Data Analysis

Data Dynamos (BQ Analysis)

For HR Data Analysis

```python
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)
```

[13] Python

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

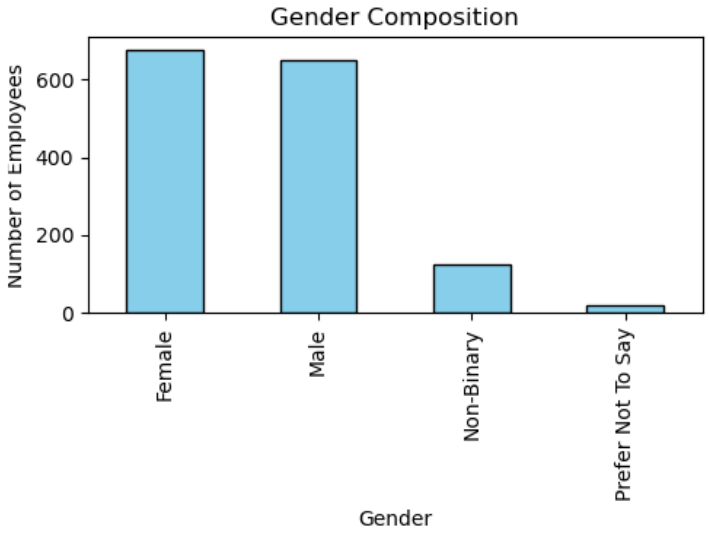
```python
import pandas as pd

# Correct file path and sheet name
file_path = r'F:\data analysis\BQ\HR Data.xlsx'
sheet_name = 'Employee'

# Load the data
employee_data = pd.read_excel(file_path, sheet_name=sheet_name)

# Display the first few rows
print(employee_data.head())
```

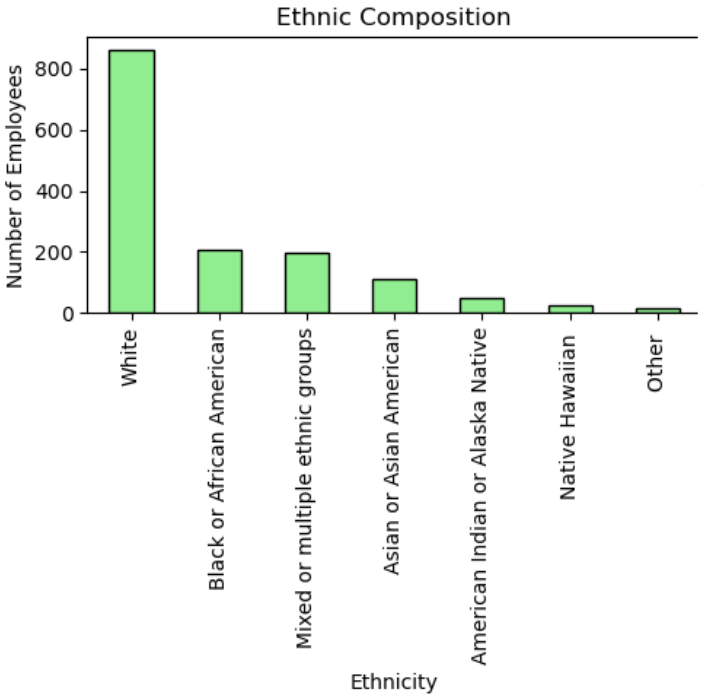
```
  EmployeeID   FirstName LastName      Gender  Age BusinessTravel  \
0  3012-1A41    Leonelle    Simco      Female   30    Some Travel
1  CBCB-9C9D     Leonerd    Aland        Male   38    Some Travel
2  95D7-1CE9       Ahmed    Sykes        Male   43    Some Travel
3  47A0-559B  Ermentrude   Berrie  Non-Binary   39    Some Travel
4  42CC-040A       Stace   Savege      Female   29    Some Travel

         Department  DistanceFromHome State                Ethnicity  ...  \
0             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  \
0       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  \
0             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?

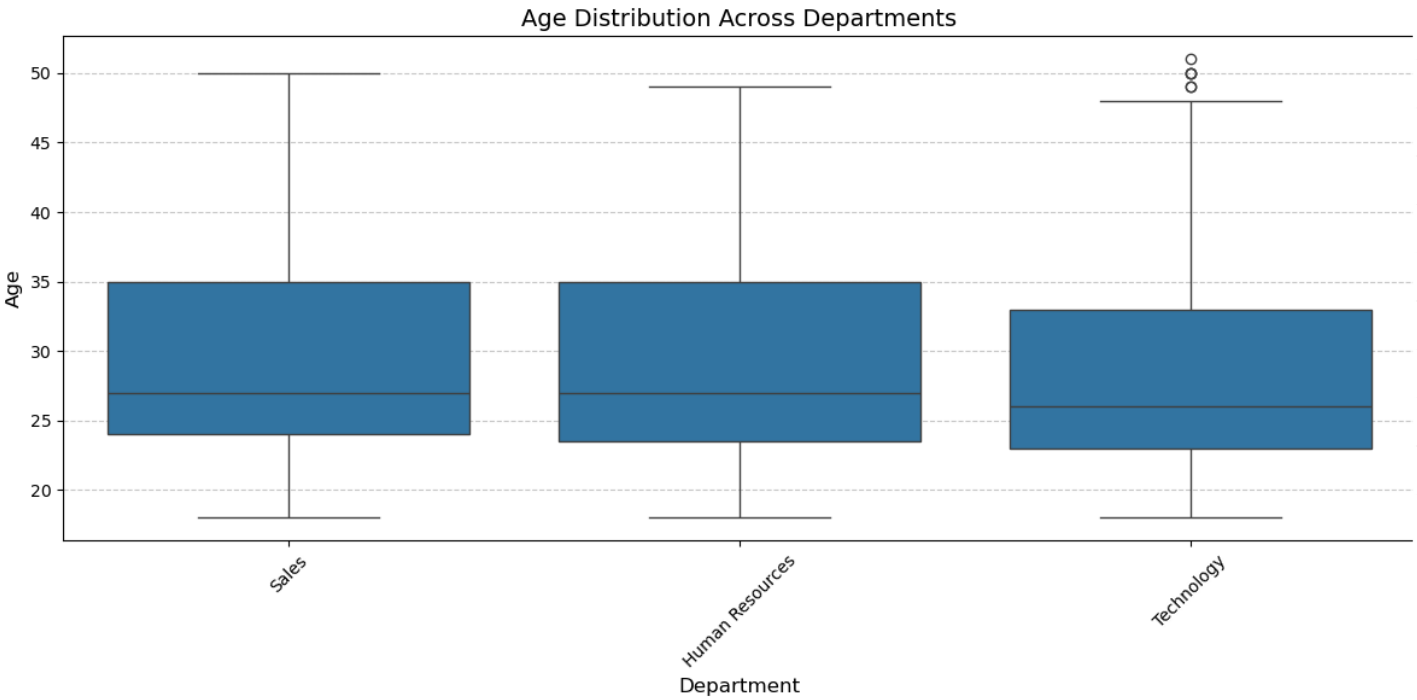
```python
import pandas as pd
import matplotlib.pyplot as plt

# Correct file path
file_path = r'F:\data analysis\BQ\HR Data.xlsx'
sheet_name = 'Employee'

# Load the data
employee_data = pd.read_excel(file_path, sheet_name=sheet_name)

# Analyze gender and ethnic composition
def workforce_demographics(data):

    gender_counts = data['Gender'].value_counts()
    ethnicity_counts = data['Ethnicity'].value_counts()

    # Plot gender composition
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    gender_counts.plot(kind='bar', color='skyblue', edgecolor='black')
    plt.title('Gender Composition')
    plt.ylabel('Number of Employees')
    plt.xlabel('Gender')

    # Plot ethnicity composition
    plt.subplot(1, 2, 2)
    ethnicity_counts.plot(kind='bar', color='lightgreen', edgecolor='black')
    plt.title('Ethnic Composition')
    plt.ylabel('Number of Employees')
    plt.xlabel('Ethnicity')

    plt.tight_layout()
    plt.savefig('workforce_demographics.png')  # Save the plot
    plt.show()

# Call the function
workforce_demographics(employee_data)

```





2.How does age distribution vary across departments?

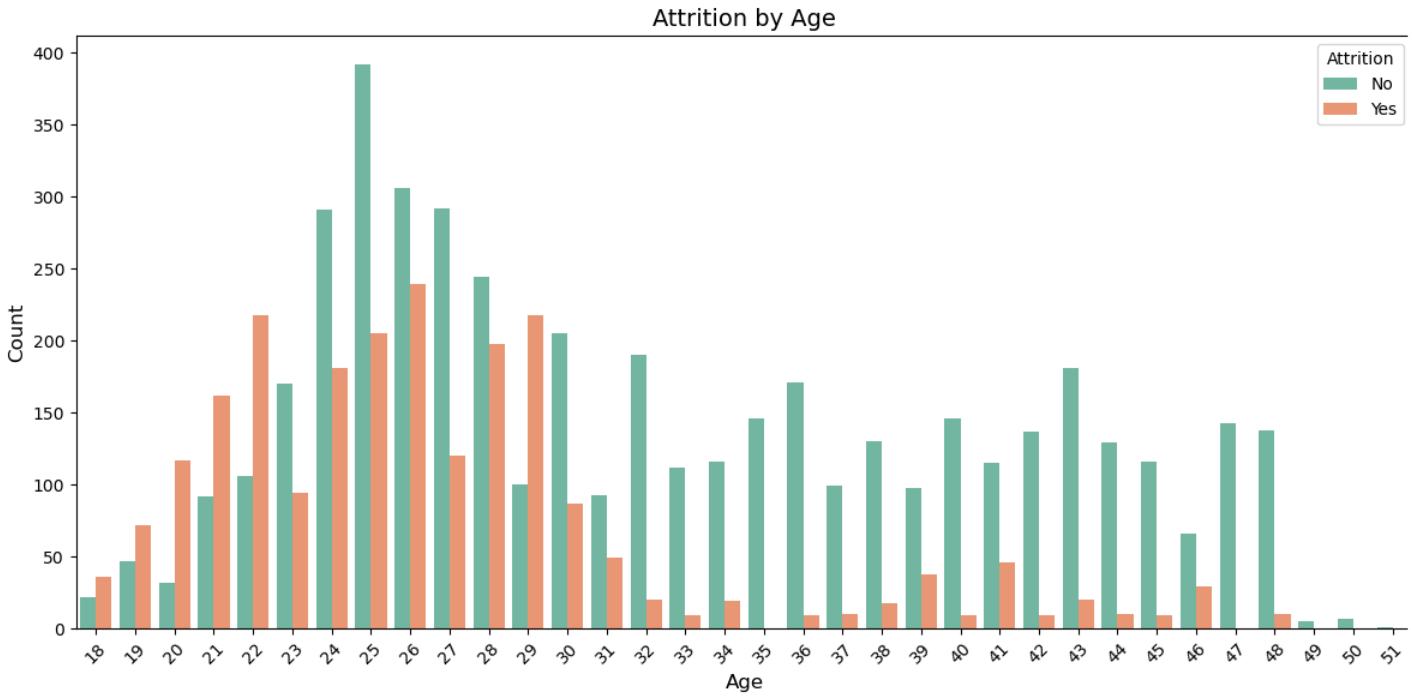
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set the file path and sheet name
file_path = r'F:\data analysis\BQ\HR Data.xlsx'
sheet_name = 'Employee'

# Load the data
employee_data = pd.read_excel(file_path, sheet_name=sheet_name)

# Ensure the required columns exist
if 'Age' in employee_data.columns and 'Department' in employee_data.columns:
    # Plot the age distribution across departments
    plt.figure(figsize=(12, 6))
    sns.boxplot(data=employee_data, x='Department', y='Age')
    plt.title('Age Distribution Across Departments', fontsize=14)
    plt.xlabel('Department', fontsize=12)
    plt.ylabel('Age', fontsize=12)
    plt.xticks(rotation=45)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()
else:
    print("Make sure the columns 'Age' and 'Department' exist in the dataset.")

```

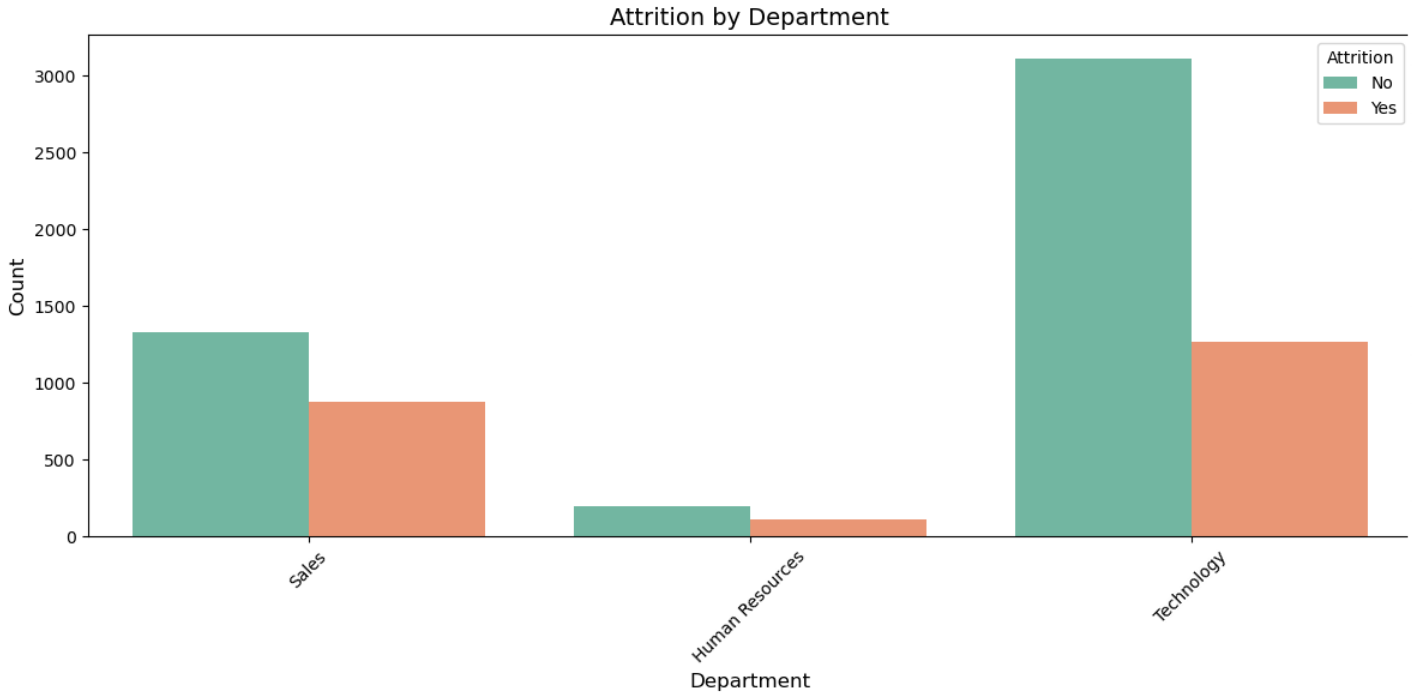

Age Distribution Across Departments
Age
50
45
40
35
30
25
20
Sales
Human Resources
Technology
Department

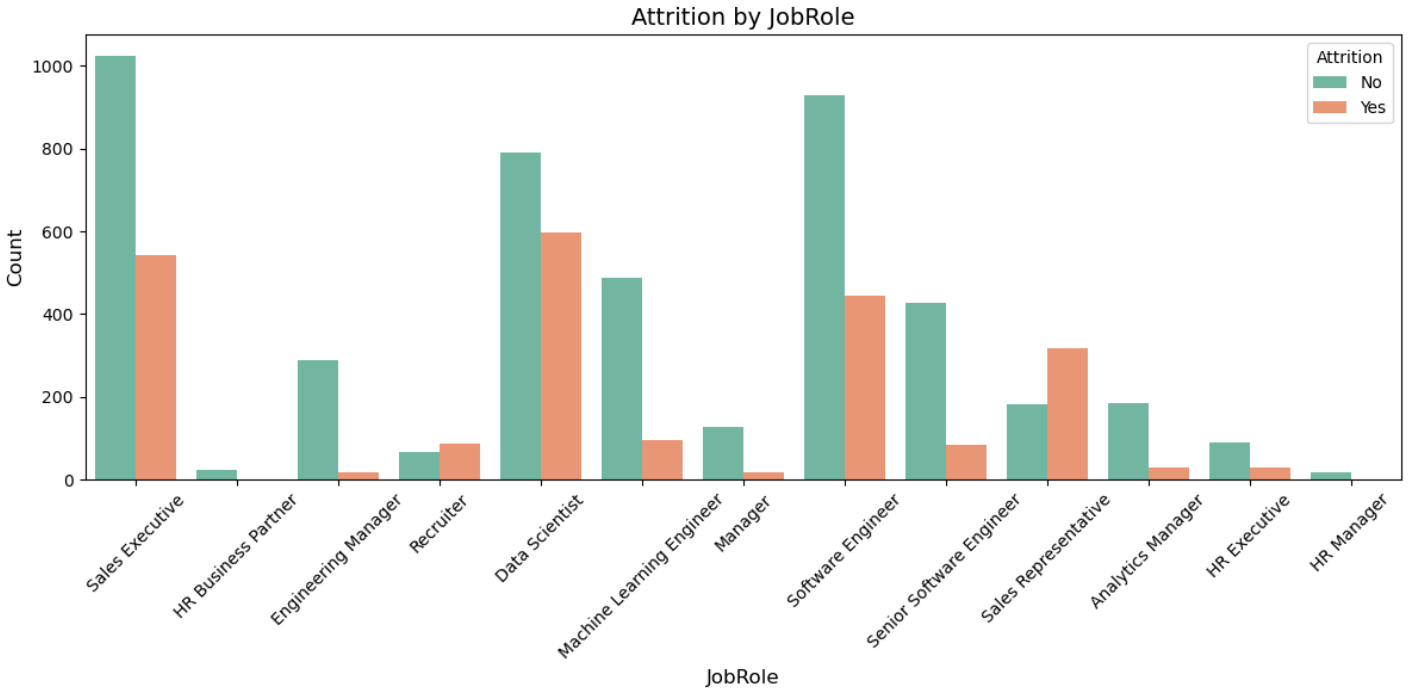
3. What are the key factors driving employee attrition?

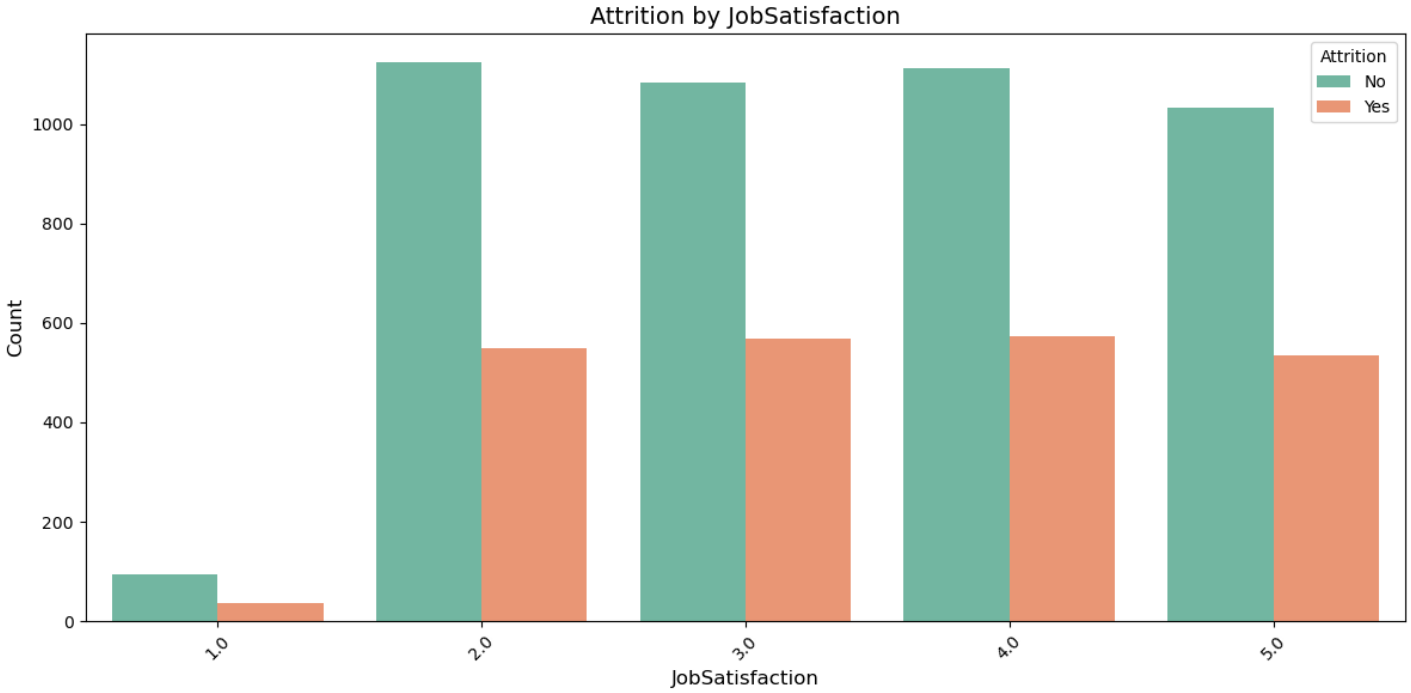
4.How does attrition vary across demographics (e.g., gender, ethnicity) and departments?

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

# Set the file path and sheet names
file_path = r'F:\data analysis\BQ\HR Data.xlsx'
employee_sheet = 'Employee'
performance_sheet = 'PerformanceRating'

# Load the data
employee_data = pd.read_excel(file_path, sheet_name=employee_sheet)
performance_data = pd.read_excel(file_path, sheet_name=performance_sheet)

# Merge performance data to get JobSatisfaction
if 'EmployeeID' in employee_data.columns and 'EmployeeID' in performance_data.columns:
    employee_data = employee_data.merge(performance_data[['EmployeeID', 'JobSatisfaction']], on='EmployeeID', how='left')
else:
    print("EmployeeID column not found in one of the datasets.")

