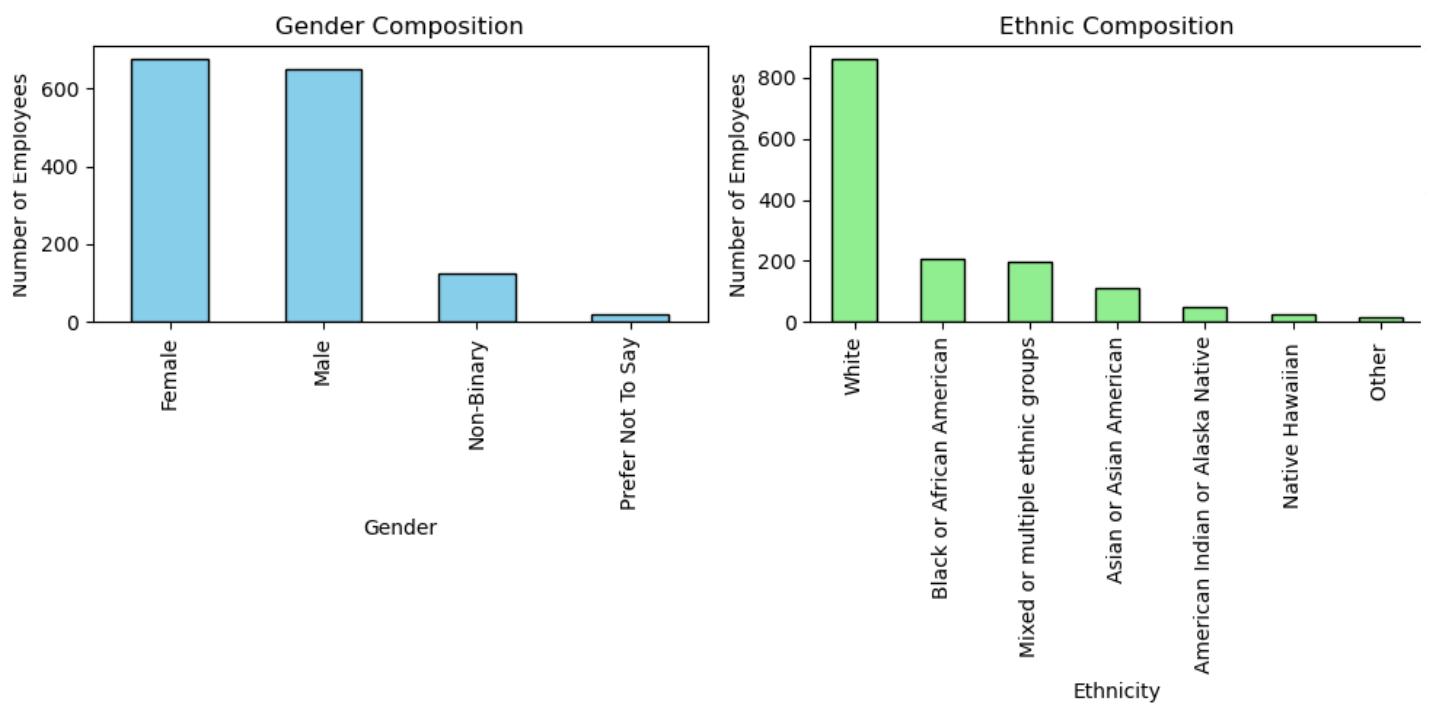
```
1 import pandas as pd  
2  
3 # Correct file path and sheet name  
4 file_path = r'F:\data analysis\BQ\HR Data.xlsx'  
5 sheet_name = 'Employee'  
6  
7 # Load the data  
8 employee_data = pd.read_excel(file_path, sheet_name=sheet_name)  
9  
10 # Display the first few rows  
11 print(employee_data.head())  
  
...  
0 EmployeeID FirstName LastName Gender Age BusinessTravel \\\n0 3012-1A41 Leonelle Simco Female 30 Some Travel  
1 CBCB-9C9D Leonerd Aland Male 38 Some Travel  
2 95D7-1CE9 Ahmed Sykes Male 43 Some Travel  
3 47A0-559B Ermentrude Berrie Non-Binary 39 Some Travel  
4 42CC-040A Stace Savege Female 29 Some Travel  
  
Department DistanceFromHome State Ethnicity ... \\\n0 Sales 27 IL White ...  
1 Sales 23 CA White ...  
2 Human Resources 29 CA Asian or Asian American ...  
3 Technology 12 IL White ...  
4 Human Resources 29 CA White ...  
  
MaritalStatus Salary StockOptionLevel Overtime HireDate Attrition \\\n0 Divorced 102059 1 No 2012-01-03 No  
1 Single 157718 0 Yes 2012-01-04 No  
2 Married 309964 1 No 2012-01-04 No  
3 Married 293132 0 No 2012-01-05 No  
4 Single 49606 0 No 2012-01-05 Yes  
  
YearsAtCompany YearsInMostRecentRole YearsSinceLastPromotion \\\n0 10 4 9  
1 10 6 10  
2 10 6 10  
...  
3 0  
4 6  
  
[5 rows x 23 columns]
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

1. What is the gender and ethnic composition of the workforce?



```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Correct file path
5 file_path = r'F:\data analysis\BQ\HR Data.xlsx'
6 sheet_name = 'Employee'
7
8 # Load the data
9 employee_data = pd.read_excel(file_path, sheet_name=sheet_name)
10
11 # Analyze gender and ethnic composition
12 def workforce_demographics(data):
13
14     gender_counts = data['Gender'].value_counts()
15     ethnicity_counts = data['Ethnicity'].value_counts()
16
17     # Plot gender composition
18     plt.figure(figsize=(10, 5))
19     plt.subplot(1, 2, 1)
20     gender_counts.plot(kind='bar', color='skyblue', edgecolor='black')
21     plt.title('Gender Composition')
22     plt.ylabel('Number of Employees')
23     plt.xlabel('Gender')
24
25     # Plot ethnicity composition
26     plt.subplot(1, 2, 2)
27     ethnicity_counts.plot(kind='bar', color='lightgreen', edgecolor='black')
28     plt.title('Ethnic Composition')
29     plt.ylabel('Number of Employees')
30     plt.xlabel('Ethnicity')
31
32     plt.tight_layout()
33     plt.savefig('workforce_demographics.png') # Save the plot
34     plt.show()
35
36 # Call the function
37 workforce_demographics(employee_data)
38
```



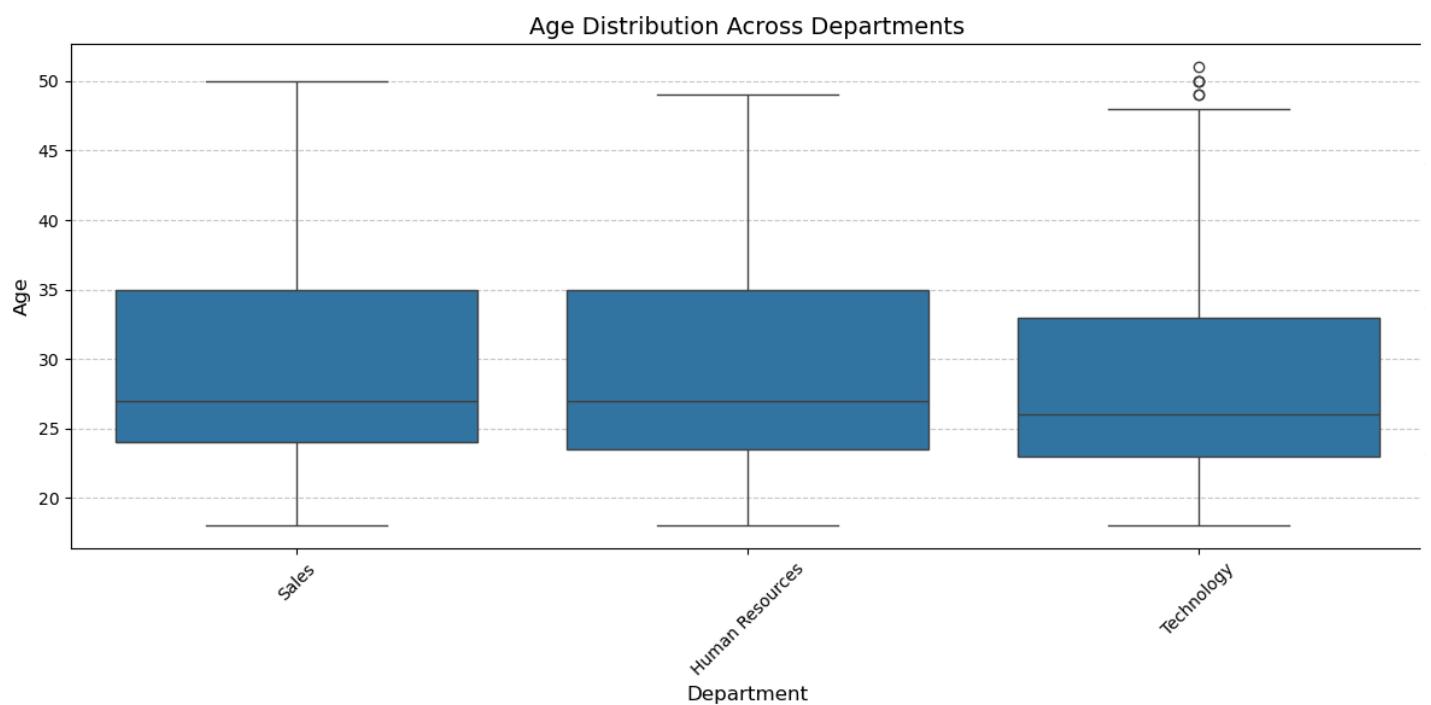
2. How does age distribution vary across departments ?

```

● ● ●

1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4
5 # Set the file path and sheet name
6 file_path = r'F:\data analysis\BQ\HR Data.xlsx'
7 sheet_name = 'Employee'
8
9 # Load the data
10 employee_data = pd.read_excel(file_path, sheet_name=sheet_name)
11
12 # Ensure the required columns exist
13 if 'Age' in employee_data.columns and 'Department' in employee_data.columns:
14     # Plot the age distribution across departments
15     plt.figure(figsize=(12, 6))
16     sns.boxplot(data=employee_data, x='Department', y='Age')
17     plt.title('Age Distribution Across Departments', fontsize=14)
18     plt.xlabel('Department', fontsize=12)
19     plt.ylabel('Age', fontsize=12)
20     plt.xticks(rotation=45)
21     plt.grid(axis='y', linestyle='--', alpha=0.7)
22     plt.tight_layout()
23     plt.show()
24 else:
25     print("Make sure the columns 'Age' and 'Department' exist in the dataset.")

```

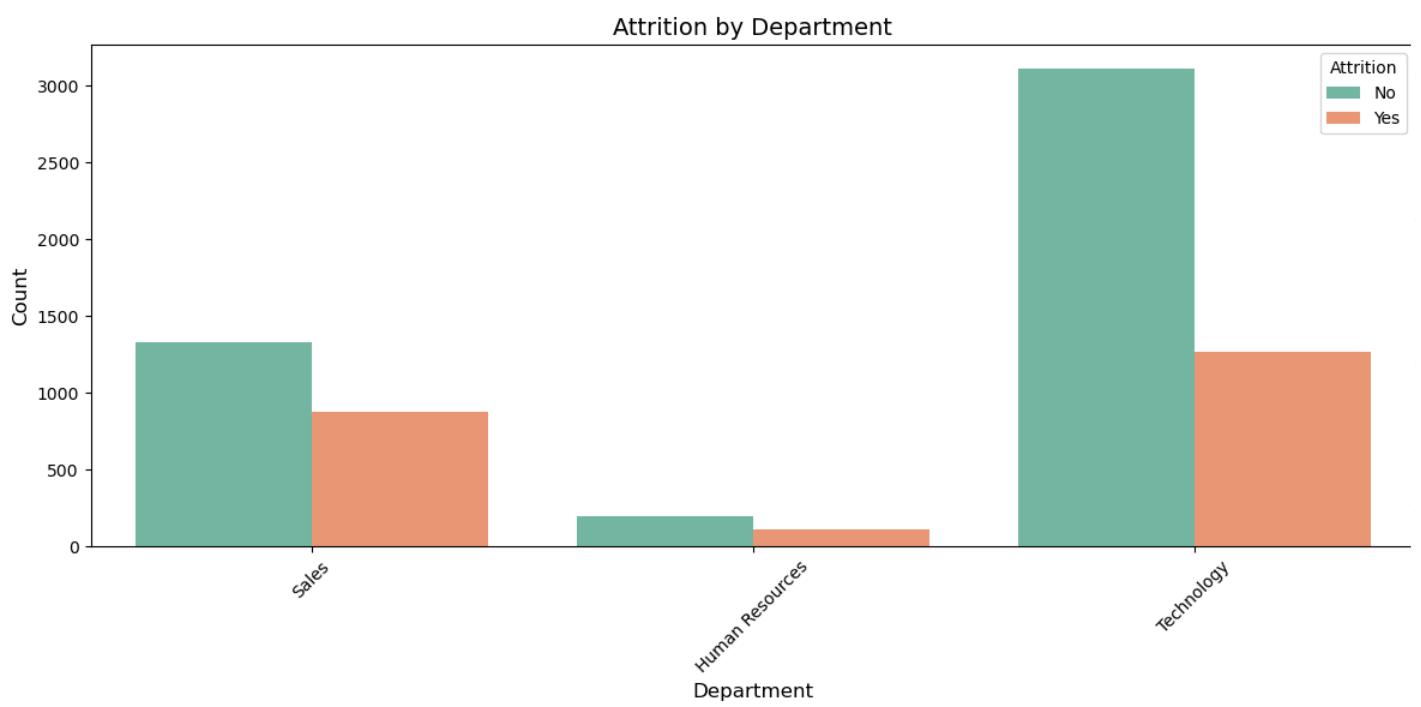
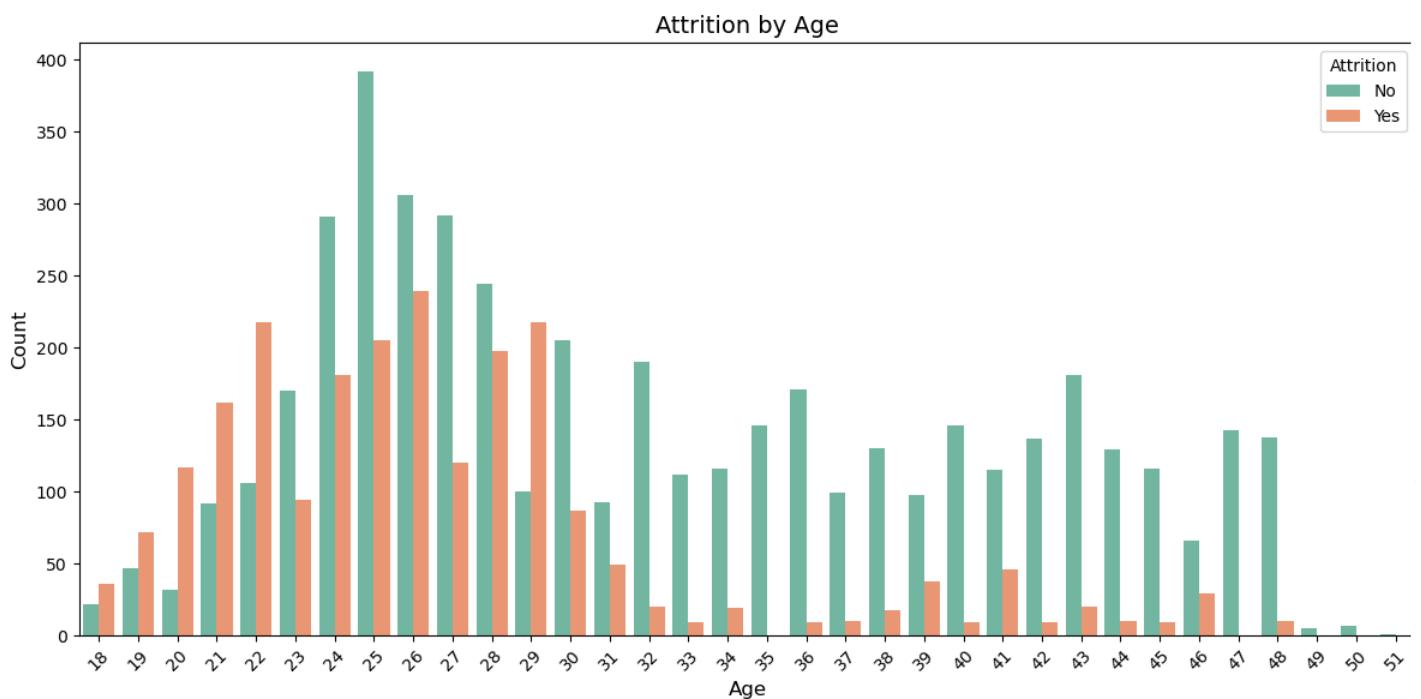


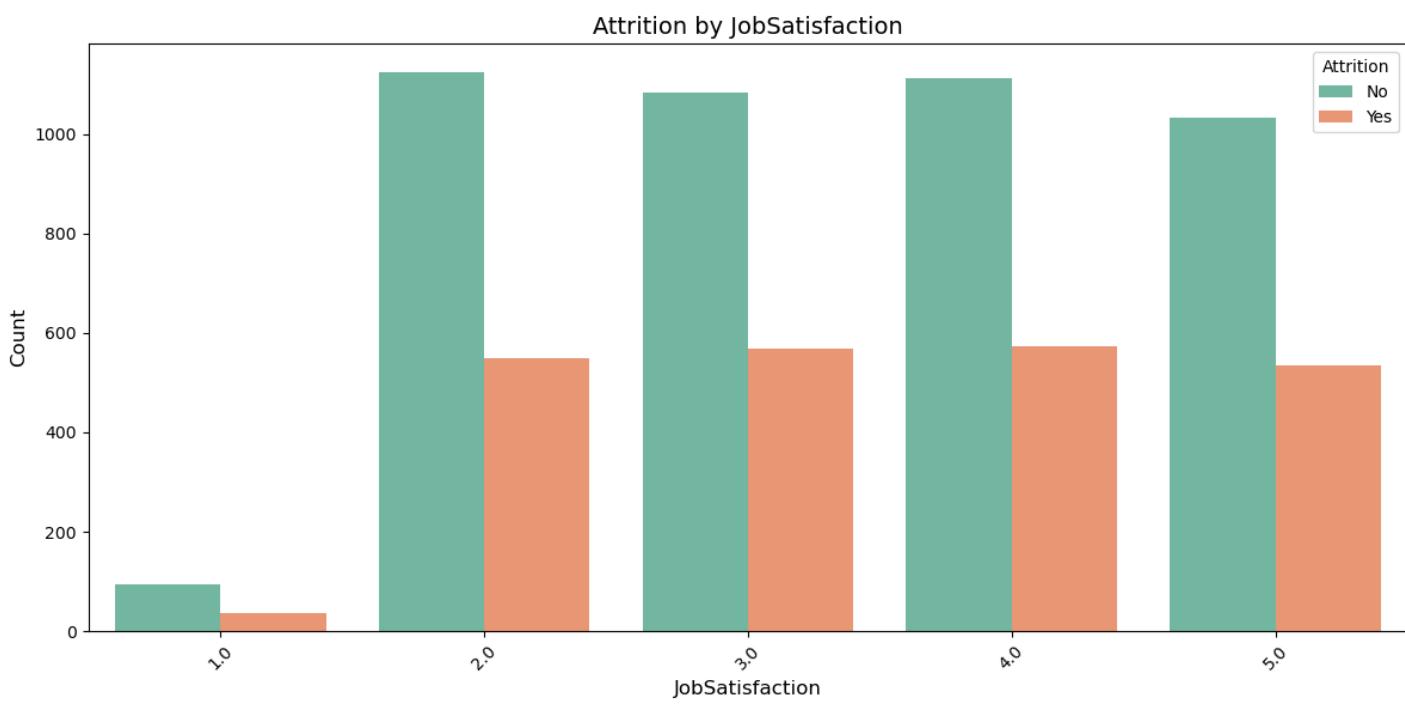
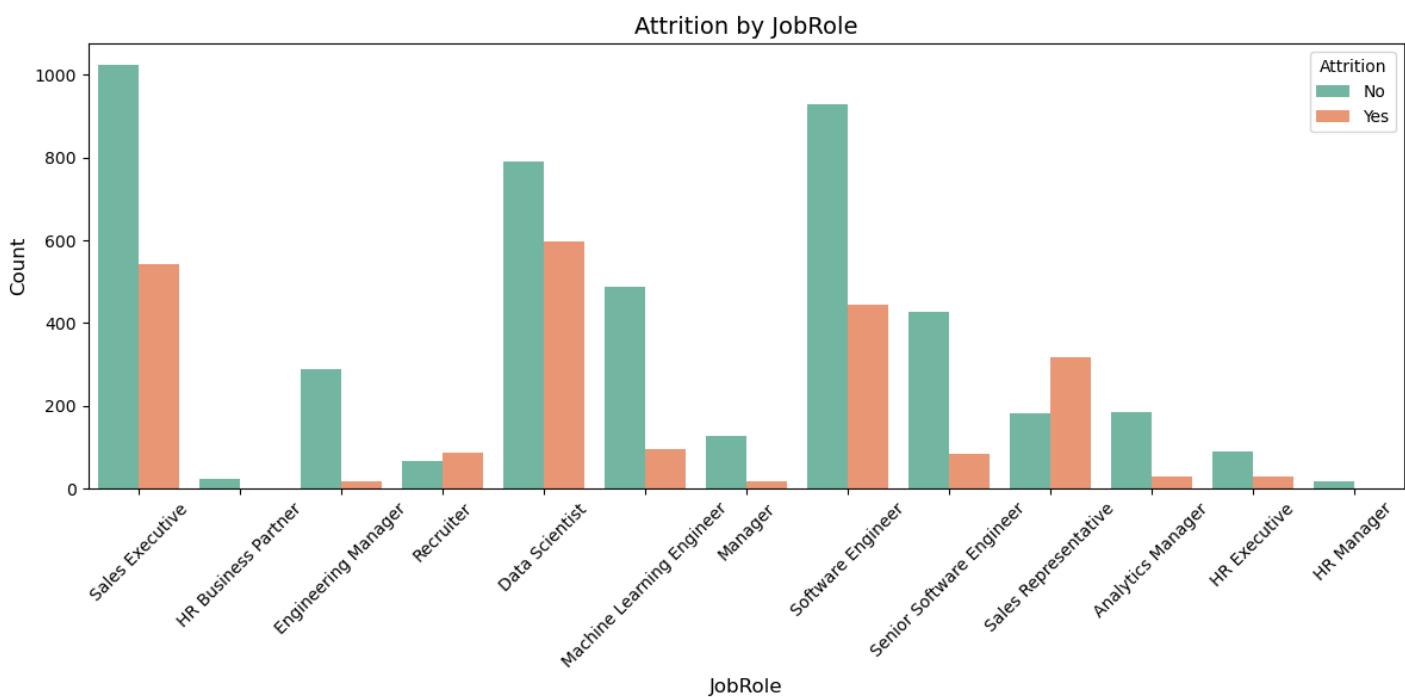
3. What are the key factors driving employee attrition?

4 .How does attrition vary across demographics (e.g., gender, ethnicity) and departments ?



```
1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4
5 # Set the file path and sheet names
6 file_path = r'F:\data analysis\BQ\HR Data.xlsx'
7 employee_sheet = 'Employee'
8 performance_sheet = 'PerformanceRating'
9
10 # Load the data
11 employee_data = pd.read_excel(file_path, sheet_name=employee_sheet)
12 performance_data = pd.read_excel(file_path, sheet_name=performance_sheet)
13
14 # Merge performance data to get JobSatisfaction
15 if 'EmployeeID' in employee_data.columns and 'EmployeeID' in performance_data.columns:
16     employee_data = employee_data.merge(performance_data[['EmployeeID', 'JobSatisfaction']], on='EmployeeID', how='left')
17 else:
18     print("EmployeeID column not found in one of the datasets.")
19
20 # Rename MonthlyIncome to salary
21 if 'MonthlyIncome' in employee_data.columns:
22     employee_data.rename(columns={'MonthlyIncome': 'Salary'}, inplace=True)
23
24 # Drop rows where salary is missing
25 employee_data = employee_data.dropna(subset=['Salary'])
26
27 # Categorize salary into bins
28 bins = [0, 3000, 7000, 12000, 20000, employee_data['Salary'].max()]
29 labels = ['Low', 'Lower-Mid', 'Mid', 'Upper-Mid', 'High']
30 employee_data['salary_category'] = pd.cut(employee_data['Salary'], bins=bins, labels=labels)
31
32 # Check for the 'Attrition' column
33 if 'Attrition' in employee_data.columns:
34     # Ensure salary_category exists before analysis
35     if 'salary_category' in employee_data.columns:
36         key_factors = ['Age', 'Department', 'JobRole', 'Salary', 'JobSatisfaction']
37     else:
38         key_factors = ['Age', 'Department', 'JobRole', 'JobSatisfaction']
39
40     for factor in key_factors:
41         if factor in employee_data.columns:
42             plt.figure(figsize=(12, 6))
43             sns.countplot(data=employee_data, x=factor, hue='Attrition', palette='Set2')
44             plt.title(f'Attrition by {factor}', fontsize=14)
45             plt.xlabel(factor, fontsize=12)
46             plt.ylabel('Count', fontsize=12)
47             plt.xticks(rotation=45)
48             plt.legend(title='Attrition', loc='upper right')
49             plt.tight_layout()
50             plt.show()
51         else:
52             print(f"Column '{factor}' not found in the dataset.")
53     else:
54         print("The dataset does not contain an 'Attrition' column.")
```





6 .Is there a significant deference in salaries across departments, genders, or other demographic groups ?

```
print(employee_data['Salary'].describe())
```

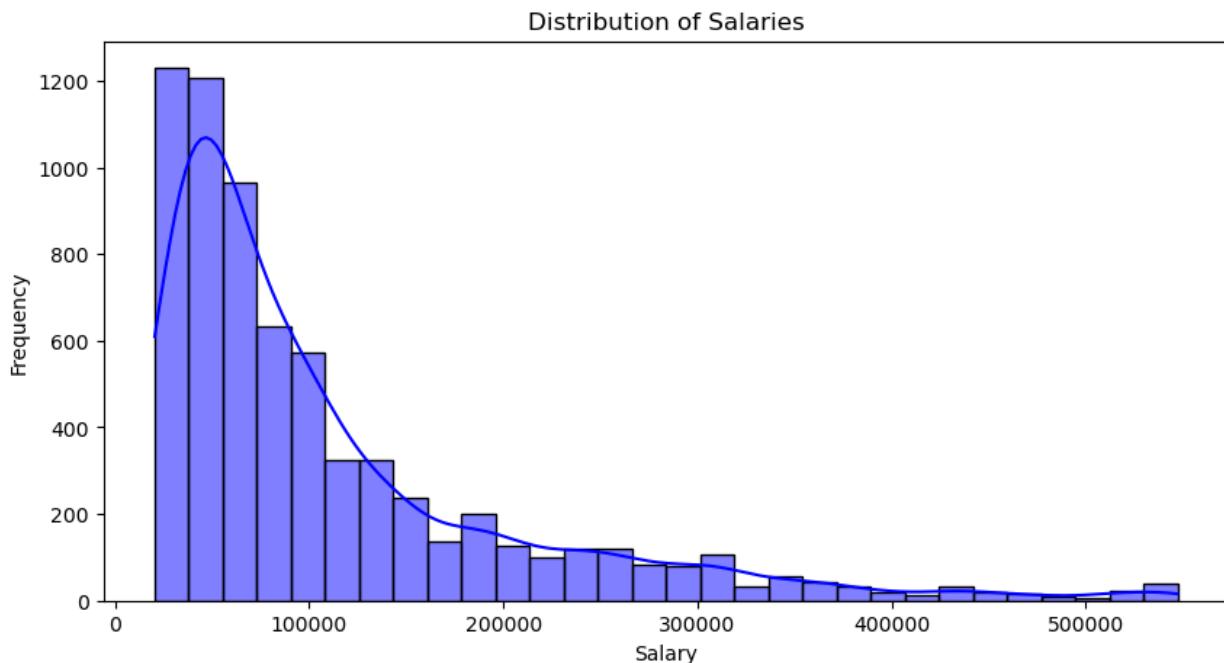
[10] ✓ 0.0s

Python

```
...    count      6899.000000
     mean      110898.374112
     std       98427.862382
     min      20387.000000
    25%      44646.000000
    50%      74458.000000
    75%     137219.500000
    max      547204.000000
Name: Salary, dtype: float64
```



```
1 plt.figure(figsize=(10, 5))
2 sns.histplot(employee_data['Salary'], bins=30, kde=True, color='blue')
3 plt.title('Distribution of Salaries')
4 plt.xlabel('Salary')
5 plt.ylabel('Frequency')
6 plt.show()
```



```
avg_salary_by_department = employee_data.groupby('Department')['salary'].mean().sort_values()  
print(avg_salary_by_department)
```

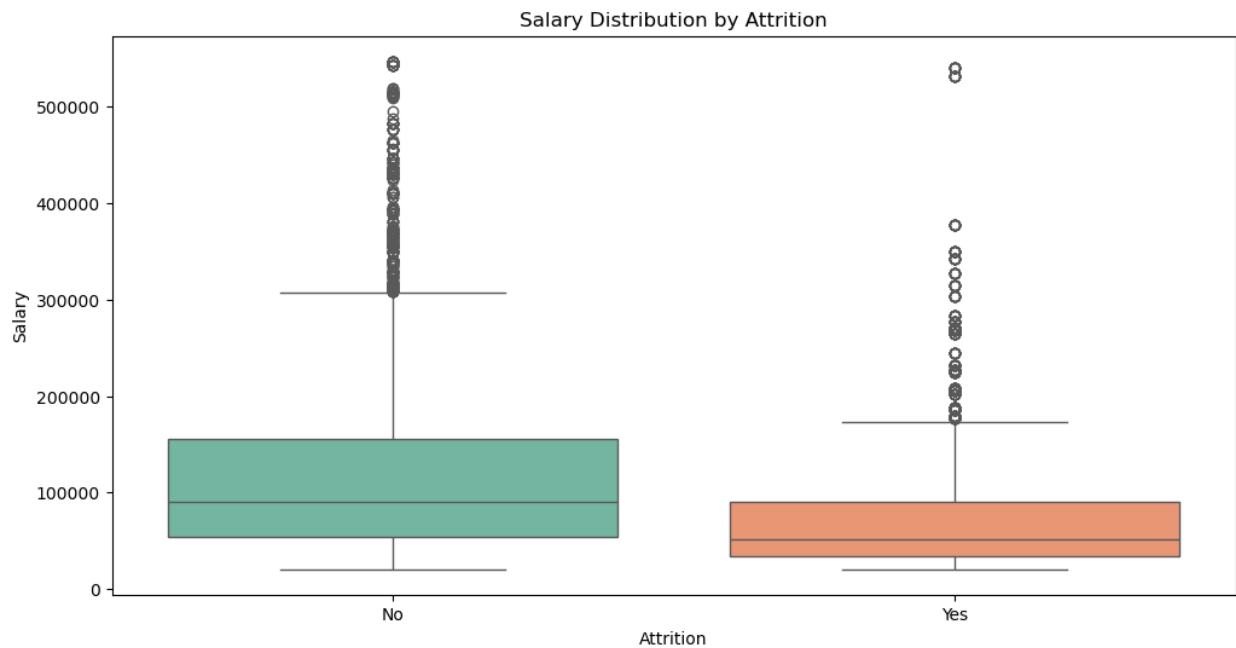
[14] ✓ 0.0s

Python

```
...   Department  
Technology      106245.325486  
Human Resources  107629.555911  
Sales           120568.309815  
Name: Salary, dtype: float64
```



```
1 plt.figure(figsize=(12, 6))  
2 sns.boxplot(data=employee_data, x='Attrition', y='Salary', palette='Set2')  
3 plt.title('Salary Distribution by Attrition')  
4 plt.xlabel('Attrition')  
5 plt.ylabel('Salary')  
6 plt.show()  
7
```



```

import scipy.stats as stats

# التحقق من أن الأعمدة المطلوبة موجودة
if 'Department' in employee_data.columns and 'Salary' in employee_data.columns:
    # إنساء قائمة من الروابط لكل قسم
    department_groups = [group["Salary"].dropna() for _, group in employee_data.groupby("Department")]

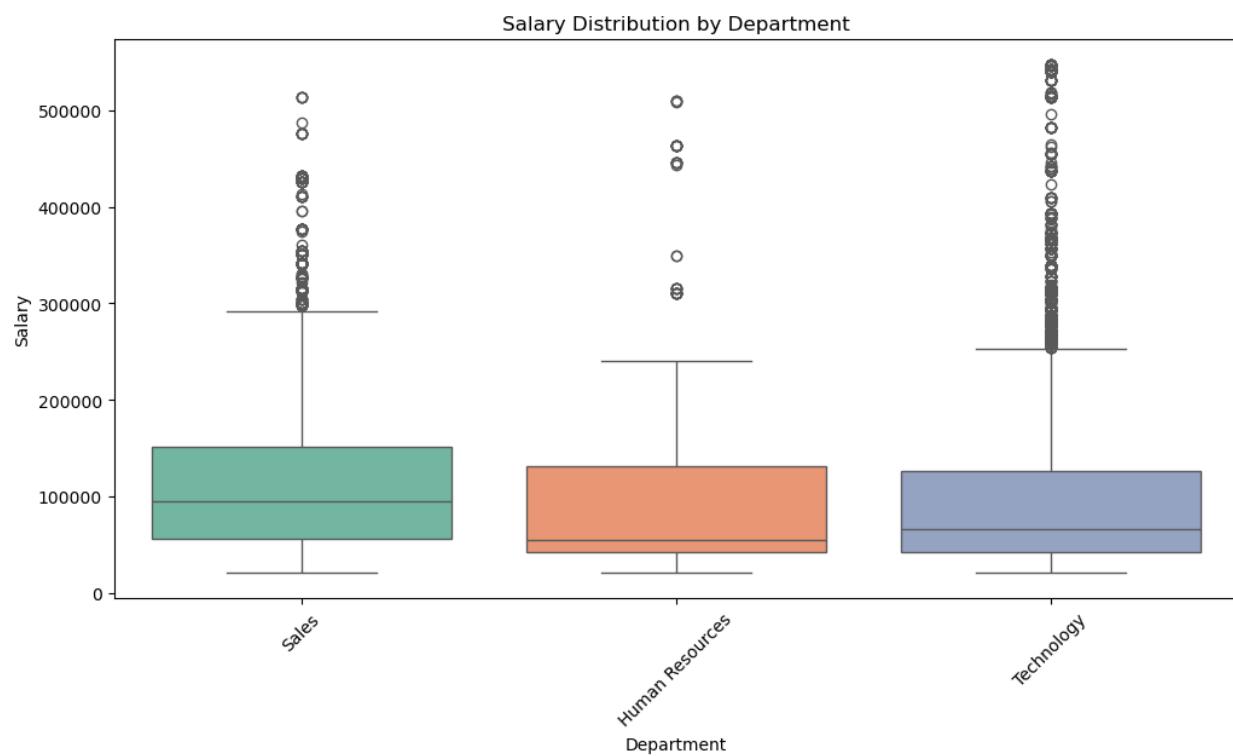
    # تأكيد اختبار ANOVA
    f_stat, p_value = stats.f_oneway(*department_groups)
    print(f"ANOVA Test for Salary across Departments:\nF-statistic: {f_stat}, P-value: {p_value}")

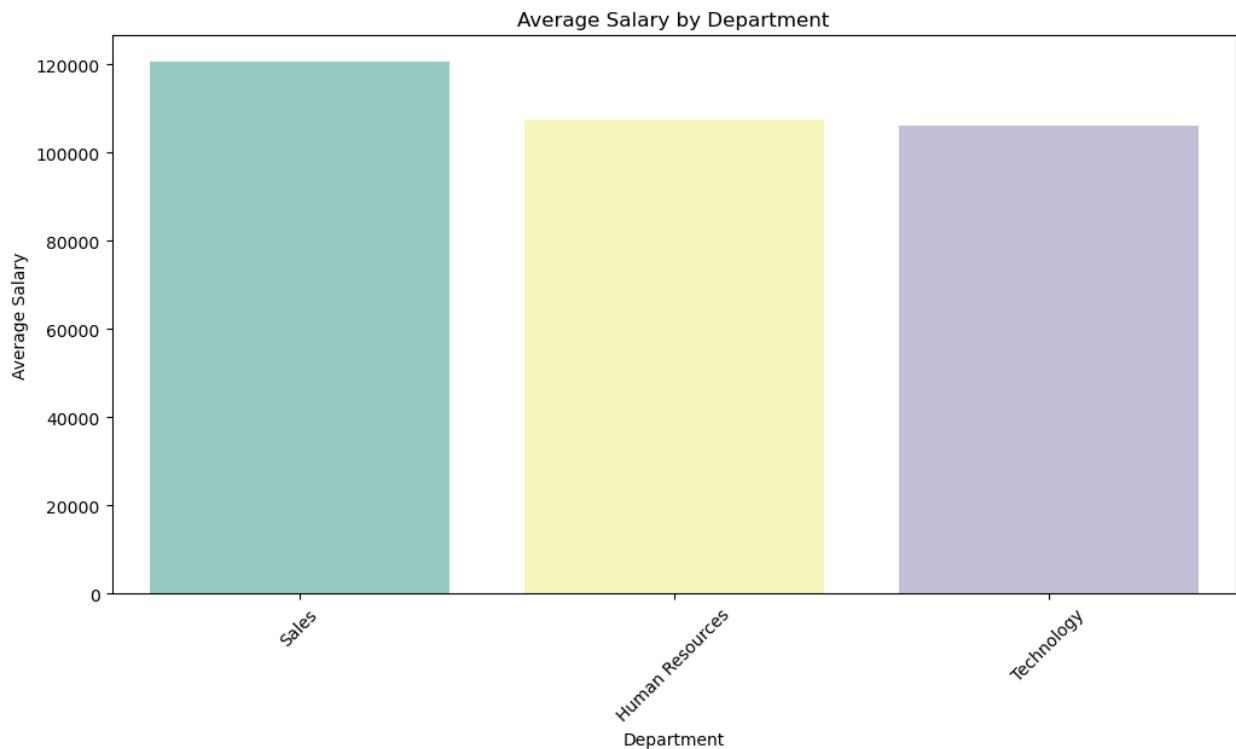
    # التأكيد على النتائج
    if p_value < 0.05:
        print("There is a significant difference in salaries across departments.")
    else:
        print("There is no significant difference in salaries across departments.")
else:
    print("Columns 'Department' or 'salary' not found in dataset.")

[27] ✓ 0.0s
... ANOVA Test for Salary across Departments:
F-statistic: 15.798869407016051, P-value: 1.4266250811861307e-07
There is a significant difference in salaries across departments.
  
```



```
1 plt.figure(figsize=(12, 6))
2 sns.boxplot(data=employee_data, x='Department', y='Salary', palette='Set2')
3 plt.title('Salary Distribution by Department')
4 plt.xlabel('Department')
5 plt.ylabel('Salary')
6 plt.xticks(rotation=45)
7 plt.show()
8
9 plt.figure(figsize=(12, 6))
10 sns.barplot(data=employee_data, x='Department', y='Salary', palette='Set3', ci=None)
11 plt.title('Average Salary by Department')
12 plt.xlabel('Department')
13 plt.ylabel('Average Salary')
14 plt.xticks(rotation=45)
15 plt.show()
16
```





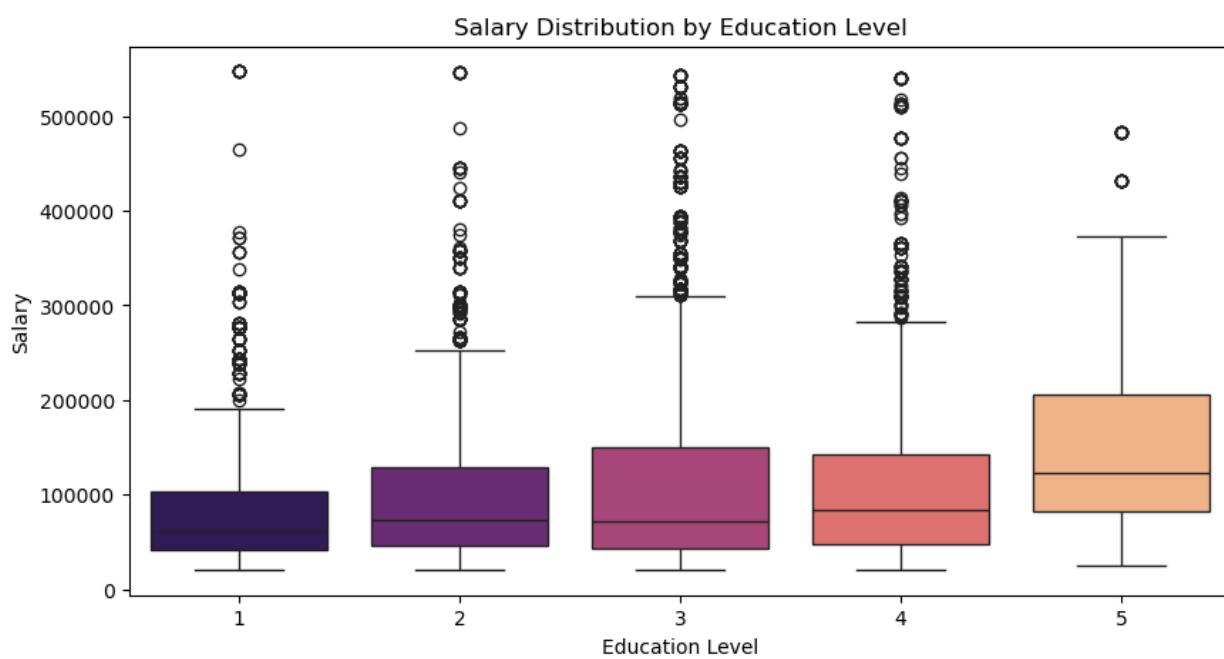
```
▷ [24] if 'Education' in employee_data.columns and 'Salary' in employee_data.columns:  
...     education_groups = [group["Salary"].dropna() for _, group in employee_data.groupby("Education")]  
...  
...     f_stat, p_value = stats.f_oneway(*education_groups)  
...     print(f"\nANOVA Test for Salary across Education Levels:\nF-statistic: {f_stat}, P-value: {p_value}")  
...  
...     if p_value < 0.05:  
...         print("There is a significant difference in salaries across education levels.")  
...     else:  
...         print("There is no significant difference in salaries across education levels.")  
[24] ✓ 0.0s
```

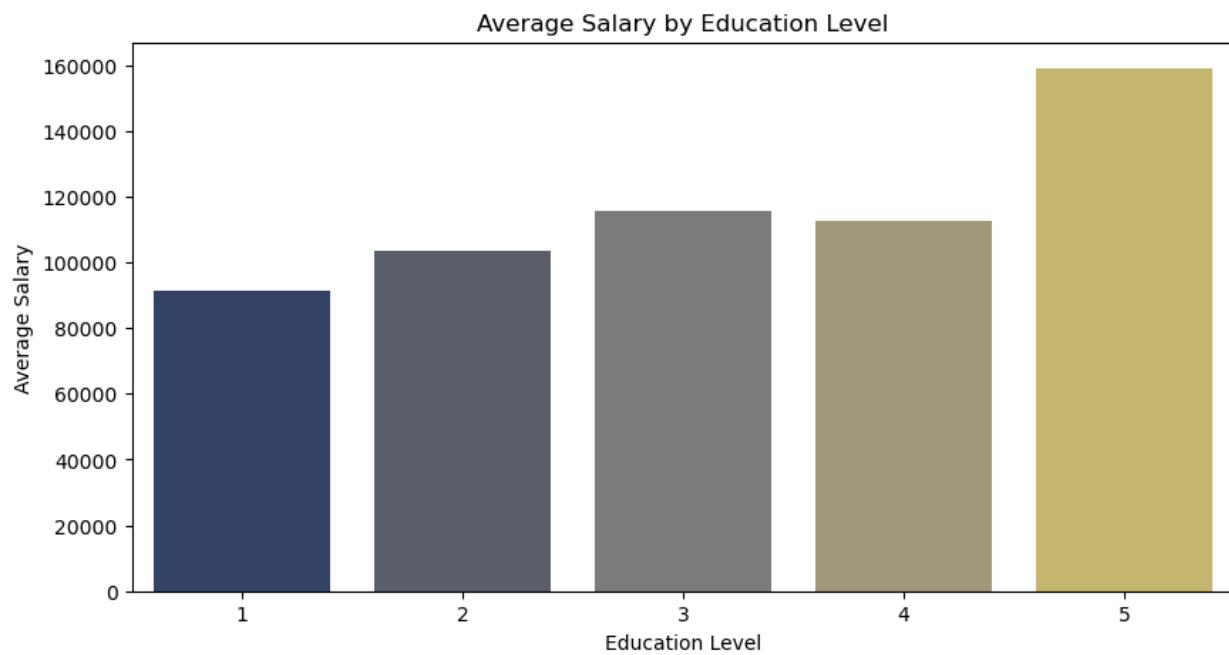
Python

```
... ANOVA Test for Salary across Education Levels:  
... F-statistic: 25.501572712591614, P-value: 5.2682176507728024e-21  
... There is a significant difference in salaries across education levels.
```



```
1 plt.figure(figsize=(10, 5))
2 sns.boxplot(data=employee_data, x='Education', y='Salary', palette='magma')
3 plt.title('Salary Distribution by Education Level')
4 plt.xlabel('Education Level')
5 plt.ylabel('Salary')
6 plt.show()
7
8 plt.figure(figsize=(10, 5))
9 sns.barplot(data=employee_data, x='Education', y='Salary', palette='cividis', ci=None)
10 plt.title('Average Salary by Education Level')
11 plt.xlabel('Education Level')
12 plt.ylabel('Average Salary')
13 plt.show()
14
```





```

if 'Gender' in employee_data.columns and 'Salary' in employee_data.columns:
    male_salaries = employee_data[employee_data['Gender'] == 'Male']['Salary'].dropna()
    female_salaries = employee_data[employee_data['Gender'] == 'Female']['Salary'].dropna()

    # T-test
    t_stat, p_value = stats.ttest_ind(male_salaries, female_salaries, equal_var=False)
    print(f"T-test for Salary by Gender:\nT-statistic: {t_stat}, P-value: {p_value}")

    if p_value < 0.05:
        print("There is a significant difference in salaries between genders.")
    else:
        print("There is no significant difference in salaries between genders.")
else:
    print("Columns 'Gender' or 'salary' not found in dataset.")

```

[29] ✓ 0.0s

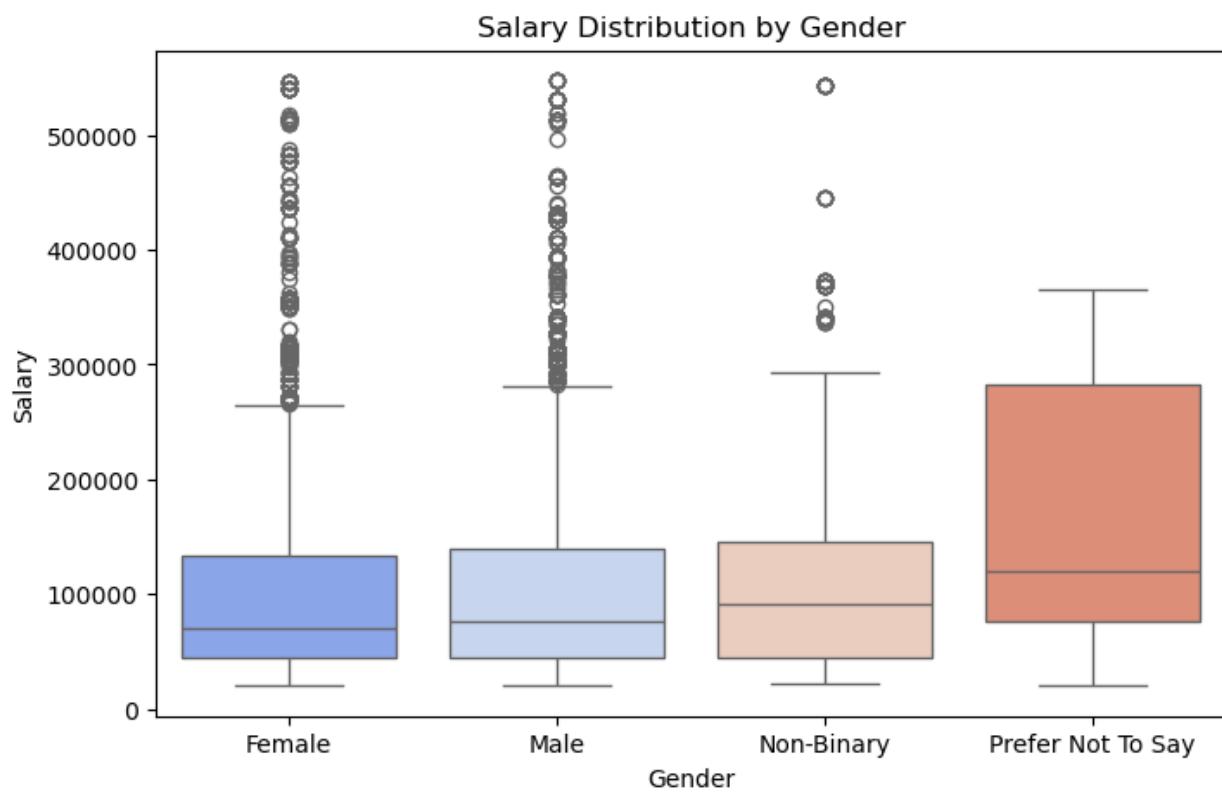
Python

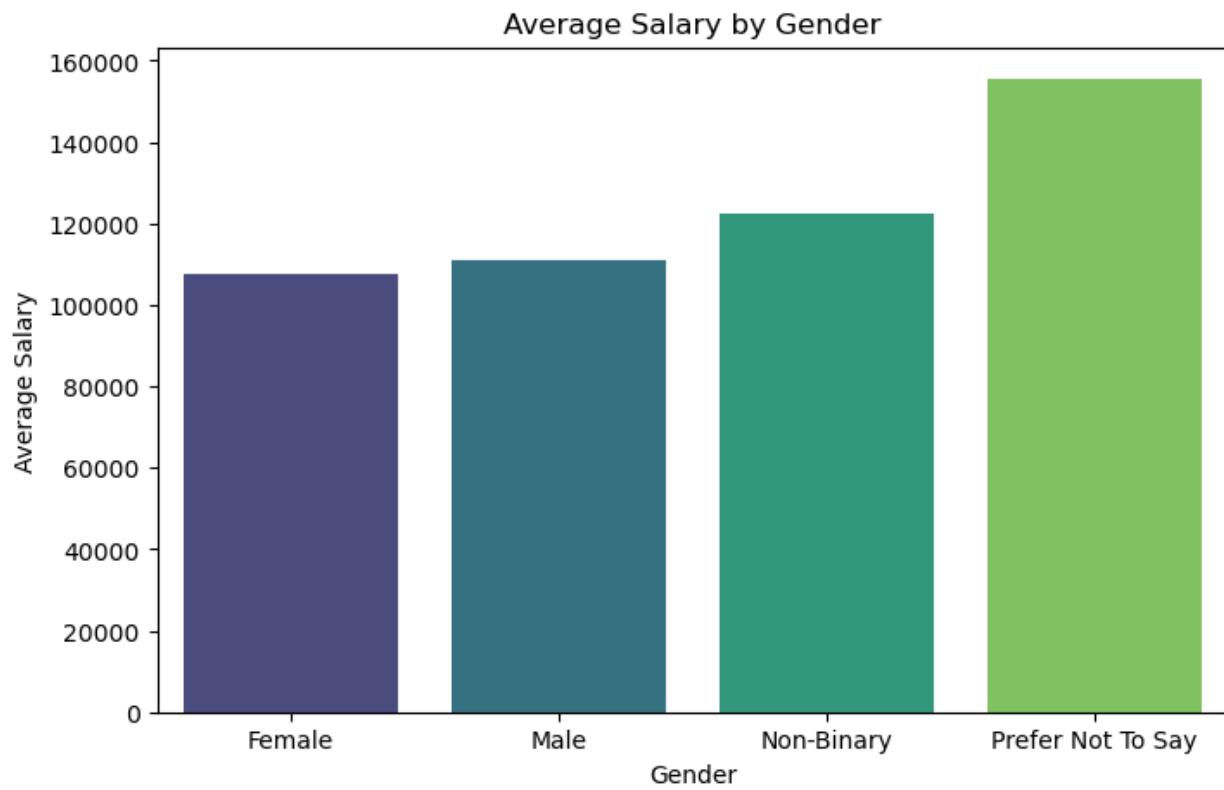
```

...
T-test for Salary by Gender:
T-statistic: 1.371114281205253, P-value: 0.17038886191669766
There is no significant difference in salaries between genders.

```