# Rename MonthlyIncome to salary
if 'MonthlyIncome' in employee_data.columns:
    employee_data.rename(columns={'MonthlyIncome': 'Salary'}, inplace=True)

    # Drop rows where salary is missing
    employee_data = employee_data.dropna(subset=['Salary'])

    # Categorize salary into bins
    bins = [0, 3000, 7000, 12000, 20000, employee_data['Salary'].max()]
    labels = ['Low', 'Lower-Mid', 'Mid', 'Upper-Mid', 'High']
    employee_data['salary_category'] = pd.cut(employee_data['Salary'], bins=bins, labels=labels)

# Check for the 'Attrition' column
if 'Attrition' in employee_data.columns:
    # Ensure salary_category exists before analysis
    if 'salary_category' in employee_data.columns:
        key_factors = ['Age', 'Department', 'JobRole', 'Salary', 'JobSatisfaction']
    else:
        key_factors = ['Age', 'Department', 'JobRole', 'JobSatisfaction']

    for factor in key_factors:
        if factor in employee_data.columns:
            plt.figure(figsize=(12, 6))
            sns.countplot(data=employee_data, x=factor, hue='Attrition', palette='Set2')
            plt.title(f'Attrition by {factor}', fontsize=14)
            plt.xlabel(factor, fontsize=12)
            plt.ylabel('Count', fontsize=12)
            plt.xticks(rotation=45)
            plt.legend(title='Attrition', loc='upper right')
            plt.tight_layout()
            plt.show()
        else:
            print(f"Column '{factor}' not found in the dataset.")
else:
    print("The dataset does not contain an 'Attrition' column.")
```


Attrition by Age
Attrition
No
Yes
Count
Age
0
50
100
150
200
250
300
350
400
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51


Attrition by Department
Attrition
No
Yes
Count
Department
0
500
1000
1500
2000
2500
3000
Sales
Human Resources
Technology


Attrition by JobRole
Attrition
No
Yes
Count
0
200
400
600
800
1000
Sales Executive
HR Business Partner
Engineering Manager
Recruiter
Data Scientist
Machine Learning Engineer
Manager
Software Engineer
Senior Software Engineer
Sales Representative
Analytics Manager
HR Executive
HR Manager
JobRole


Attrition by JobSatisfaction
Attrition
No
Yes
Count
0
200
400
600
800
1000
1.0
2.0
3.0
4.0
5.0
JobSatisfaction

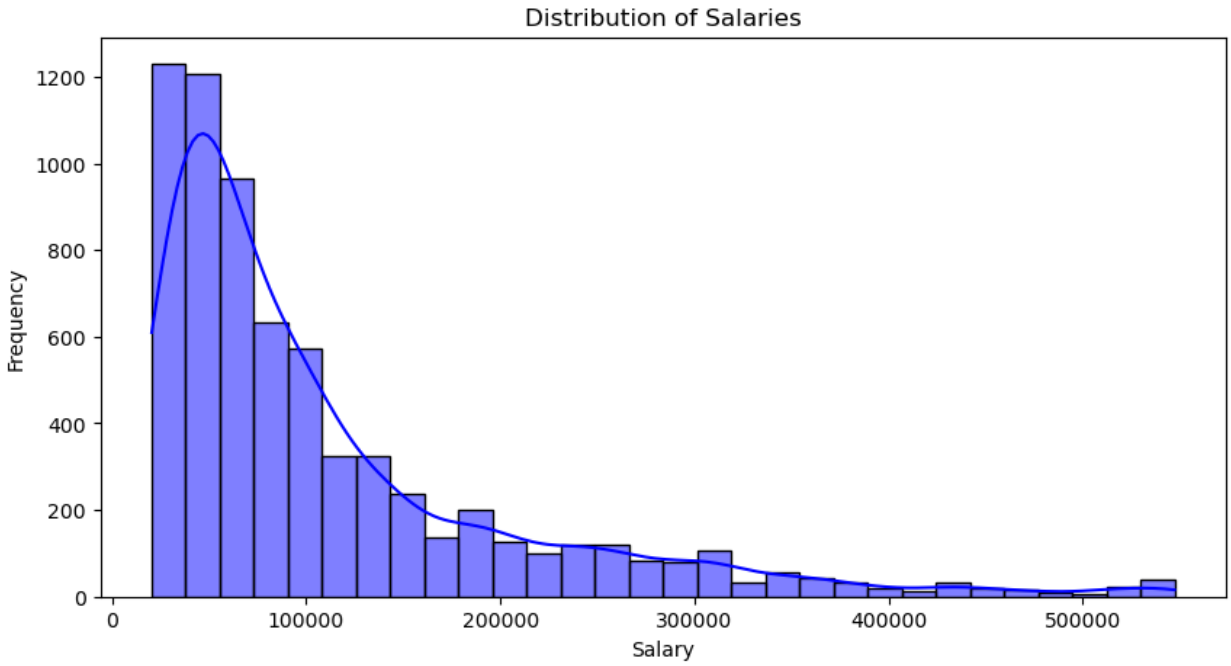
6.Is there a significant deference in salaries across departments, genders, or other demographic groups?

```
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
```

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

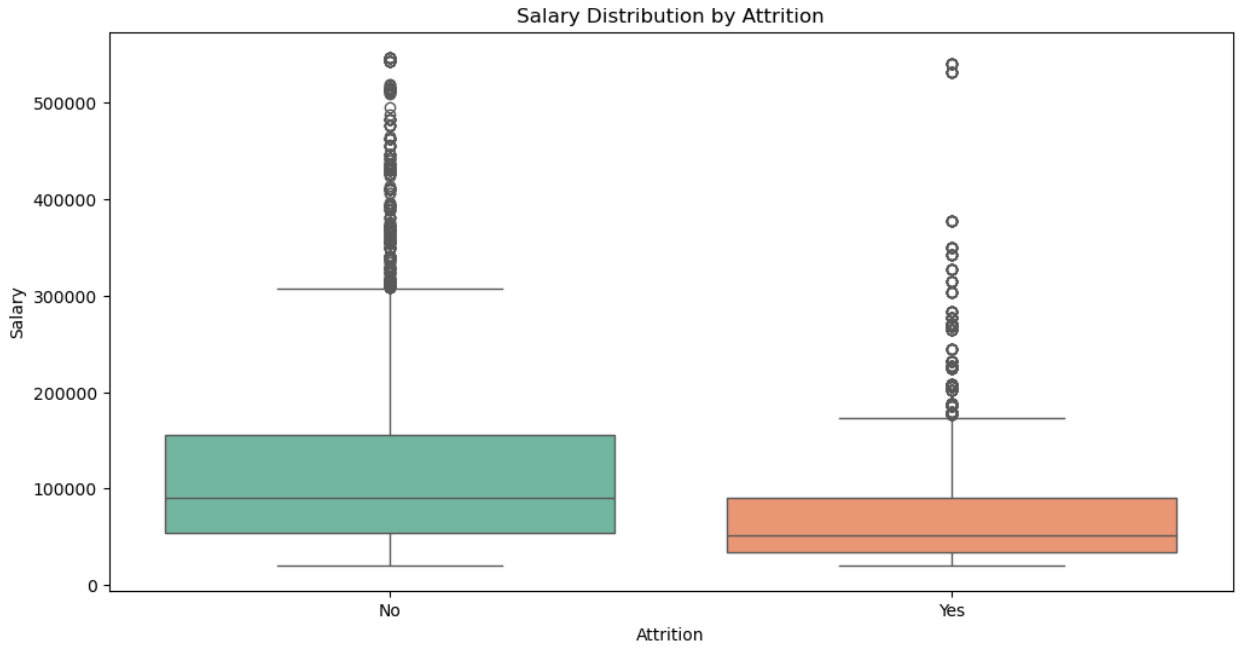



```
avg_salary_by_department = employee_data.groupby('Department')['Salary'].mean().sort_values()
print(avg_salary_by_department)
```

[14] ✓ 0.0s Python

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

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

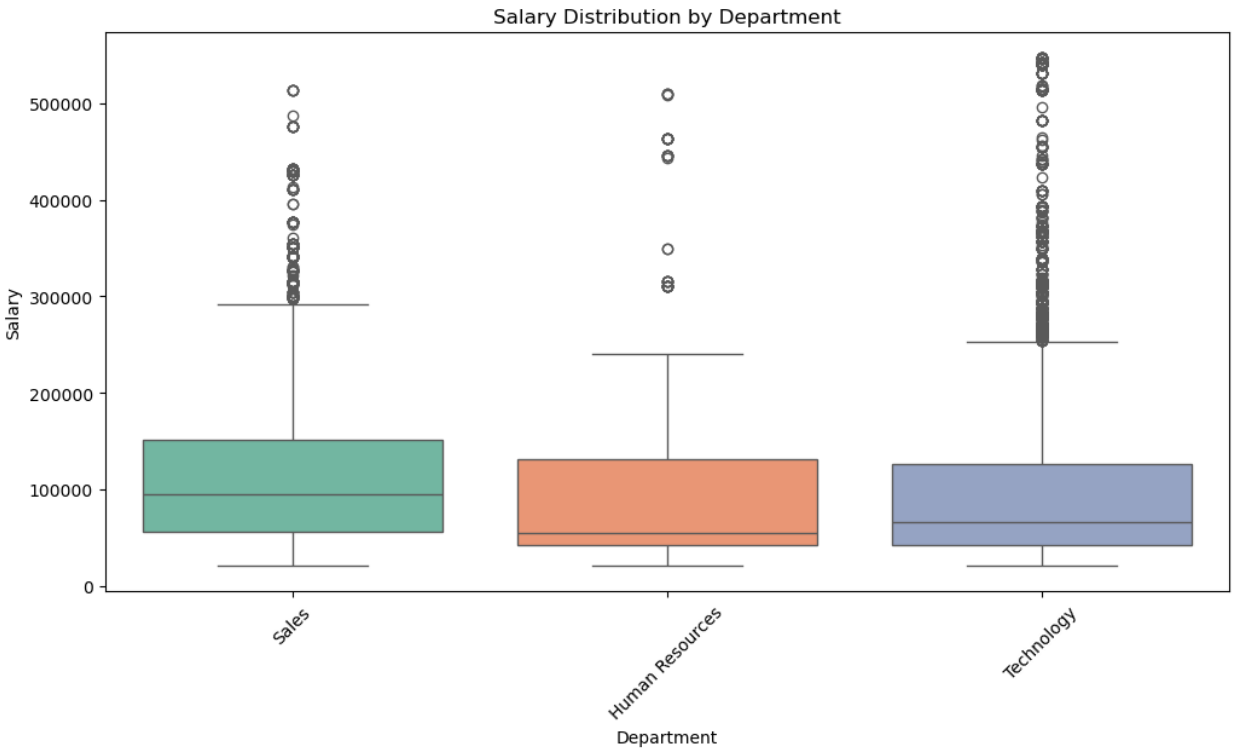



```python
import scipy.stats as stats

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

    # تنفيذ 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}")

    # تفسير النتيجة
    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 Python

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

```python
plt.figure(figsize=(12, 6))
sns.boxplot(data=employee_data, x='Department', y='Salary', palette='Set2')
plt.title('Salary Distribution by Department')
plt.xlabel('Department')
plt.ylabel('Salary')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(12, 6))
sns.barplot(data=employee_data, x='Department', y='Salary', palette='Set3', ci=None)
plt.title('Average Salary by Department')
plt.xlabel('Department')
plt.ylabel('Average Salary')
plt.xticks(rotation=45)
plt.show()

```

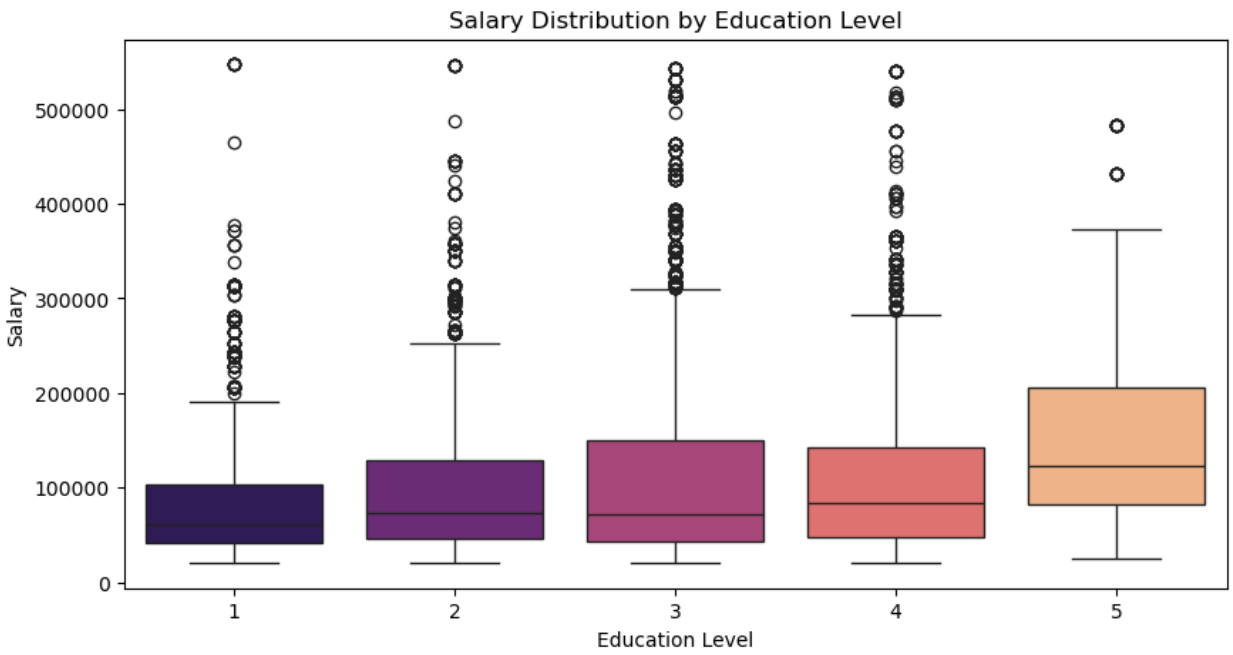


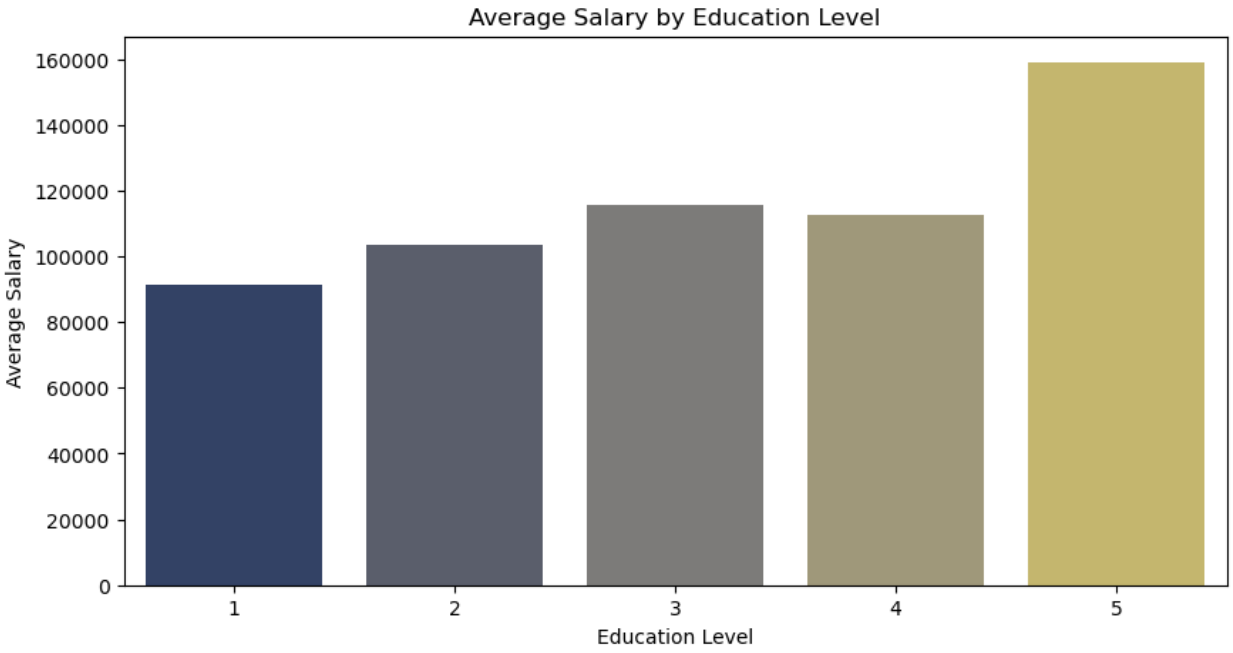

```python
if 'Education' in employee_data.columns and 'Salary' in employee_data.columns:
    education_groups = [group["Salary"].dropna() for _, group in employee_data.groupby("Education")]

    f_stat, p_value = stats.f_oneway(*education_groups)
    print(f"ANOVA 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.
```

```python
plt.figure(figsize=(10, 5))
sns.boxplot(data=employee_data, x='Education', y='Salary', palette='magma')
plt.title('Salary Distribution by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Salary')
plt.show()

plt.figure(figsize=(10, 5))
sns.barplot(data=employee_data, x='Education', y='Salary', palette='cividis', ci=None)
plt.title('Average Salary by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Average Salary')
plt.show()

```



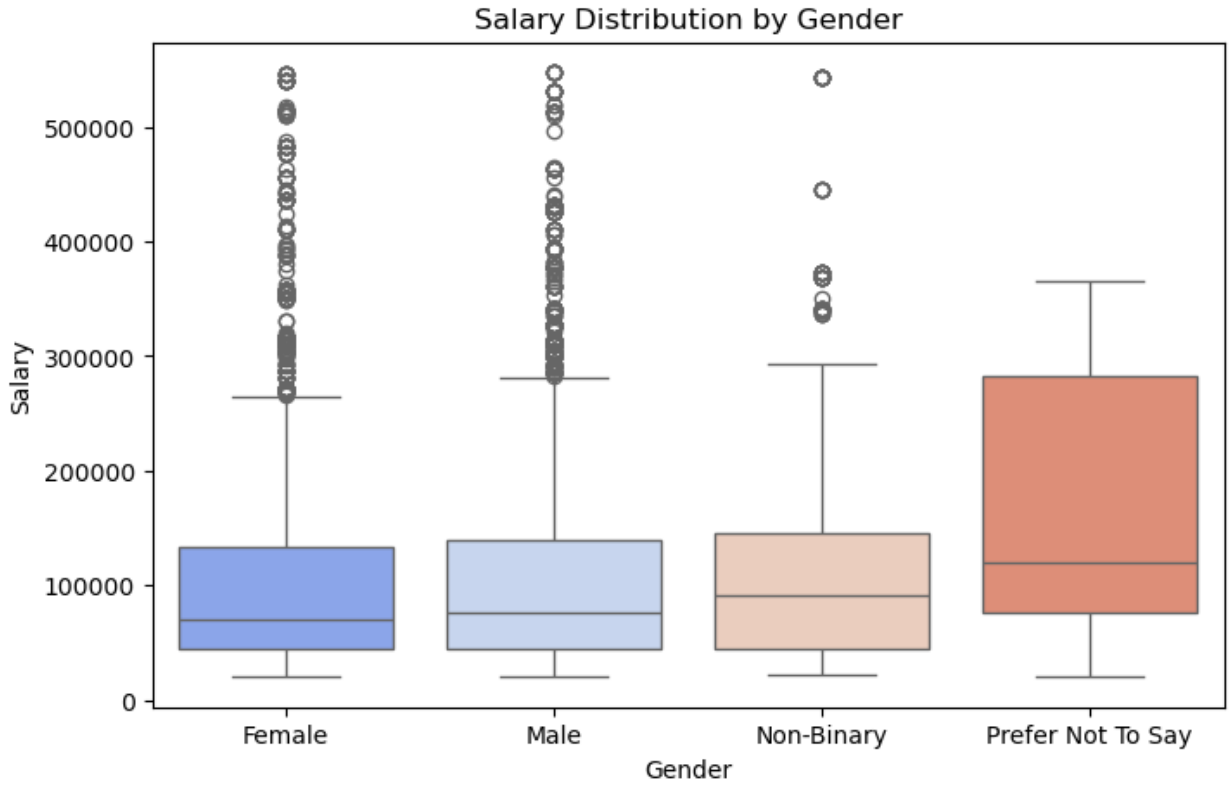


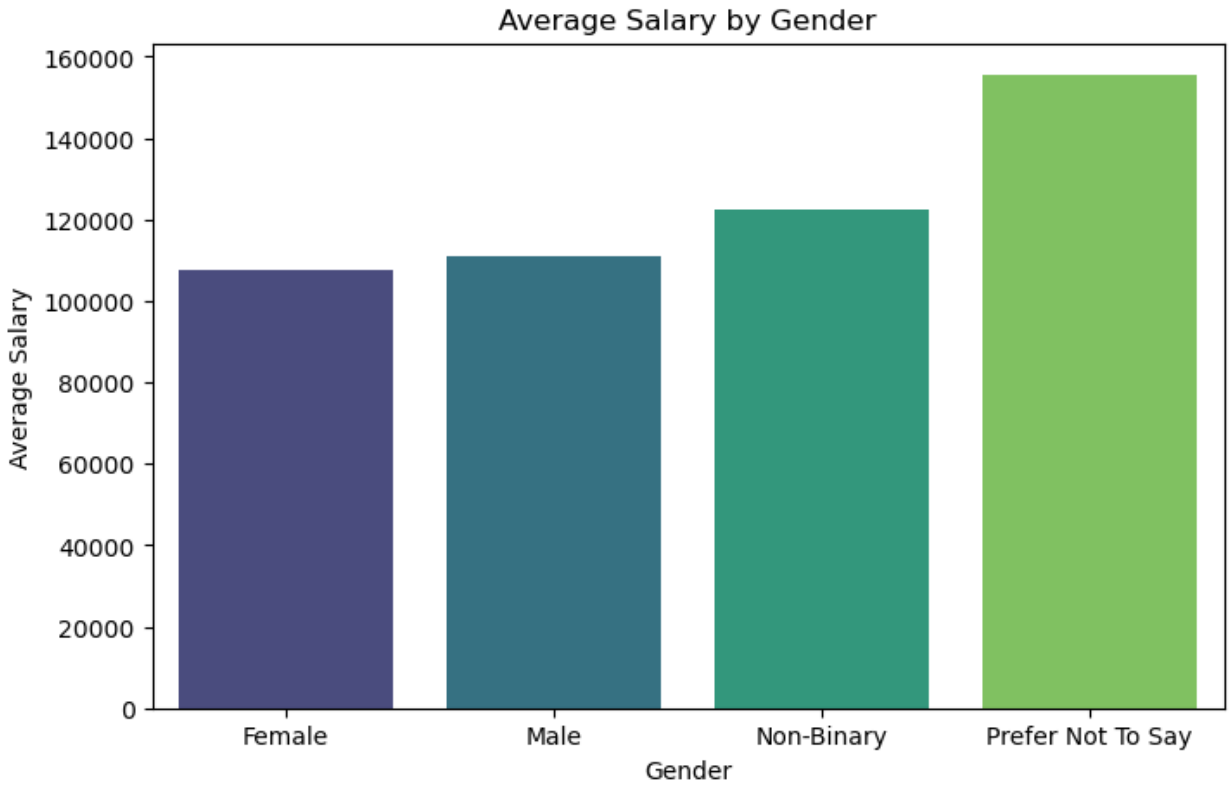

```python
if 'Gender' in employee_data.columns and 'Salary' in employee_data.columns:
    male_salaries = employee_data[employee_data['Gender'] == 'Male']['Salary'].dropna()
    female_salaries = employee_data[employee_data['Gender'] == 'Female']['Salary'].dropna()

    # 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.
```

```python
plt.figure(figsize=(8, 5))
sns.boxplot(data=employee_data, x='Gender', y='Salary', palette='coolwarm')
plt.title('Salary Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Salary')
plt.show()

plt.figure(figsize=(8, 5))
sns.barplot(data=employee_data, x='Gender', y='Salary', palette='viridis', ci=None)
plt.title('Average Salary by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Salary')
plt.show()