```
● ● ●  
1 plt.figure(figsize=(8, 5))  
2 sns.boxplot(data=employee_data, x='Gender', y='Salary', palette='coolwarm')  
3 plt.title('Salary Distribution by Gender')  
4 plt.xlabel('Gender')  
5 plt.ylabel('Salary')  
6 plt.show()  
7  
8 plt.figure(figsize=(8, 5))  
9 sns.barplot(data=employee_data, x='Gender', y='Salary', palette='viridis', ci=None)  
10 plt.title('Average Salary by Gender')  
11 plt.xlabel('Gender')  
12 plt.ylabel('Average Salary')  
13 plt.show()  
14
```



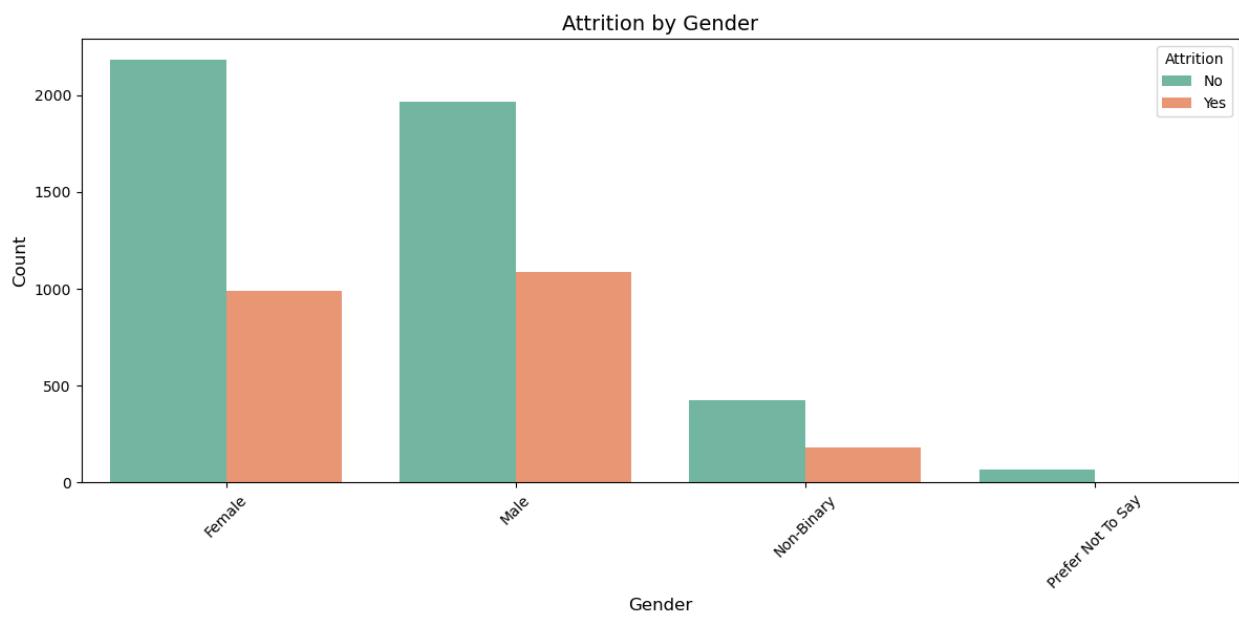
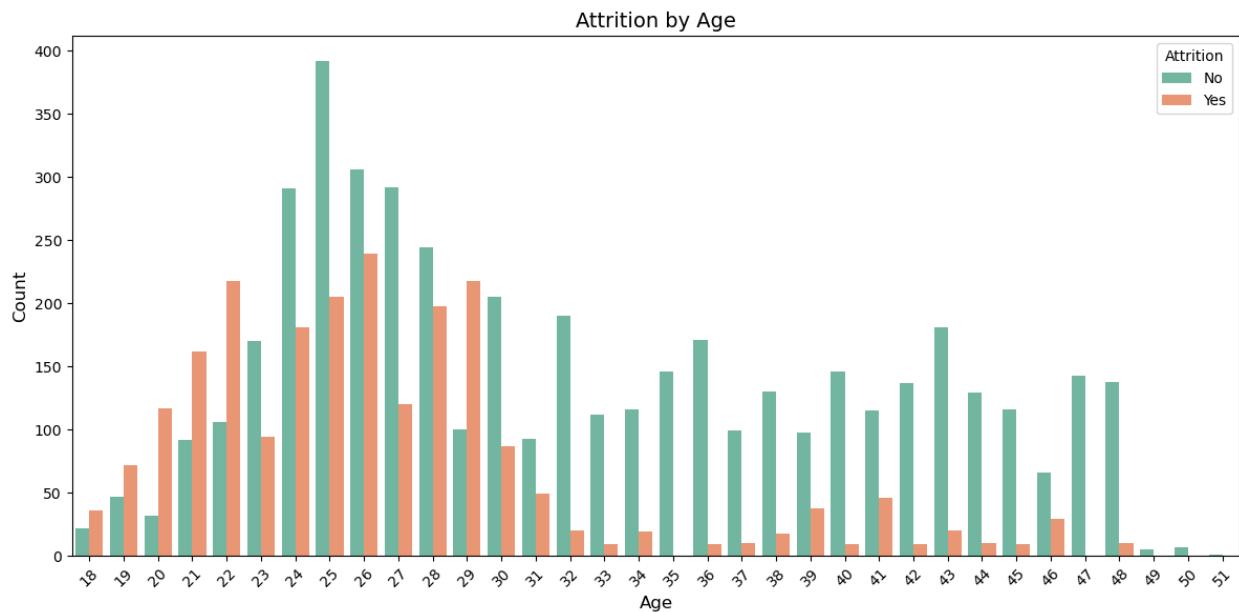


4 . How does attrition vary across demographics (e.g., gender, ethnicity) and departments?

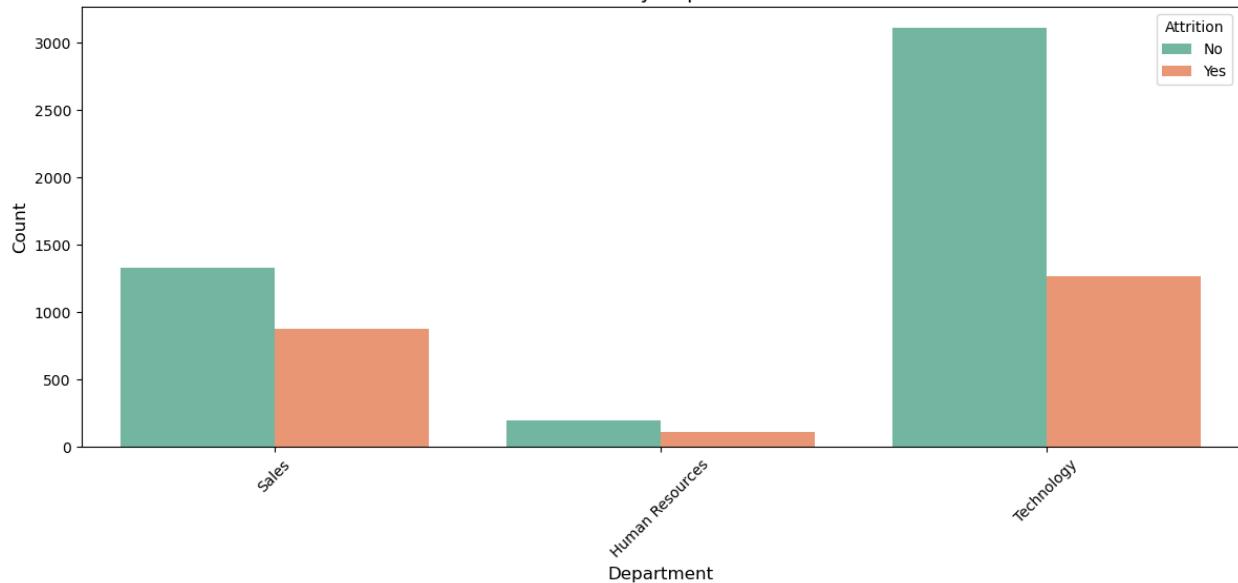
7 . How does marital status influence attrition rates?



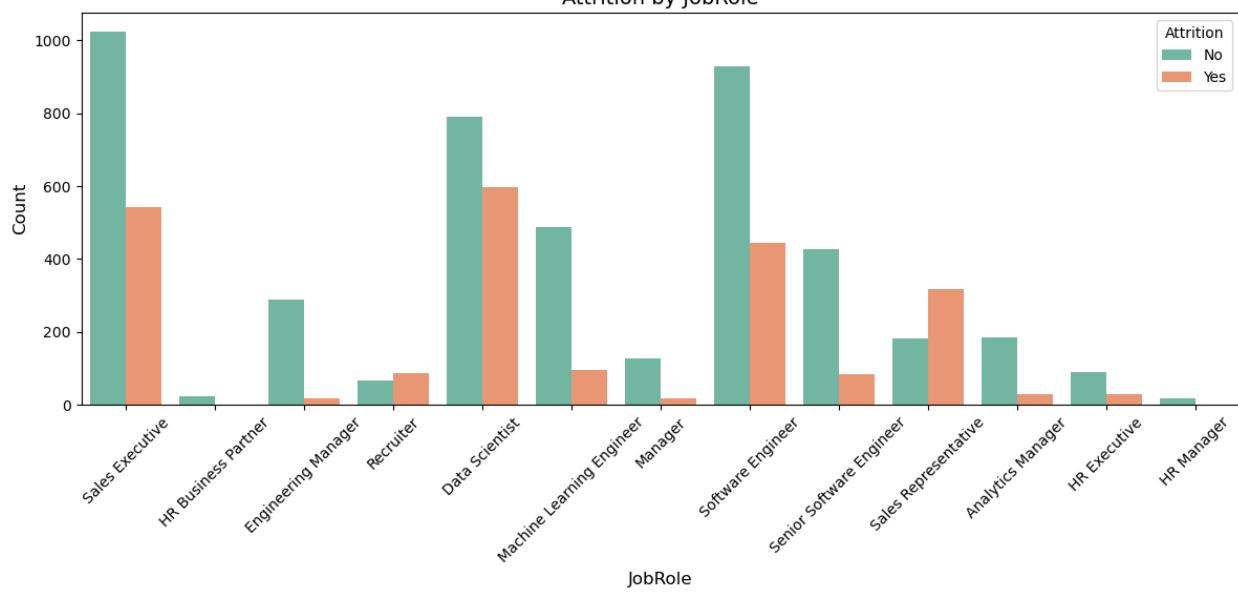
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # List of key factors to analyze
6 key_factors = [
7     'Age', 'Gender', 'Department', 'JobRole', 'DistanceFromHome',
8     'MaritalStatus', 'OverTime', 'Ethnicity'
9 ]
10
11 # Loop through each factor and visualize attrition impact
12 for factor in key_factors:
13     if factor in employee_data.columns:
14         plt.figure(figsize=(12, 6))
15         sns.countplot(data=employee_data, x=factor, hue='Attrition', palette='Set2')
16         plt.title(f'Attrition by {factor}', fontsize=14)
17         plt.xlabel(factor, fontsize=12)
18         plt.ylabel('Count', fontsize=12)
19         plt.xticks(rotation=45)
20         plt.legend(title='Attrition', loc='upper right')
21         plt.tight_layout()
22         plt.show()
23     else:
24         print(f"Column '{factor}' not found in the dataset.")
```



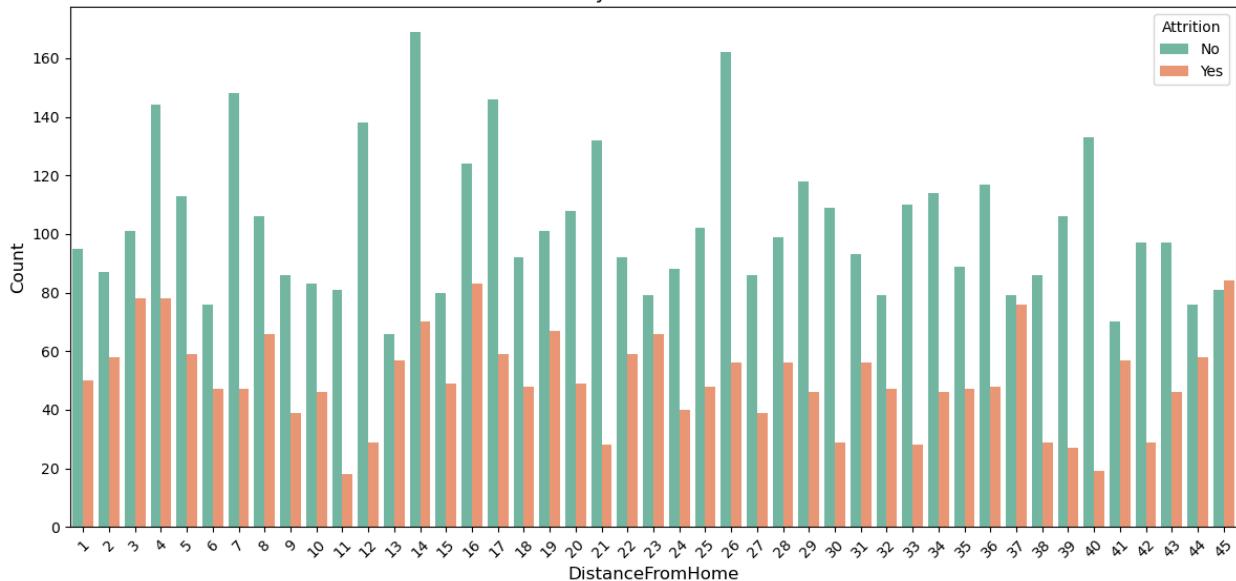
Attrition by Department



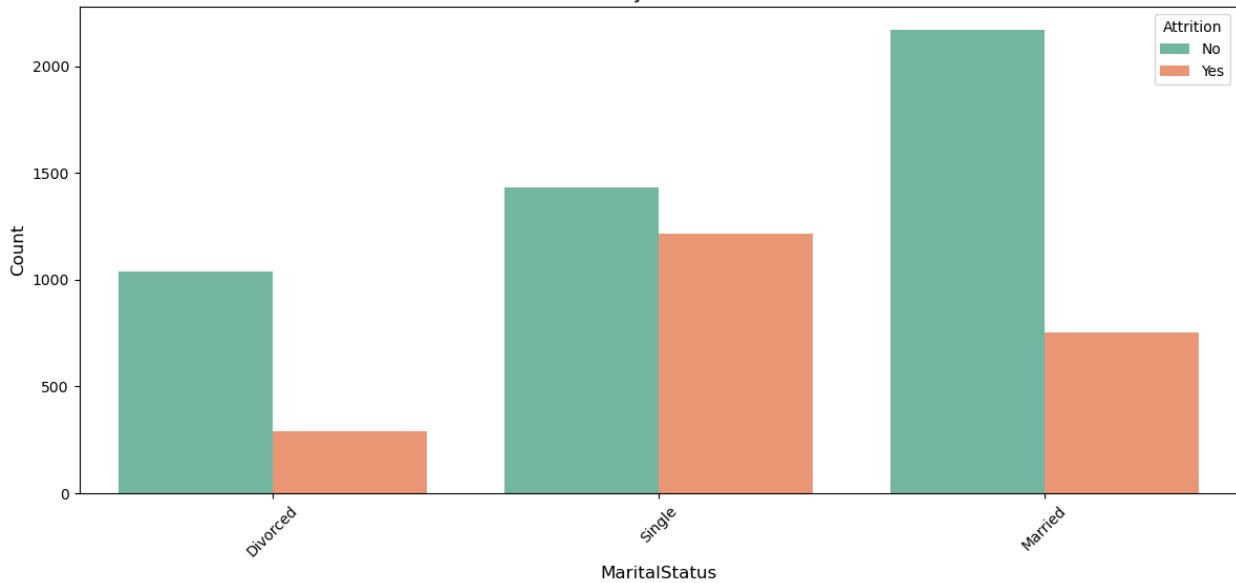
Attrition by JobRole

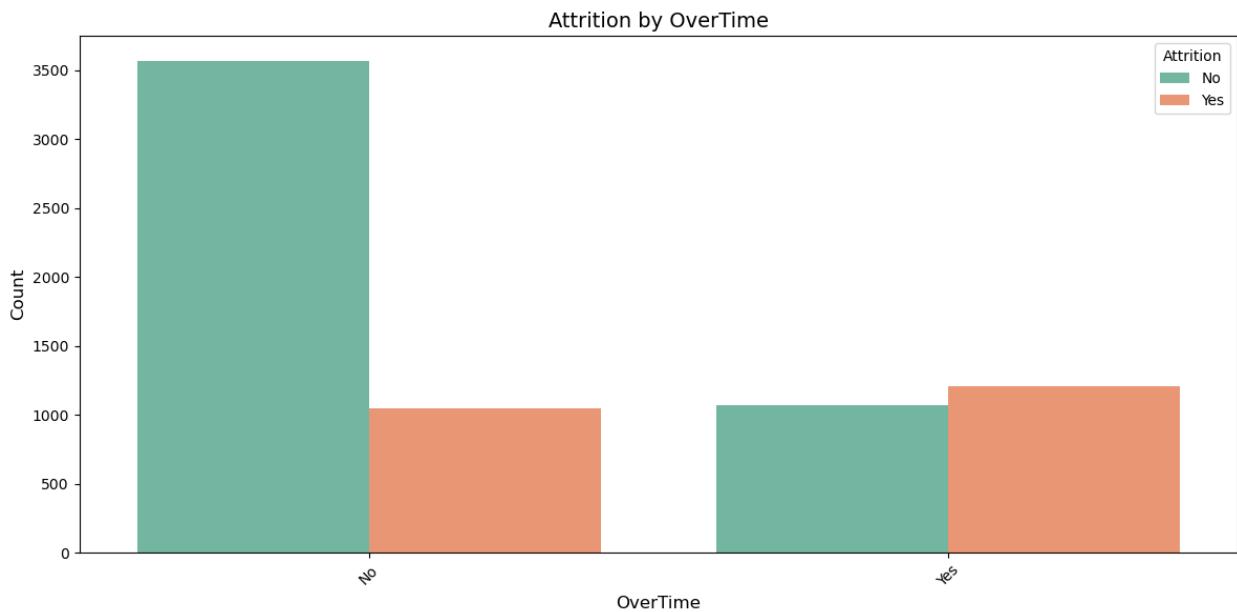


Attrition by DistanceFromHome



Attrition by MaritalStatus





5 . Is there a correlation between DistanceFromHome and attrition?

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
file_path = r'F:\data analysis\BQ\HR Data.xlsx'
sheet_name = 'Employee'
employee_data = pd.read_excel(file_path, sheet_name=sheet_name)

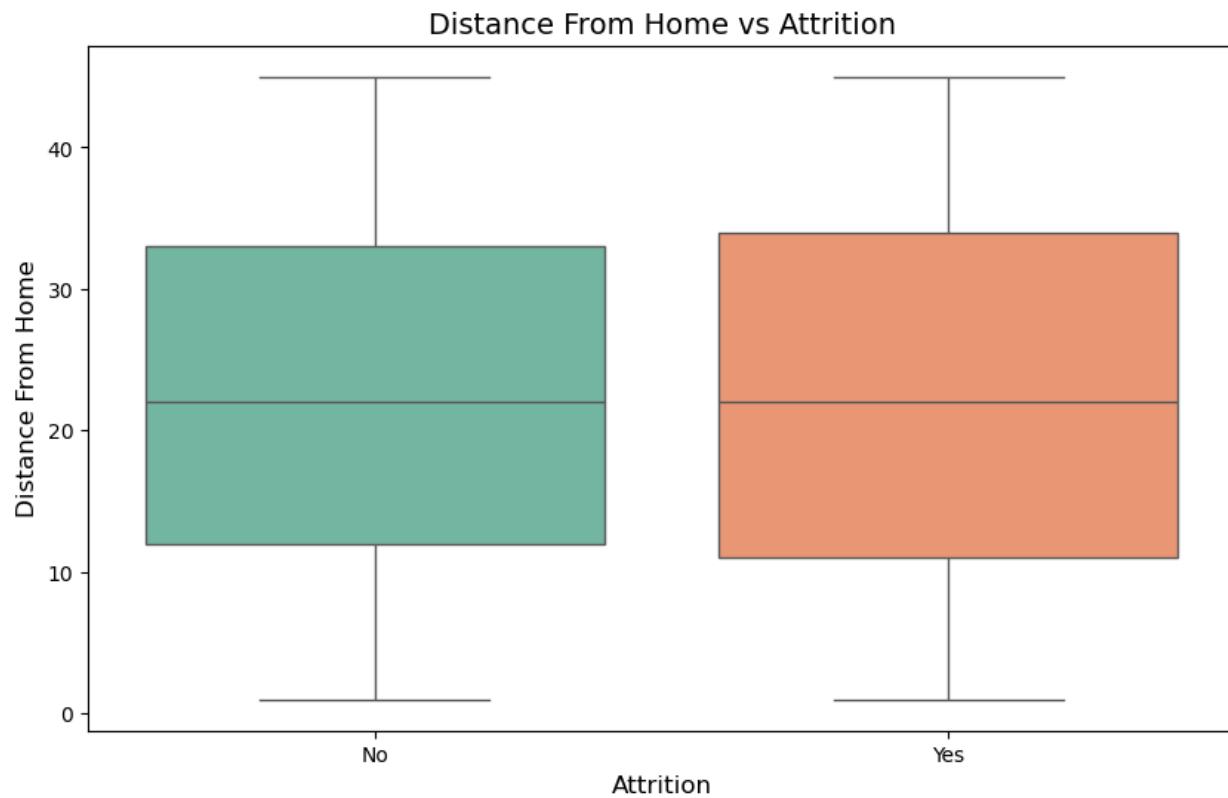
# Ensure the columns exist
if 'DistanceFromHome' in employee_data.columns and 'Attrition' in employee_data.columns:
    # Convert Attrition to numerical values (e.g., Yes=1, No=0)
    employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})

    # Calculate correlation
    correlation = employee_data['DistanceFromHome'].corr(employee_data['AttritionNumeric'])
    print(f"Correlation between DistanceFromHome and Attrition: {correlation:.2f}")

    # Visualize the relationship
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Attrition', y='DistanceFromHome', data=employee_data, hue='Attrition', palette='Set2')
    plt.title('Distance From Home vs Attrition', fontsize=14)
    plt.xlabel('Attrition', fontsize=12)
    plt.ylabel('Distance From Home', fontsize=12)
    plt.show()
else:
    print("Required columns 'DistanceFromHome' or 'Attrition' are not found in the dataset.")

[ ] Python
... Correlation between DistanceFromHome and Attrition: -0.01

```



6. *Is there a significant difference in salaries across departments, genders, or other demographic groups ?*

7. *How does stock option level influence employee retention and satisfaction?*

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from scipy import stats
5
6 # Load your dataset
7 file_path = r'F:\data analysis\BQ\HR Data.xlsx' # Update with your file path
8 sheet_name = 'Employee' # Adjust if necessary
9 employee_data = pd.read_excel(file_path, sheet_name=sheet_name)
10
11 # Ensure necessary columns exist
12 required_columns = ['Attrition', 'Gender', 'Ethnicity', 'Department', 'Salary', 'StockOptionLevel', 'Overtime']
13 if all(col in employee_data.columns for col in required_columns):
14
15     # Convert Attrition to numerical values (Yes=1, No=0) if not already done
16     if 'AttritionNumeric' not in employee_data.columns:
17         employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})
18
19     # 6. Is there a significant difference in salaries across departments, genders, or other demographic groups?
20
21     print("Salary Comparison Across Demographics and Departments:")
22
23     # Salary by Department (ANOVA)
24     department_salary = employee_data.groupby('Department')['Salary'].mean()
25     print("\nAverage Salary by Department:")
26     print(department_salary)
27
28     # ANOVA Test for Salary by Department
29     department_salary_groups = [group['Salary'].values for name, group in employee_data.groupby('Department')]
30     f_stat, p_val = stats.f_oneway(*department_salary_groups)
31     print(f"\nANOVA Test Result for Salary by Department: F-statistic = {f_stat:.2f}, P-value = {p_val:.4f}")
32
33     # Salary by Gender (T-test)
34     male_salary = employee_data[employee_data['Gender'] == 'Male']['Salary']
35     female_salary = employee_data[employee_data['Gender'] == 'Female']['Salary']
36     t_stat, p_val_gender = stats.ttest_ind(male_salary, female_salary)
37     print(f"\nT-test for Salary by Gender: T-statistic = {t_stat:.2f}, P-value = {p_val_gender:.4f}")
38
39     # Visualize Salary by Department and Gender (with hue to avoid FutureWarning)
40     plt.figure(figsize=(12, 8))
41     sns.boxplot(x='Department', y='Salary', data=employee_data, hue='Attrition', palette='Set2', legend=False)
42     plt.title('Salary Distribution by Department')
43     plt.xticks(rotation=45)
44     plt.show()
45
46     plt.figure(figsize=(12, 8))
47     sns.boxplot(x='Gender', y='Salary', data=employee_data, hue='Attrition', palette='Set2', legend=False)
48     plt.title('Salary Distribution by Gender')
49     plt.show()
50
51     # 7. How does stock option level influence employee retention and satisfaction?
52
53     # Correlation between StockOptionLevel and Attrition (Retention)
54     print("\nStock Option Level and Retention (Attrition):")
55     correlation_stock = employee_data['StockOptionLevel'].corr(employee_data['AttritionNumeric'])
56     print(f"Correlation between StockOptionLevel and Attrition: {correlation_stock:.2f}")
57
58     # Visualize StockOptionLevel vs Attrition
59     plt.figure(figsize=(10, 6))
60     sns.boxplot(x='AttritionNumeric', y='StockOptionLevel', data=employee_data, palette='Set2')
61     plt.title('Stock Option Level vs Attrition')
62     plt.xlabel('Attrition')
63     plt.ylabel('Stock Option Level')
64     plt.show()
65
66 else:
67     print("Some required columns are missing.")
68

```

Salary Comparison Across Demographics and Departments:

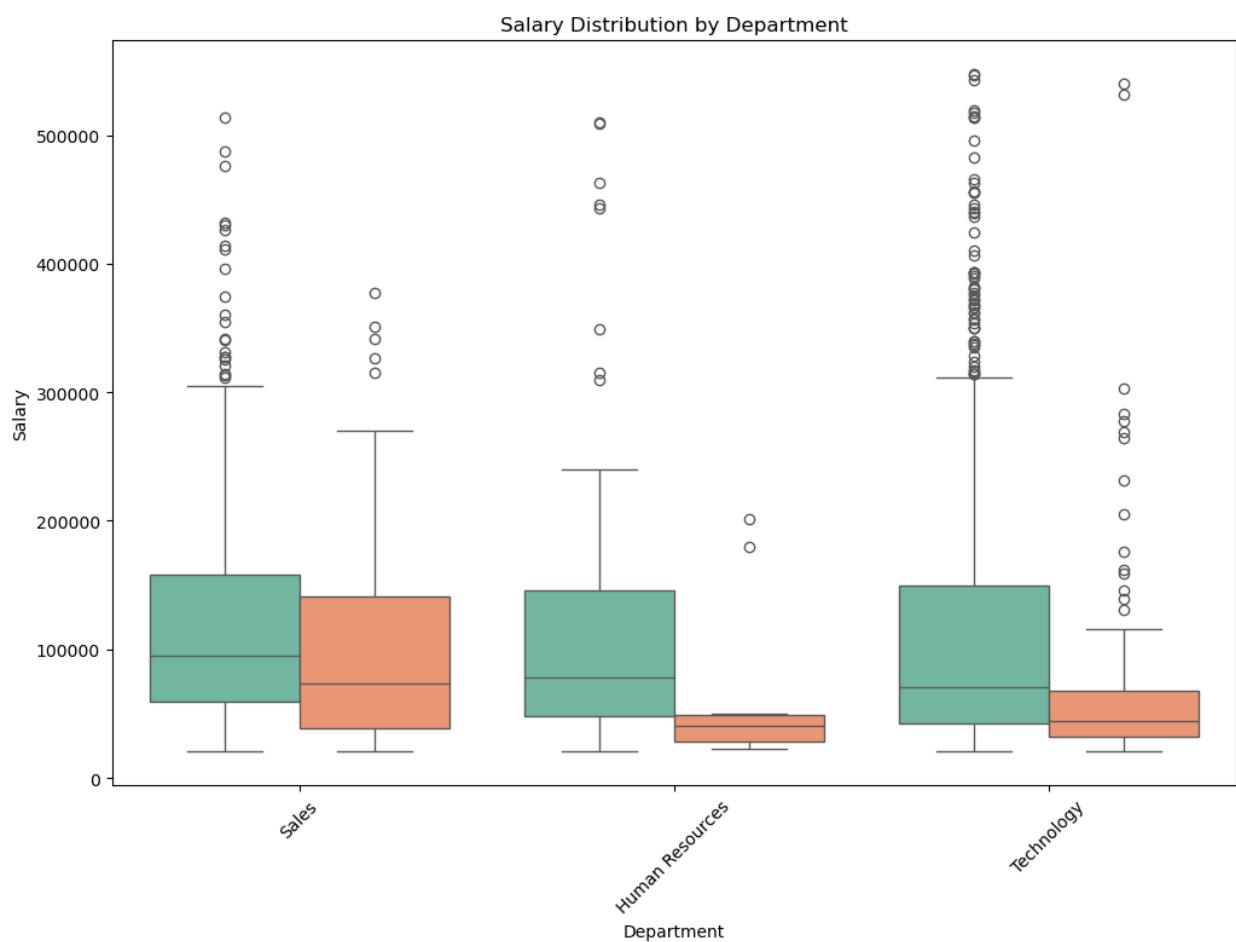
Average Salary by Department:

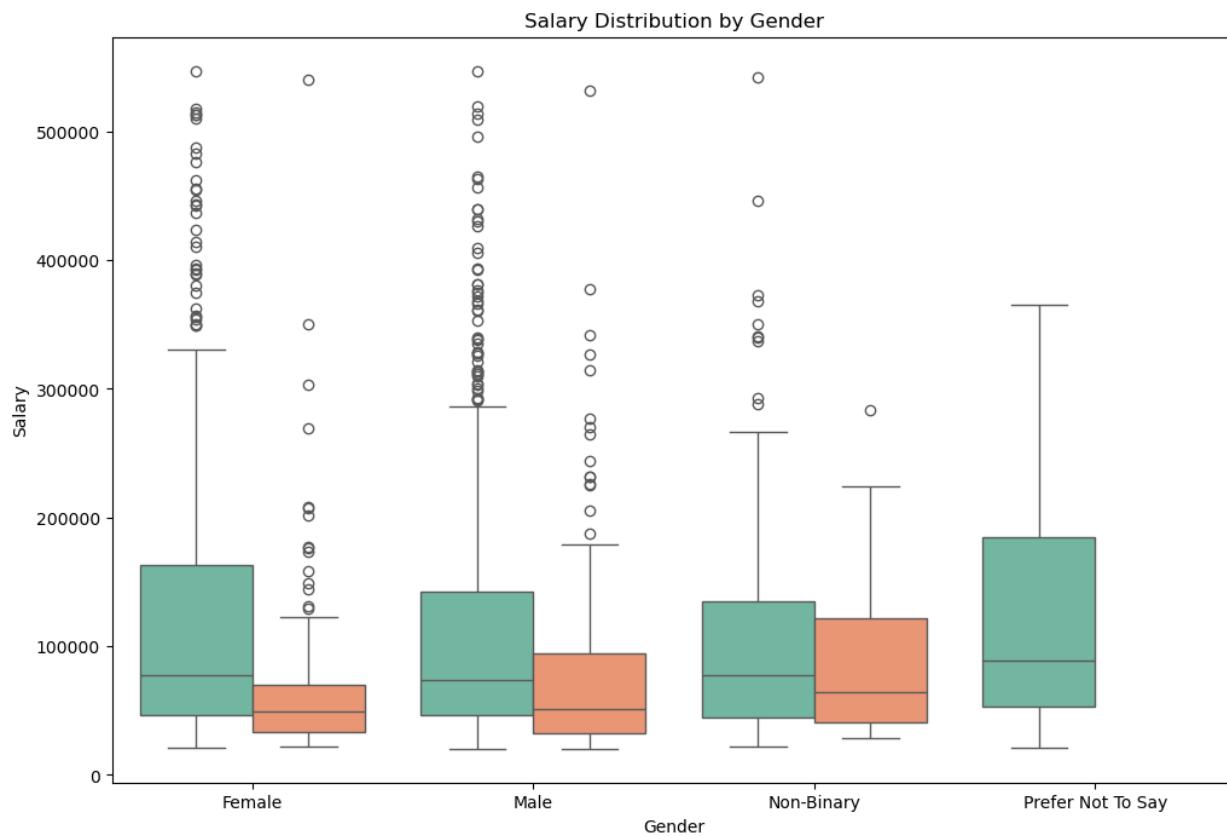
Department

Human Resources	119698.809524
Sales	119117.609865
Technology	109655.122789
Name: Salary, dtype: float64	