```

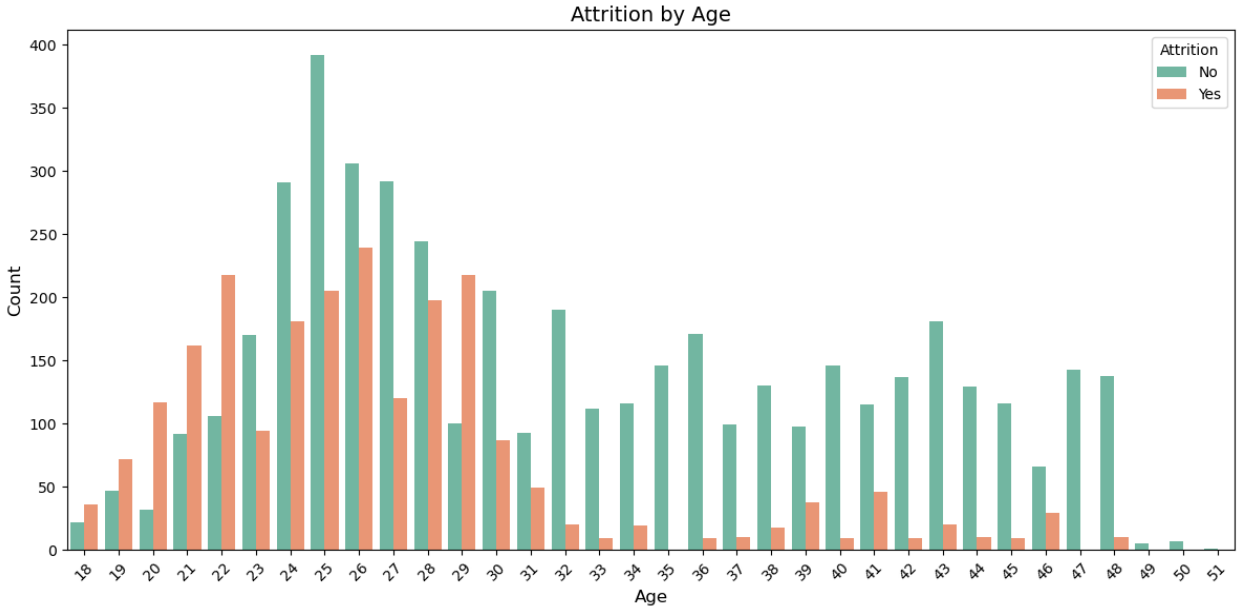


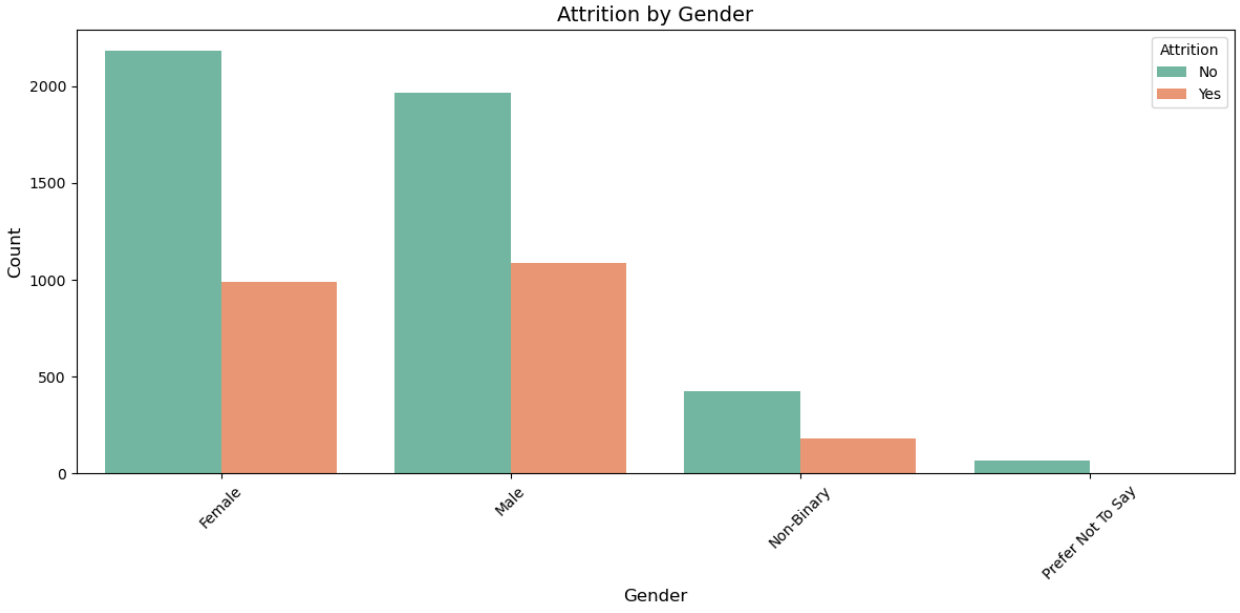

Average Salary by Gender
160000
140000
120000
100000
80000
60000
40000
20000
0
Average Salary
Female
Male
Non-Binary
Prefer Not To Say
Gender

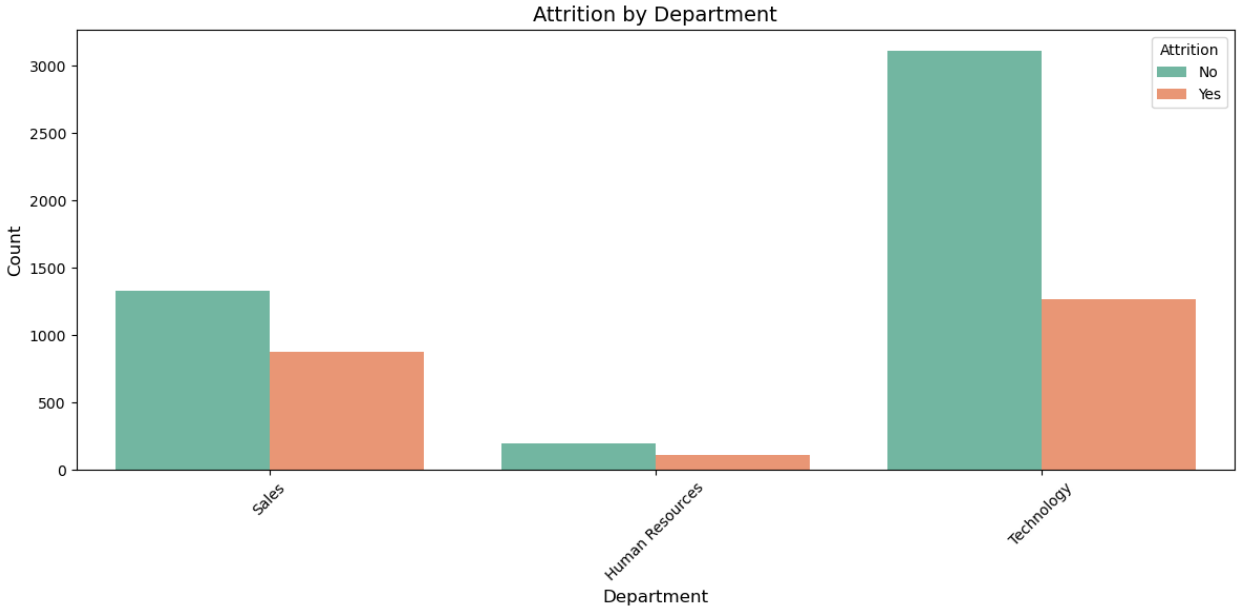
4. How does attrition vary across demographics (e.g., gender, ethnicity) and departments?

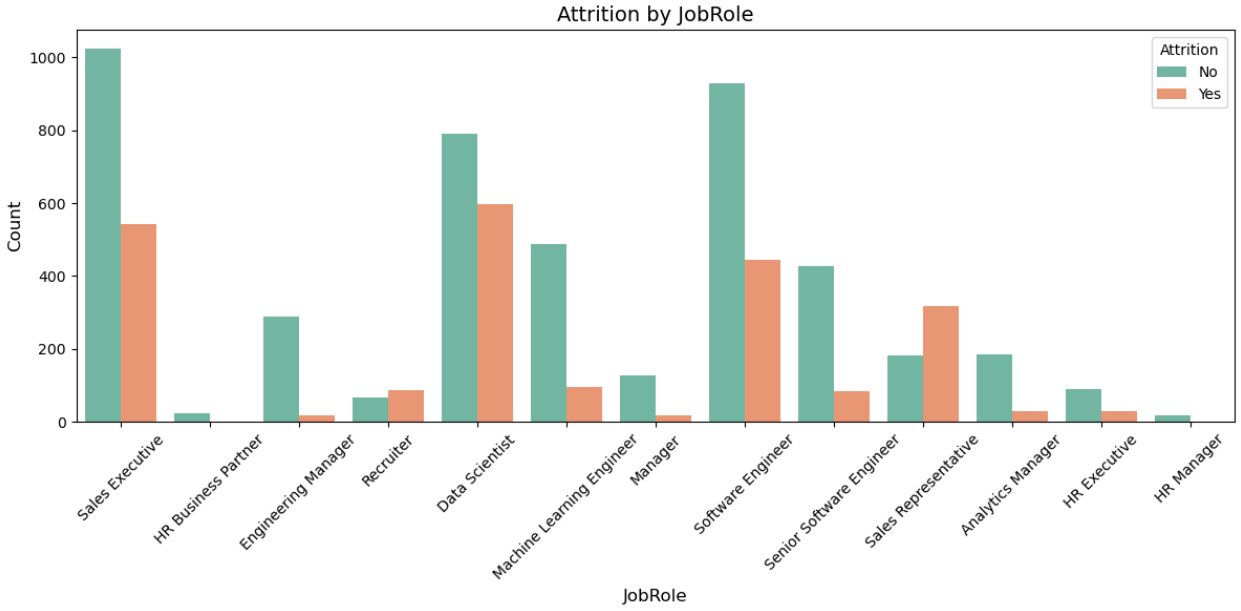
7. How does marital status influence attrition rates?

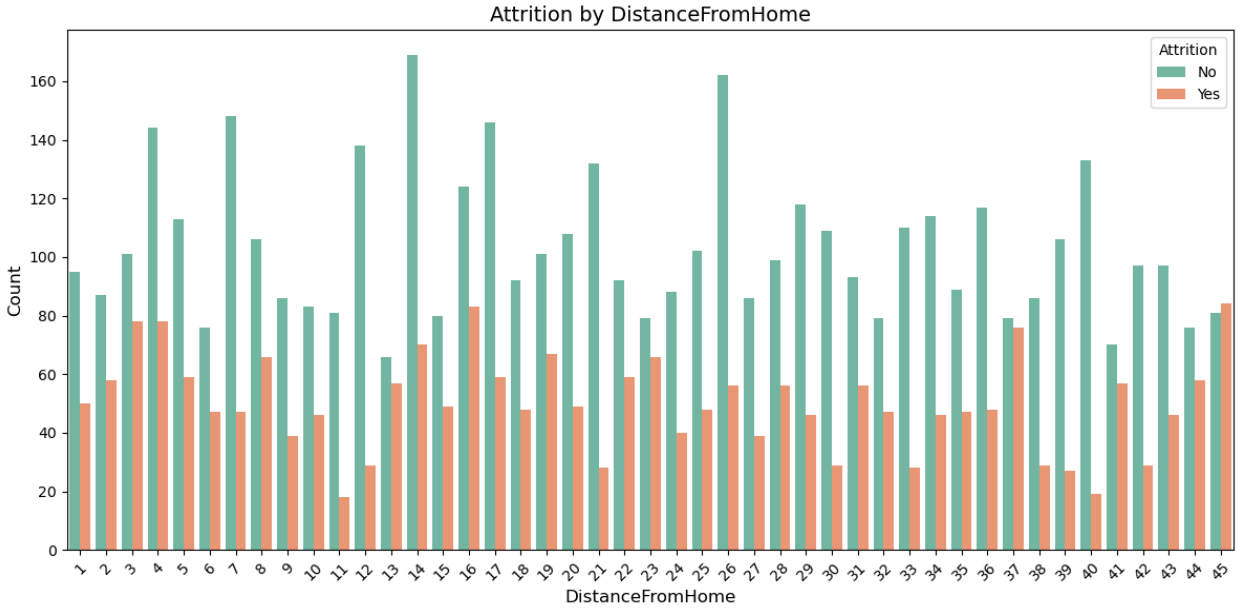
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# List of key factors to analyze
key_factors = [
    'Age', 'Gender', 'Department', 'JobRole', 'DistanceFromHome',
    'MaritalStatus', 'OverTime','Ethnicity'
]

# Loop through each factor and visualize attrition impact
for factor in key_factors:
    if factor in employee_data.columns:
        plt.figure(figsize=(12, 6))
        sns.countplot(data=employee_data, x=factor, hue='Attrition', palette='Set2')
        plt.title(f'Attrition by {factor}', fontsize=14)
        plt.xlabel(factor, fontsize=12)
        plt.ylabel('Count', fontsize=12)
        plt.xticks(rotation=45)
        plt.legend(title='Attrition', loc='upper right')
        plt.tight_layout()
        plt.show()
    else:
        print(f"Column '{factor}' not found in the dataset.")

```

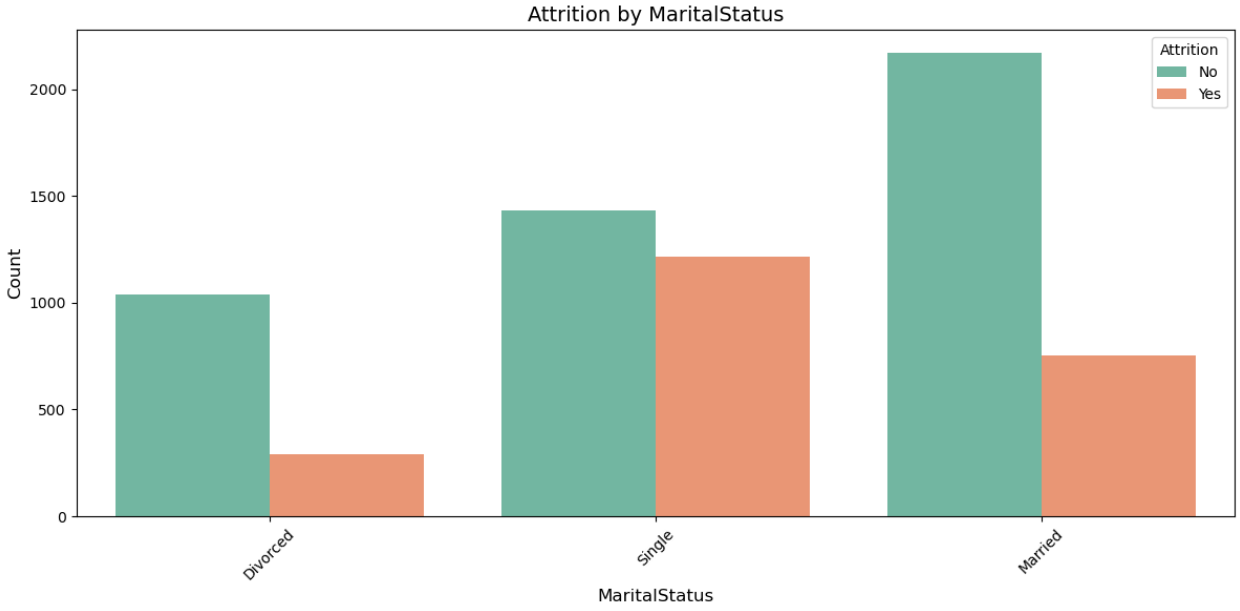

Attrition by Age
Attrition
No
Yes
Count
Age
0
50
100
150
200
250
300
350
400
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

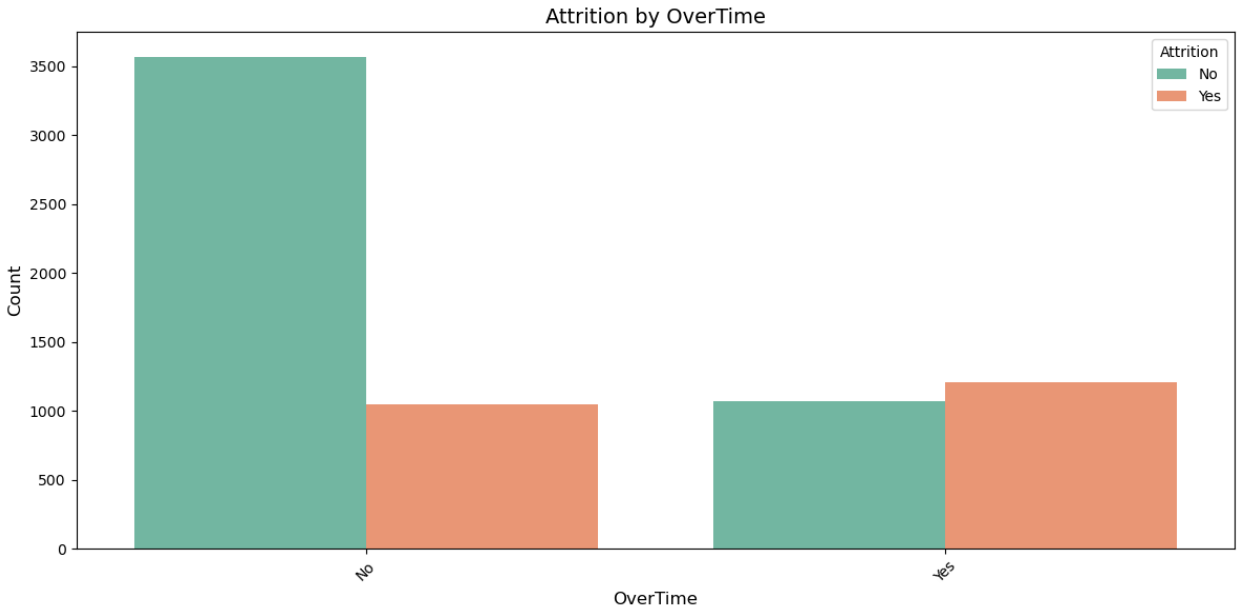

Attrition by Gender
Attrition
No
Yes
Count
Gender
0
500
1000
1500
2000
Female
Male
Non-Binary
Prefer Not To Say


Attrition by Department
Attrition
No
Yes
Count
3000
2500
2000
1500
1000
500
0
Sales
Human Resources
Technology
Department


Attrition by JobRole
Attrition
No
Yes
Count
1000
800
600
400
200
0
Sales Executive
HR Business Partner
Engineering Manager
Recruiter
Data Scientist
Machine Learning Engineer
Manager
Software Engineer
Senior Software Engineer
Sales Representative
Analytics Manager
HR Executive
HR Manager
JobRole


Attrition by DistanceFromHome
Attrition
No
Yes
Count
DistanceFromHome


Attrition by MaritalStatus
Attrition
No
Yes
Count
Divorced
Single
Married
MaritalStatus

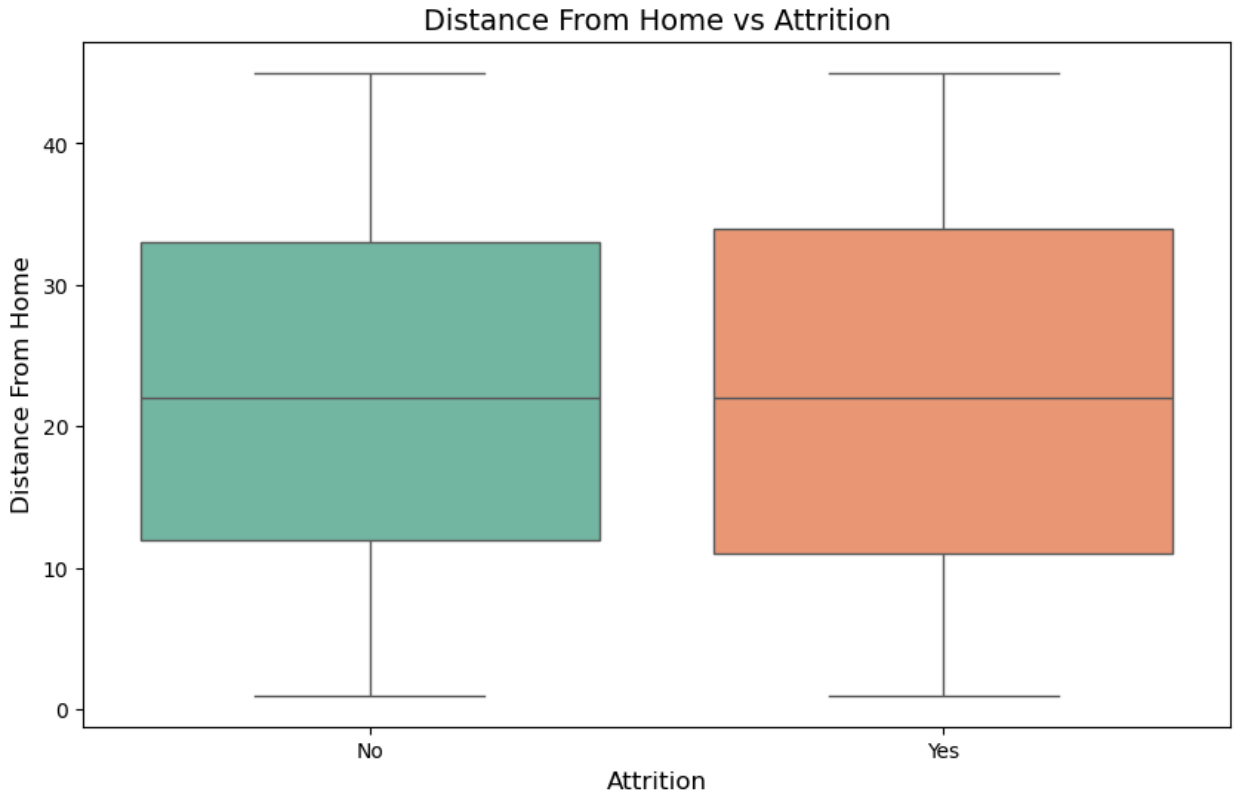


5. Is there a correlation between DistanceFromHome and attrition?