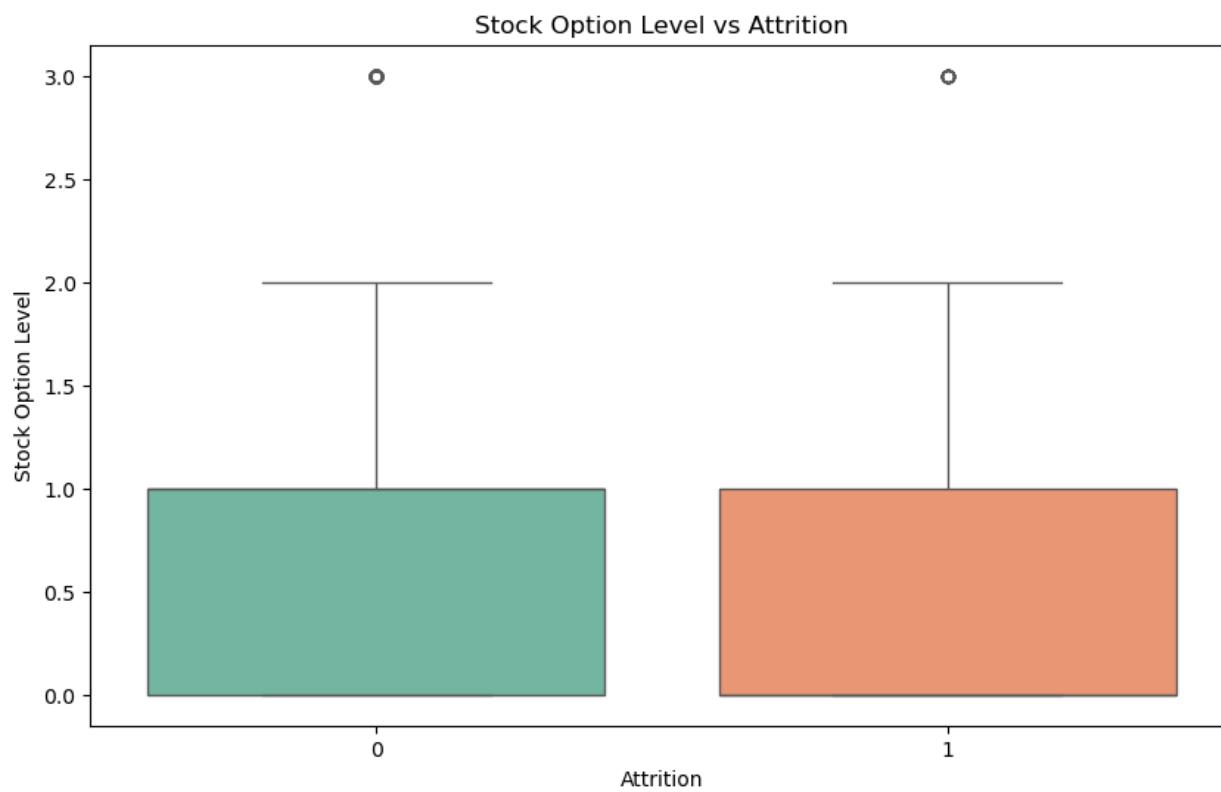
ANOVA Test Result for Salary by Department: F-statistic = 1.42, P-value = 0.2426

T-test for Salary by Gender: T-statistic = -0.38, P-value = 0.7046





...
Stock Option Level and Retention (Attrition):
 Correlation between StockOptionLevel and Attrition: -0.14



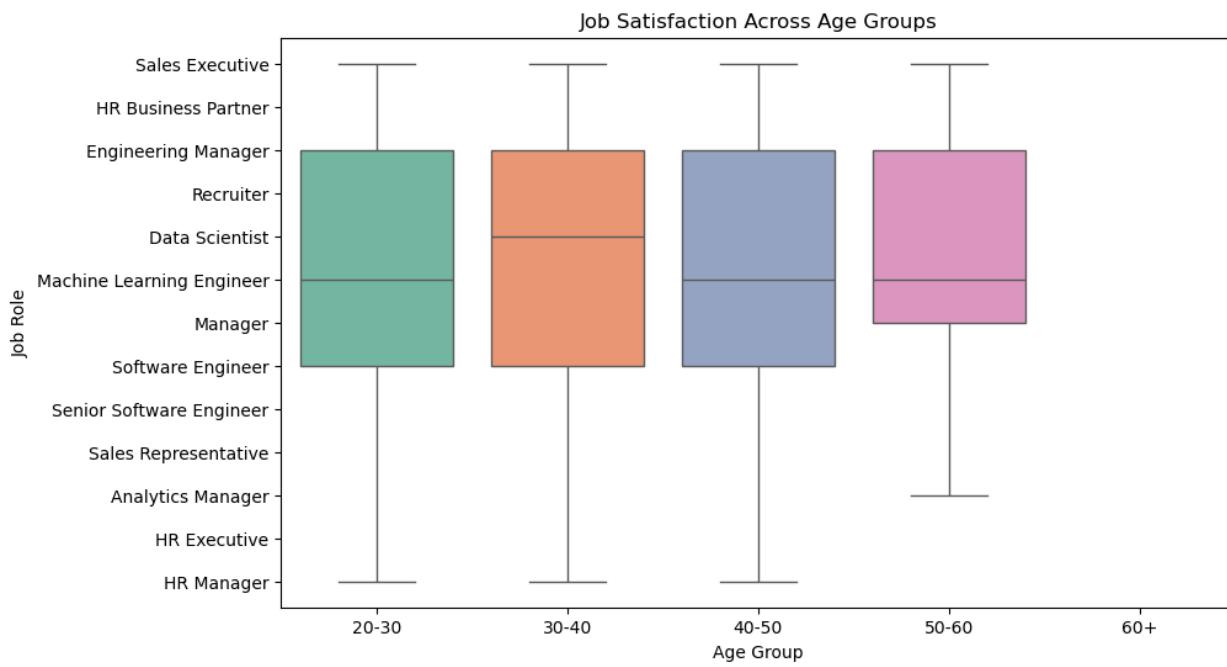
3) Compensation Analysis:

Age Impact on Job Satisfaction and Attrition:

- How does job satisfaction vary across different age groups?
- What is the attrition rate for different age groups?



```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from scipy import stats
5
6 # Load your dataset
7 file_path = r'F:\data analysis\BQ\HR Data.xlsx'
8 sheet_name = 'Employee'
9 employee_data = pd.read_excel(file_path, sheet_name=sheet_name)
10
11 # Ensure necessary columns exist
12 required_columns = ['Attrition', 'Age', 'JobRole', 'Department', 'Gender']
13 if all(col in employee_data.columns for col in required_columns):
14
15     # Convert Attrition to numerical values (Yes=1, No=0) if not already done
16     if 'AttritionNumeric' not in employee_data.columns:
17         employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})
18
19     # 8. Age Impact on Job Satisfaction and Attrition:
20     print("\nAge Impact on Job Satisfaction and Attrition:")
21
22     # Create age groups (e.g., 20-30, 30-40, 40-50, 50+)
23     age_bins = [20, 30, 40, 50, 60, 70]
24     age_labels = ['20-30', '30-40', '40-50', '50-60', '60+']
25     employee_data['AgeGroup'] = pd.cut(employee_data['Age'], bins=age_bins, labels=age_labels, right=False)
26
27     # 1. How does job satisfaction vary across different age groups?
28     # Assuming 'JobRole' or other columns are related to job satisfaction
29     # If you have a specific column for job satisfaction, you can use it here
30     plt.figure(figsize=(10, 6))
31     sns.boxplot(x='AgeGroup', y='JobRole', data=employee_data, palette='Set2')
32     plt.title('Job Satisfaction Across Age Groups')
33     plt.xlabel('Age Group')
34     plt.ylabel('Job Role')
35     plt.show()
36
37     # 2. What is the attrition rate for different age groups?
38     attrition_rate_by_age = employee_data.groupby('AgeGroup')['AttritionNumeric'].mean()
39     print("\nAttrition Rate by Age Group:")
40     print(attrition_rate_by_age)
41
42     # Visualize Attrition Rate by Age Group
43     plt.figure(figsize=(10, 6))
44     sns.barplot(x=attrition_rate_by_age.index, y=attrition_rate_by_age.values, palette='Set2')
45     plt.title('Attrition Rate by Age Group')
46     plt.xlabel('Age Group')
47     plt.ylabel('Attrition Rate')
48     plt.show()
49
50 else:
51     print("Some required columns are missing.")
```

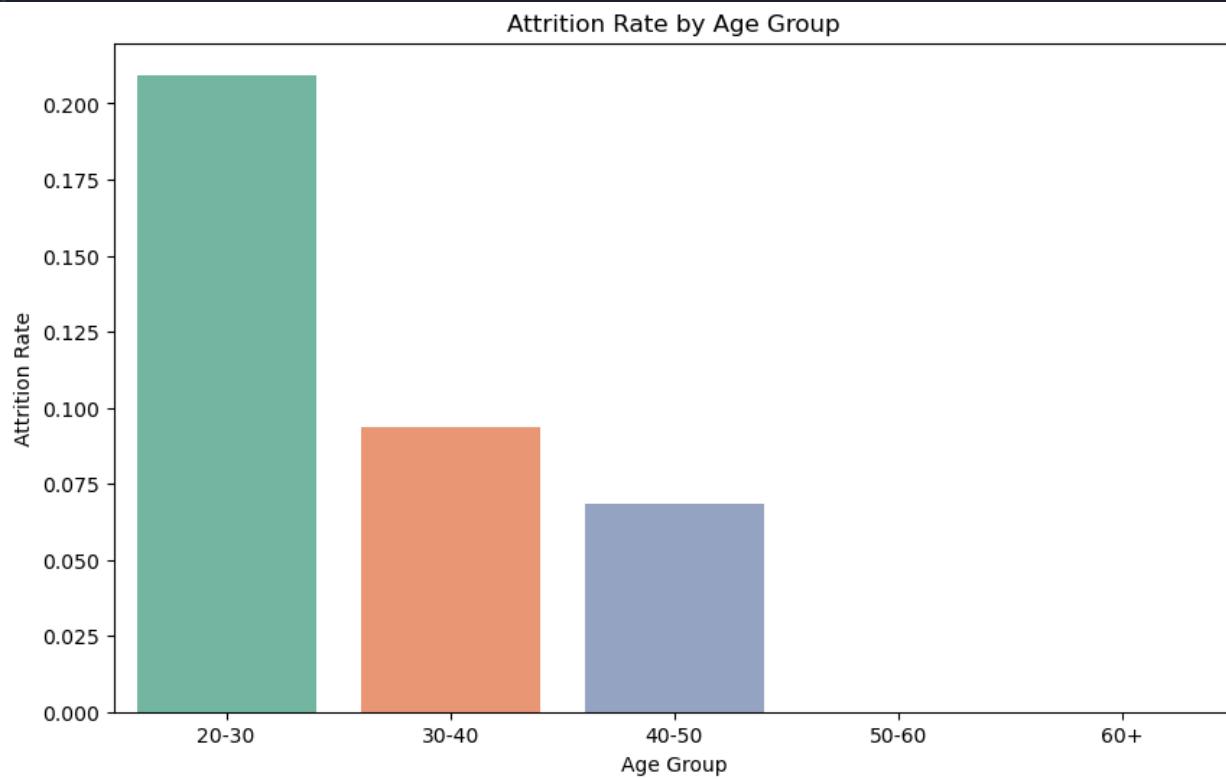


Attrition Rate by Age Group:

AgeGroup

```
20-30    0.209382
30-40    0.093426
40-50    0.068493
50-60    0.000000
60+      NaN
```

Name: AttritionNumeric, dtype: float64



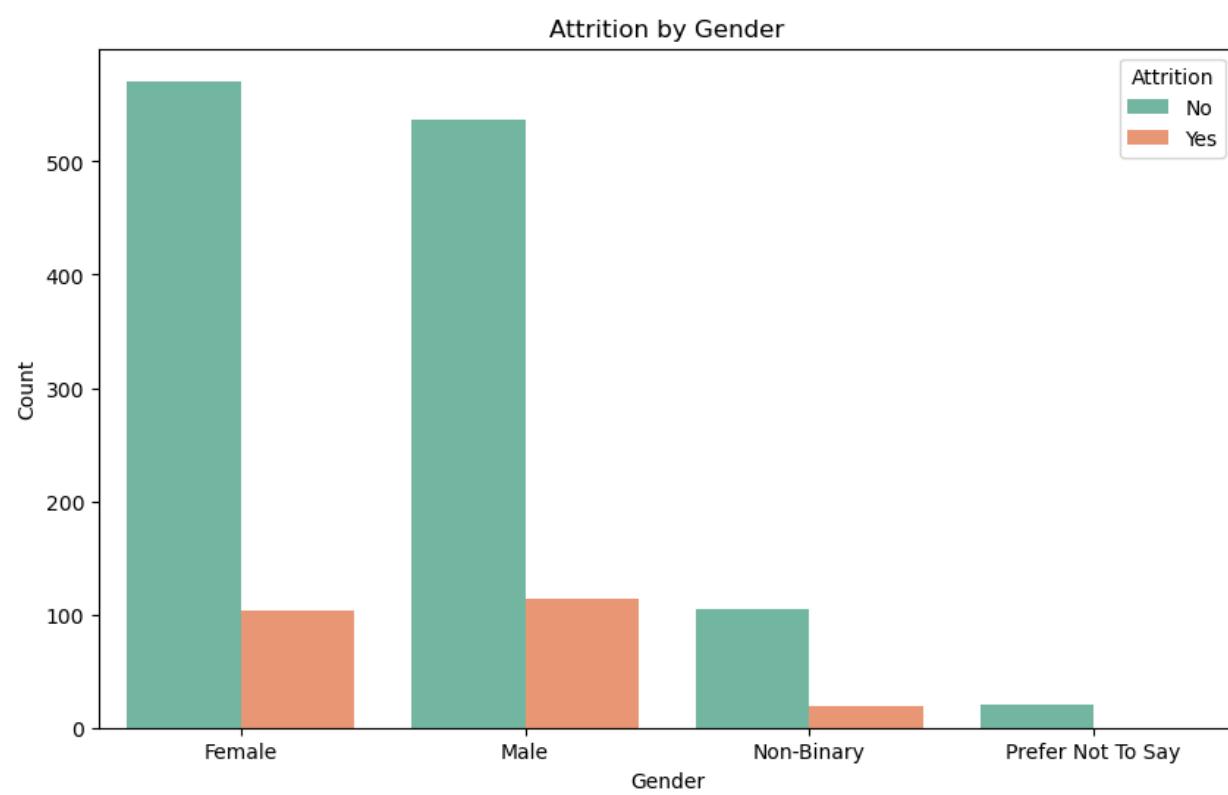
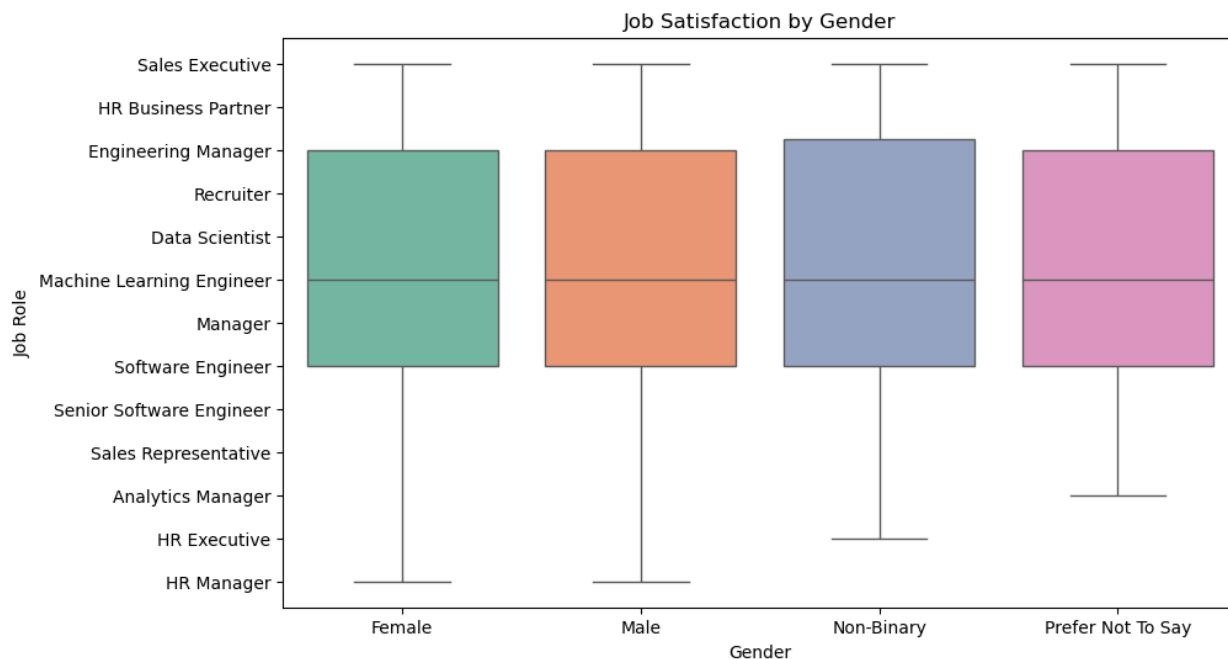
3) Compensation Analysis:

Gender Differences in Job Satisfaction and Performance:

- *How does job satisfaction vary between male and female employees?*
- *Is there a difference in performance ratings between male and female employees?*

```
● ● ●

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from scipy import stats
5
6 # Load your dataset
7 file_path = r'F:\data analysis\BQ\HR Data.xlsx'
8 sheet_name = 'Employee'
9 employee_data = pd.read_excel(file_path, sheet_name=sheet_name)
10
11 # Ensure necessary columns exist
12 required_columns = ['Attrition', 'Gender', 'JobRole', 'Age']
13 if all(col in employee_data.columns for col in required_columns):
14
15     # Convert Attrition to numerical values (Yes=1, No=0) if not already done
16     if 'AttritionNumeric' not in employee_data.columns:
17         employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})
18
19     # 9. Gender Differences in Job Satisfaction and Performance:
20     print("\nGender Differences in Job Satisfaction and Performance:")
21
22     # 1. How does job satisfaction vary between male and female employees?
23     # Using JobRole as a proxy for job satisfaction
24     plt.figure(figsize=(10, 6))
25     sns.boxplot(x='Gender', y='JobRole', data=employee_data, palette='Set2')
26     plt.title('Job Satisfaction by Gender')
27     plt.xlabel('Gender')
28     plt.ylabel('Job Role')
29     plt.show()
30
31     # 2. Is there a difference in retention (Attrition) by Gender?
32     plt.figure(figsize=(10, 6))
33     sns.countplot(x='Gender', hue='Attrition', data=employee_data, palette='Set2')
34     plt.title('Attrition by Gender')
35     plt.xlabel('Gender')
36     plt.ylabel('Count')
37     plt.show()
38
39     # Statistical Test: Chi-Square Test for Gender and Attrition
40     gender_attrition_crosstab = pd.crosstab(employee_data['Gender'], employee_data['Attrition'])
41     chi2_stat, p_val_gender_attrition, dof, expected = stats.chi2_contingency(gender_attrition_crosstab)
42     print(f"\nChi-Square Test for Gender and Attrition: Chi2-statistic = {chi2_stat:.2f}, P-value = {p_val_gender_attrition:.4f}")
43
44 else:
45     print("Some required columns are missing.")
```



...

Chi-Square Test for Gender and Attrition: Chi2-statistic = 5.09, P-value = 0.1655

3) Compensation Analysis:

Education Level and its Impact on Performance and Salary:

- *How does education level influence employee performance and salary variation?*
- *How does education level impact employee performance?*
- *How does salary vary by education level?*
- *How does education level impact employee performance by manager rating?*

Marital Status and Attrition:

How does marital status influence attrition rates and job satisfaction?

- *Does marital status affect attrition rates?*
- *Is there a relationship between marital status and job satisfaction?*

```
● ○ ●
```

```
1 import pandas as pd
2 from scipy.stats import f_oneway, ttest_ind
3
4 # Define file path
5 file_path = r'F:\data analysis\BQ\HR Data.xlsx'
6
7 # Load the sheets
8 employee_data = pd.read_excel(file_path, sheet_name='Employee')
9 performance_data = pd.read_excel(file_path, sheet_name='PerformanceRating')
10
11 # Merge the two datasets on 'EmployeeID'
12 merged_data = pd.merge(employee_data, performance_data, on='EmployeeID', how='inner')
13
14 # Add 'AttritionNumeric' column based on 'Attrition' (Yes=1, No=0)
15 merged_data['AttritionNumeric'] = merged_data['Attrition'].map({'Yes': 1, 'No': 0})
16
17 # 6. Salary Analysis Across Demographics and Departments
18 print("\nSalary Analysis Across Demographics and Departments:")
19
20 # Average salary by department
21 avg_salary_department = merged_data.groupby('Department')['Salary'].mean()
22 print("Average Salary by Department:")
23 print(avg_salary_department)
24
25 # ANOVA for salary by department
26 anova_salary_department = f_oneway(
27     *[merged_data.loc[merged_data['Department'] == dept, 'Salary']
28       for dept in merged_data['Department'].unique()])
29
30 print(f"ANOVA Test for Salary by Department: F-statistic = {anova_salary_department.statistic:.2f}, P-value = {anova_salary_department.pvalue:.4f}")
31
32 # T-test for salary by gender
33 male_salary = merged_data.loc[merged_data['Gender'] == 'Male', 'Salary']
34 female_salary = merged_data.loc[merged_data['Gender'] == 'Female', 'Salary']
35 t_test_gender_salary = ttest_ind(male_salary, female_salary)
36 print(f"T-test for Salary by Gender: T-statistic = {t_test_gender_salary.statistic:.2f}, P-value = {t_test_gender_salary.pvalue:.4f}")
37
38 # 7. Stock Option Level and Retention (Attrition)
39 print("\nStock Option Level and Retention (Attrition):")
40 correlation_stock = merged_data['StockOptionLevel'].corr(merged_data['AttritionNumeric'])
41 print(f"Correlation between StockOptionLevel and Attrition: {correlation_stock:.2f}")
42
43 # 8. Age Impact on Job Satisfaction and Attrition
44 print("\nAge Impact on Job Satisfaction and Attrition:")
45
46 # Job satisfaction by age group
47 merged_data['AgeGroup'] = pd.cut(merged_data['Age'], bins=[18, 30, 40, 50, 60], labels=['18-30', '31-40', '41-50', '51-60'])
48 avg_job_satisfaction_age = merged_data.groupby('AgeGroup')['JobSatisfaction'].mean()
49 print("Average Job Satisfaction by Age Group:")
50 print(avg_job_satisfaction_age)
```

```

52 # Attrition rate by age group
53 attrition_rate_age = merged_data.groupby('AgeGroup')['AttritionNumeric'].mean()
54 print("Attrition Rate by Age Group:")
55 print(attrition_rate_age)
56
57 # 9. Gender Differences in Job Satisfaction and Performance
58 print("\nGender Differences in Job Satisfaction and Performance:")
59
60 # Job satisfaction by gender
61 avg_job_satisfaction_gender = merged_data.groupby('Gender')['JobSatisfaction'].mean()
62 print("Average Job Satisfaction by Gender:")
63 print(avg_job_satisfaction_gender)
64
65 # Manager rating by gender
66 avg_manager_rating_gender = merged_data.groupby('Gender')['ManagerRating'].mean()
67 print("Average Manager Rating by Gender:")
68 print(avg_manager_rating_gender)
69
70 # 10. Education Level and its Impact on Performance and Salary
71 print("\nEducation Level and its Impact on Performance and Salary:")
72
73 # Salary by education level
74 avg_salary_education = merged_data.groupby('Education')['Salary'].mean()
75 print("Average Salary by Education Level:")
76 print(avg_salary_education)
77
78 # Manager rating by education level
79 avg_manager_rating_education = merged_data.groupby('Education')['ManagerRating'].mean()
80 print("Average Manager Rating by Education Level:")
81 print(avg_manager_rating_education)
82
83 # 11. Marital Status and Attrition
84 print("\nMarital Status and Attrition:")
85
86 # Attrition rate by marital status
87 attrition_rate_marital = merged_data.groupby('MaritalStatus')['AttritionNumeric'].mean()
88 print("Attrition Rate by Marital Status:")
89 print(attrition_rate_marital)
90
91 # Job satisfaction by marital status
92 avg_job_satisfaction_marital = merged_data.groupby('MaritalStatus')['JobSatisfaction'].mean()
93 print("Job Satisfaction by Marital Status:")
94 print(avg_job_satisfaction_marital)
95
96 # Visualizations
97 sns.boxplot(x='Department', y='Salary', data=merged_data, palette='Set2')
98 plt.title("Salary Distribution by Department")
99 plt.show()
100
101 sns.barplot(x='AgeGroup', y='JobSatisfaction', data=merged_data, palette='viridis')
102 plt.title("Job Satisfaction by Age Group")
103 plt.show()
104
105

```

```
...
Salary Analysis Across Demographics and Departments:
Average Salary by Department:
Department
Human Resources      105804.270627
Sales                121045.081899
Technology           106396.216115
Name: Salary, dtype: float64
ANOVA Test for Salary by Department: F-statistic = 16.40, P-value = 0.0000
T-test for Salary by Gender: T-statistic = 1.61, P-value = 0.1079
```

```
Stock Option Level and Retention (Attrition):
Correlation between StockOptionLevel and Attrition: -0.17
```

```
Age Impact on Job Satisfaction and Attrition:
```

```
Average Job Satisfaction by Age Group:
```

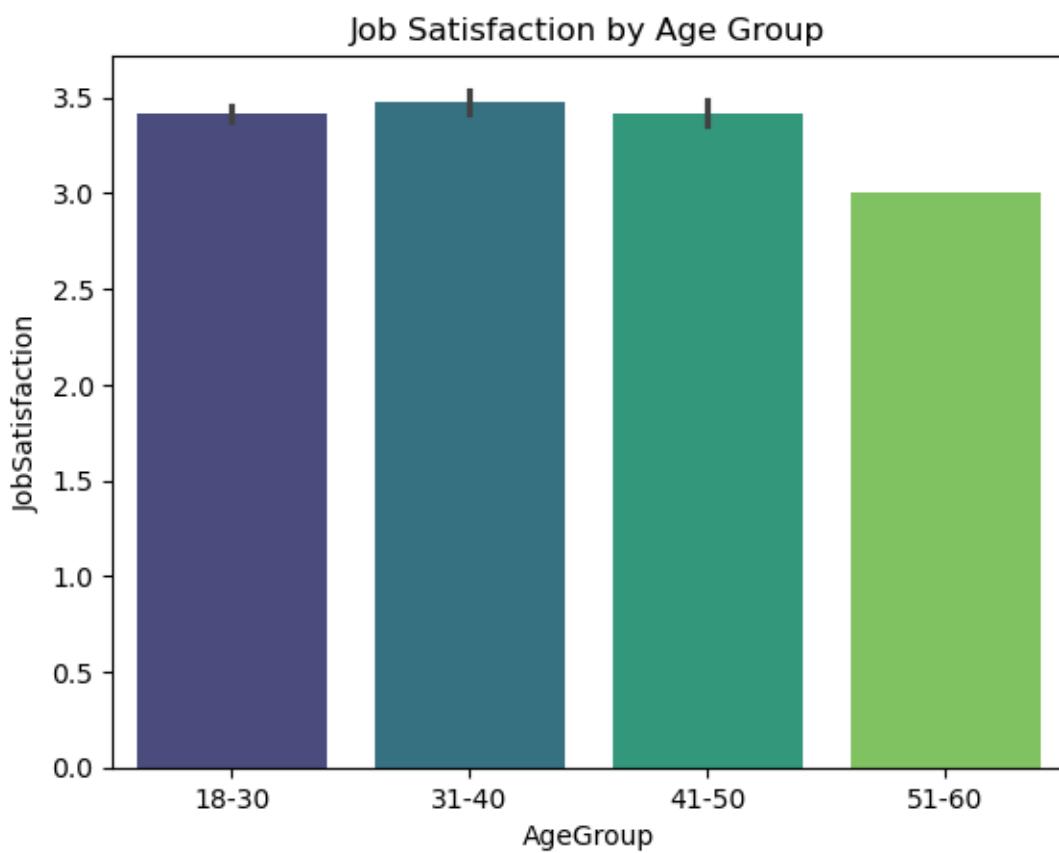
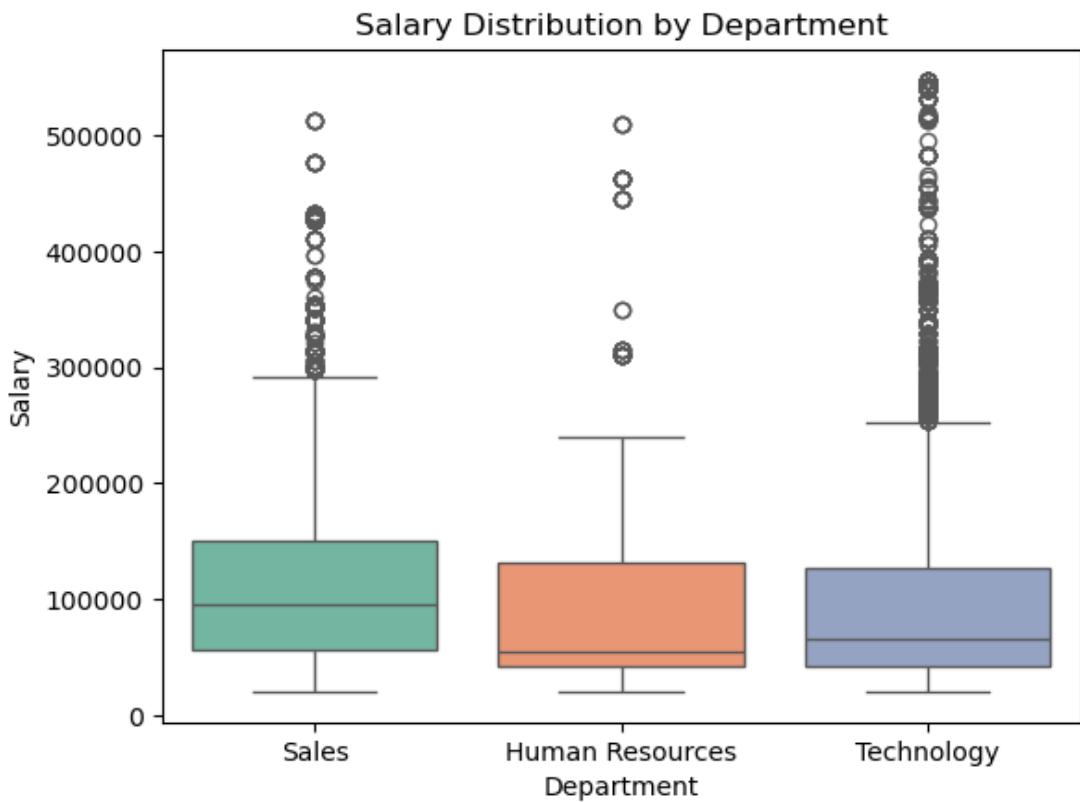
```
AgeGroup
18-30    3.416400
31-40    3.477931
41-50    3.420327
51-60    3.000000
```

```
Name: JobSatisfaction, dtype: float64
```

```
Attrition Rate by Age Group:
```

```
AgeGroup
18-30    0.470574
...
Divorced  3.405199
Married   3.440098
Single    3.433018
```

```
Name: JobSatisfaction, dtype: float64
```



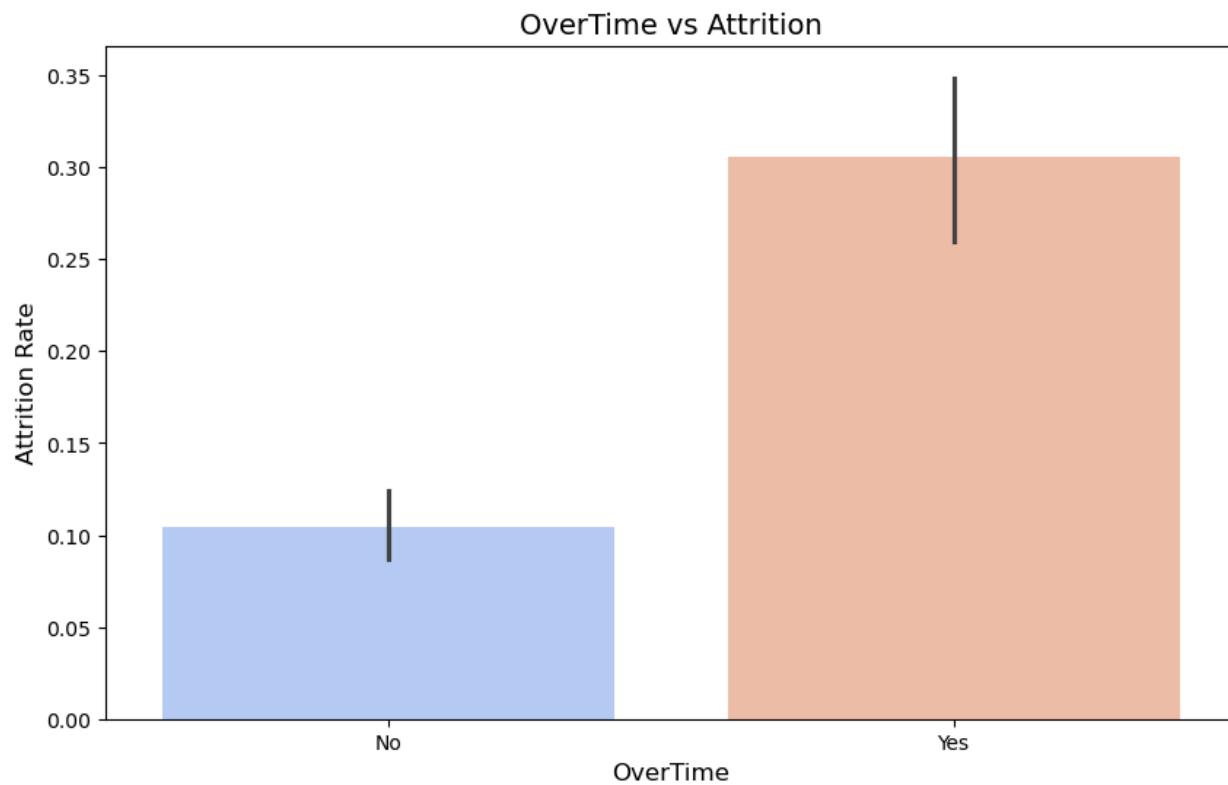
2. Geographic Insights:

4) Work Environment:

- 8 .Is there a correlation between overtime work and performance or attrition ?
- 9 .How does business travel affect employee satisfaction and retention ?

```
● ● ●  
1 import pandas as pd  
2 import seaborn as sns  
3 import matplotlib.pyplot as plt  
4  
5 # Load the dataset  
6 file_path = r'F:\data analysis\BQ\HR Data.xlsx'  
7 sheet_name = 'Employee'  
8 employee_data = pd.read_excel(file_path, sheet_name=sheet_name)  
9  
10 # Check columns  
11 if 'OverTime' in employee_data.columns and 'Attrition' in employee_data.columns:  
12     # Convert OverTime and Attrition to numeric  
13     employee_data['OverTimeNumeric'] = employee_data['OverTime'].map({'Yes': 1, 'No': 0})  
14     employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})  
15  
16     # Correlation between OverTime and Attrition  
17     correlation_attrition = employee_data['OverTimeNumeric'].corr(employee_data['AttritionNumeric'])  
18     print(f"Correlation between OverTime and Attrition: {correlation_attrition:.2f}")  
19  
20     # Visualization for Attrition  
21     plt.figure(figsize=(10, 6))  
22     sns.barplot(x='OverTime', y='AttritionNumeric', data=employee_data, palette='coolwarm')  
23     plt.title('OverTime vs Attrition', fontsize=14)  
24     plt.xlabel('OverTime', fontsize=12)  
25     plt.ylabel('Attrition Rate', fontsize=12)  
26     plt.show()  
27  
28     # Analyze correlation with performance if available  
29     if 'Performance' in employee_data.columns:  
30         correlation_performance = employee_data['OverTimeNumeric'].corr(employee_data['Performance'])  
31         print(f"Correlation between OverTime and Performance: {correlation_performance:.2f}")  
32     else:  
33         print("Required columns 'OverTime' or 'Attrition' not found in the dataset.")  
34
```

```
... Correlation between OverTime and Attrition: 0.25
```



```

1 # Import required libraries
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import numpy as np
6 from scipy.stats import pearsonr
7 from scipy.stats import ttest_ind
8 from sklearn.preprocessing import LabelEncoder
9 from sklearn.model_selection import train_test_split
10 from sklearn.preprocessing import StandardScaler
11 from sklearn.linear_model import LogisticRegression
12 from sklearn.ensemble import RandomForestClassifier
13 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
14
15 # Load data from both sheets
16 df_employee = pd.read_excel('Data.xlsx', sheet_name='Employee')
17 df_performance = pd.read_excel('Data.xlsx', sheet_name='PerformanceRating')
18
19 # Merge the two dataframes on EmployeeID
20 df = pd.merge(df_employee, df_performance, on='EmployeeID', how='inner')
21
22 print(df.head())
23 # Check for missing values in the dataset
24 print(df.isnull().sum())
25 # Display basic information about the dataset (column types, memory usage, etc.)
26 print(df.info())
27
28
29 # Preprocessing categorical columns for analysis
30 # Convert 'OverTime', 'Attrition', and 'BusinessTravel' to numeric
31 df['OverTime'] = df['OverTime'].map({'Yes': 1, 'No': 0})
32 df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})
33 label_enc = LabelEncoder()
34 df['BusinessTravel'] = label_enc.fit_transform(df['BusinessTravel'])

```

	EmployeeID	FirstName	LastName	Gender	Age	BusinessTravel	Department	
0	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	
1	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	
2	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	
3	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	
4	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	
	DistanceFromHome	State	Ethnicity	...	ReviewDate	EnvironmentSatisfaction		
0	27	IL	White	...	10/30/2016	3		
1	27	IL	White	...	10/30/2017	4		
2	27	IL	White	...	10/30/2018	5		
3	27	IL	White	...	10/30/2019	1		
4	27	IL	White	...	10/31/2014	3		
	JobSatisfaction	RelationshipSatisfaction	TrainingOpportunitiesWithinYear					
0	3		2			3		
1	4		5			3		
2	5		4			3		
3	3		2			3		
4	4		2			1		
	TrainingOpportunitiesTaken	WorkLifeBalance	SelfRating	ManagerRating				
0		0	4	3		3		
1		1	2	3		2		
2		0	4	5		5		
..	DataQualityStatus		6709 non-null	object				
	dtypes:	datetime64[ns](1),	int64(17),	object(16)				
	memory usage:	1.7+ MB						
	None							

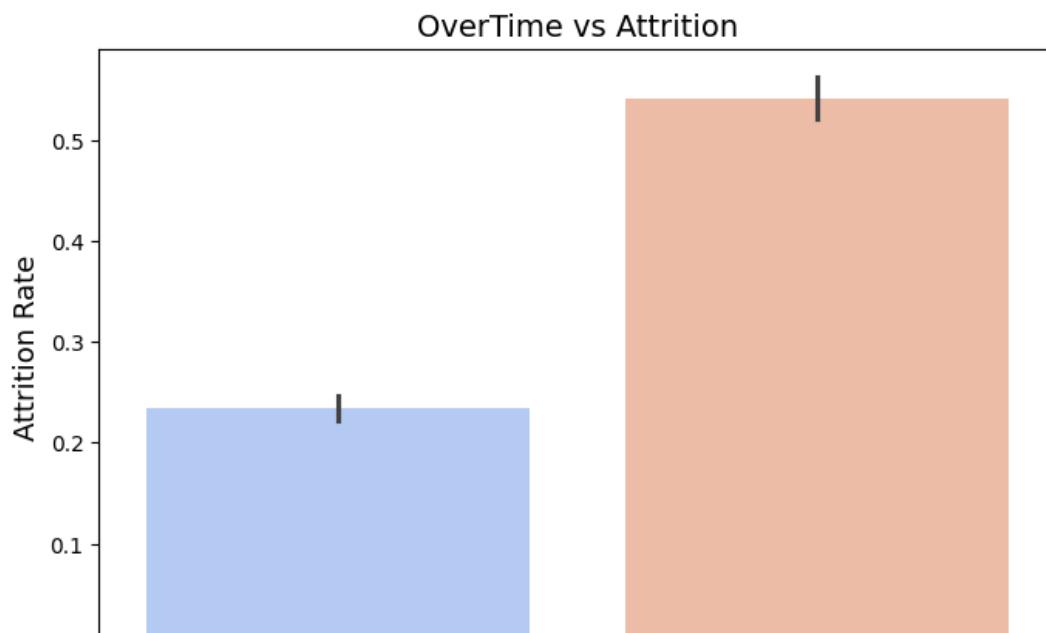
2.Geographic Insights:

18.Is there a correlation between overtime work and performance or attrition?

Approach: We'll analyze the relationship between `OverTime` and `Attrition`.

```
● ● ●  
1 # Calculate correlation between OverTime and Attrition  
2 correlation = df[['OverTime', 'Attrition']].corr()  
3 print(f"Correlation between OverTime and Attrition:\n{correlation}")  
4  
5  
6 # Visualization  
7 plt.figure(figsize=(8, 5))  
8 sns.barplot(x='OverTime', y='Attrition', data=df, palette='coolwarm')  
9 plt.title('OverTime vs Attrition', fontsize=14)  
10 plt.xlabel('OverTime', fontsize=13)  
11 plt.ylabel('Attrition Rate', fontsize=13)  
12 plt.show()
```

```
Correlation between OverTime and Attrition:  
          OverTime  Attrition  
OverTime    1.000000   0.305746  
Attrition   0.305746   1.000000
```



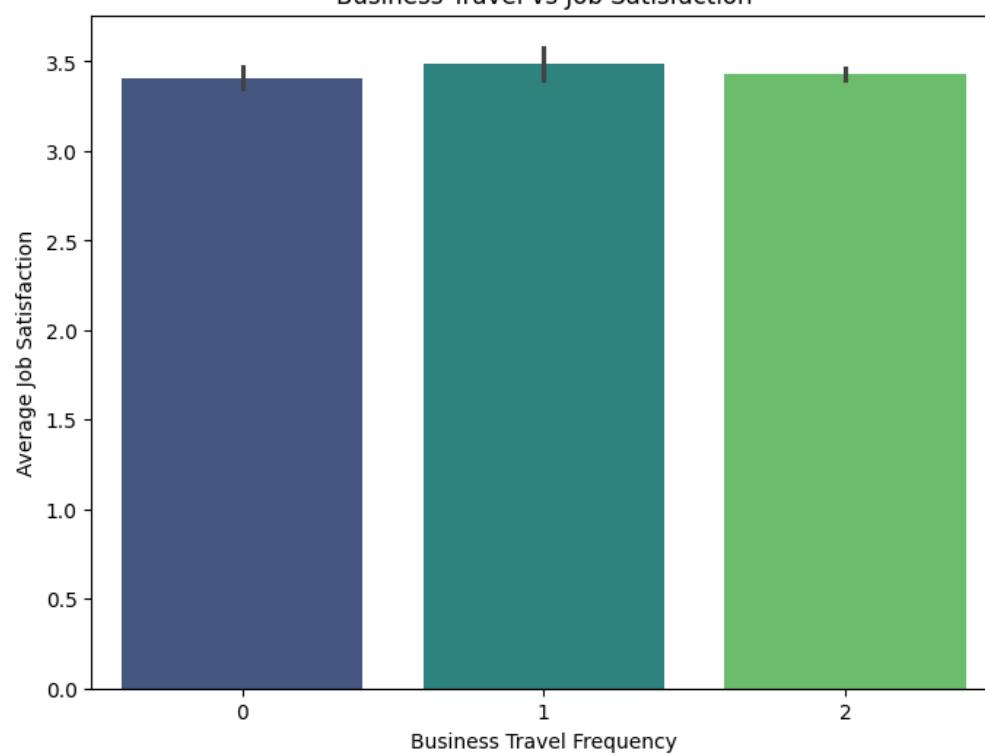
19. How does business travel affect employee satisfaction and retention?

We analyzed how `BusinessTravel` impacts `JobSatisfaction` and `Attrition` by computing their averages for each category and visualizing the trends using bar plots.

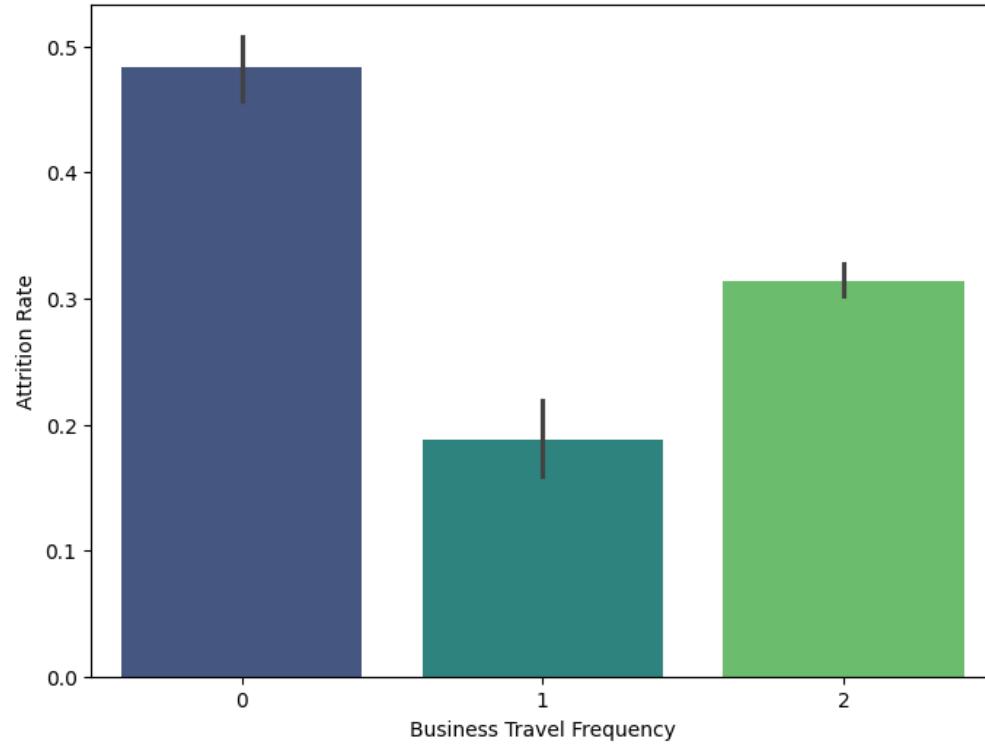
```
● ● ●  
1 # Average Job Satisfaction and Attrition by Business Travel category  
2 travel_satisfaction = df.groupby('BusinessTravel')[['JobSatisfaction', 'Attrition']].mean()  
3 print(f"Business Travel and Job Satisfaction:\n{travel_satisfaction}")  
4  
5 # Visualization: Business Travel vs Job Satisfaction  
6 plt.figure(figsize=(8, 6))  
7 sns.barplot(x='BusinessTravel', y='JobSatisfaction', data=df, palette='viridis')  
8 plt.title('Business Travel vs Job Satisfaction')  
9 plt.xlabel('Business Travel Frequency')  
10 plt.ylabel('Average Job Satisfaction')  
11 plt.show()  
12  
13 # Visualization: Business Travel vs Attrition  
14 plt.figure(figsize=(8, 6))  
15 sns.barplot(x='BusinessTravel', y='Attrition', data=df, palette='viridis')  
16 plt.title('Business Travel vs Attrition')  
17 plt.xlabel('Business Travel Frequency')  
18 plt.ylabel('Attrition Rate')  
19 plt.show()
```

```
Business Travel and Job Satisfaction:  
          JobSatisfaction  Attrition  
BusinessTravel  
0                  3.410162  0.483063  
1                  3.489362  0.188216  
2                  3.428903  0.314346
```

Business Travel vs Job Satisfaction



Business Travel vs Attrition



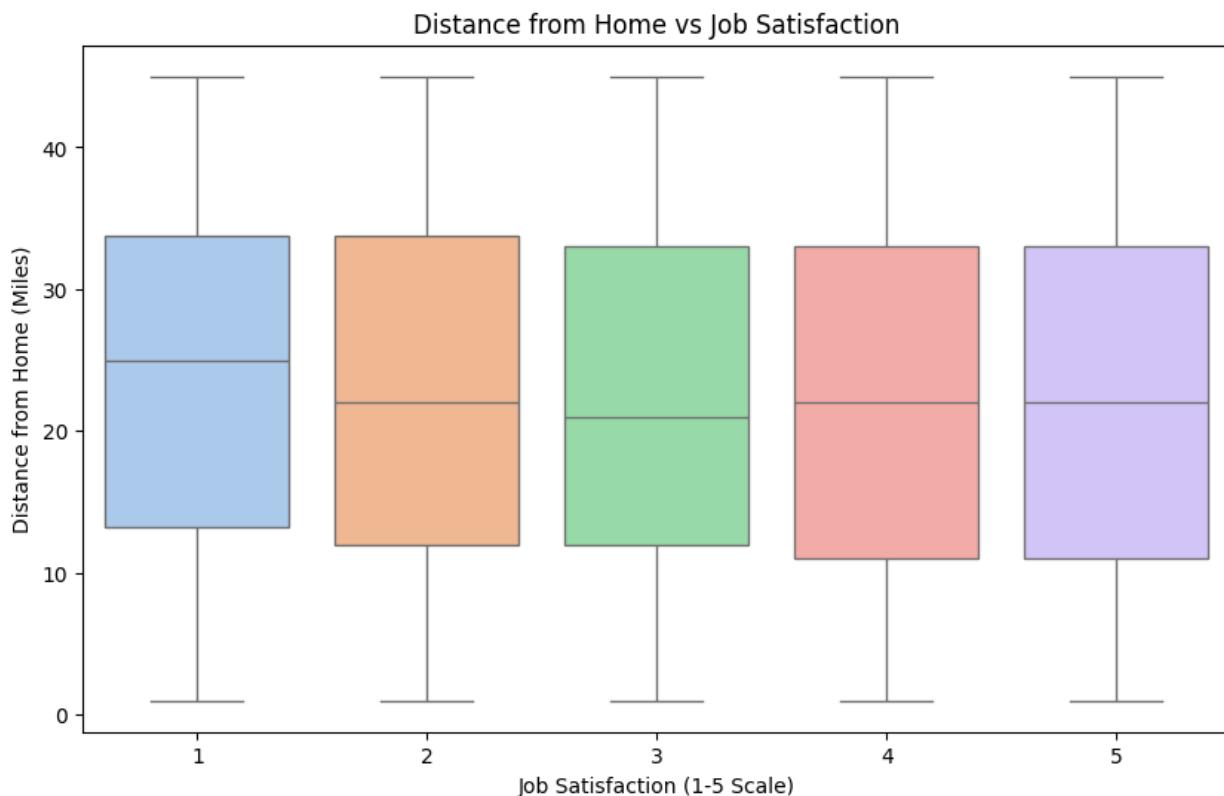
20. How does the distance from home affect job satisfaction?

We analyze the relationship between `DistanceFromHome` and `JobSatisfaction` by computing their correlation and visualizing the distribution using a box plot.

```
● ● ●  
1 # Correlation between DistanceFromHome and Job Satisfaction  
2 correlation_distance = df[['DistanceFromHome', 'JobSatisfaction']].corr()  
3 print(f"Correlation between Distance from Home and Job Satisfaction:\n{correlation_distance}")  
4  
5 # Assuming 'JobSatisfaction' is a column in the dataset  
6 plt.figure(figsize=(10, 6))  
7 sns.boxplot(x='JobSatisfaction', y='DistanceFromHome', data=df, palette='pastel')  
8 plt.title('Distance from Home vs Job Satisfaction')  
9 plt.xlabel('Job Satisfaction (1-5 Scale)')  
10 plt.ylabel('Distance from Home (Miles)')  
11 plt.show()
```

Correlation between Distance from Home and Job Satisfaction:

	DistanceFromHome	JobSatisfaction
DistanceFromHome	1.000000	-0.011745
JobSatisfaction	-0.011745	1.000000



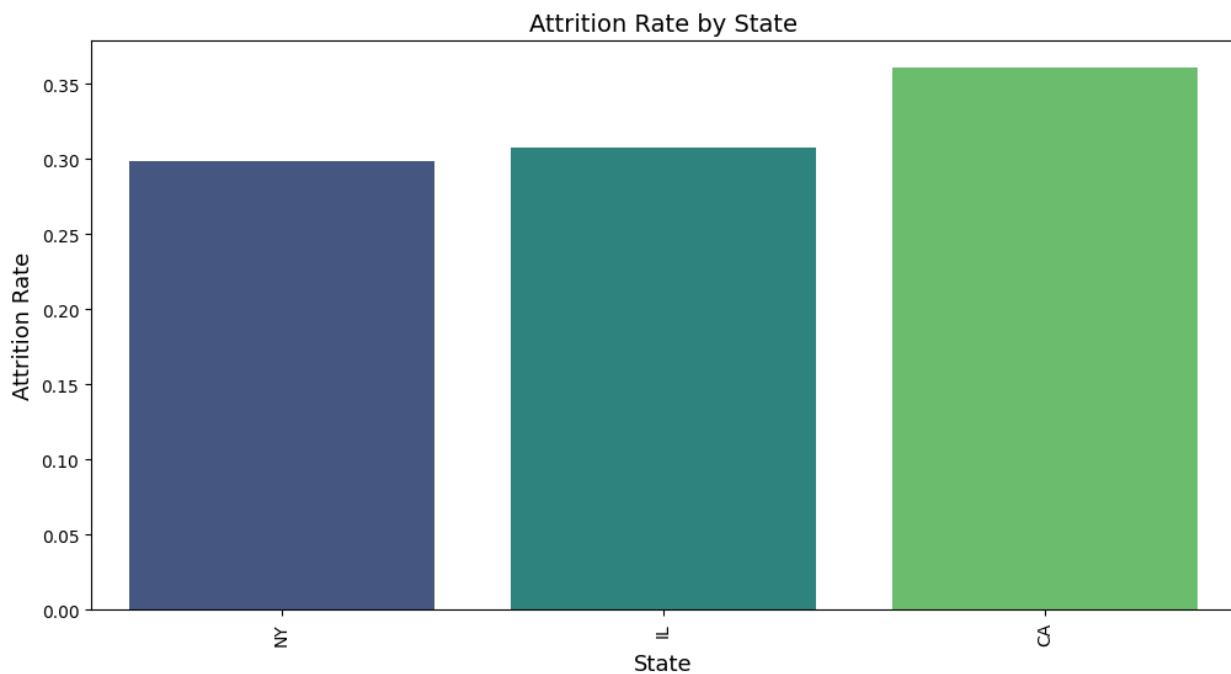
21.Which states have the highest and lowest attrition rates?

We'll group by the `State` column and calculate the `attrition` rate for each state.

```
● ● ●  
1 # Group by State and calculate the attrition rate  
2 state_attrition = df.groupby('State')['Attrition'].mean().sort_values()  
3  
4 # Print the attrition rates by state  
5 print(f"Attrition Rate by State:\n{state_attrition}")  
6  
7 # Visualization  
8 plt.figure(figsize=(12, 6))  
9 sns.barplot(x=state_attrition.index, y=state_attrition.values, palette='viridis')  
10 plt.title('Attrition Rate by State', fontsize=14)  
11 plt.xlabel('State', fontsize=13)  
12 plt.ylabel('Attrition Rate', fontsize=13)  
13 plt.xticks(rotation=90)  
14 plt.show()
```

Attrition Rate by State:

State	Attrition Rate
NY	0.298378
IL	0.307407
CA	0.360583



22.Which states have the highest percentage of employees leaving due to excessive overtime?

We can compute the `Attrition` rate among `OverTime` workers by filtering employees where `Attrition = 1` and `OverTime = 1`, then calculating the percentage per `State`.