```python
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
```

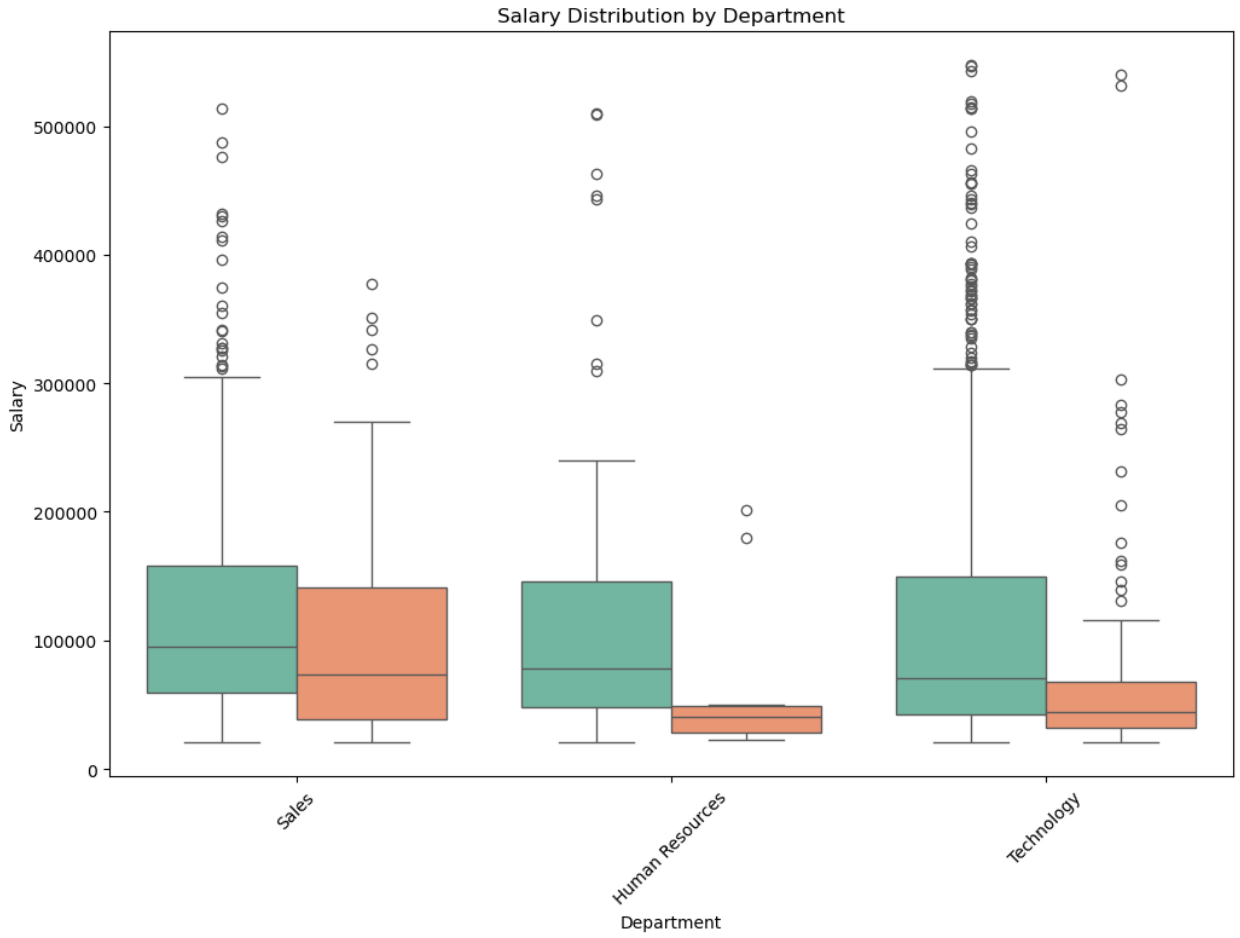


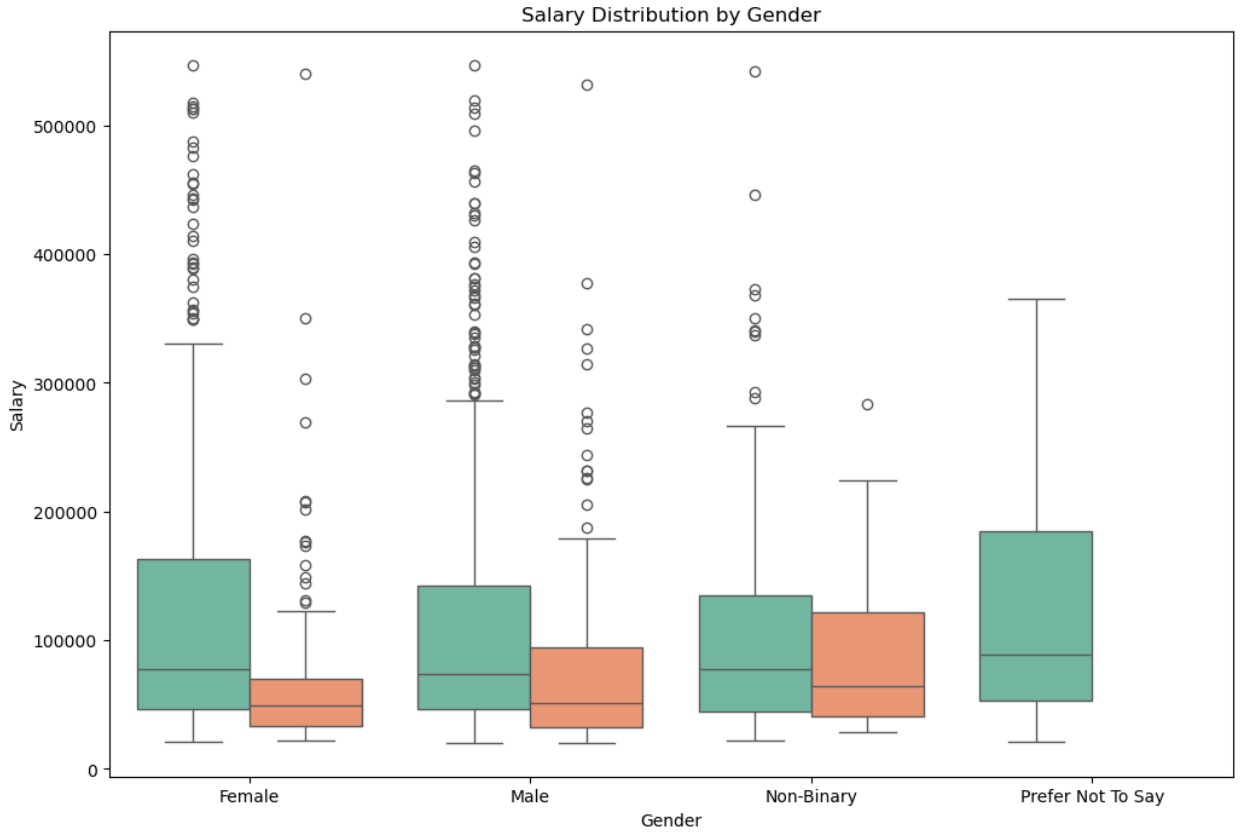
6.Is there a significant deference in salaries across departments, genders, or other demographic groups?

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

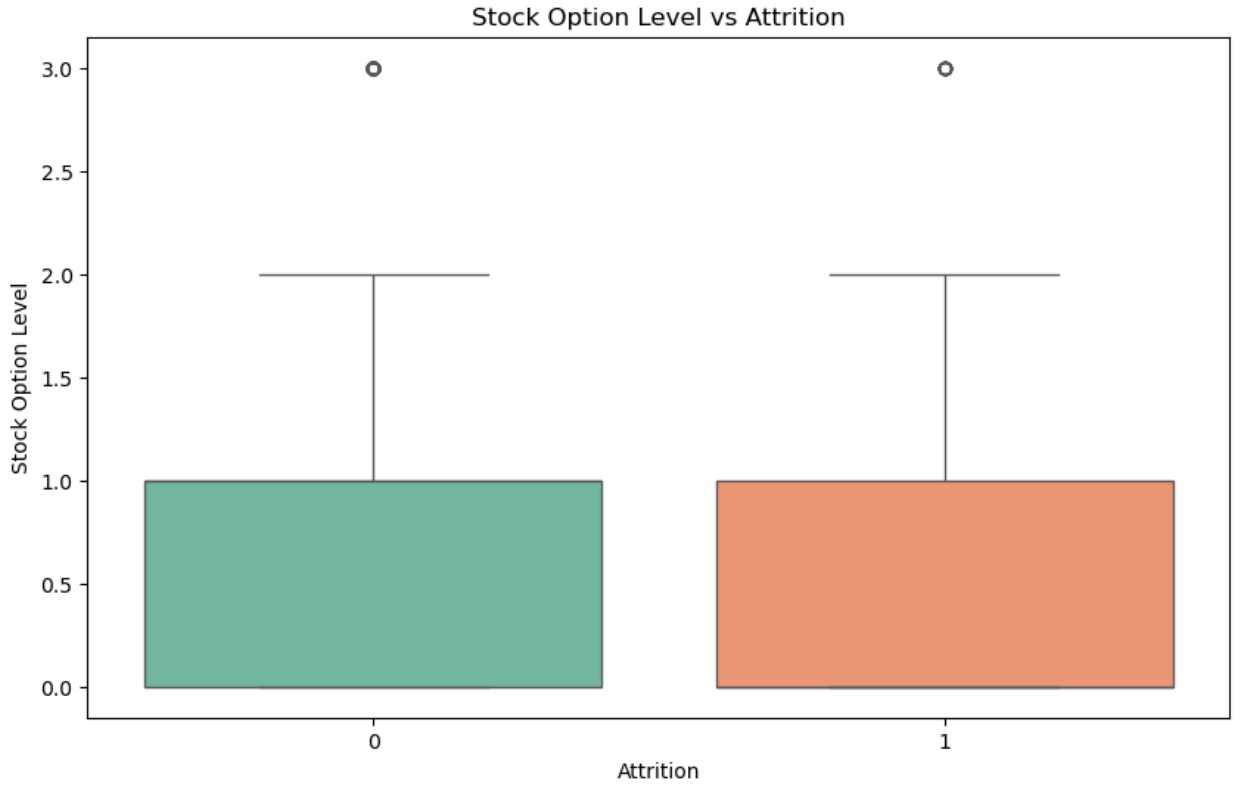
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Load your dataset
file_path = r'F:\data analysis\BQ\HR Data.xlsx'  # Update with your file path
sheet_name = 'Employee'  # Adjust if necessary
employee_data = pd.read_excel(file_path, sheet_name=sheet_name)

# Ensure necessary columns exist
required_columns = ['Attrition', 'Gender', 'Ethnicity', 'Department', 'Salary', 'StockOptionLevel', 'OverTime']
if all(col in employee_data.columns for col in required_columns):

    # Convert Attrition to numerical values (Yes=1, No=0) if not already done
    if 'AttritionNumeric' not in employee_data.columns:
        employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})

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

    print("Salary Comparison Across Demographics and Departments:")

    # Salary by Department (ANOVA)
    department_salary = employee_data.groupby('Department')['Salary'].mean()
    print("\nAverage Salary by Department:")
    print(department_salary)

    # ANOVA Test for Salary by Department
    department_salary_groups = [group['Salary'].values for name, group in employee_data.groupby('Department')]
    f_stat, p_val = stats.f_oneway(*department_salary_groups)
    print(f"\nANOVA Test Result for Salary by Department: F-statistic = {f_stat:.2f}, P-value = {p_val:.4f}")

    # Salary by Gender (T-test)
    male_salary = employee_data[employee_data['Gender'] == 'Male']['Salary']
    female_salary = employee_data[employee_data['Gender'] == 'Female']['Salary']
    t_stat, p_val_gender = stats.ttest_ind(male_salary, female_salary)
    print(f"\nT-test for Salary by Gender: T-statistic = {t_stat:.2f}, P-value = {p_val_gender:.4f}")

    # Visualize Salary by Department and Gender (with hue to avoid FutureWarning)
    plt.figure(figsize=(12, 8))
    sns.boxplot(x='Department', y='Salary', data=employee_data, hue='Attrition', palette='Set2', legend=False)
    plt.title('Salary Distribution by Department')
    plt.xticks(rotation=45)
    plt.show()

    plt.figure(figsize=(12, 8))
    sns.boxplot(x='Gender', y='Salary', data=employee_data, hue='Attrition', palette='Set2', legend=False)
    plt.title('Salary Distribution by Gender')
    plt.show()

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

    # Correlation between StockOptionLevel and Attrition (Retention)
    print("\nStock Option Level and Retention (Attrition):")
    correlation_stock = employee_data['StockOptionLevel'].corr(employee_data['AttritionNumeric'])
    print(f"Correlation between StockOptionLevel and Attrition: {correlation_stock:.2f}")

    # Visualize StockOptionLevel vs Attrition
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='AttritionNumeric', y='StockOptionLevel', data=employee_data, palette='Set2')
    plt.title('Stock Option Level vs Attrition')
    plt.xlabel('Attrition')
    plt.ylabel('Stock Option Level')
    plt.show()

else:
    print("Some required columns are missing.")

```

```
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
```



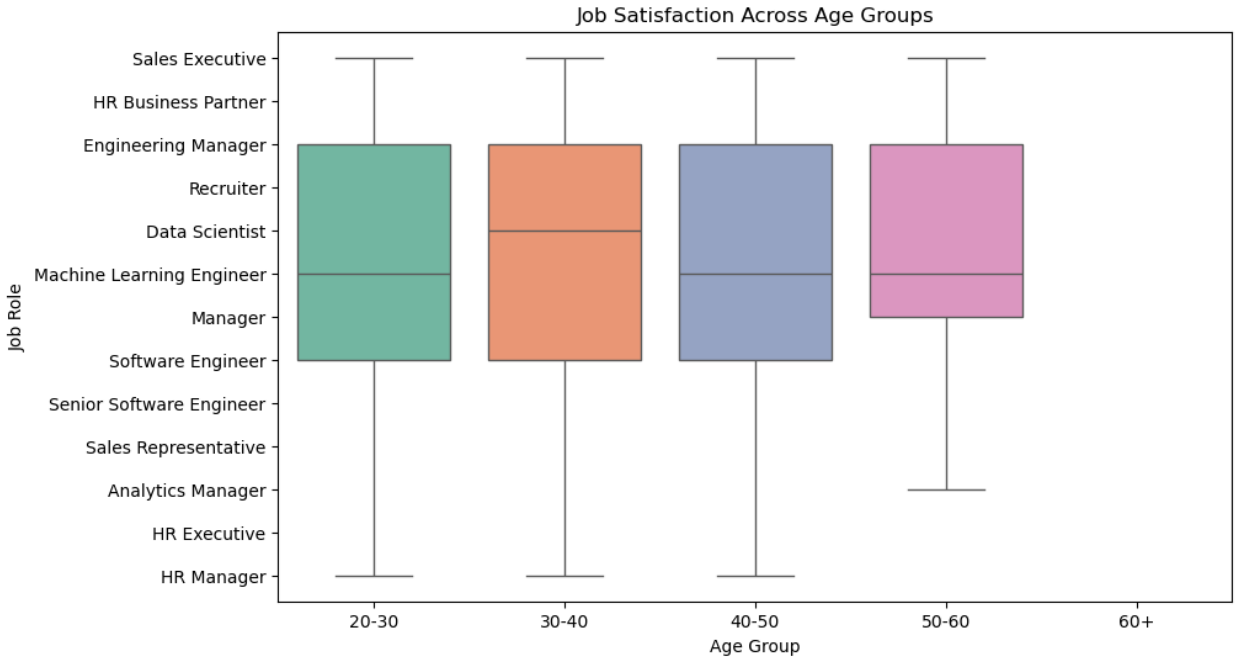
3) Compensation Analysis:

Age Impact on Job Satisfaction and Attrition:

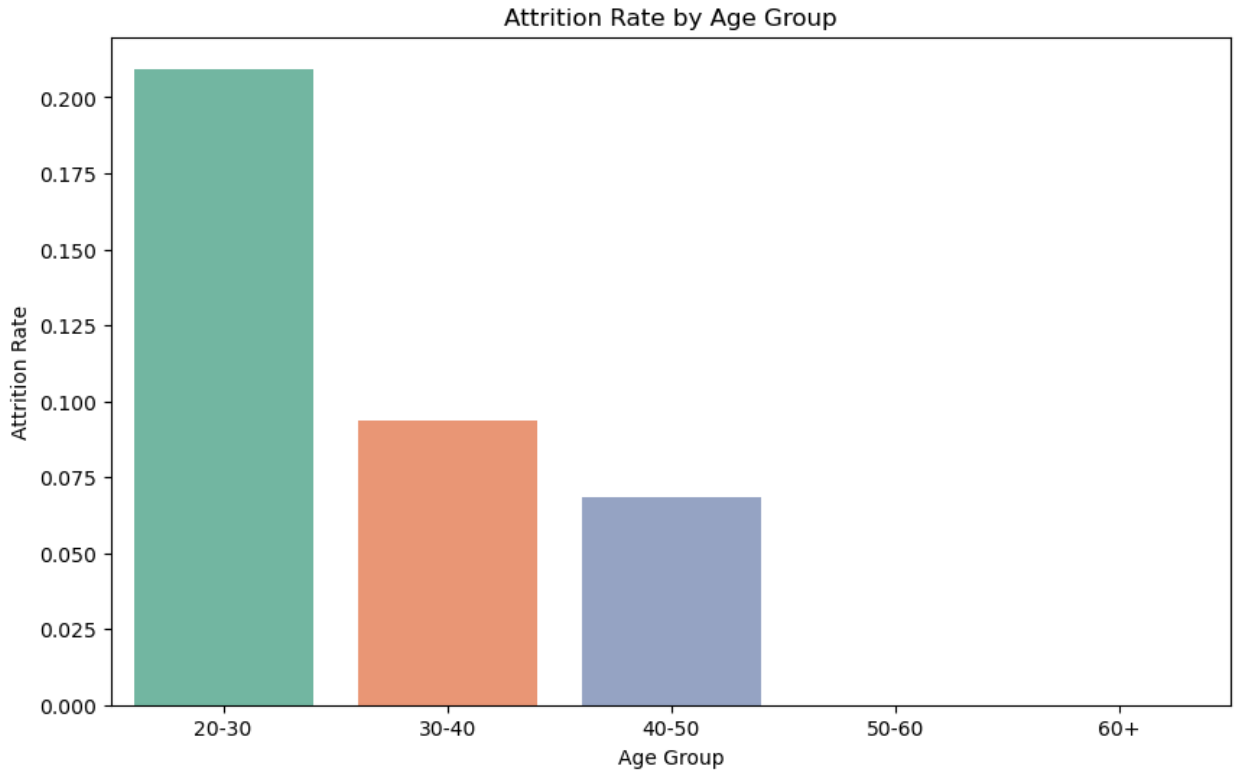
How does job satisfaction vary across different age groups?

What is the attrition rate for different age groups?

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# 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 necessary columns exist
required_columns = ['Attrition', 'Age', 'JobRole', 'Department', 'Gender']
if all(col in employee_data.columns for col in required_columns):

    # Convert Attrition to numerical values (Yes=1, No=0) if not already done
    if 'AttritionNumeric' not in employee_data.columns:
        employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})

    # 8. Age Impact on Job Satisfaction and Attrition:
    print("\nAge Impact on Job Satisfaction and Attrition:")

    # Create age groups (e.g., 20-30, 30-40, 40-50, 50+)
    age_bins = [20, 30, 40, 50, 60, 70]
    age_labels = ['20-30', '30-40', '40-50', '50-60', '60+']
    employee_data['AgeGroup'] = pd.cut(employee_data['Age'], bins=age_bins, labels=age_labels, right=False)

    # 1. How does job satisfaction vary across different age groups?
    # Assuming 'JobRole' or other columns are related to job satisfaction
    # If you have a specific column for job satisfaction, you can use it here
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='AgeGroup', y='JobRole', data=employee_data, palette='Set2')
    plt.title('Job Satisfaction Across Age Groups')
    plt.xlabel('Age Group')
    plt.ylabel('Job Role')
    plt.show()

    # 2. What is the attrition rate for different age groups?
    attrition_rate_by_age = employee_data.groupby('AgeGroup')['AttritionNumeric'].mean()
    print("\nAttrition Rate by Age Group:")
    print(attrition_rate_by_age)

    # Visualize Attrition Rate by Age Group
    plt.figure(figsize=(10, 6))
    sns.barplot(x=attrition_rate_by_age.index, y=attrition_rate_by_age.values, palette='Set2')
    plt.title('Attrition Rate by Age Group')
    plt.xlabel('Age Group')
    plt.ylabel('Attrition Rate')
    plt.show()

else:
    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
```

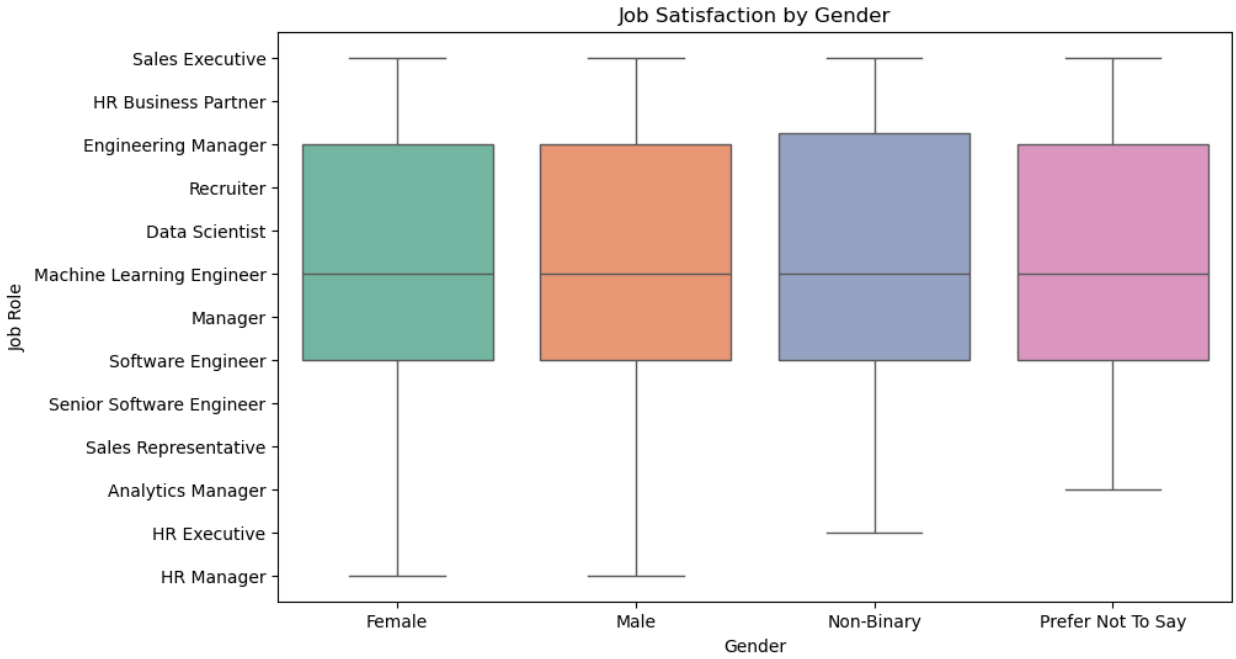


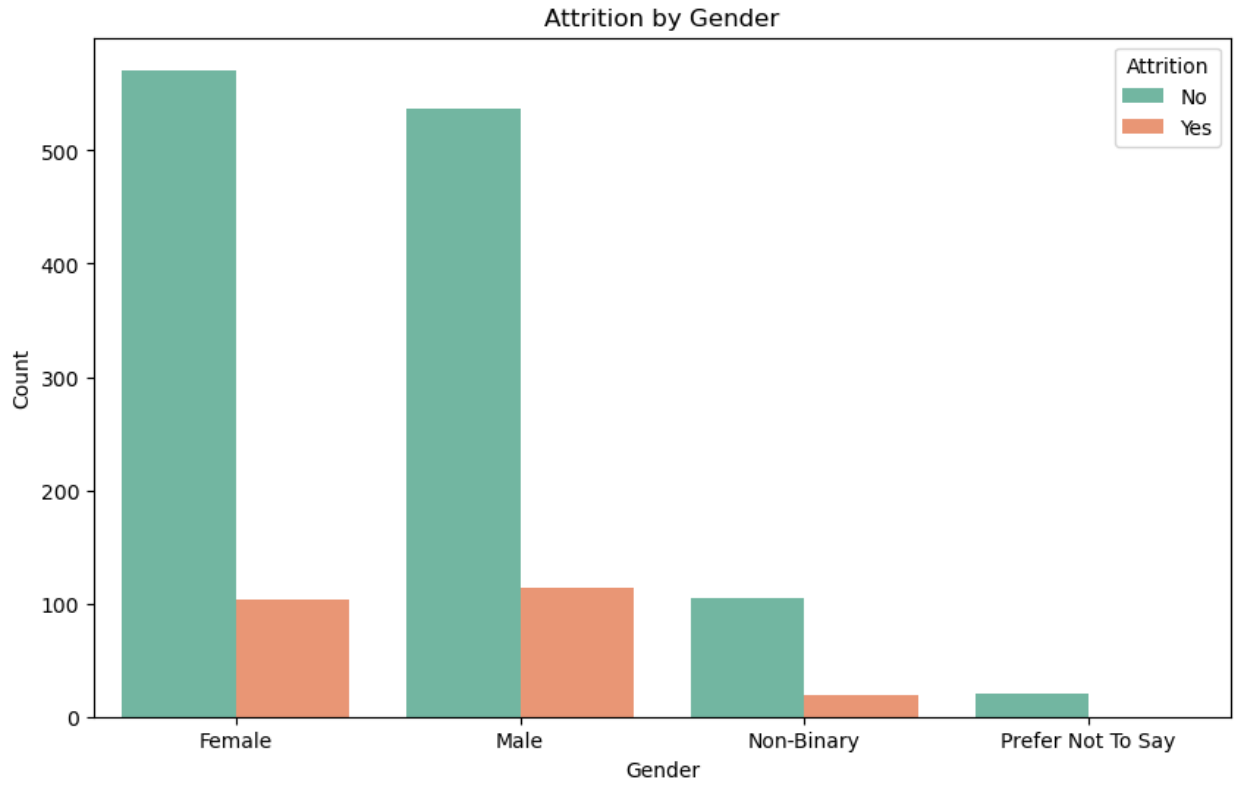
3) Compensation Analysis:

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?*

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# 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 necessary columns exist
required_columns = ['Attrition', 'Gender', 'JobRole', 'Age']
if all(col in employee_data.columns for col in required_columns):

    # Convert Attrition to numerical values (Yes=1, No=0) if not already done
    if 'AttritionNumeric' not in employee_data.columns:
        employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})

    # 9. Gender Differences in Job Satisfaction and Performance:
    print("\nGender Differences in Job Satisfaction and Performance:")

    # 1. How does job satisfaction vary between male and female employees?
    # Using JobRole as a proxy for job satisfaction
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Gender', y='JobRole', data=employee_data, palette='Set2')
    plt.title('Job Satisfaction by Gender')
    plt.xlabel('Gender')
    plt.ylabel('Job Role')
    plt.show()

    # 2. Is there a difference in retention (Attrition) by Gender?
    plt.figure(figsize=(10, 6))
    sns.countplot(x='Gender', hue='Attrition', data=employee_data, palette='Set2')
    plt.title('Attrition by Gender')
    plt.xlabel('Gender')
    plt.ylabel('Count')
    plt.show()

    # Statistical Test: Chi-Square Test for Gender and Attrition
    gender_attrition_crosstab = pd.crosstab(employee_data['Gender'], employee_data['Attrition'])
    chi2_stat, p_val_gender_attrition, dof, expected = stats.chi2_contingency(gender_attrition_crosstab)
    print(f"\nChi-Square Test for Gender and Attrition: Chi2-statistic = {chi2_stat:.2f}, P-value = {p_val_gender_attrition:.4f}")

else:
    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:

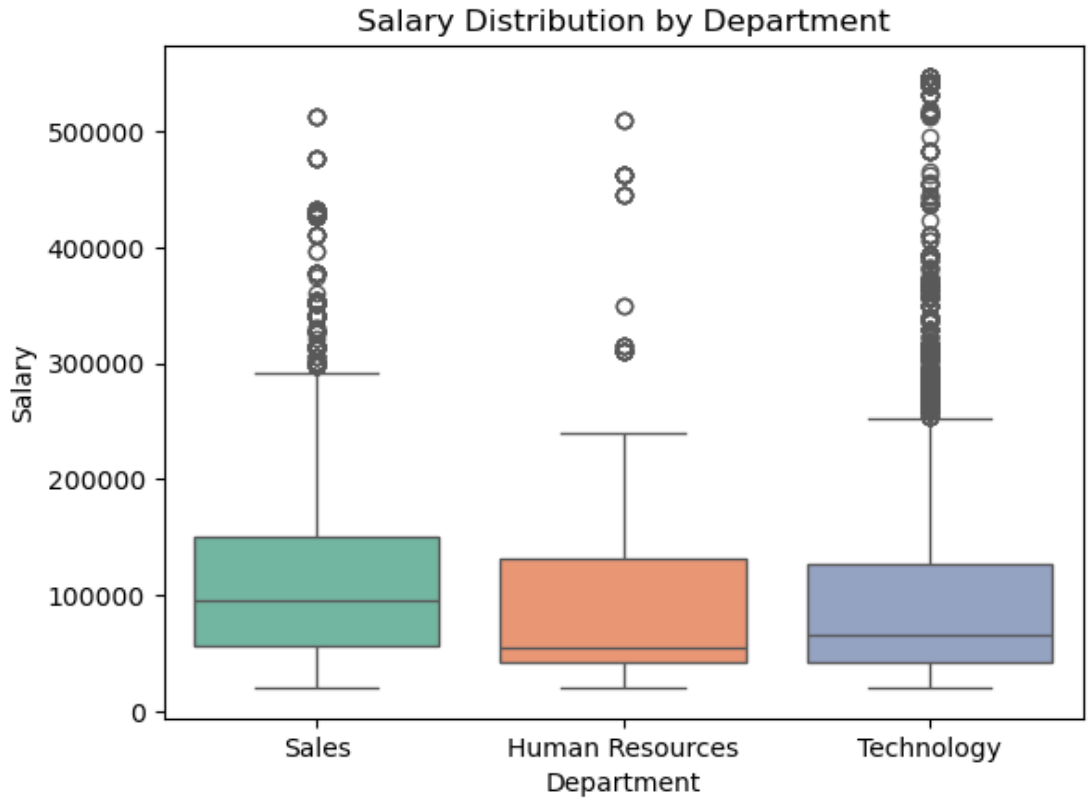
How does marital status influence attrition rates and job satisfaction?

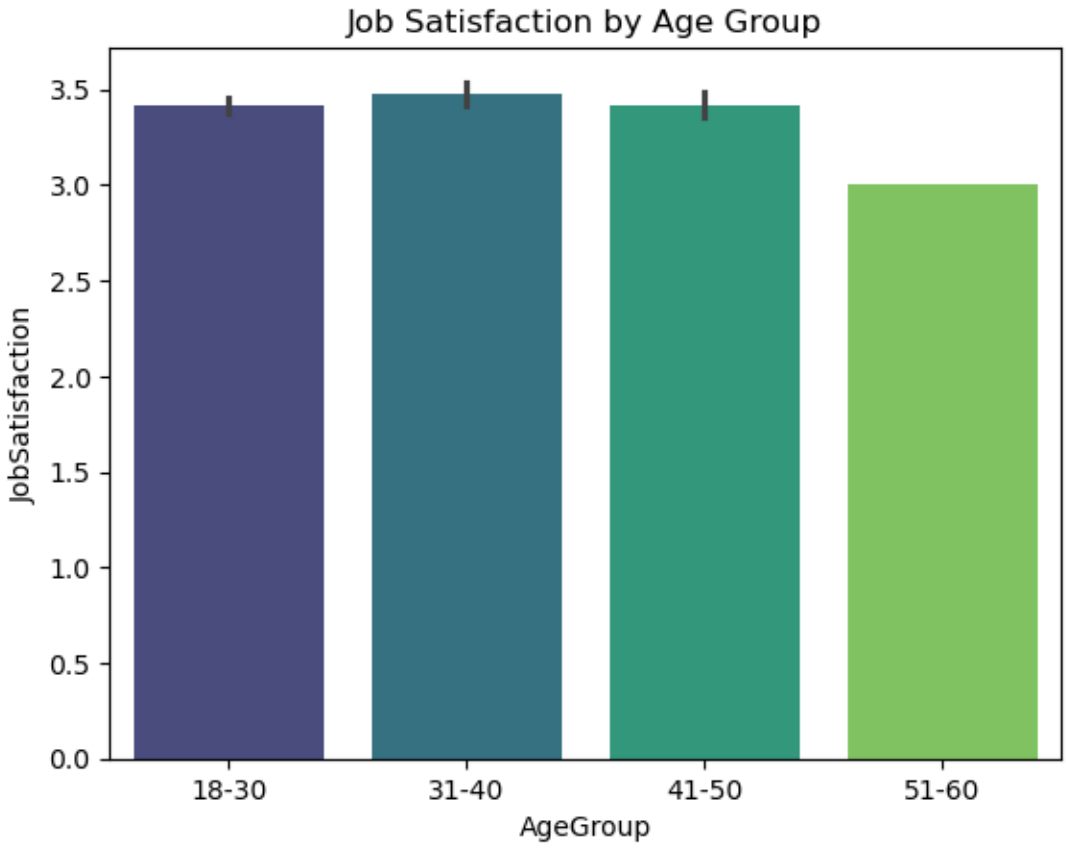
- *Does marital status affect attrition rates?*
- *Is there a relationship between marital status and job satisfaction?*

```python
import pandas as pd
from scipy.stats import f_oneway, ttest_ind

# Define file path
file_path = r'F:\data analysis\BQ\HR Data.xlsx'

# Load the sheets
employee_data = pd.read_excel(file_path, sheet_name='Employee')
performance_data = pd.read_excel(file_path, sheet_name='PerformanceRating')

# Merge the two datasets on 'EmployeeID'
merged_data = pd.merge(employee_data, performance_data, on='EmployeeID', how='inner')

# Add 'AttritionNumeric' column based on 'Attrition' (Yes=1, No=0)
merged_data['AttritionNumeric'] = merged_data['Attrition'].map({'Yes': 1, 'No': 0})

# 6. Salary Analysis Across Demographics and Departments
print("\nSalary Analysis Across Demographics and Departments:")

# Average salary by department
avg_salary_department = merged_data.groupby('Department')['Salary'].mean()
print("Average Salary by Department:")
print(avg_salary_department)

# ANOVA for salary by department
anova_salary_department = f_oneway(
    *[merged_data.loc[merged_data['Department'] == dept, 'Salary']
      for dept in merged_data['Department'].unique()]
)
print(f"ANOVA Test for Salary by Department: F-statistic = {anova_salary_department.statistic:.2f}, P-value = {anova_salary_department.pvalue:.4f}")

# T-test for salary by gender
male_salary = merged_data.loc[merged_data['Gender'] == 'Male', 'Salary']
female_salary = merged_data.loc[merged_data['Gender'] == 'Female', 'Salary']
t_test_gender_salary = ttest_ind(male_salary, female_salary)
print(f"T-test for Salary by Gender: T-statistic = {t_test_gender_salary.statistic:.2f}, P-value = {t_test_gender_salary.pvalue:.4f}")

# 7. Stock Option Level and Retention (Attrition)
print("\nStock Option Level and Retention (Attrition):")
correlation_stock = merged_data['StockOptionLevel'].corr(merged_data['AttritionNumeric'])
print(f"Correlation between StockOptionLevel and Attrition: {correlation_stock:.2f}")

# 8. Age Impact on Job Satisfaction and Attrition
print("\nAge Impact on Job Satisfaction and Attrition:")

# Job satisfaction by age group
merged_data['AgeGroup'] = pd.cut(merged_data['Age'], bins=[18, 30, 40, 50, 60], labels=['18-30', '31-40', '41-50', '51-60'])
avg_job_satisfaction_age = merged_data.groupby('AgeGroup')['JobSatisfaction'].mean()
print("Average Job Satisfaction by Age Group:")
print(avg_job_satisfaction_age)

```

```python
# Attrition rate by age group
attrition_rate_age = merged_data.groupby('AgeGroup')['AttritionNumeric'].mean()
print("Attrition Rate by Age Group:")
print(attrition_rate_age)

# 9. Gender Differences in Job Satisfaction and Performance
print("\nGender Differences in Job Satisfaction and Performance:")

# Job satisfaction by gender
avg_job_satisfaction_gender = merged_data.groupby('Gender')['JobSatisfaction'].mean()
print("Average Job Satisfaction by Gender:")
print(avg_job_satisfaction_gender)

# Manager rating by gender
avg_manager_rating_gender = merged_data.groupby('Gender')['ManagerRating'].mean()
print("Average Manager Rating by Gender:")
print(avg_manager_rating_gender)

# 10. Education Level and its Impact on Performance and Salary
print("\nEducation Level and its Impact on Performance and Salary:")

# Salary by education level
avg_salary_education = merged_data.groupby('Education')['Salary'].mean()
print("Average Salary by Education Level:")
print(avg_salary_education)

# Manager rating by education level
avg_manager_rating_education = merged_data.groupby('Education')['ManagerRating'].mean()
print("Average Manager Rating by Education Level:")
print(avg_manager_rating_education)

# 11. Marital Status and Attrition
print("\nMarital Status and Attrition:")

# Attrition rate by marital status
attrition_rate_marital = merged_data.groupby('MaritalStatus')['AttritionNumeric'].mean()
print("Attrition Rate by Marital Status:")
print(attrition_rate_marital)

# Job satisfaction by marital status
avg_job_satisfaction_marital = merged_data.groupby('MaritalStatus')['JobSatisfaction'].mean()
print("Job Satisfaction by Marital Status:")
print(avg_job_satisfaction_marital)

# Visualizations
sns.boxplot(x='Department', y='Salary', data=merged_data, palette='Set2')
plt.title("Salary Distribution by Department")
plt.show()

sns.barplot(x='AgeGroup', y='JobSatisfaction', data=merged_data, palette='viridis')
plt.title("Job Satisfaction by Age Group")
plt.show()
```

```
...
    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
```


Salary Distribution by Department
Salary
500000
400000
300000
200000
100000
0
Sales
Human Resources
Technology
Department


Job Satisfaction by Age Group
JobSatisfaction
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
18-30
31-40
41-50
51-60
AgeGroup

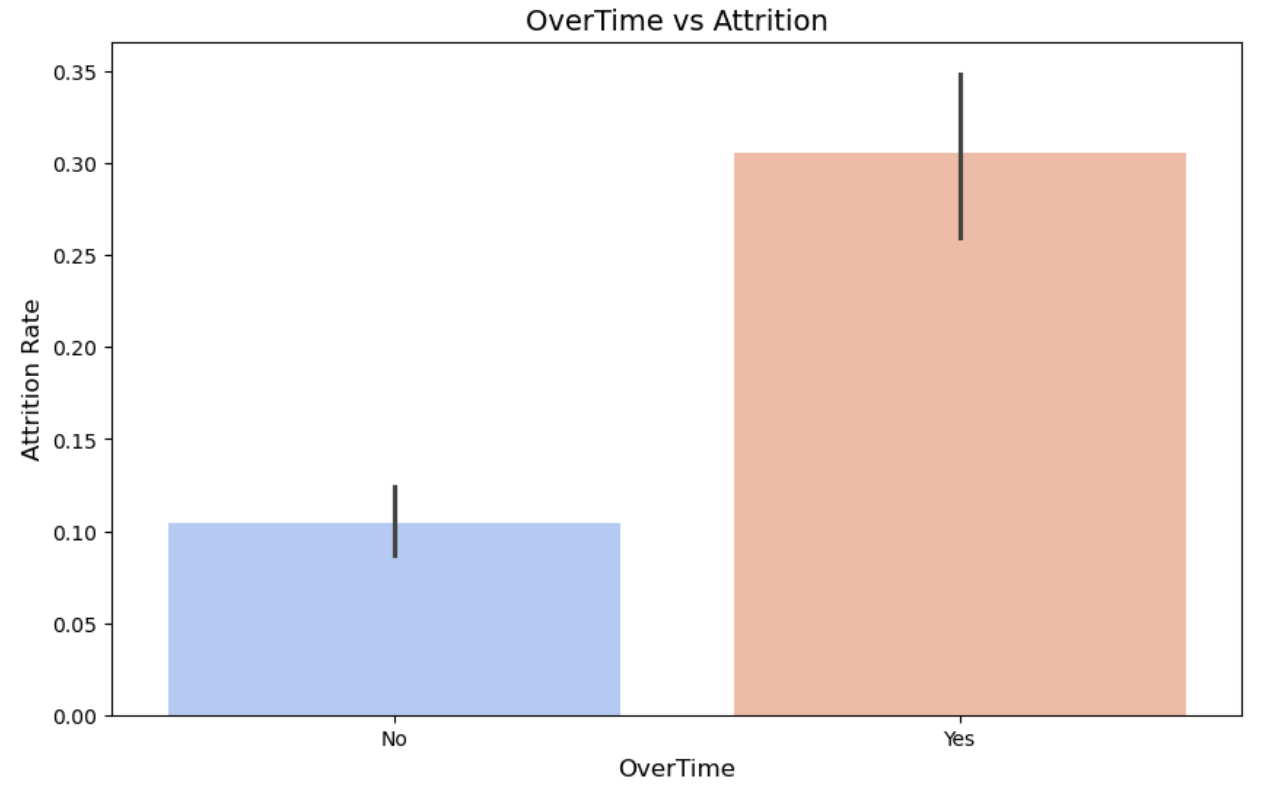
2. Geographic Insights:

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?*

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

# Load the 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)

# Check columns
if 'OverTime' in employee_data.columns and 'Attrition' in employee_data.columns:
    # Convert OverTime and Attrition to numeric
    employee_data['OverTimeNumeric'] = employee_data['OverTime'].map({'Yes': 1, 'No': 0})
    employee_data['AttritionNumeric'] = employee_data['Attrition'].map({'Yes': 1, 'No': 0})

    # Correlation between OverTime and Attrition
    correlation_attrition = employee_data['OverTimeNumeric'].corr(employee_data['AttritionNumeric'])
    print(f"Correlation between OverTime and Attrition: {correlation_attrition:.2f}")

    # Visualization for Attrition
    plt.figure(figsize=(10, 6))
    sns.barplot(x='OverTime', y='AttritionNumeric', data=employee_data, palette='coolwarm')
    plt.title('OverTime vs Attrition', fontsize=14)
    plt.xlabel('OverTime', fontsize=12)
    plt.ylabel('Attrition Rate', fontsize=12)
    plt.show()

    # Analyze correlation with performance if available
    if 'Performance' in employee_data.columns:
        correlation_performance = employee_data['OverTimeNumeric'].corr(employee_data['Performance'])
        print(f"Correlation between OverTime and Performance: {correlation_performance:.2f}")
else:
    print("Required columns 'OverTime' or 'Attrition' not found in the dataset.")