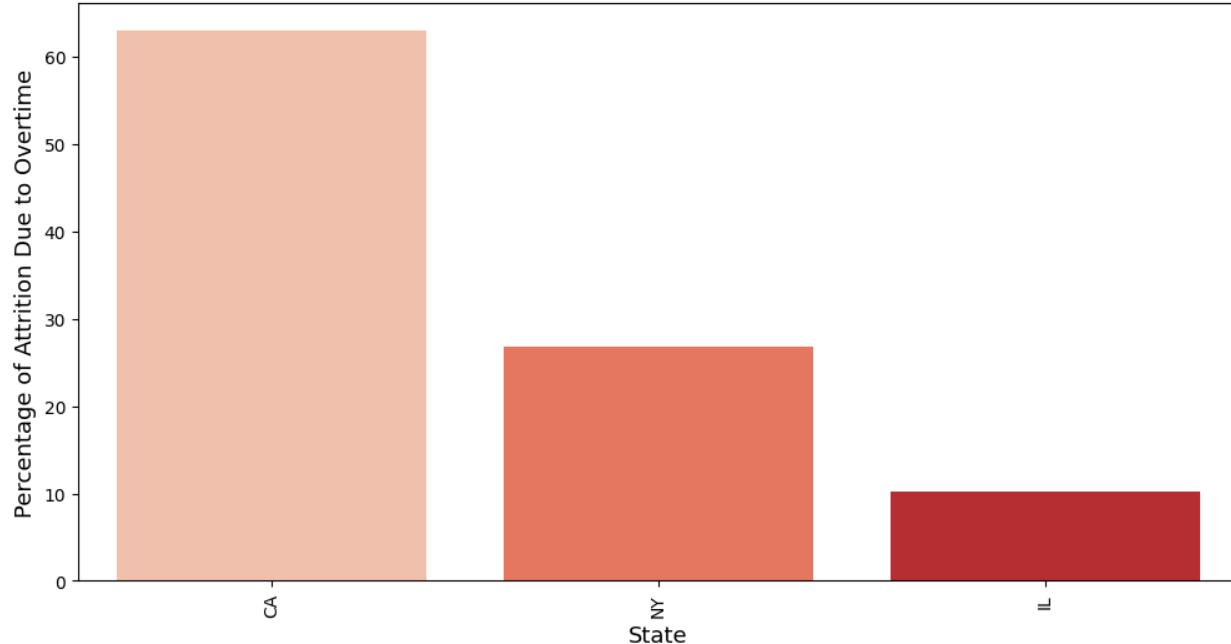


```
1 # Filter for employees who work overtime and calculate attrition rate by state
2 overtime_attrition = df[df['OverTime'] == 1].groupby('State')['Attrition'].mean().sort_values()
3
4 print(f'Overtime Attrition Rate by State:\n{overtime_attrition}')
5
6 # Filter employees who left and worked overtime
7 overtime_attrition = df[(df['Attrition'] == 1) & (df['OverTime'] == 1)]
8
9 # Calculate percentage of attrition due to overtime per state
10 state_counts = (overtime_attrition['State'].value_counts(normalize=True) * 100).sort_values(ascending=False)
11
12 # Visualization - Bar Plot
13 plt.figure(figsize=(12, 6))
14 sns.barplot(x=state_counts.index, y=state_counts.values, palette='Reds')
15 plt.title('States with Highest Attrition Due to Overtime', fontsize=14)
16 plt.xlabel('State', fontsize=13)
17 plt.ylabel('Percentage of Attrition Due to Overtime', fontsize=13)
18 plt.xticks(rotation=90)
19 plt.show()
20
21 # Visualization - Pie Chart
22 plt.figure(figsize=(10, 6))
23 state_counts.plot(kind='pie', autopct='%1.1f%%')
24 plt.title('States with Attrition Due to Overtime')
25 plt.ylabel('')
26 plt.show()
```

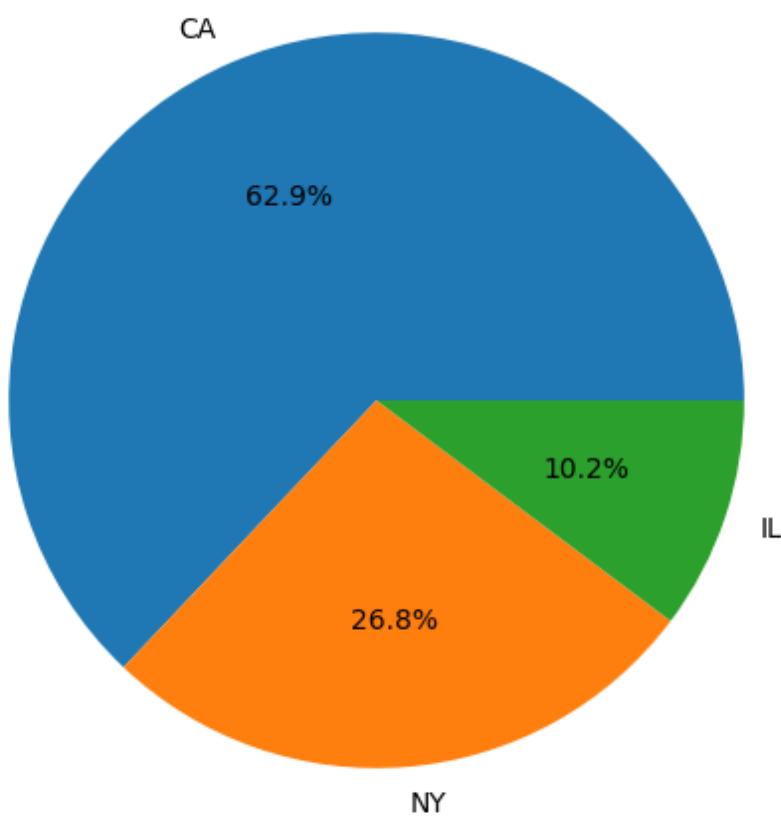
Overtime Attrition Rate by State:

State	Attrition Rate
NY	0.507812
CA	0.554585
IL	0.556054

States with Highest Attrition Due to Overtime



States with Attrition Due to Overtime



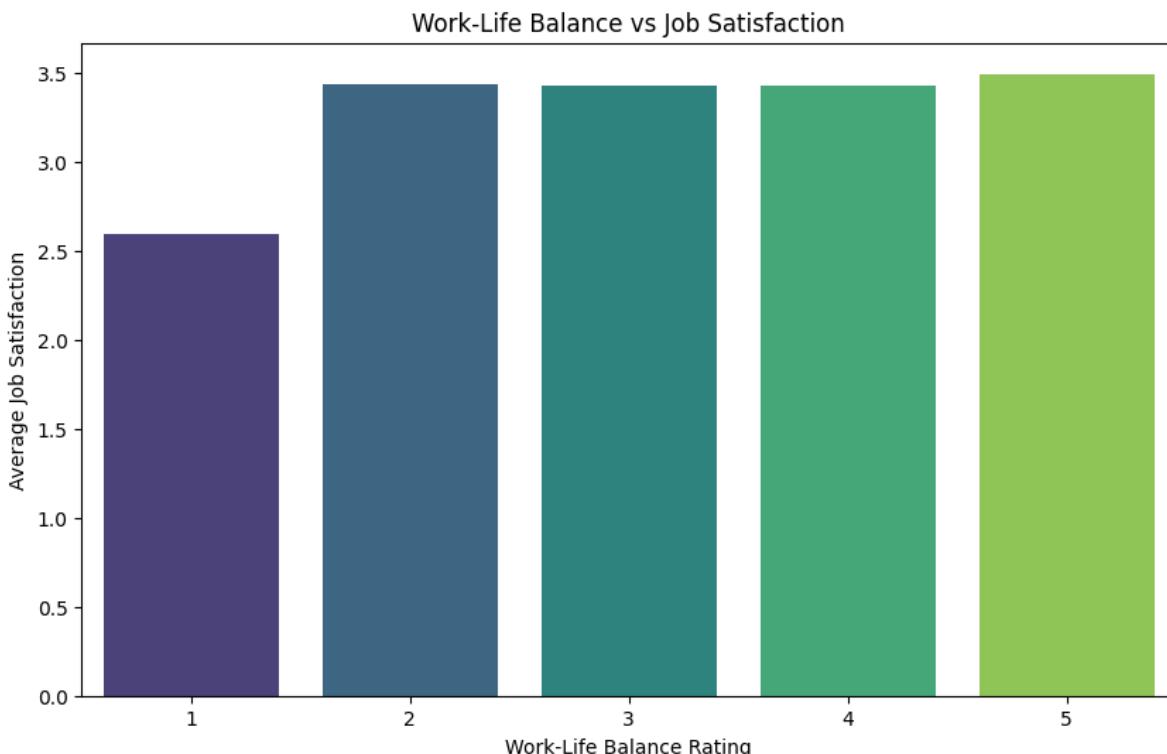
23. How does work-life balance affect overall job satisfaction?

We can use BusinessTravel to analyze its impact on `WorkLifeBalance` ratings and visualize the distribution of work-life balance across different business travel frequencies, while highlighting how `DistanceFromHome` relates to `ManagerRating` and how `WorkLifeBalance` influences JobSatisfaction.

```
● ● ●  
1  
2 # Calculate correlation between DistanceFromHome and ManagerRating  
3 distance_performance_correlation = df[['DistanceFromHome', 'ManagerRating']].corr()  
4 print(f"Correlation between Distance from Home and Manager Rating:\n{distance_performance_correlation}")  
5  
6 # Assuming 'WorkLifeBalance' and 'JobSatisfaction' are ordinal columns (1-5)  
7 plt.figure(figsize=(10, 6))  
8 sns.barplot(x='WorkLifeBalance', y='JobSatisfaction', data=df, ci=None, palette='viridis')  
9 plt.title('Work-Life Balance vs Job Satisfaction')  
10 plt.xlabel('Work-Life Balance Rating')  
11 plt.ylabel('Average Job Satisfaction')  
12 plt.show()
```

Correlation between Distance from Home and Manager Rating:

	DistanceFromHome	ManagerRating
DistanceFromHome	1.000000	0.025832
ManagerRating	0.025832	1.000000



24. Is there a relationship between work-life balance and business travel frequency?

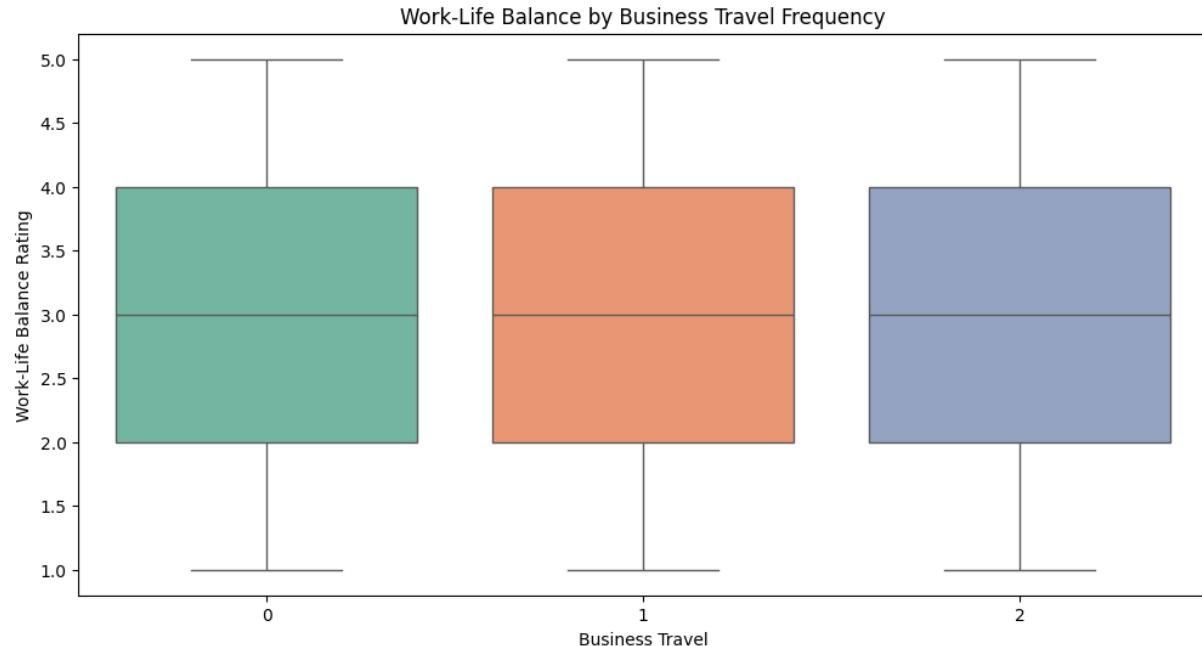
We can use `BusinessTravel` to analyze its impact on `WorkLifeBalance` ratings and visualize the distribution of work-life balance across different business travel frequencies.



```
1 # Calculate the average Work-Life Balance rating for each Business Travel frequency
2 average_work_life_balance = df.groupby('BusinessTravel')['WorkLifeBalance'].mean().reset_index()
3
4 # Display the average Work-Life Balance values
5 print("Average Work-Life Balance by Business Travel Frequency:")
6 print(average_work_life_balance)
7
8 # Visualize the relationship between Business Travel frequency and Work-Life Balance ratings
9 plt.figure(figsize=(12, 6))
10 sns.boxplot(x='BusinessTravel', y='WorkLifeBalance', data=df, palette='Set2')
11 plt.title('Work-Life Balance by Business Travel Frequency')
12 plt.xlabel('Business Travel')
13 plt.ylabel('Work-Life Balance Rating')
14 plt.show()
15
```

Average Work-Life Balance by Business Travel Frequency:

BusinessTravel	WorkLifeBalance
0	3.400589
1	3.402619
2	3.420253



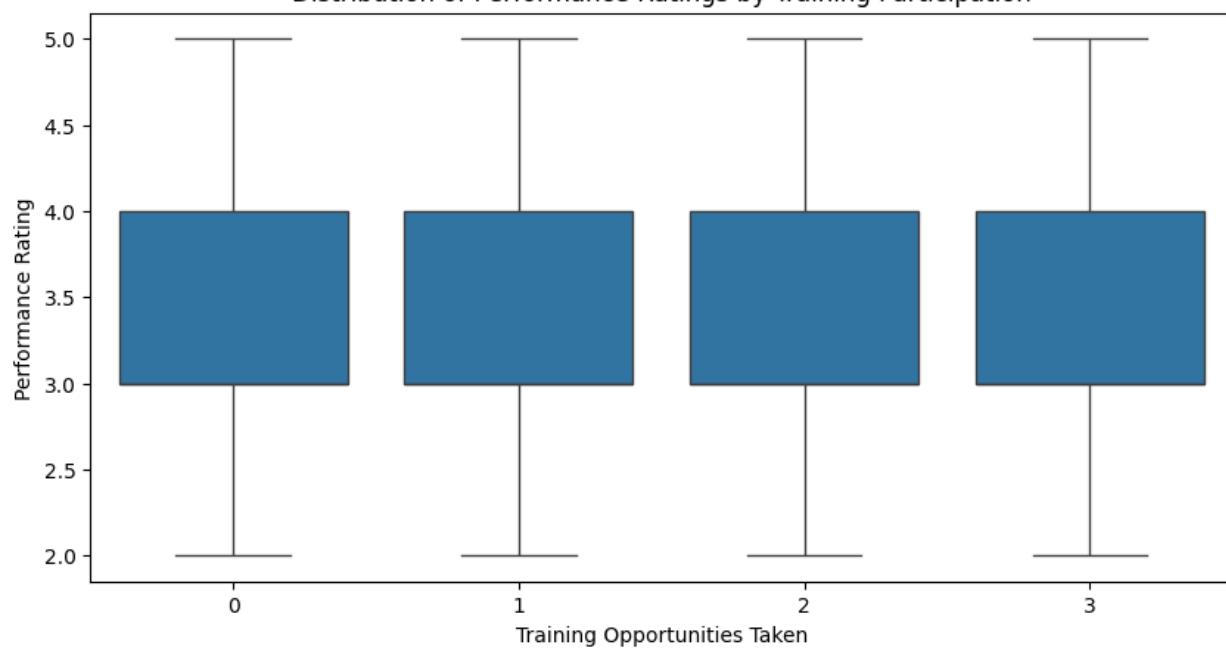
25. How does participation in training programs impact employee performance?

We can use `Education` as a proxy for training and skill level to analyze its impact on `Salary`.

```
● ● ●  
1 # Statistics  
2 corr, p_value = stats.pearsonr(  
3     df['TrainingOpportunitiesTaken'],  
4     df['ManagerRating'])  
5 )  
6 print(f'''  
7 [Training vs Performance]  
8 Correlation Coefficient: {corr:.3f}  
9 P-value: {p_value:.4f}  
10 Significant at 95% confidence: {'Yes' if p_value < 0.05 else 'No'}  
11 ''')  
12  
13 # Visualization  
14 plt.figure(figsize=(10, 5))  
15 sns.boxplot(x='TrainingOpportunitiesTaken', y='ManagerRating', data=df)  
16 plt.title("Distribution of Performance Ratings by Training Participation")  
17 plt.xlabel("Training Opportunities Taken")  
18 plt.ylabel("Performance Rating")  
19 plt.show()
```

```
[Training vs Performance]  
Correlation Coefficient: 0.006  
P-value: 0.6039  
Significant at 95% confidence: No
```

Distribution of Performance Ratings by Training Participation



26. Is there a connection between training and job satisfaction or promotion rates?

We can use `TrainingOpportunitiesTaken` to evaluate how training affects `JobSatisfaction` and `YearsSinceLastPromotion`.



```
1 # Calculate average Job Satisfaction and Years Since Last Promotion based on Training Opportunities Taken
2 training_analysis = df.groupby('TrainingOpportunitiesTaken').agg({
3     'JobSatisfaction': 'mean',
4     'YearsSinceLastPromotion': 'mean'
5 }).reset_index()
6
7 print("Average Job Satisfaction and Years Since Last Promotion based on Training Opportunities Taken:")
8 print(training_analysis)
9
10
11 # Visualization
12 sns.barplot(x='TrainingOpportunitiesTaken', y='JobSatisfaction', data=df, ci=None)
13 plt.title('Training vs Job Satisfaction')
14 plt.xlabel('Training Opportunities Taken')
15 plt.ylabel('Average Job Satisfaction')
16 plt.show()
17
18 plt.figure(figsize=(10, 6))
19 sns.barplot(x='TrainingOpportunitiesTaken', y='YearsSinceLastPromotion', data=df, ci=None)
20 plt.title('Training vs Time Since Last Promotion')
21 plt.xlabel('Training Opportunities Taken')
22 plt.ylabel('Years Since Last Promotion')
23 plt.show()
```

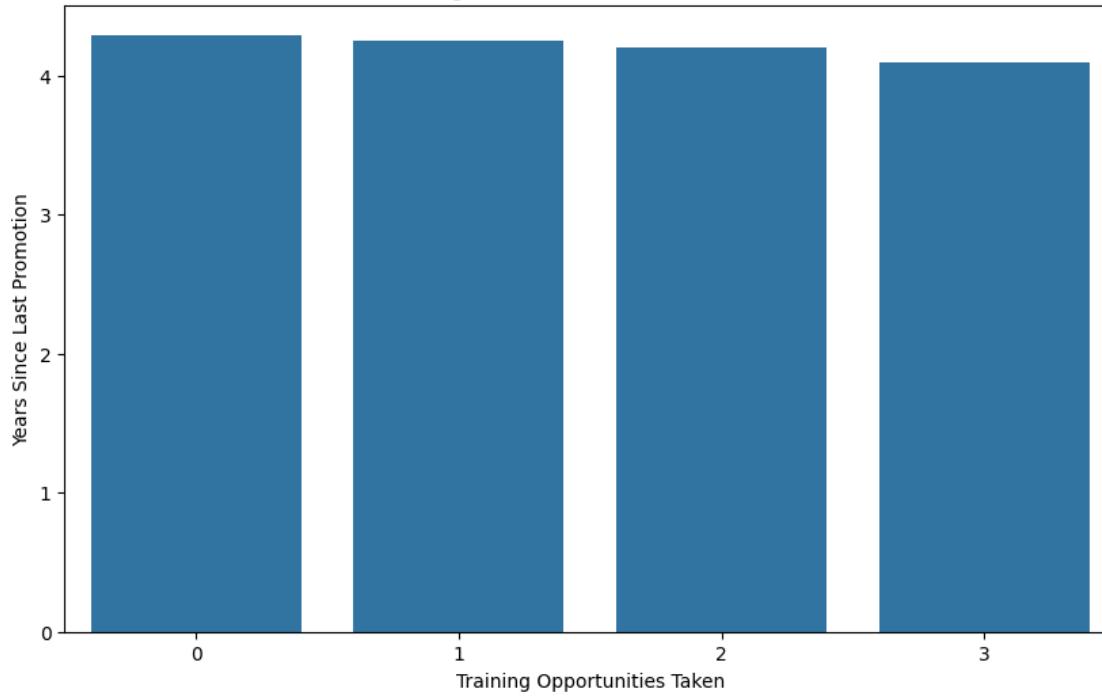
Average Job Satisfaction and Years Since Last Promotion based on Training Opportunities Taken:

TrainingOpportunitiesTaken	JobSatisfaction	YearsSinceLastPromotion
0	3.440135	4.291315
1	3.425171	4.253668
2	3.430712	4.203184
3	3.400990	4.094059

Training vs Job Satisfaction



Training vs Time Since Last Promotion



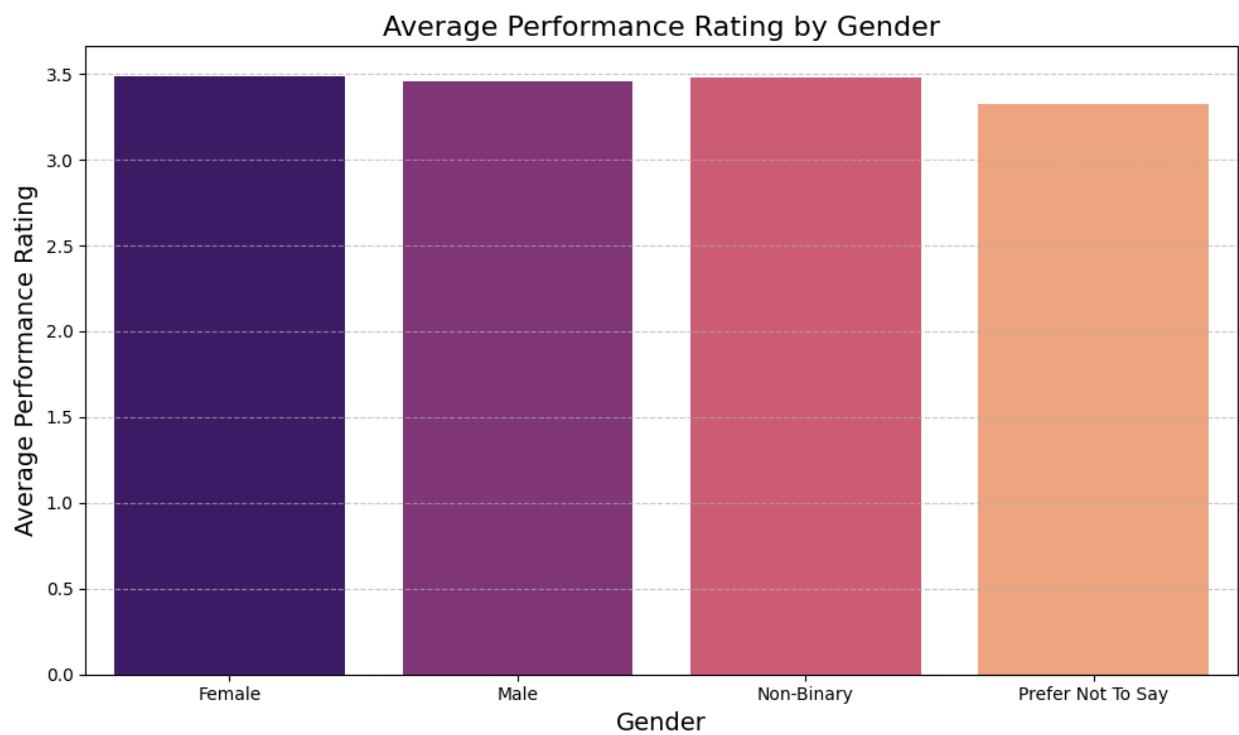
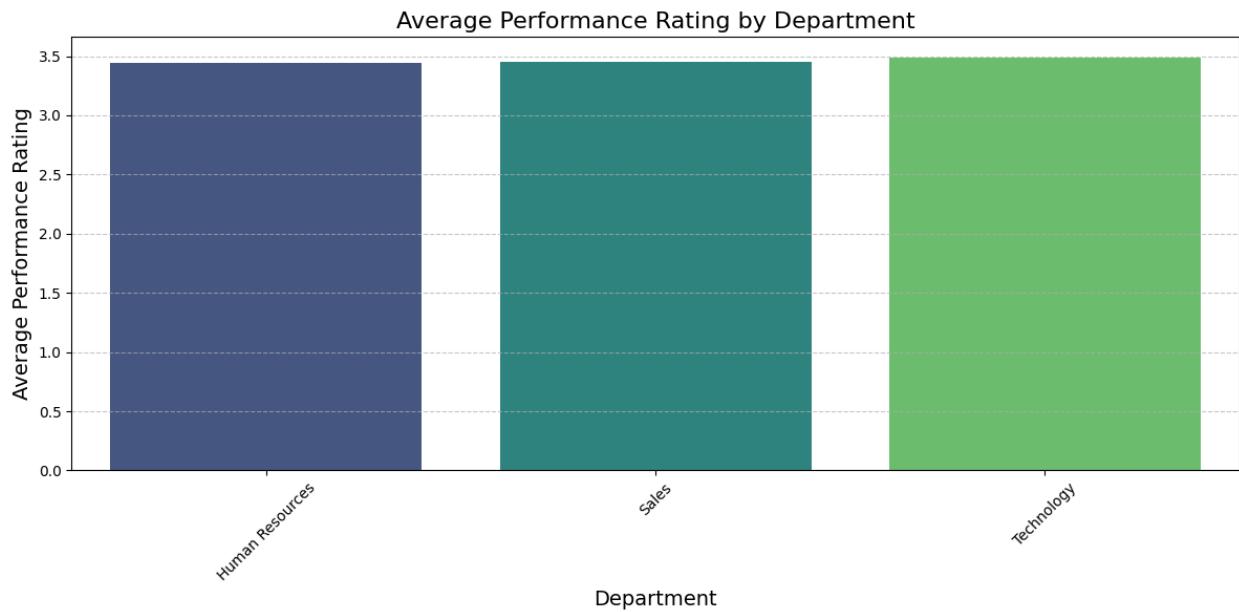
4. Behavioral Insights:

27.What are the trends in performance ratings across departments and demographics?

We will analyze how `PerformanceRating` varies across `Department` and `Gender`.

```
● ● ●
1 # Performance ratings by department
2 dept_performance = df.groupby('Department')['ManagerRating'].mean().reset_index()
3 print(f"Performance Rating by Department:\n{dept_performance}")
4
5 # Performance ratings by gender
6 gender_performance = df.groupby('Gender')['ManagerRating'].mean().reset_index()
7 print(f"Performance Rating by Gender:\n{gender_performance}")
8
9 # Visualization
10
11 # Performance Ratings by Department
12 plt.figure(figsize=(12, 6))
13 sns.barplot(x='Department', y='ManagerRating', data=dept_performance, palette='viridis')
14 plt.title('Average Performance Rating by Department', fontsize=16)
15 plt.xlabel('Department', fontsize=14)
16 plt.ylabel('Average Performance Rating', fontsize=14)
17 plt.xticks(rotation=45)
18 plt.grid(axis='y', linestyle='--', alpha=0.7) # Adding grid for better readability
19 plt.tight_layout()
20 plt.show()
21
22 # Performance Ratings by Gender
23 plt.figure(figsize=(10, 6))
24 sns.barplot(x='Gender', y='ManagerRating', data=gender_performance, palette='magma')
25 plt.title('Average Performance Rating by Gender', fontsize=16)
26 plt.xlabel('Gender', fontsize=14)
27 plt.ylabel('Average Performance Rating', fontsize=14)
28 plt.grid(axis='y', linestyle='--', alpha=0.7) # Adding grid for better readability
29 plt.tight_layout()
30 plt.show()
31
```

```
Performance Rating by Department:
   Department  ManagerRating
0  Human Resources      3.442244
1          Sales      3.449977
2    Technology      3.487432
Performance Rating by Gender:
           Gender  ManagerRating
0        Female      3.486574
1         Male      3.460916
2  Non-Binary      3.482173
3 Prefer Not To Say      3.327869
```



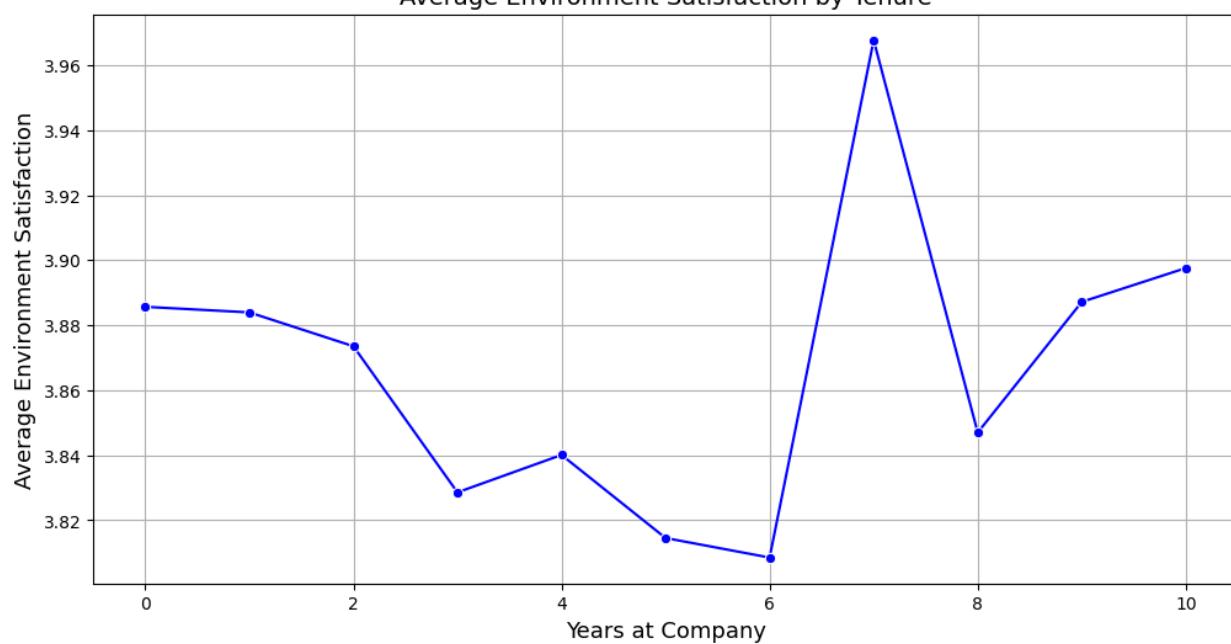
28. How does tenure (YearsAtCompany) impact performance and promotion rates?

To examine this, we can analyze how `YearsAtCompany` affects both `PerformanceRating` and `YearsSinceLastPromotion`.