```

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


OverTime vs Attrition
Attrition Rate
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
No
Yes
OverTime

```python
# Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import pearsonr
from scipy.stats import ttest_ind
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load data from both sheets
df_employee = pd.read_excel('Data.xlsx', sheet_name='Employee')
df_performance = pd.read_excel('Data.xlsx', sheet_name='PerformanceRating')

# Merge the two dataframes on EmployeeID
df = pd.merge(df_employee, df_performance, on='EmployeeID', how='inner')

print(df.head())
# Check for missing values in the dataset
print(df.isnull().sum())
# Display basic information about the dataset (column types, memory usage, etc.)
print(df.info())


# Preprocessing categorical columns for analysis
# Convert 'OverTime', 'Attrition', and 'BusinessTravel' to numeric
df['OverTime'] = df['OverTime'].map({'Yes': 1, 'No': 0})
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})
label_enc = LabelEncoder()
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
...
 33  DataQualityStatus               6709 non-null   object
dtypes: datetime64[ns](1), int64(17), object(16)
memory usage: 1.7+ MB
None
```

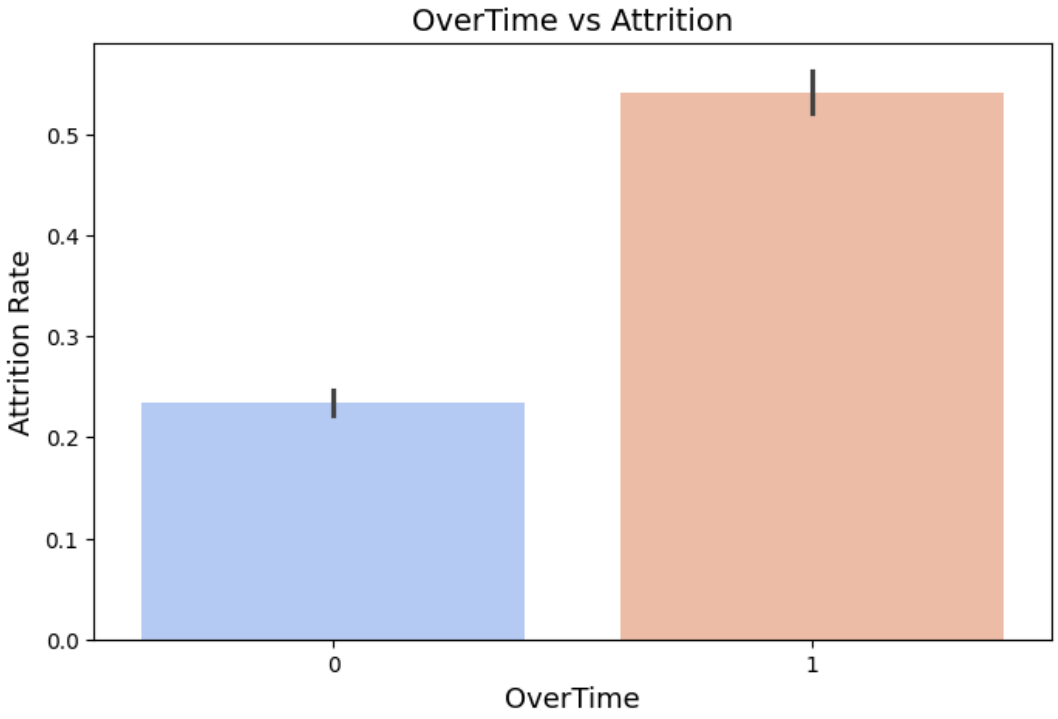
2.Geographic Insights:

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

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

```python
# Calculate correlation between OverTime and Attrition
correlation = df[['OverTime', 'Attrition']].corr()
print(f"Correlation between OverTime and Attrition:\n{correlation}")


# Visualization
plt.figure(figsize=(8, 5))
sns.barplot(x='OverTime', y='Attrition', data=df, palette='coolwarm')
plt.title('OverTime vs Attrition', fontsize=14)
plt.xlabel('OverTime', fontsize=13)
plt.ylabel('Attrition Rate', fontsize=13)
plt.show()
```

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

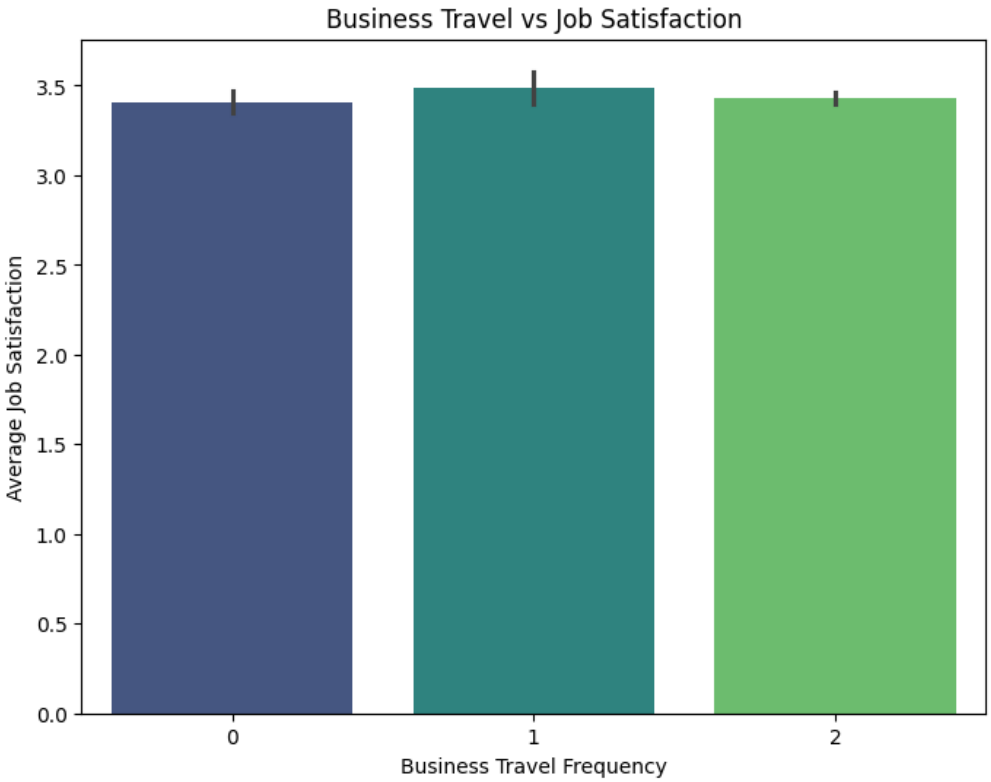


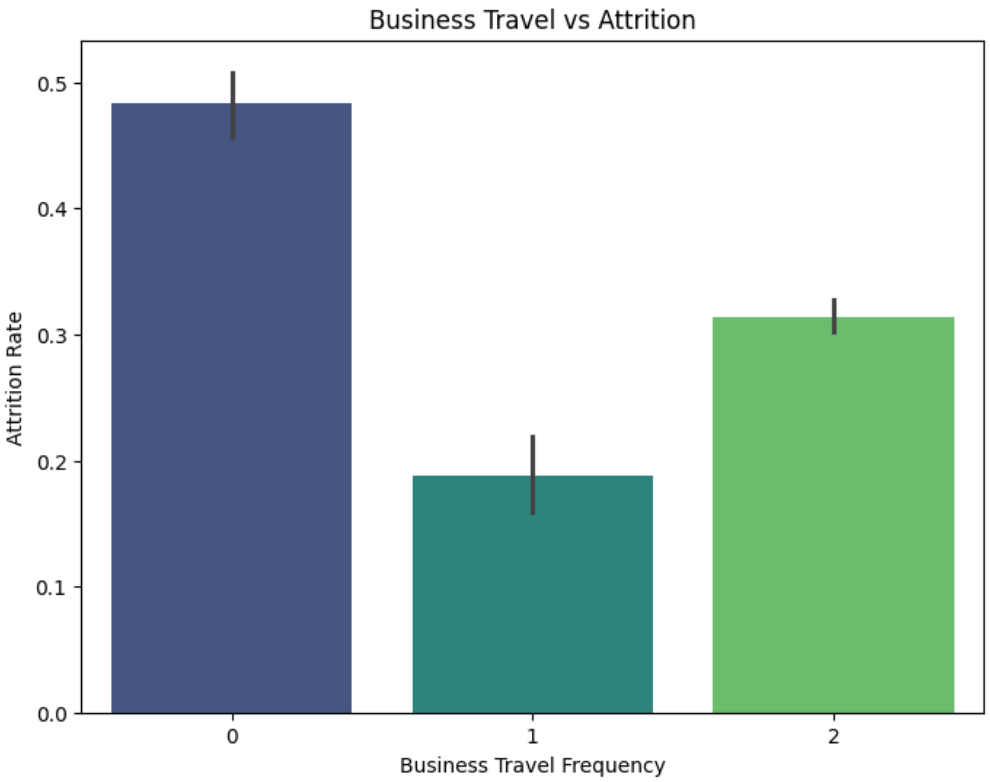
19.How does business travel affect employee satisfaction and retention?

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

```python
# Average Job Satisfaction and Attrition by Business Travel category
travel_satisfaction = df.groupby('BusinessTravel')[['JobSatisfaction', 'Attrition']].mean()
print(f"Business Travel and Job Satisfaction:\n{travel_satisfaction}")

# Visualization: Business Travel vs Job Satisfaction
plt.figure(figsize=(8, 6))
sns.barplot(x='BusinessTravel', y='JobSatisfaction', data=df, palette='viridis')
plt.title('Business Travel vs Job Satisfaction')
plt.xlabel('Business Travel Frequency')
plt.ylabel('Average Job Satisfaction')
plt.show()

# Visualization: Business Travel vs Attrition
plt.figure(figsize=(8, 6))
sns.barplot(x='BusinessTravel', y='Attrition', data=df, palette='viridis')
plt.title('Business Travel vs Attrition')
plt.xlabel('Business Travel Frequency')
plt.ylabel('Attrition Rate')
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
Average Job Satisfaction
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
Business Travel Frequency

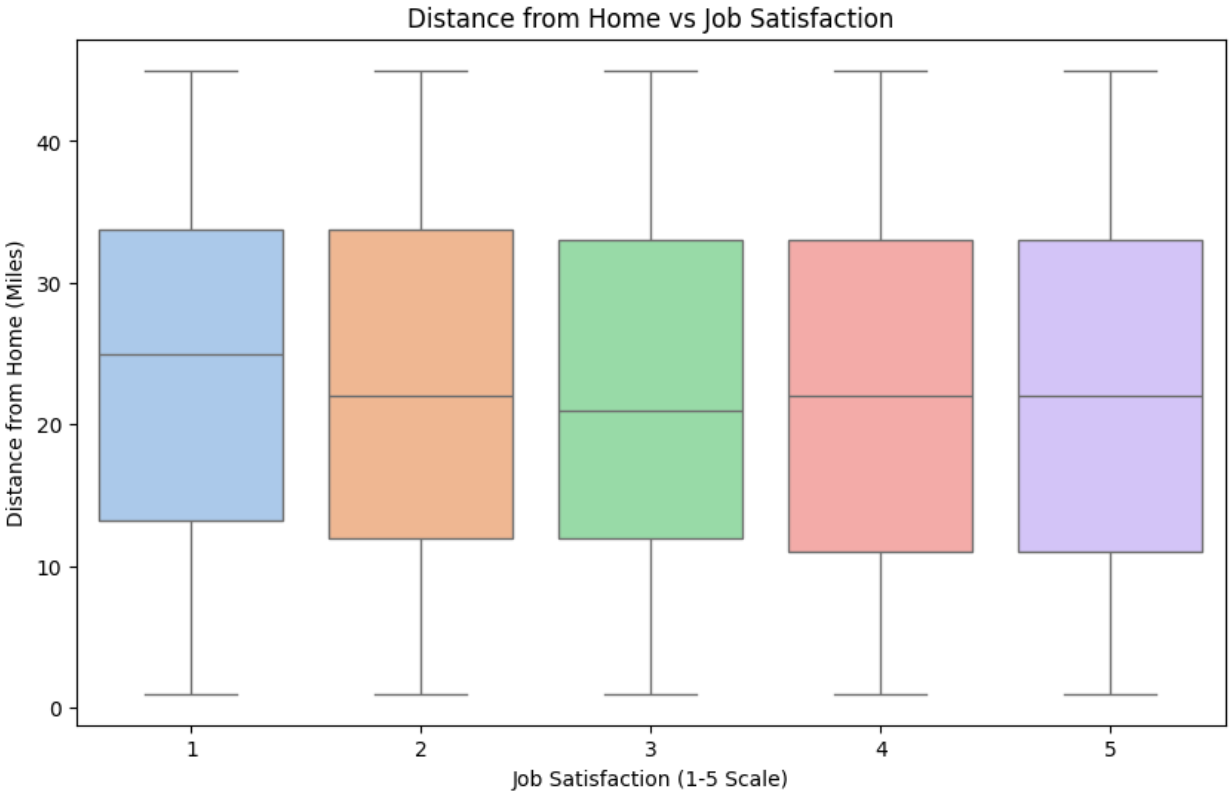

Business Travel vs Attrition
Attrition Rate
0.0
0.1
0.2
0.3
0.4
0.5
0
1
2
Business Travel Frequency

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.

```python
# Correlation between DistanceFromHome and Job Satisfaction
correlation_distance = df[['DistanceFromHome', 'JobSatisfaction']].corr()
print(f"Correlation between Distance from Home and Job Satisfaction:\n{correlation_distance}")

# Assuming 'JobSatisfaction' is a column in the dataset
plt.figure(figsize=(10, 6))
sns.boxplot(x='JobSatisfaction', y='DistanceFromHome', data=df, palette='pastel')
plt.title('Distance from Home vs Job Satisfaction')
plt.xlabel('Job Satisfaction (1-5 Scale)')
plt.ylabel('Distance from Home (Miles)')
plt.show()
```

```
Correlation between Distance from Home and Job Satisfaction:
                  DistanceFromHome  JobSatisfaction
DistanceFromHome          1.000000        -0.011745
JobSatisfaction          -0.011745         1.000000
```

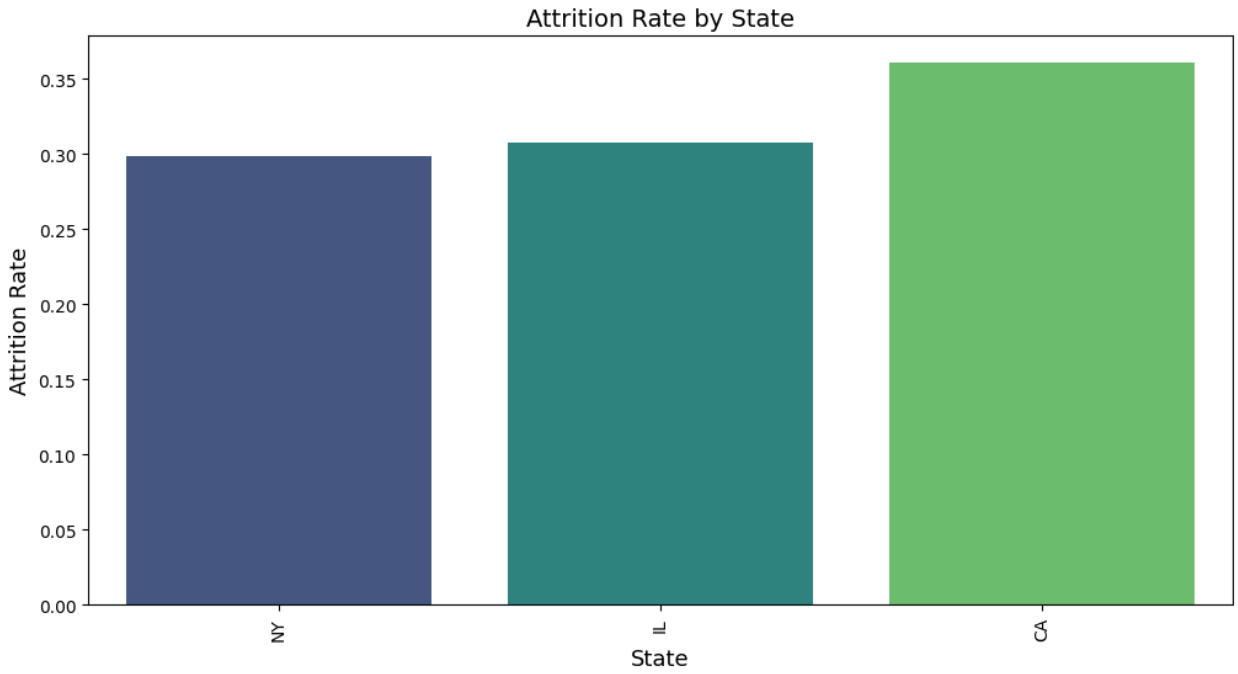


21.Which states have the highest and lowest attrition rates?

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

```python
# Group by State and calculate the attrition rate
state_attrition = df.groupby('State')['Attrition'].mean().sort_values()

# Print the attrition rates by state
print(f"Attrition Rate by State:\n{state_attrition}")

# Visualization
plt.figure(figsize=(12, 6))
sns.barplot(x=state_attrition.index, y=state_attrition.values, palette='viridis')
plt.title('Attrition Rate by State', fontsize=14)
plt.xlabel('State', fontsize=13)
plt.ylabel('Attrition Rate', fontsize=13)
plt.xticks(rotation=90)
plt.show()
```

```
Attrition Rate by State:
State
NY    0.298378
IL    0.307407
CA    0.360583
```

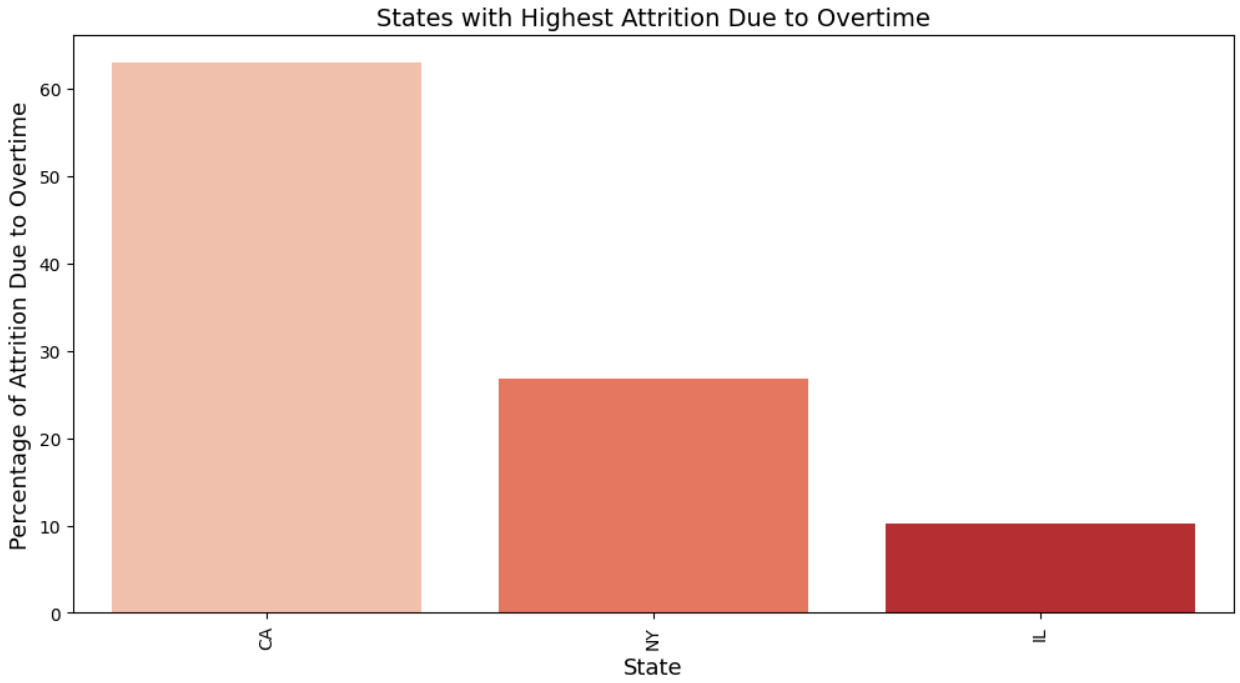


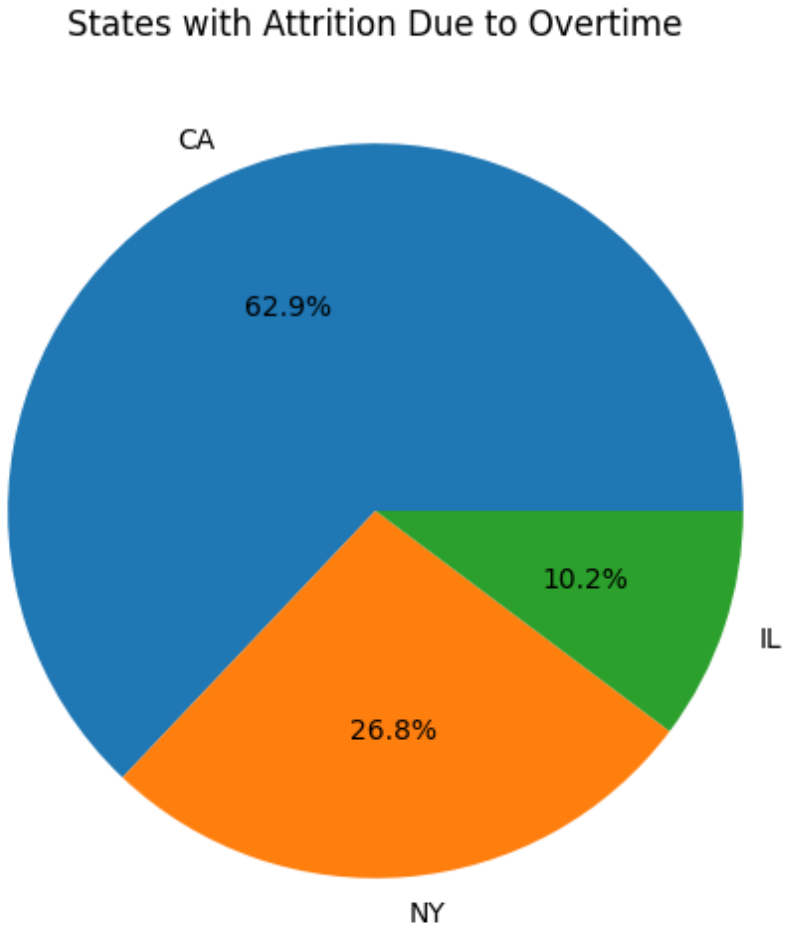
22.Which states have the highest percentage of employees leaving due to excessive overtime?

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

```python
# Filter for employees who work overtime and calculate attrition rate by state
overtime_attrition = df[df['OverTime'] == 1].groupby('State')['Attrition'].mean().sort_values()

print(f"Overtime Attrition Rate by State:\n{overtime_attrition}")

# Filter employees who left and worked overtime
overtime_attrition = df[(df['Attrition'] == 1) & (df['OverTime'] == 1)]

# Calculate percentage of attrition due to overtime per state
state_counts = (overtime_attrition['State'].value_counts(normalize=True) * 100).sort_values(ascending=False)

# Visualization - Bar Plot
plt.figure(figsize=(12, 6))
sns.barplot(x=state_counts.index, y=state_counts.values, palette='Reds')
plt.title('States with Highest Attrition Due to Overtime', fontsize=14)
plt.xlabel('State', fontsize=13)
plt.ylabel('Percentage of Attrition Due to Overtime', fontsize=13)
plt.xticks(rotation=90)
plt.show()

# Visualization -Pie Chart
plt.figure(figsize=(10, 6))
state_counts.plot(kind='pie', autopct='%1.1f%%')
plt.title('States with Attrition Due to Overtime')
plt.ylabel('')
plt.show()
```

```
Overtime Attrition Rate by State:
State
NY    0.507812
CA    0.554585
IL    0.556054
```

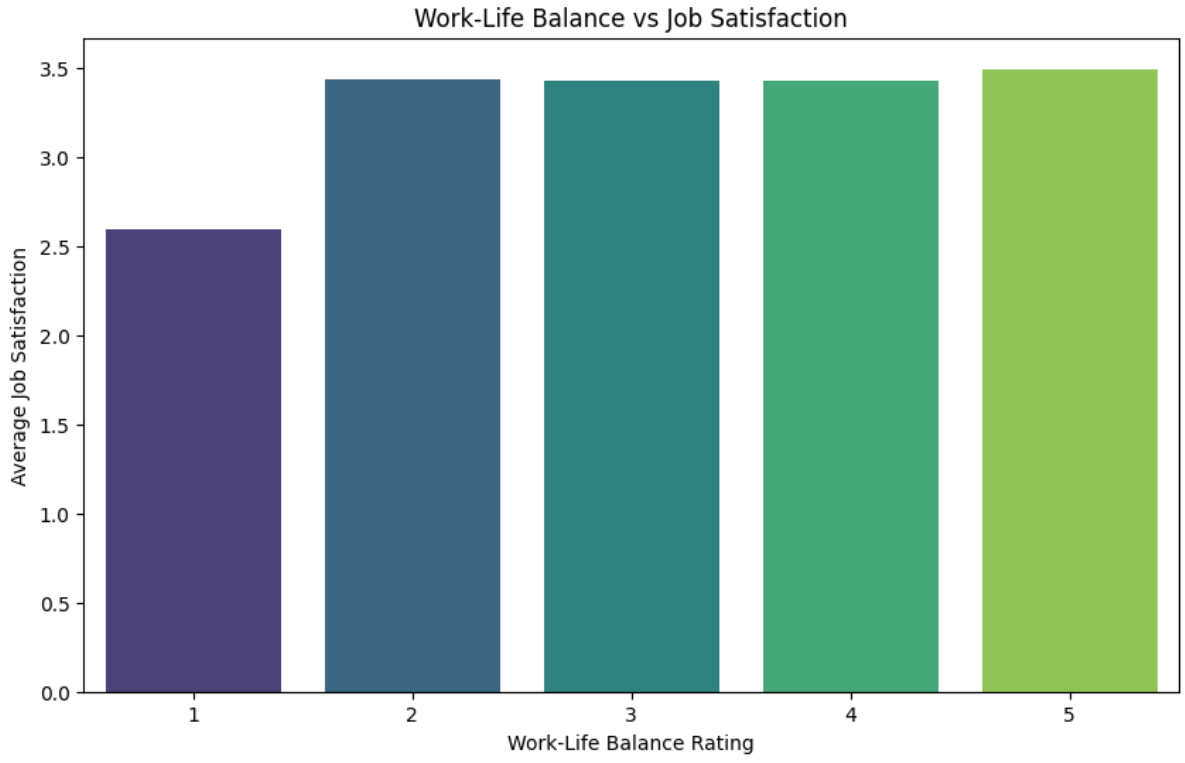

States with Highest Attrition Due to Overtime
Percentage of Attrition Due to Overtime
0
10
20
30
40
50
60
CA
NY
IL
State


States with Attrition Due to Overtime
CA
62.9%
10.2%
IL
26.8%
NY

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.

```python

# Calculate correlation between DistanceFromHome and ManagerRating
distance_performance_correlation = df[['DistanceFromHome', 'ManagerRating']].corr()
print(f"Correlation between Distance from Home and Manager Rating:\n{distance_performance_correlation}")

# Assuming 'WorkLifeBalance' and 'JobSatisfaction' are ordinal columns (1-5)
plt.figure(figsize=(10, 6))
sns.barplot(x='WorkLifeBalance', y='JobSatisfaction', data=df, ci=None, palette='viridis')
plt.title('Work-Life Balance vs Job Satisfaction')
plt.xlabel('Work-Life Balance Rating')
plt.ylabel('Average Job Satisfaction')
plt.show()
```



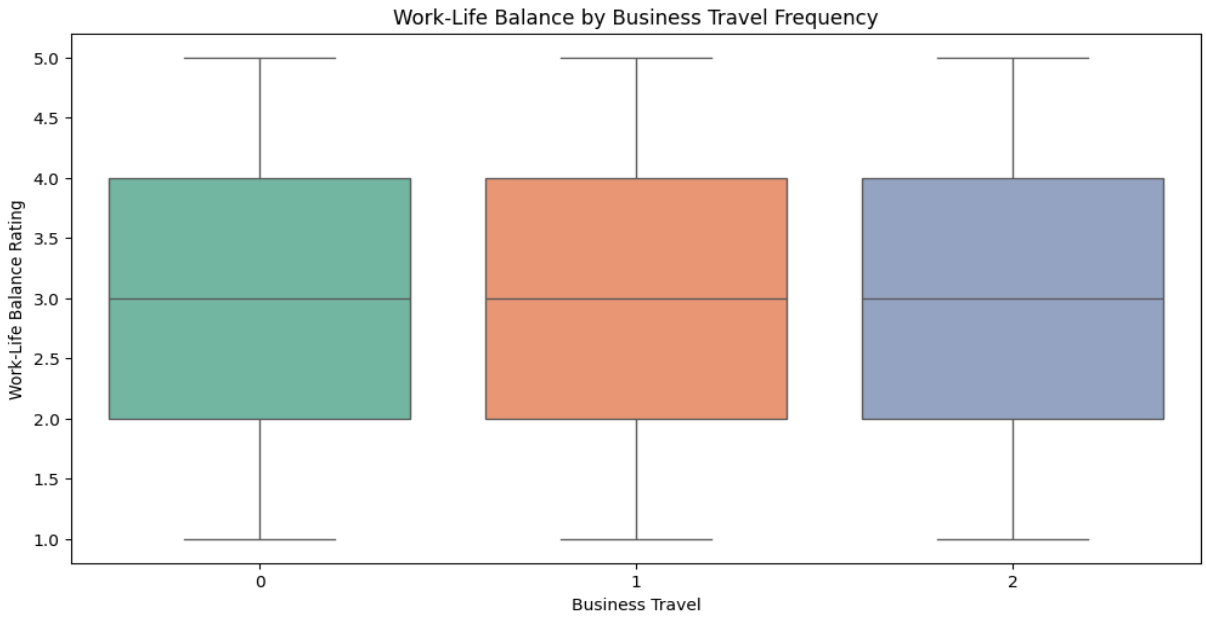

```
Correlation between Distance from Home and Manager Rating:
                  DistanceFromHome  ManagerRating
DistanceFromHome          1.000000       0.025832
ManagerRating             0.025832       1.000000
```

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.

```python
# Calculate the average Work-Life Balance rating for each Business Travel frequency
average_work_life_balance = df.groupby('BusinessTravel')['WorkLifeBalance'].mean().reset_index()

# Display the average Work-Life Balance values
print("Average Work-Life Balance by Business Travel Frequency:")
print(average_work_life_balance)

# Visualize the relationship between Business Travel frequency and Work-Life Balance ratings
plt.figure(figsize=(12, 6))
sns.boxplot(x='BusinessTravel', y='WorkLifeBalance', data=df, palette='Set2')
plt.title('Work-Life Balance by Business Travel Frequency')
plt.xlabel('Business Travel')
plt.ylabel('Work-Life Balance Rating')
plt.show()
```

```
Average Work-Life Balance by Business Travel Frequency:
   BusinessTravel  WorkLifeBalance
0               0         3.400589
1               1         3.402619
2               2         3.420253
```

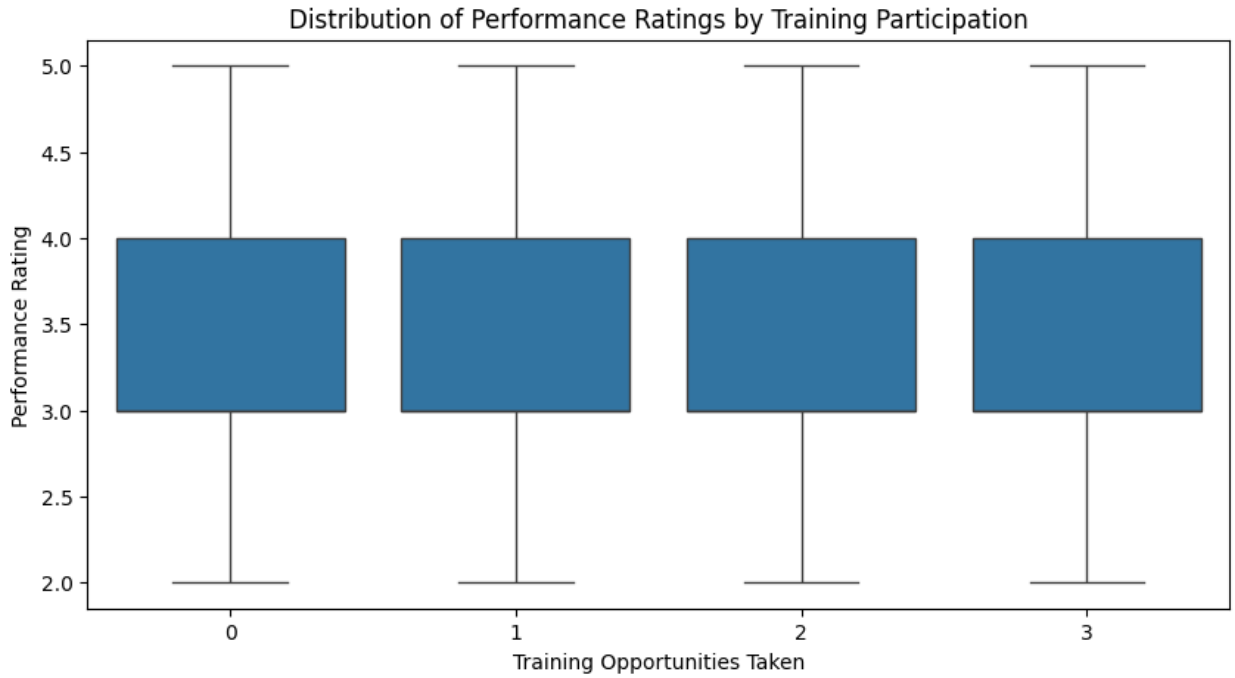


25.How does participation in training programs impact employee performance?

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

```python
# Statistics
corr, p_value = stats.pearsonr(
    df['TrainingOpportunitiesTaken'],
    df['ManagerRating']
)
print(f'''
[Training vs Performance]
Correlation Coefficient: {corr:.3f}
P-value: {p_value:.4f}
Significant at 95% confidence: {'Yes' if p_value < 0.05 else 'No'}
''')

# Visualization
plt.figure(figsize=(10, 5))
sns.boxplot(x='TrainingOpportunitiesTaken', y='ManagerRating', data=df)
plt.title("Distribution of Performance Ratings by Training Participation")
plt.xlabel("Training Opportunities Taken")
plt.ylabel("Performance Rating")
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
Performance Rating
5.0
4.5
4.0
3.5
3.0
2.5
2.0
0
1
2
3
Training Opportunities Taken

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`.

```python
# Calculate average Job Satisfaction and Years Since Last Promotion based on Training Opportunities Taken
training_analysis = df.groupby('TrainingOpportunitiesTaken').agg({
    'JobSatisfaction': 'mean',
    'YearsSinceLastPromotion': 'mean'
}).reset_index()

print("Average Job Satisfaction and Years Since Last Promotion based on Training Opportunities Taken:")
print(training_analysis)


# Visualization
sns.barplot(x='TrainingOpportunitiesTaken', y='JobSatisfaction', data=df, ci=None)
plt.title('Training vs Job Satisfaction')
plt.xlabel('Training Opportunities Taken')
plt.ylabel('Average Job Satisfaction')
plt.show()

plt.figure(figsize=(10, 6))
sns.barplot(x='TrainingOpportunitiesTaken', y='YearsSinceLastPromotion', data=df, ci=None)
plt.title('Training vs Time Since Last Promotion')
plt.xlabel('Training Opportunities Taken')
plt.ylabel('Years Since Last Promotion')
plt.show()
```

```
Average Job Satisfaction and Years Since Last Promotion based on Training Opportunities Taken:
   TrainingOpportunitiesTaken  JobSatisfaction  YearsSinceLastPromotion
0                           0         3.440135                 4.291315
1                           1         3.425171                 4.253668
2                           2         3.430712                 4.203184
3                           3         3.400990                 4.094059
```


Training vs Job Satisfaction
Average Job Satisfaction
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3
Training Opportunities Taken


Training vs Time Since Last Promotion
Years Since Last Promotion
0
1
2
3
4
0
1
2
3
Training Opportunities Taken

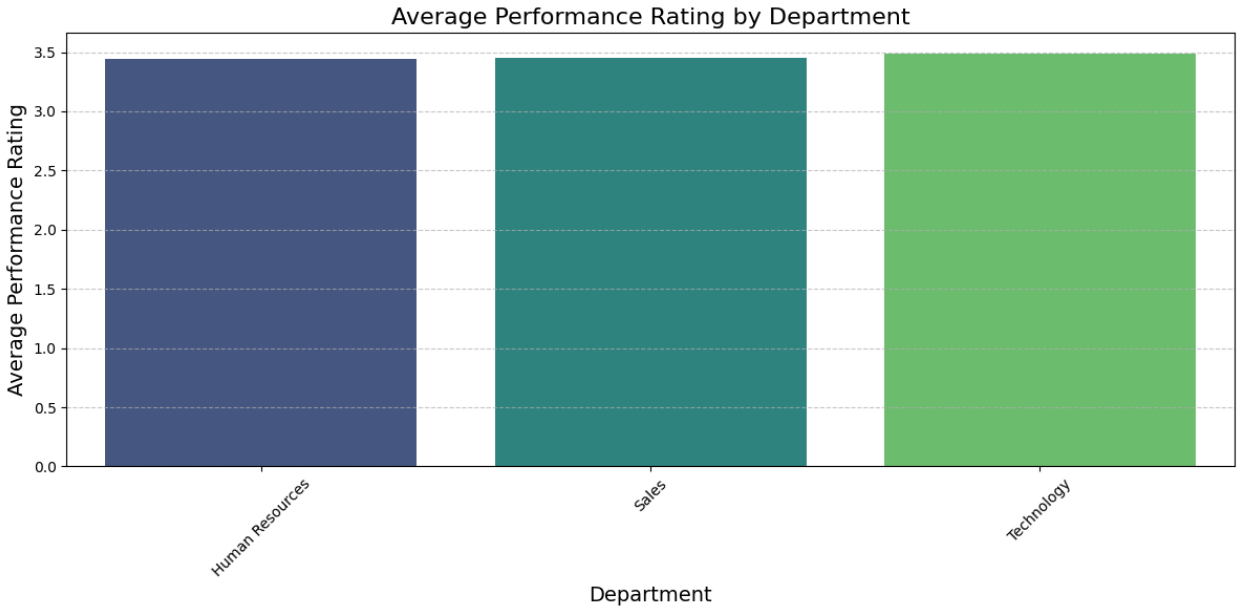
4. Behavioral Insights:

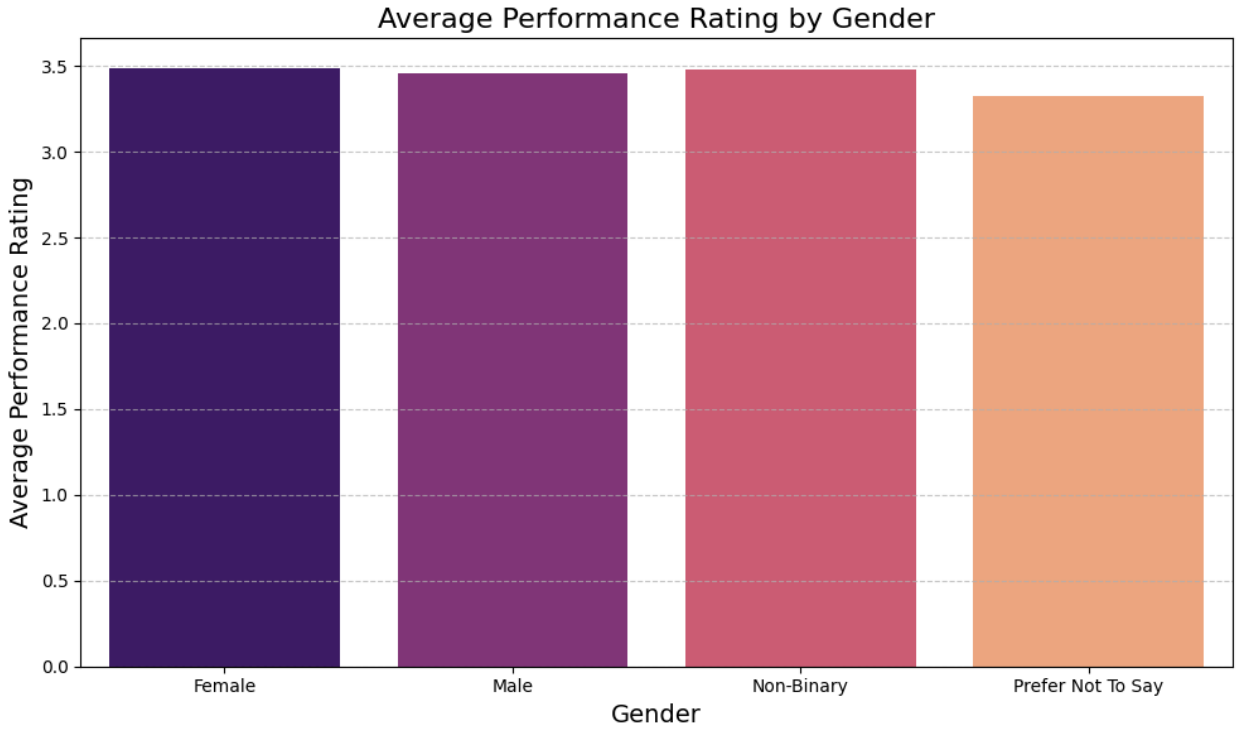
27.What are the trends in performance ratings across departments and demographics?