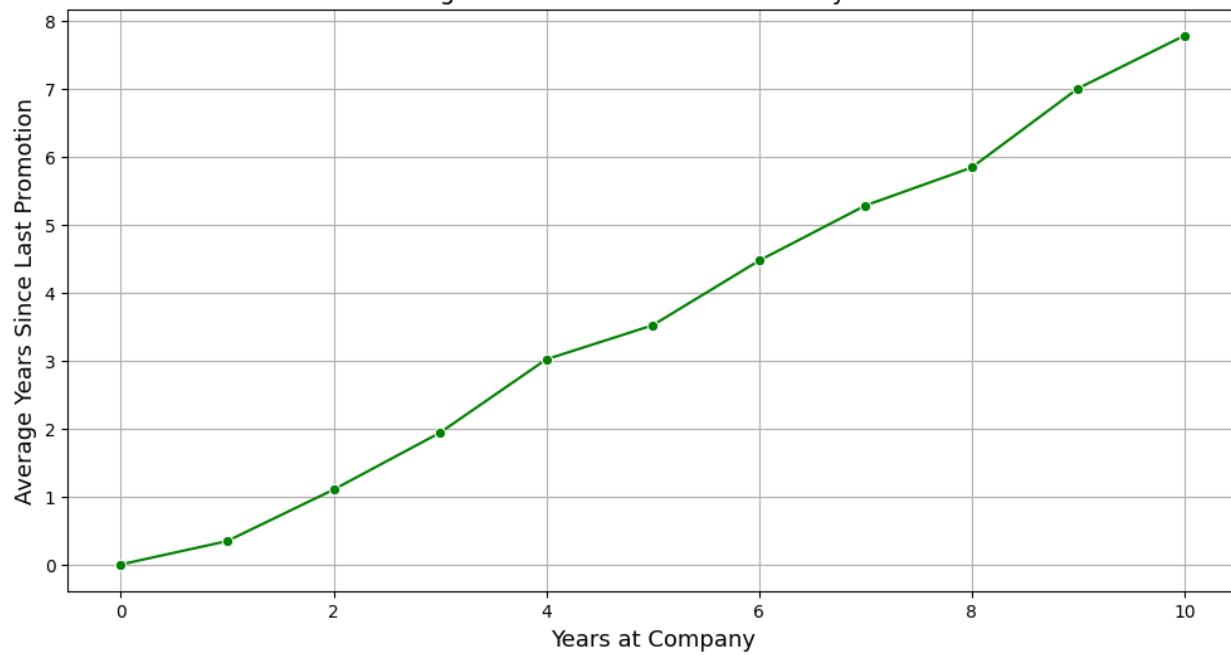
```
● ● ●  
1 # Group by 'YearsAtCompany' and calculate average performance rating and average years since last promotion  
2 tenure_performance = df.groupby('YearsAtCompany')[['EnvironmentSatisfaction']].mean() # Use 'EnvironmentSatisfaction' as proxy for performance  
3 tenure_promotion = df.groupby('YearsAtCompany')[['YearsSinceLastPromotion']].mean()  
4  
5 # Print results  
6 print(f"Average Environment Satisfaction by Tenure (YearsAtCompany):\n{tenure_performance}")  
7 print(f"Average Years Since Last Promotion by Tenure (YearsAtCompany):\n{tenure_promotion}")  
8  
9 # Visualization for performance rating vs. tenure  
10 plt.figure(figsize=(12, 6))  
11 sns.lineplot(x=tenure_performance.index, y=tenure_performance.values, marker='o', color='blue')  
12 plt.title('Average Environment Satisfaction by Tenure', fontsize=14)  
13 plt.xlabel('Years at Company', fontsize=13)  
14 plt.ylabel('Average Environment Satisfaction', fontsize=13)  
15 plt.grid(True)  
16 plt.show()  
17  
18 # Visualization for years since last promotion vs. tenure  
19 plt.figure(figsize=(12, 6))  
20 sns.lineplot(x=tenure_promotion.index, y=tenure_promotion.values, marker='o', color='green')  
21 plt.title('Average Years Since Last Promotion by Tenure', fontsize=14)  
22 plt.xlabel('Years at Company', fontsize=13)  
23 plt.ylabel('Average Years Since Last Promotion', fontsize=13)  
24 plt.grid(True)  
25 plt.show()  
26
```

```
Average Environment Satisfaction by Tenure (YearsAtCompany):  
YearsAtCompany  
0    3.885662  
1    3.883943  
2    3.873529  
3    3.828571  
4    3.840085  
5    3.814532  
6    3.808511  
7    3.967692  
8    3.846980  
9    3.887173  
10   3.897585  
Name: EnvironmentSatisfaction, dtype: float64  
Average Years Since Last Promotion by Tenure (YearsAtCompany):  
YearsAtCompany  
0    0.000000  
1    0.343402  
2    1.100000  
3    1.936264  
4    3.014925  
5    3.516252  
6    4.465957  
7    5.276923  
8    5.836242  
9    6.998812  
10   7.774879  
Name: YearsSinceLastPromotion, dtype: float64
```

Average Environment Satisfaction by Tenure



Average Years Since Last Promotion by Tenure



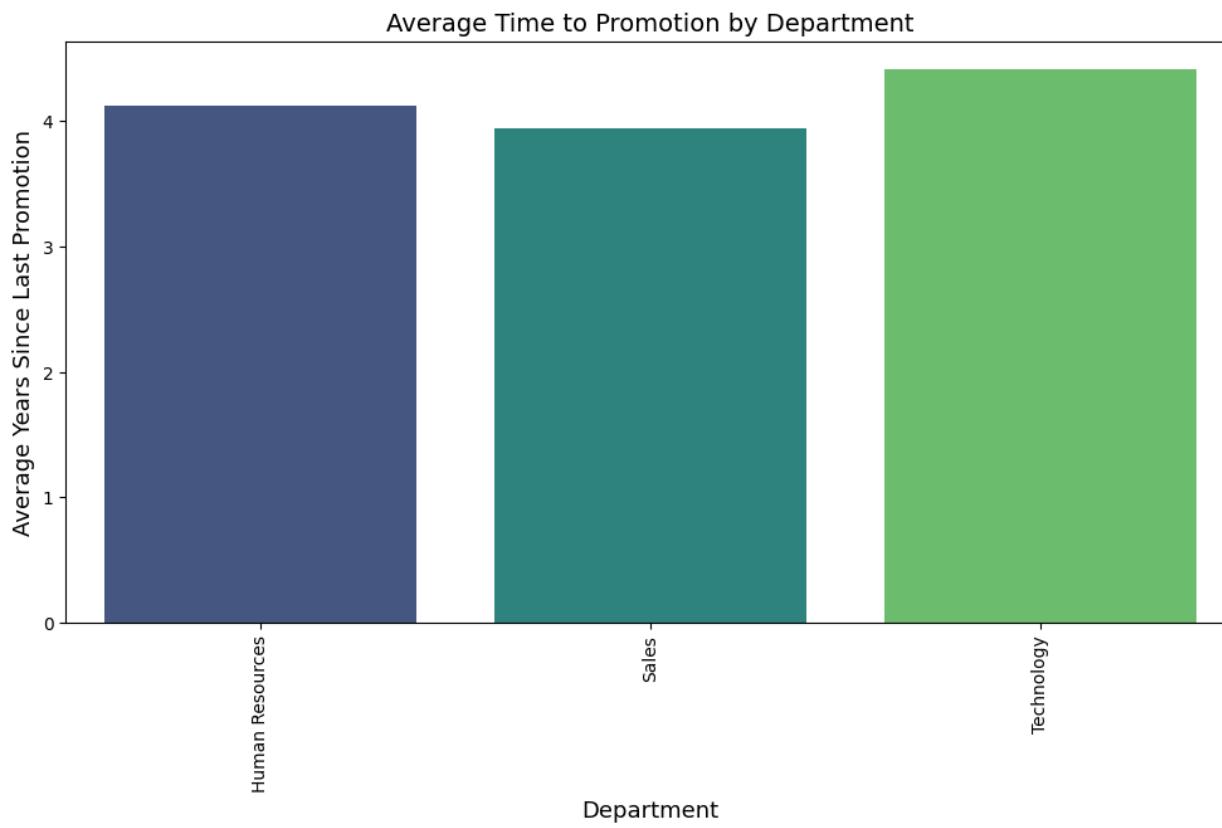
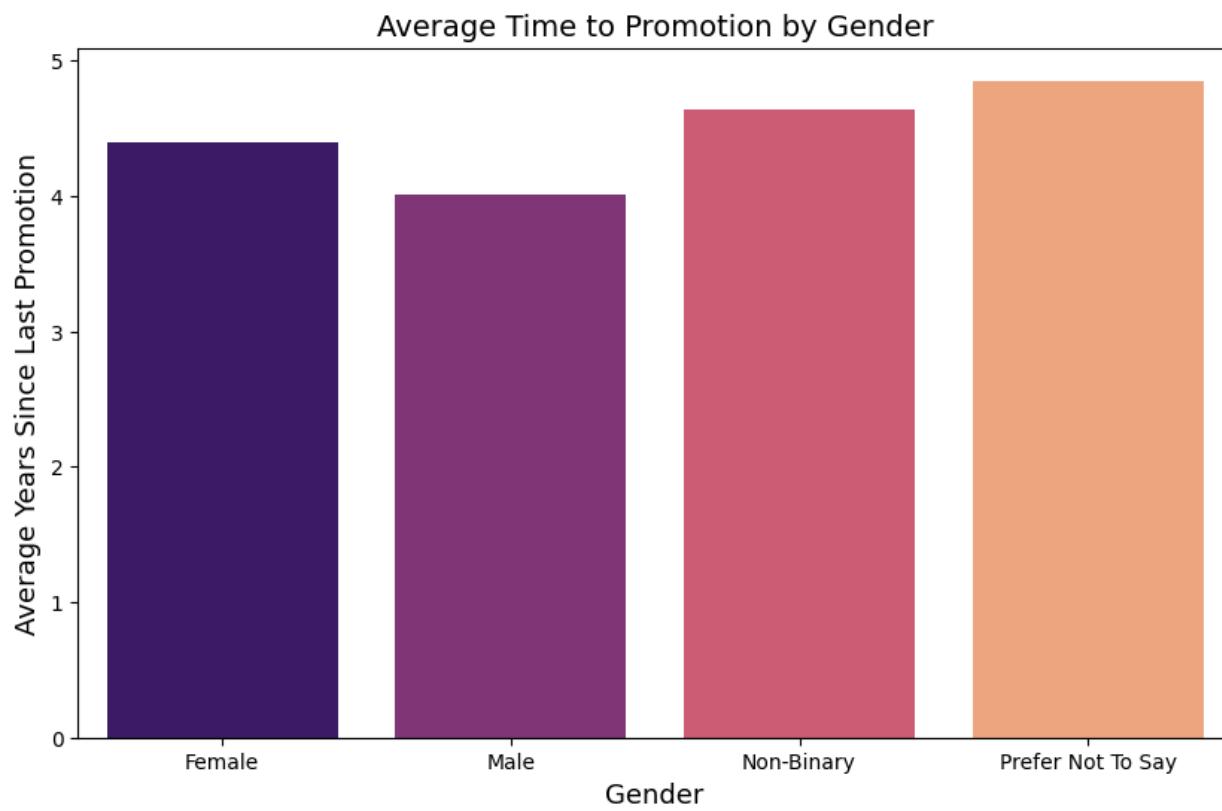
29.What is the average time to promotion for employees, and does it vary by department or gender?

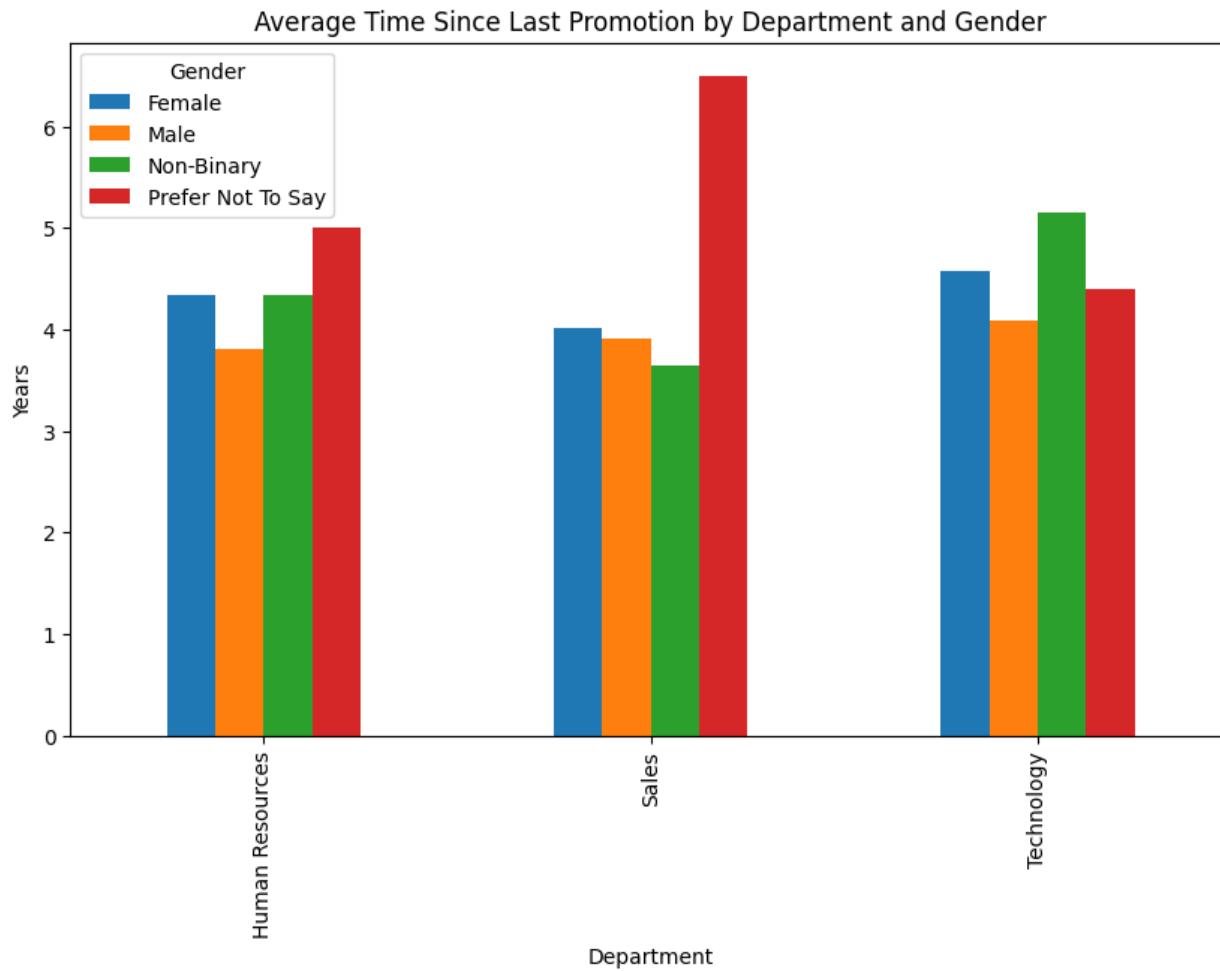
We'll calculate the average `YearsSinceLastPromotion` and group the data by `Department` and `Gender`.



```
1 # Calculate average years since last promotion
2 promotion_time = df.groupby(['Department', 'Gender'])['YearsSinceLastPromotion'].mean().unstack()
3 print("Average Time Since Last Promotion:\n", promotion_time)
4
5 # Visualization for average time to promotion by department
6 plt.figure(figsize=(12, 6))
7 sns.barplot(x=dept_promotion_time.index, y=dept_promotion_time.values, palette='viridis')
8 plt.title('Average Time to Promotion by Department', fontsize=14)
9 plt.xlabel('Department', fontsize=13)
10 plt.ylabel('Average Years Since Last Promotion', fontsize=13)
11 plt.xticks(rotation=90)
12 plt.show()
13
14 # Visualization for average time to promotion by gender
15 plt.figure(figsize=(10, 6))
16 sns.barplot(x=gender_promotion_time.index, y=gender_promotion_time.values, palette='magma')
17 plt.title('Average Time to Promotion by Gender', fontsize=14)
18 plt.xlabel('Gender', fontsize=13)
19 plt.ylabel('Average Years Since Last Promotion', fontsize=13)
20 plt.show()
21
22
23 # Visualization
24 promotion_time.plot(kind='bar', figsize=(10, 6))
25 plt.title('Average Time Since Last Promotion by Department and Gender')
26 plt.ylabel('Years')
27 plt.show()
```

Gender	Female	Male	Non-Binary	Prefer Not To Say
Department				
Human Resources	4.333333	3.800000	4.333333	5.0
Sales	4.019438	3.909715	3.640625	6.5
Technology	4.582127	4.083333	5.155844	4.4



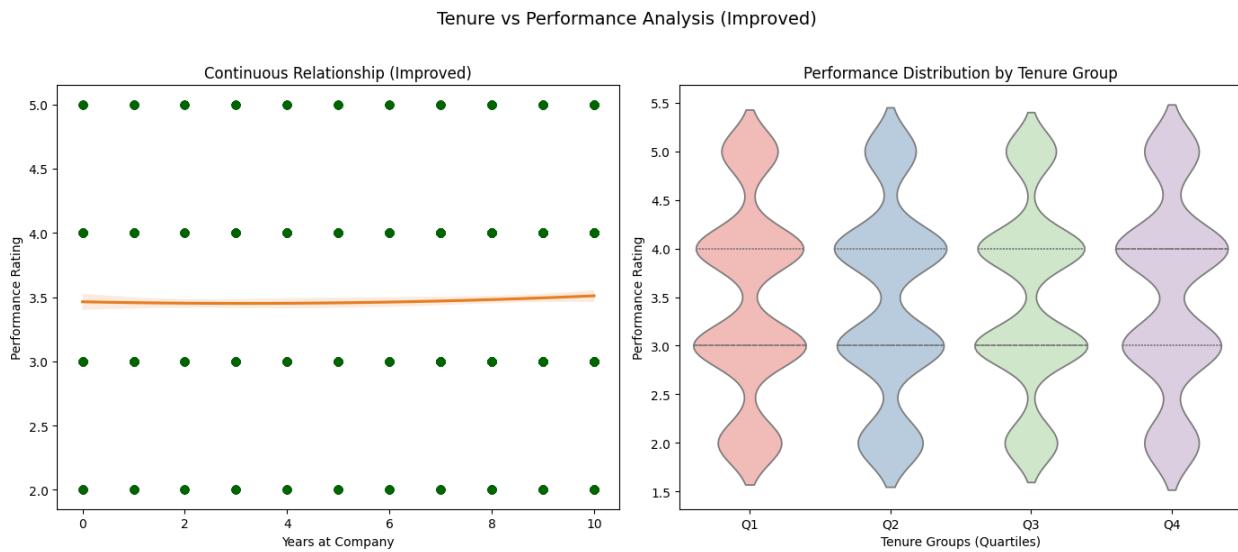


30. Are employees with longer tenure more likely to have higher performance ratings?

We analyze the correlation between `YearsAtCompany` and `ManagerRating` using regression and quartile-based grouping. Columns used: `YearsAtCompany`, `ManagerRating`, `TenureGroup`.

```
 1 # --- Statistical Analysis ---
 2 # Compute Pearson correlation
 3 corr, p_value = stats.pearsonr(df['YearsAtCompany'], df['ManagerRating'])
 4 print(f'\n[Tenure vs Performance Analysis]')
 5 print(f'Correlation Coefficient: {corr:.3f}')
 6 print(f'P-value: {p_value:.4f}')
 7 print(f'Statistically Significant at 95% Confidence: {'Yes' if p_value < 0.05 else 'No'})'
 8
 9
10
11 # --- Visualization ---
12 plt.figure(figsize=(14, 6))
13 # --- Scatter Plot with Regression (Left Plot) ---
14 plt.subplot(1, 2, 1)
15 sns.regplot(
16     x='YearsAtCompany',
17     y='ManagerRating',
18     data=df,
19     scatter_kws={'alpha':0.4, 'color':'darkgreen'}, # Make points semi-transparent
20     line_kws={'color':'#e67e22'}, # Regression line color
21     order=2 # Use polynomial regression of degree 2 for better fit
22 )
23 plt.title('Continuous Relationship (Improved)', fontsize=12)
24 plt.xlabel('Years at Company', fontsize=10)
25 plt.ylabel('Performance Rating', fontsize=10)
26
27 # --- Violin Plot (Right Plot) ---
28 plt.subplot(1, 2, 2)
29 # Create tenure quartiles
30 df['TenureGroup'] = pd.qcut(df['YearsAtCompany'], q=4, labels=['Q1', 'Q2', 'Q3', 'Q4'])
31
32 sns.violinplot(
33     x='TenureGroup',
34     y='ManagerRating',
35     data=df,
36     palette='Pastel1',
37     inner='quartile' # Show quartile lines inside the violin
38 )
39 plt.title('Performance Distribution by Tenure Group', fontsize=12)
40 plt.xlabel('Tenure Groups (Quartiles)', fontsize=10)
41 plt.ylabel('Performance Rating', fontsize=10)
42
43 # Set overall title
44 plt.suptitle('Tenure vs Performance Analysis (Improved)', fontsize=14, y=1.02)
45 plt.tight_layout()
46 plt.show()
```

[Tenure vs Performance Analysis]
Correlation Coefficient: 0.018
P-value: 0.1434
Statistically Significant at 95% Confidence: No

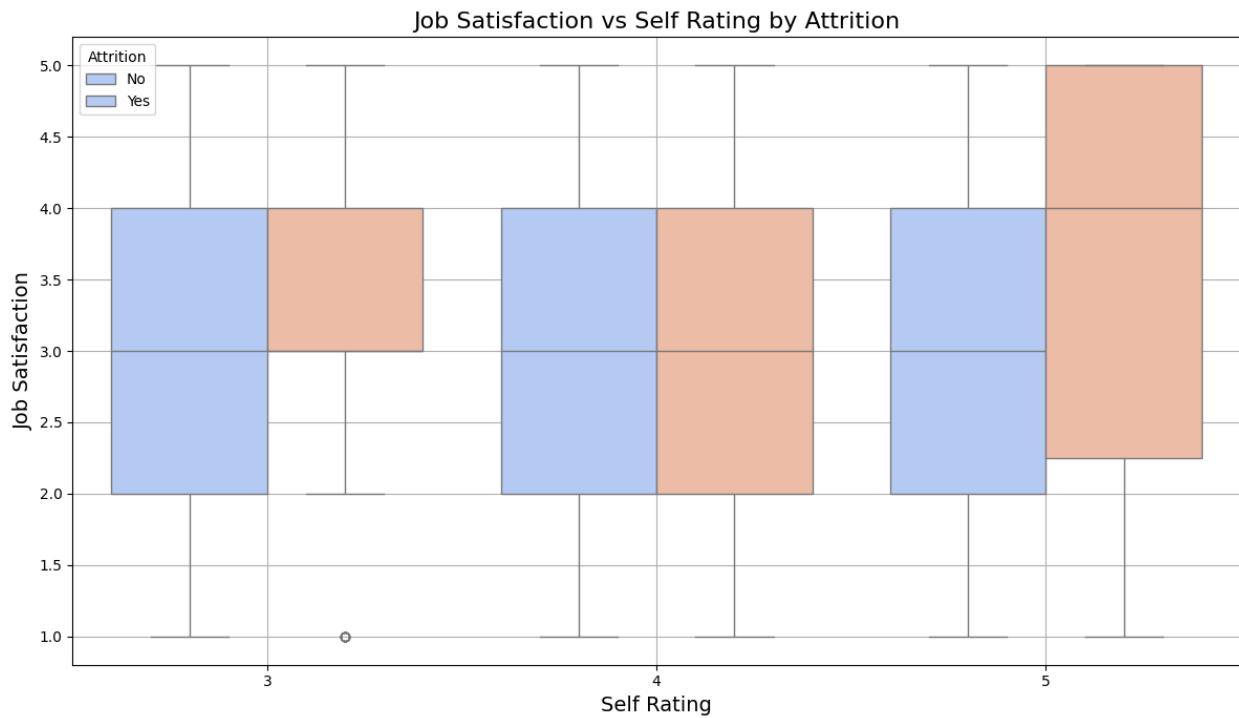


31. How does performance rating correlate with job satisfaction?

We can calculate the average `jobSatisfaction` for each `SelfRating` and create a boxplot to visualize the distribution of job satisfaction based on self-rating, highlighting differences based on attrition status.

```
1
2 # Calculate the average job satisfaction based on self-rating
3 avg_job_satisfaction = df.groupby('SelfRating')['JobSatisfaction'].mean()
4 print("Average Job Satisfaction by Self Rating:")
5 print(avg_job_satisfaction)
6
7 # Create a boxplot for job satisfaction based on self-rating
8 plt.figure(figsize=(12, 7))
9 ax = sns.boxplot(
10     x='SelfRating',
11     y='JobSatisfaction',
12     hue='Attrition', # Add attrition information to the plot
13     data=df,
14     palette='coolwarm'
15 )
16
17 # Customize the appearance of the boxplot using matplotlib
18 for patch in ax.artists:
19     patch.set_facecolor('lightblue') # Change the box color
20     patch.set_edgecolor('black') # Box edge color
21     patch.set_linewidth(1.5) # Edge line width
22
23 plt.title('Job Satisfaction vs Self Rating by Attrition', fontsize=16)
24 plt.xlabel('Self Rating', fontsize=14)
25 plt.ylabel('Job Satisfaction', fontsize=14)
26 plt.grid(True) # Add gridlines for better visualization
27
28 # Add a legend to improve understanding
29 plt.legend(title='Attrition', loc='upper left', labels=['No', 'Yes'])
30 plt.tight_layout() # Improve layout
31 plt.show()
```

```
Average Job Satisfaction by Self Rating:
SelfRating
3    3.457642
4    3.399821
5    3.433807
```



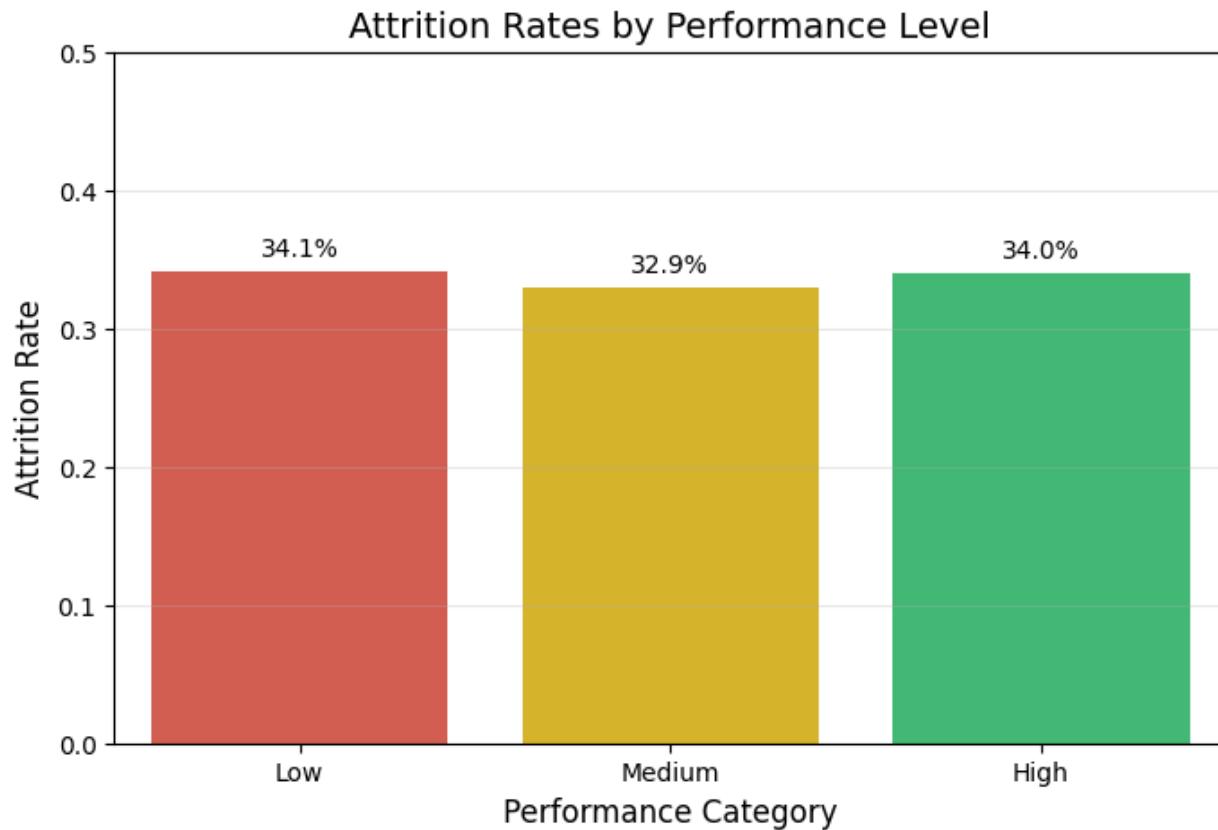
32.What is the attrition rate for employees with different performance ratings?

We can group the data by `PerformanceRating` and calculate the attrition rate for each rating level.

```
● ● ●

1 # Statistics
2 low = df[df['Performance'] == 'Low']['Attrition']
3 high = df[df['Performance'] == 'High']['Attrition']
4 t_stat, p_value = stats.ttest_ind(low, high)
5 print(f'''
6 [Attrition by Performance]
7 T-statistic: {t_stat:.2f}
8 P-value: {p_value:.4f}
9 Significant difference: {'Yes' if p_value < 0.05 else 'No'}
10 ''')
11
12
13 # Create performance categories
14 df['Performance'] = pd.qcut(
15     df['ManagerRating'],
16     q=3,
17     labels=['Low', 'Medium', 'High']
18 )
19
20 # Visualization
21 plt.figure(figsize=(8, 5))
22 ax = sns.barplot(
23     x='Performance',
24     y='Attrition',
25     data=df,
26     order=['Low', 'Medium', 'High'],
27     palette=['#e74c3c', '#f1c40f', '#2ecc71'],
28     errorbar=None
29 )
30
31 # Add percentages
32 for p in ax.patches:
33     ax.annotate(
34         f'{p.get_height():.1%}',
35         (p.get_x() + p.get_width()/2., p.get_height()),
36         ha='center',
37         va='center',
38         xytext=(0, 9),
39         textcoords='offset points'
40     )
41
42 plt.title('Attrition Rates by Performance Level', fontsize=14)
43 plt.xlabel('Performance Category', fontsize=12)
44 plt.ylabel('Attrition Rate', fontsize=12)
45 plt.ylim(0, 0.5)
46 plt.grid(axis='y', alpha=0.3)
47 plt.show()
48
```

[Attrition by Performance]
T-statistic: 0.08
P-value: 0.9380
Significant difference: No



33.What is the impact of overtime on job satisfaction?

Since we do not have a direct measure of job satisfaction, we can analyze the impact of `OverTime` on `PerformanceRating` and use that as a proxy for satisfaction.

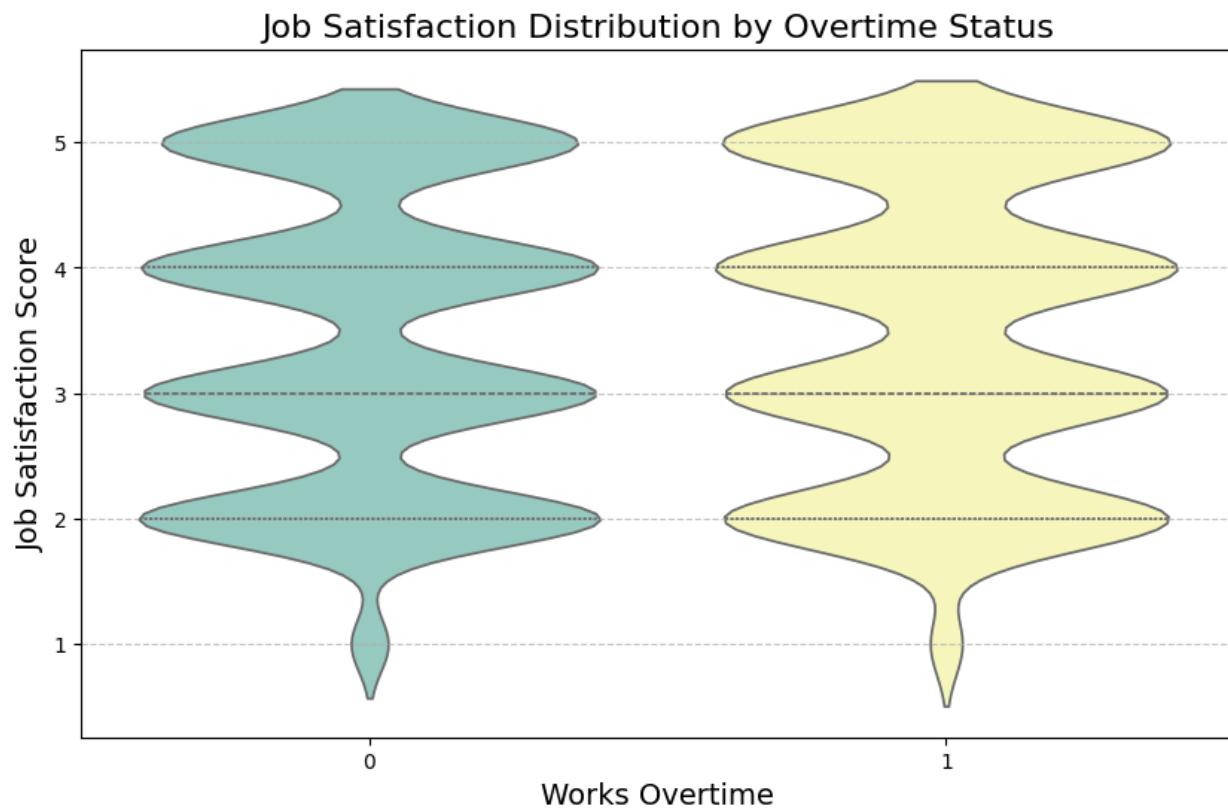
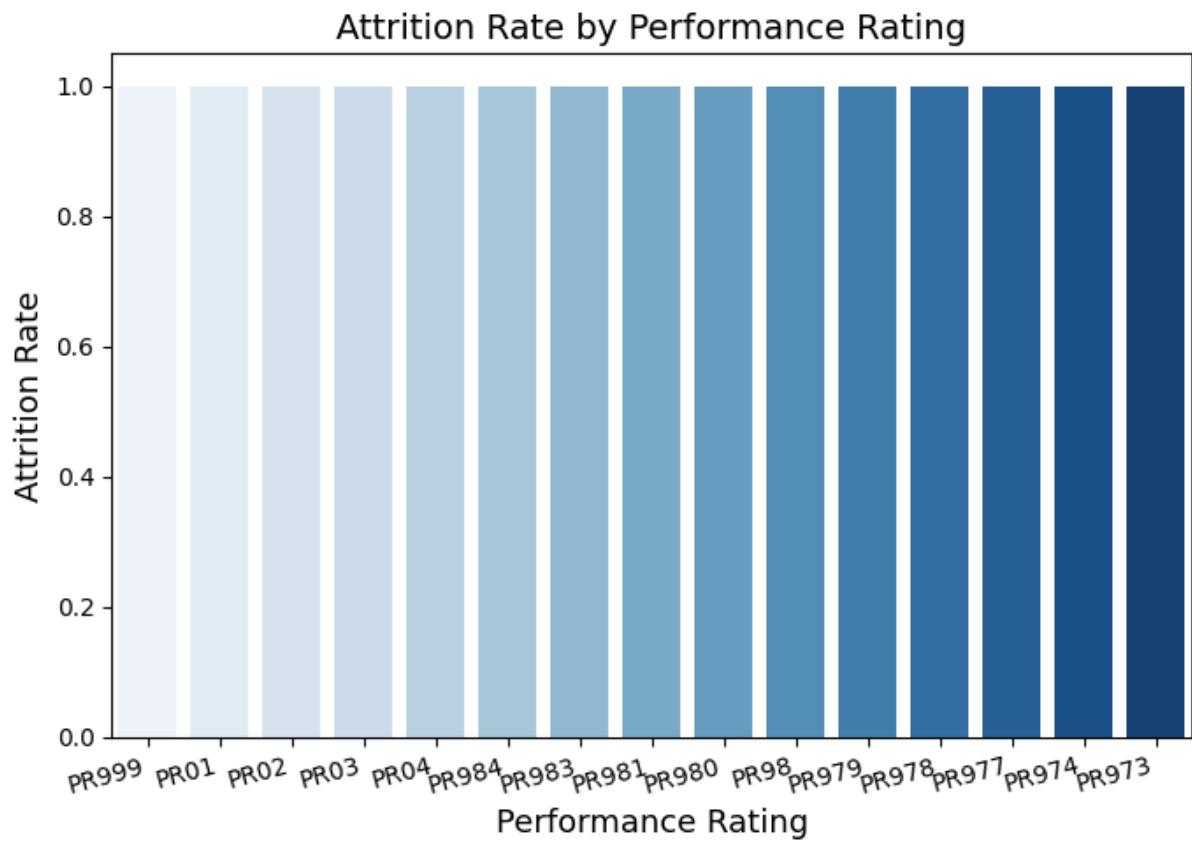


```
1 # Group by PerformanceID and calculate attrition rate
2 performance_attrition_rate = df.groupby('PerformanceID')['Attrition'].mean().sort_values(ascending=False).head(15)
3
4
5 # Display all performance ratings
6 print(f"Attrition Rate by Performance Rating:\n{performance_attrition_rate}")
7
8 # Visualization for attrition rate by performance rating
9 plt.figure(figsize=(7, 5)) # Adjust figure size
10 sns.barplot(x=performance_attrition_rate.index, y=performance_attrition_rate.values, palette='Blues')
11
12 plt.title('Attrition Rate by Performance Rating', fontsize=14)
13 plt.xlabel('Performance Rating', fontsize=13)
14 plt.ylabel('Attrition Rate', fontsize=13)
15
16 # Rotate x-axis labels slightly for better readability
17 plt.xticks(rotation=15, ha='right')
18
19 # Adjust layout to ensure labels fit
20 plt.tight_layout()
21
22 plt.show()
23
24 # Violin plot for Job Satisfaction by Overtime Status
25 plt.figure(figsize=(10, 6))
26 sns.violinplot(
27     x='OverTime',
28     y='JobSatisfaction',
29     data=df,
30     palette='Set3',
31     inner='quartile' # Display quartiles within the violin plot
32 )
33 plt.title('Job Satisfaction Distribution by Overtime Status', fontsize=16)
34 plt.xlabel('Works Overtime', fontsize=14)
35 plt.ylabel('Job Satisfaction Score', fontsize=14)
36 plt.xticks(rotation=0) # Rotate x-axis labels for better readability
37 plt.grid(axis='y', linestyle='--', alpha=0.7) # Add grid for better readability
38 plt.show()
39
40 # Statistical test
41 overtime_yes = df[df['OverTime'] == 'Yes']['JobSatisfaction']
42 overtime_no = df[df['OverTime'] == 'No']['JobSatisfaction']
43 t_stat, p_value = stats.ttest_ind(overtime_yes, overtime_no, nan_policy='omit')
44 print(f"Overtime Satisfaction Difference: t={t_stat:.2f}, p={p_value:.4f}")
45
```

Attrition Rate by Performance Rating:

PerformanceID

PR999	1.0
PR01	1.0
PR02	1.0
PR03	1.0
PR04	1.0
PR984	1.0
PR983	1.0
PR981	1.0
PR980	1.0
PR98	1.0
PR979	1.0
PR978	1.0
PR977	1.0
PR974	1.0
PR973	1.0



34. Is there a relationship between overtime and performance?

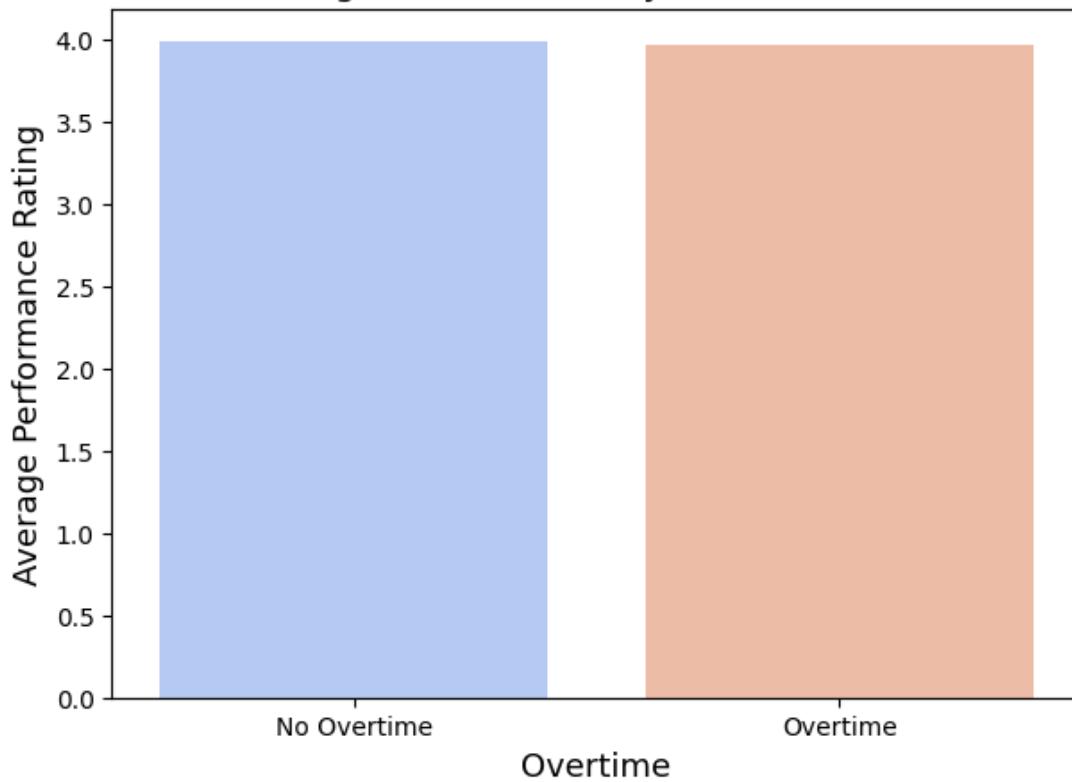
We analyze whether working `overtime` impacts `SelfRating` by comparing average ratings and conducting a statistical test.



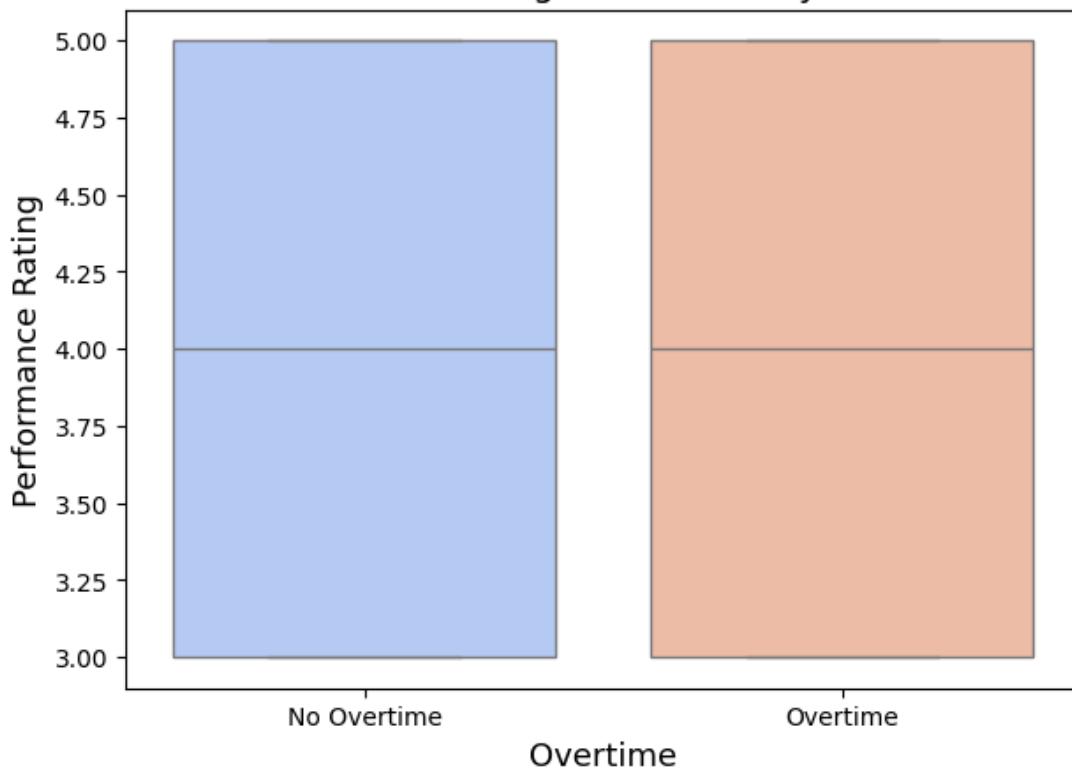
```
1 # Group by 'OverTime' and calculate mean performance
2 overtime_performance = df.groupby('OverTime')['SelfRating'].mean()
3 print(f"Average Performance by Overtime Status:\n{overtime_performance}")
4
5 # Statistical test: Is there a significant difference in performance?
6 overtime_yes = df[df['OverTime'] == 1]['SelfRating']
7 overtime_no = df[df['OverTime'] == 0]['SelfRating']
8 t_stat, p_value = ttest_ind(overtime_yes, overtime_no, equal_var=False)
9 print(f"T-test Results: t-statistic = {t_stat:.4f}, p-value = {p_value:.4f}")
10
11 # Visualization: Bar Plot
12 plt.figure(figsize=(7, 5))
13 sns.barplot(x=overtime_performance.index, y=overtime_performance.values, palette='coolwarm')
14 plt.xticks([0, 1], ['No Overtime', 'Overtime'])
15 plt.title('Average Performance by Overtime Status', fontsize=14)
16 plt.xlabel('Overtime', fontsize=13)
17 plt.ylabel('Average Performance Rating', fontsize=13)
18 plt.show()
19
20 # Visualization: Box Plot
21 plt.figure(figsize=(7, 5))
22 sns.boxplot(x=df['OverTime'], y=df['SelfRating'], palette='coolwarm')
23 plt.xticks([0, 1], ['No Overtime', 'Overtime'])
24 plt.title('Performance Rating Distribution by Overtime', fontsize=14)
25 plt.xlabel('Overtime', fontsize=13)
26 plt.ylabel('Performance Rating', fontsize=13)
27 plt.show()
```

```
Average Performance by Overtime Status:
OverTime
0    3.989937
1    3.972284
Name: SelfRating, dtype: float64
T-test Results: t-statistic = -0.8383, p-value = 0.4019
```

Average Performance by Overtime Status



Performance Rating Distribution by Overtime



35. Can we predict the likelihood of attrition based on employee attributes?

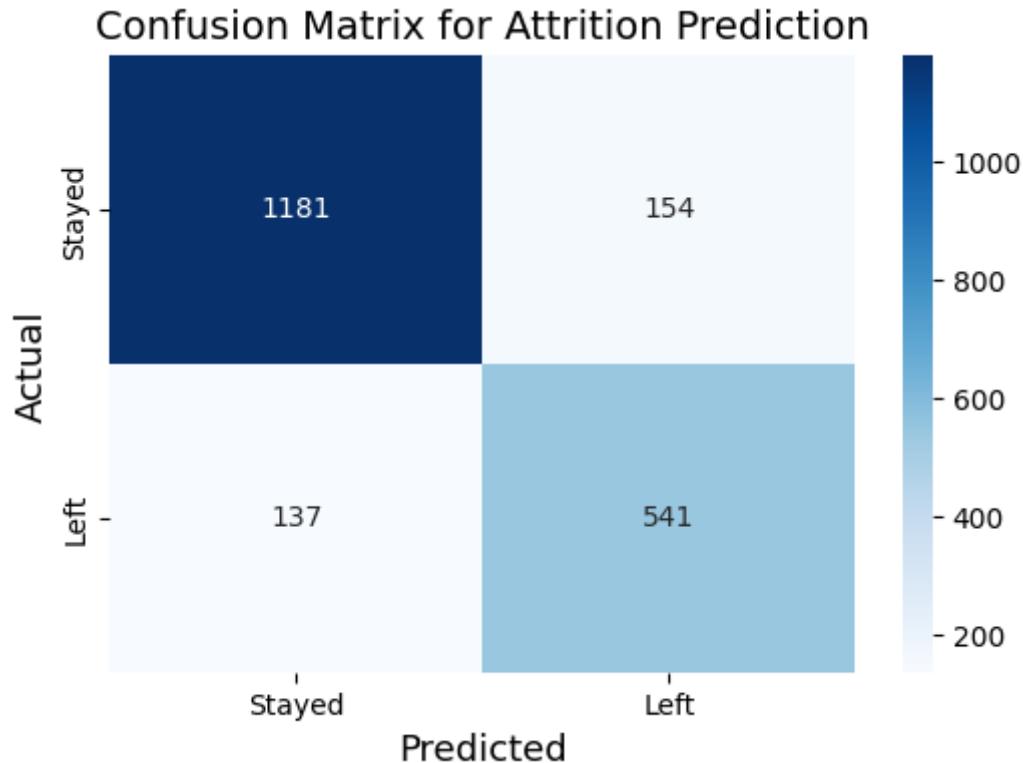
We can create a predictive model to determine the likelihood of attrition (`Attrition`) using attributes like `YearsAtCompany`, `OverTime`, `PerformanceRating`, etc.

We'll use Logistic Regression for this predictive task.

```
● ● ●
1 # Select features and target
2 X = df[['YearsAtCompany', 'OverTime', 'SelfRating']]
3 y = df['Attrition']
4
5 # Handle missing values (fill with median for numerical columns)
6 X.fillna(X.median(), inplace=True)
7
8 # Split data into training and testing sets
9 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
10
11 # Standardize the features
12 scaler = StandardScaler()
13 X_train = scaler.fit_transform(X_train)
14 X_test = scaler.transform(X_test)
15
16 # Train Logistic Regression model
17 model = LogisticRegression()
18 model.fit(X_train, y_train)
19
20 # Make predictions
21 y_pred = model.predict(X_test)
22
23 # Evaluate the model
24 print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
25 print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
26
27 # Confusion Matrix
28 plt.figure(figsize=(6, 4))
29 conf_matrix = confusion_matrix(y_test, y_pred)
30 sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Stayed', 'Left'], yticklabels=['Stayed', 'Left'])
31 plt.title('Confusion Matrix for Attrition Prediction', fontsize=14)
32 plt.xlabel('Predicted', fontsize=13)
33 plt.ylabel('Actual', fontsize=13)
34 plt.show()
35
```

```
Accuracy: 0.8554
Classification Report:
              precision    recall  f1-score   support
             0       0.90      0.88      0.89     1335
             1       0.78      0.80      0.79      678

      accuracy                           0.86     2013
     macro avg       0.84      0.84      0.84     2013
  weighted avg       0.86      0.86      0.86     2013
```



36. Are there any leading indicators of high-performing employees?

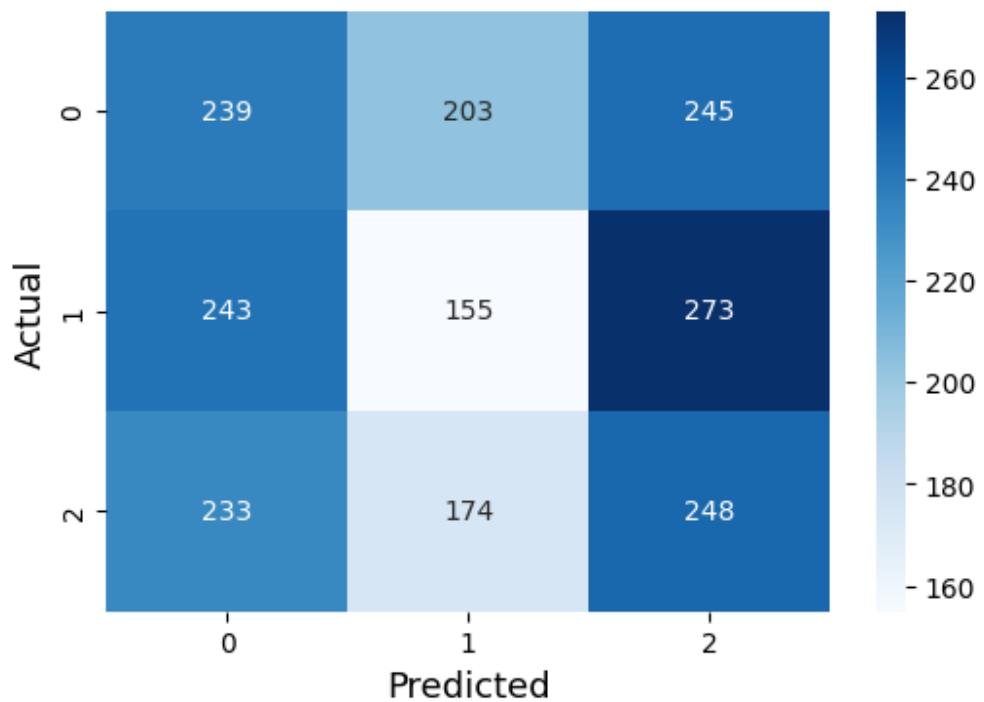
We can try to identify features that predict high performance
(`PerformanceRating`).



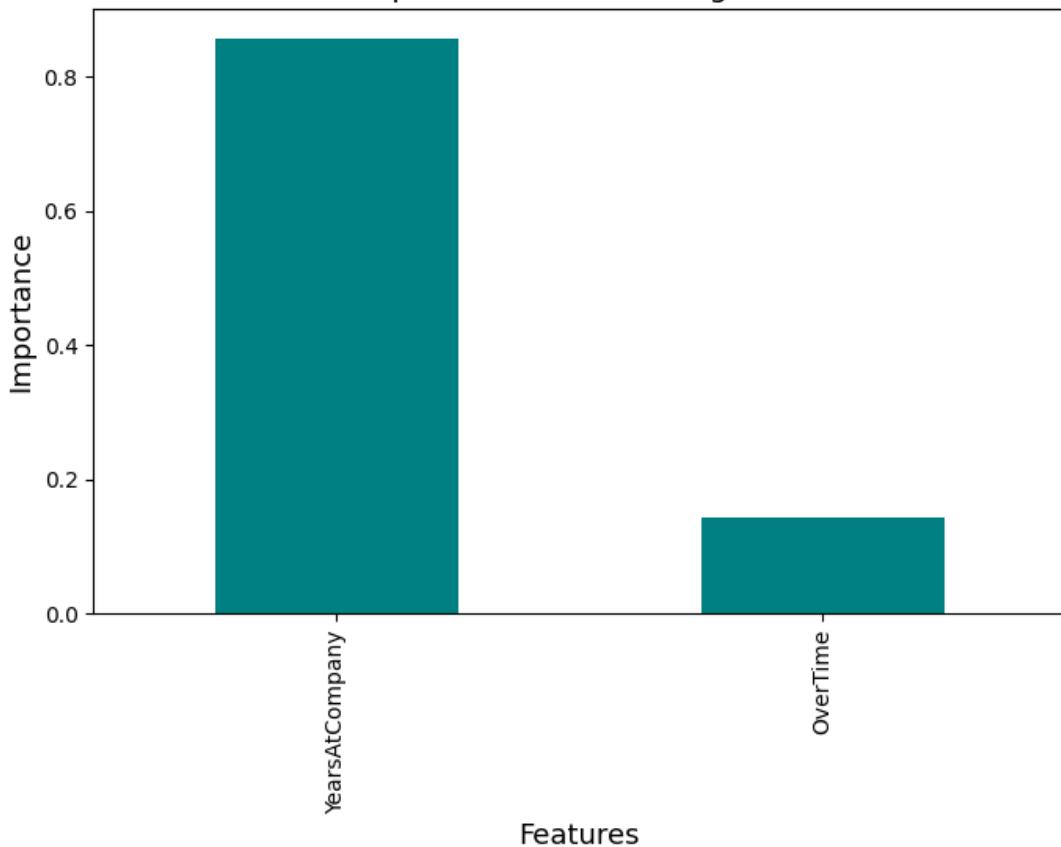
```
1 # Select features and target (predicting performance)
2 X = df[['YearsAtCompany', 'OverTime']]
3 y = df['SelfRating']
4
5 # Handle missing values
6 X.fillna(X.median(), inplace=True)
7 y.fillna(y.median(), inplace=True)
8
9 # Split data into training and testing sets
10 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
11
12 # Standardize the features
13 scaler = StandardScaler()
14 X_train = scaler.fit_transform(X_train)
15 X_test = scaler.transform(X_test)
16
17 # Train Random Forest model
18 model = RandomForestClassifier(n_estimators=100, random_state=42)
19 model.fit(X_train, y_train)
20
21 # Make predictions
22 y_pred = model.predict(X_test)
23
24 # Evaluate the model
25 print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
26 print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
27
28 # Confusion Matrix
29 plt.figure(figsize=(6, 4))
30 conf_matrix = confusion_matrix(y_test, y_pred)
31 sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
32 plt.title('Confusion Matrix for Performance Prediction', fontsize=14)
33 plt.xlabel('Predicted', fontsize=13)
34 plt.ylabel('Actual', fontsize=13)
35 plt.show()
36
37 # Feature Importance
38 feature_importance = pd.Series(model.feature_importances_, index=['YearsAtCompany', 'OverTime'])
39 feature_importance.sort_values(ascending=False).plot(kind='bar', color='teal', figsize=(8, 5))
40 plt.title('Feature Importance in Predicting Performance', fontsize=14)
41 plt.xlabel('Features', fontsize=13)
42 plt.ylabel('Importance', fontsize=13)
43 plt.show()
44
```

```
X.fillna(X.median(), inplace=True)
Accuracy: 0.3189
Classification Report:
precision    recall   f1-score   support
3          0.33      0.35      0.34      687
4          0.29      0.23      0.26      671
5          0.32      0.38      0.35      655
accuracy                           0.32      2013
macro avg       0.32      0.32      0.32      2013
weighted avg    0.32      0.32      0.32      2013
```

Confusion Matrix for Performance Prediction



Feature Importance in Predicting Performance



THANKES