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

```python
# Performance ratings by department
dept_performance = df.groupby('Department')['ManagerRating'].mean().reset_index()
print(f"Performance Rating by Department:\n{dept_performance}")

# Performance ratings by gender
gender_performance = df.groupby('Gender')['ManagerRating'].mean().reset_index()
print(f"Performance Rating by Gender:\n{gender_performance}")

# Visualization

# Performance Ratings by Department
plt.figure(figsize=(12, 6))
sns.barplot(x='Department', y='ManagerRating', data=dept_performance, palette='viridis')
plt.title('Average Performance Rating by Department', fontsize=16)
plt.xlabel('Department', fontsize=14)
plt.ylabel('Average Performance Rating', fontsize=14)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Adding grid for better readability
plt.tight_layout()
plt.show()

# Performance Ratings by Gender
plt.figure(figsize=(10, 6))
sns.barplot(x='Gender', y='ManagerRating', data=gender_performance, palette='magma')
plt.title('Average Performance Rating by Gender', fontsize=16)
plt.xlabel('Gender', fontsize=14)
plt.ylabel('Average Performance Rating', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Adding grid for better readability
plt.tight_layout()
plt.show()

```

```
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
```


Average Performance Rating by Department
Average Performance Rating
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Human Resources
Sales
Technology
Department

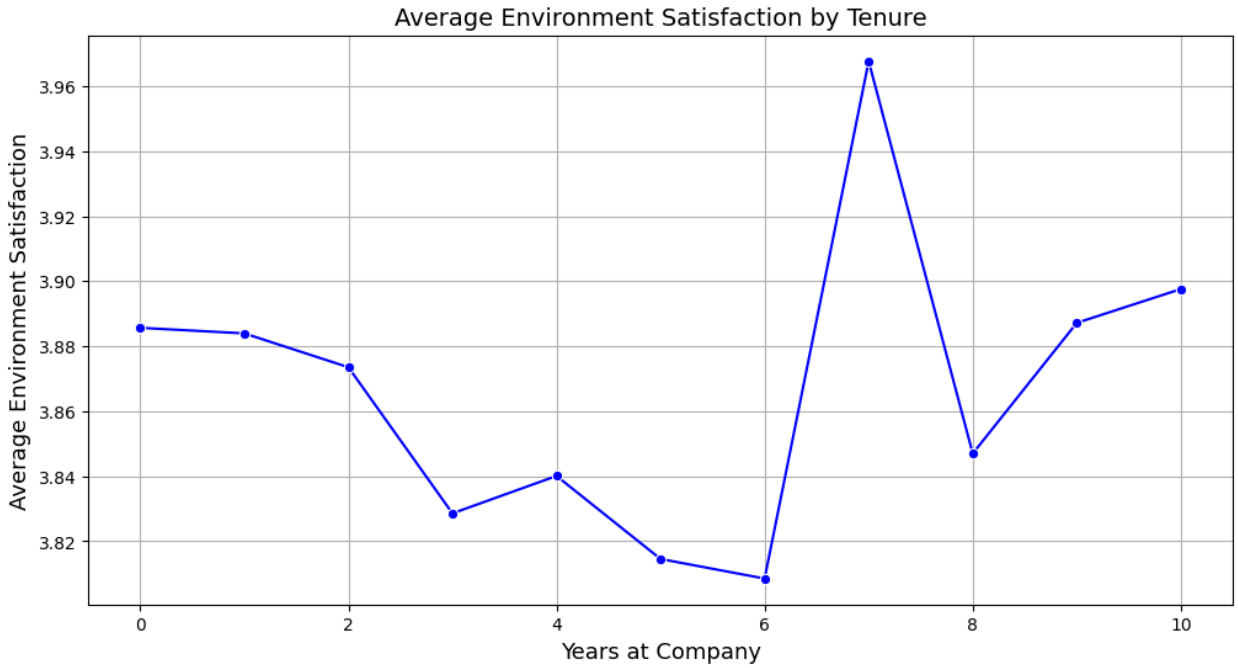

Average Performance Rating by Gender
Average Performance Rating
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Female
Male
Non-Binary
Prefer Not To Say
Gender

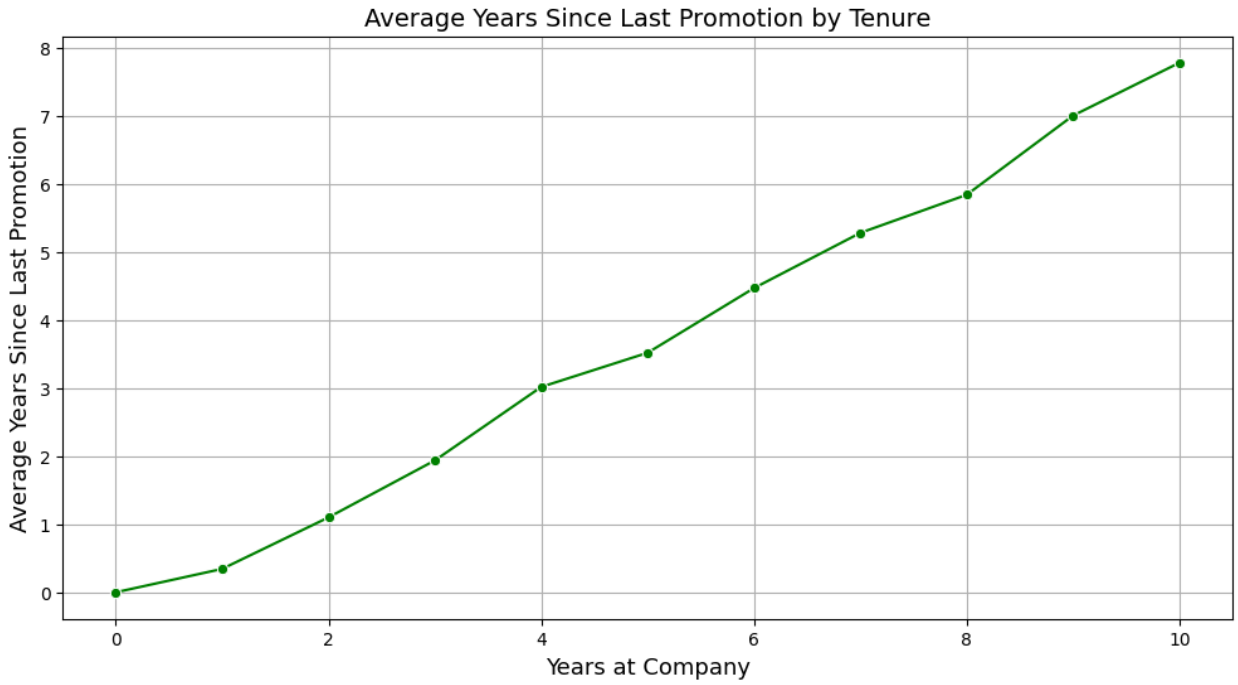
28.How does tenure (YearsAtCompany) impact performance and promotion rates?

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

```python
# Group by 'YearsAtCompany' and calculate average performance rating and average years since last promotion
tenure_performance = df.groupby('YearsAtCompany')['EnvironmentSatisfaction'].mean()  # Use 'EnvironmentSatisfaction' as proxy for performance
tenure_promotion = df.groupby('YearsAtCompany')['YearsSinceLastPromotion'].mean()

# Print results
print(f"Average Environment Satisfaction by Tenure (YearsAtCompany):\n{tenure_performance}")
print(f"Average Years Since Last Promotion by Tenure (YearsAtCompany):\n{tenure_promotion}")

# Visualization for performance rating vs. tenure
plt.figure(figsize=(12, 6))
sns.lineplot(x=tenure_performance.index, y=tenure_performance.values, marker='o', color='blue')
plt.title('Average Environment Satisfaction by Tenure', fontsize=14)
plt.xlabel('Years at Company', fontsize=13)
plt.ylabel('Average Environment Satisfaction', fontsize=13)
plt.grid(True)
plt.show()

# Visualization for years since last promotion vs. tenure
plt.figure(figsize=(12, 6))
sns.lineplot(x=tenure_promotion.index, y=tenure_promotion.values, marker='o', color='green')
plt.title('Average Years Since Last Promotion by Tenure', fontsize=14)
plt.xlabel('Years at Company', fontsize=13)
plt.ylabel('Average Years Since Last Promotion', fontsize=13)
plt.grid(True)
plt.show()
```

```
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
```



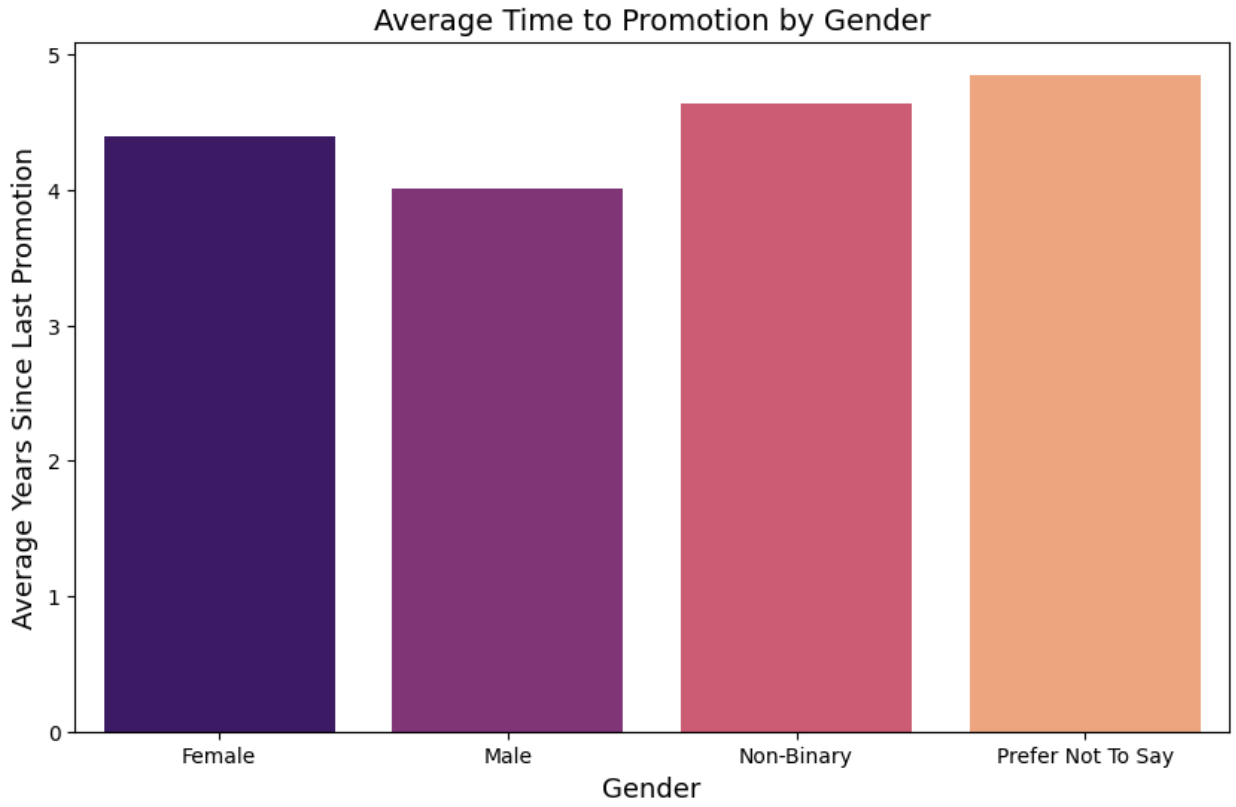

Average Years Since Last Promotion by Tenure
Average Years Since Last Promotion
Years at Company
0
1
2
3
4
5
6
7
8
0
2
4
6
8
10

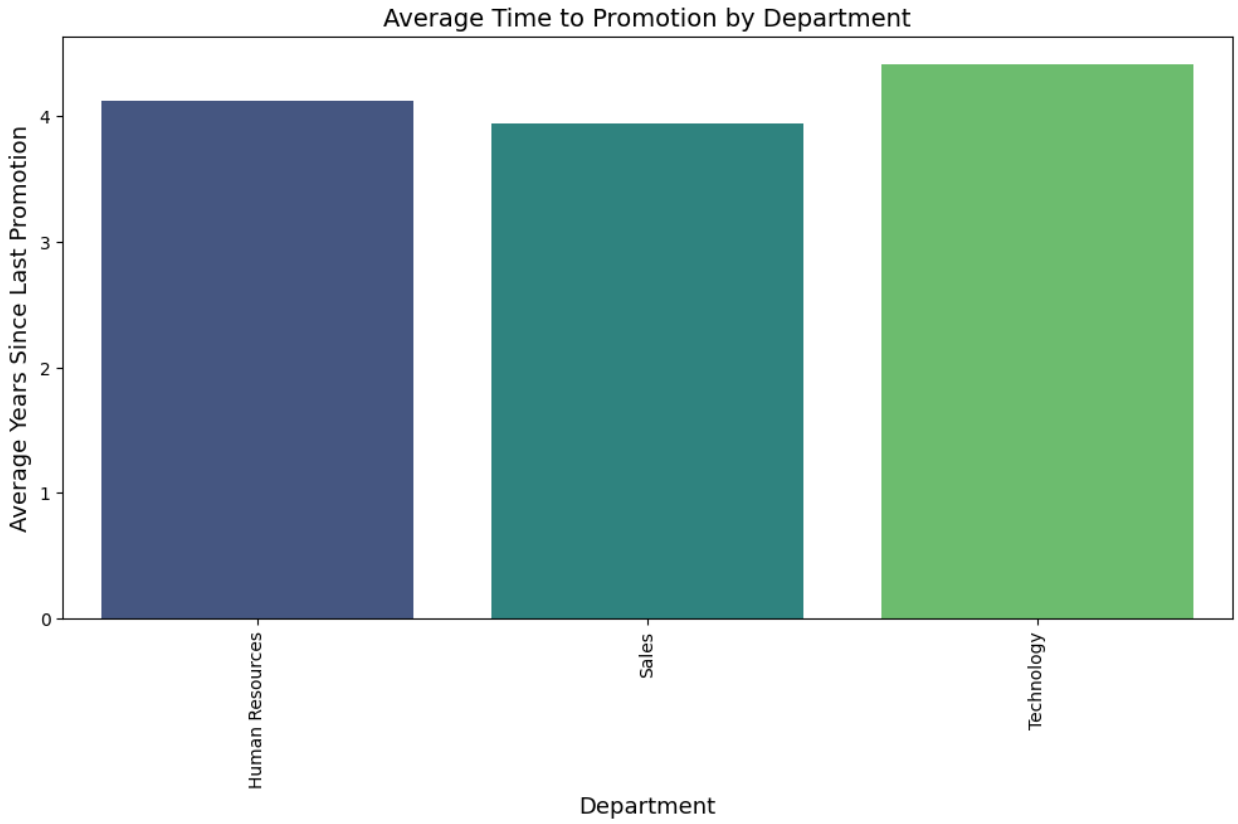
29.What is the average time to promotion for employees, and does it vary by department or gender?

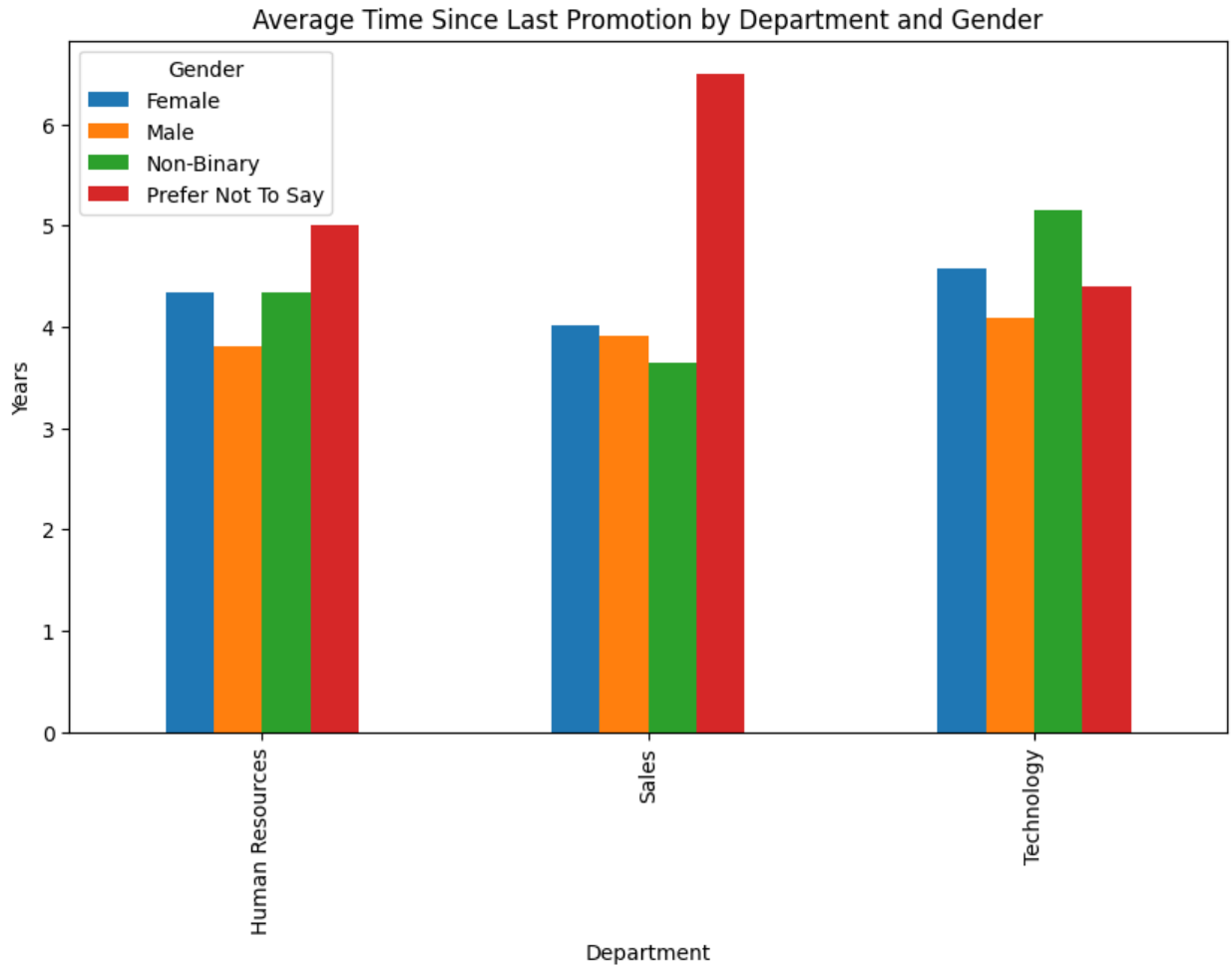
<u>We'll calculate the average `YearsSinceLastPromotion` and group the data by `Department` and `Gender`.</u>

```python
# Calculate average years since last promotion
promotion_time = df.groupby(['Department', 'Gender'])['YearsSinceLastPromotion'].mean().unstack()
print("Average Time Since Last Promotion:\n", promotion_time)

# Visualization for average time to promotion by department
plt.figure(figsize=(12, 6))
sns.barplot(x=dept_promotion_time.index, y=dept_promotion_time.values, palette='viridis')
plt.title('Average Time to Promotion by Department', fontsize=14)
plt.xlabel('Department', fontsize=13)
plt.ylabel('Average Years Since Last Promotion', fontsize=13)
plt.xticks(rotation=90)
plt.show()

# Visualization for average time to promotion by gender
plt.figure(figsize=(10, 6))
sns.barplot(x=gender_promotion_time.index, y=gender_promotion_time.values, palette='magma')
plt.title('Average Time to Promotion by Gender', fontsize=14)
plt.xlabel('Gender', fontsize=13)
plt.ylabel('Average Years Since Last Promotion', fontsize=13)
plt.show()


# Visualization
promotion_time.plot(kind='bar', figsize=(10, 6))
plt.title('Average Time Since Last Promotion by Department and Gender')
plt.ylabel('Years')
plt.show()
```

```
Average Time Since Last Promotion:
 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
```


Average Time to Promotion by Gender
Average Years Since Last Promotion
0
1
2
3
4
5
Female
Male
Non-Binary
Prefer Not To Say
Gender


Average Time to Promotion by Department
Average Years Since Last Promotion
0
1
2
3
4
Human Resources
Sales
Technology
Department

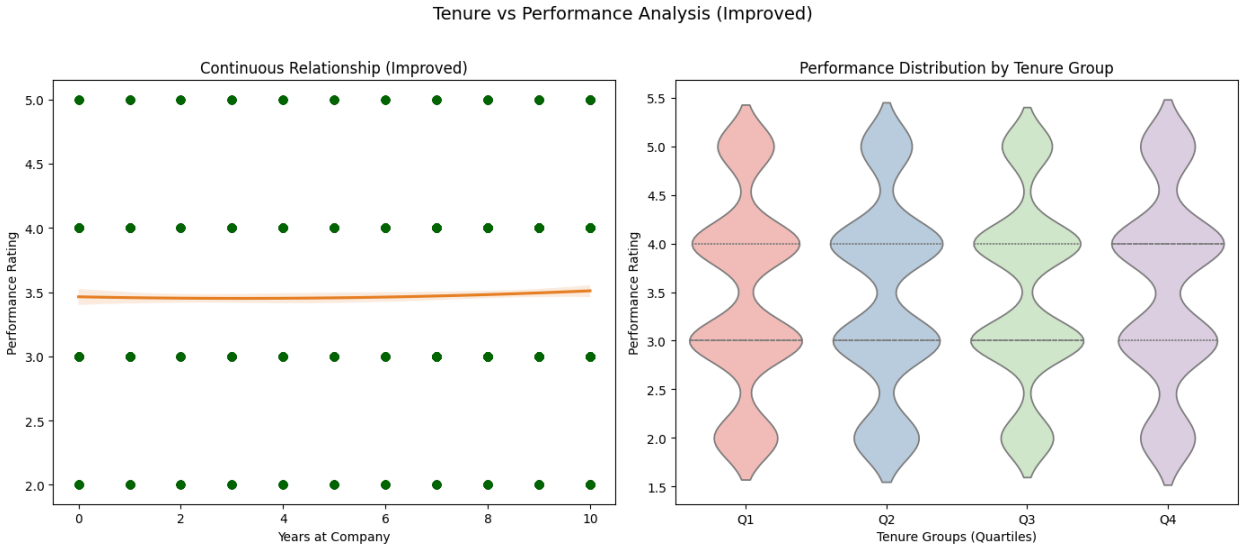


30.Are employees with longer tenure more likely to have higher performance ratings?

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

```python
# --- Statistical Analysis ---
# Compute Pearson correlation
corr, p_value = stats.pearsonr(df['YearsAtCompany'], df['ManagerRating'])
print(f'\n[Tenure vs Performance Analysis]')
print(f'Correlation Coefficient: {corr:.3f}')
print(f'P-value: {p_value:.4f}')
print(f'Statistically Significant at 95% Confidence: {'Yes' if p_value < 0.05 else 'No'}')



# --- Visualization ---
plt.figure(figsize=(14, 6))
# --- Scatter Plot with Regression (Left Plot) ---
plt.subplot(1, 2, 1)
sns.regplot(
    x='YearsAtCompany',
    y='ManagerRating',
    data=df,
    scatter_kws={'alpha':0.4, 'color':'darkgreen'},  # Make points semi-transparent
    line_kws={'color':'#e67e22'},  # Regression line color
    order=2  # Use polynomial regression of degree 2 for better fit
)
plt.title('Continuous Relationship (Improved)', fontsize=12)
plt.xlabel('Years at Company', fontsize=10)
plt.ylabel('Performance Rating', fontsize=10)

# --- Violin Plot (Right Plot) ---
plt.subplot(1, 2, 2)
# Create tenure quartiles
df['TenureGroup'] = pd.qcut(df['YearsAtCompany'], q=4, labels=['Q1', 'Q2', 'Q3', 'Q4'])

sns.violinplot(
    x='TenureGroup',
    y='ManagerRating',
    data=df,
    palette='Pastel1',
    inner='quartile'  # Show quartile lines inside the violin
)
plt.title('Performance Distribution by Tenure Group', fontsize=12)
plt.xlabel('Tenure Groups (Quartiles)', fontsize=10)
plt.ylabel('Performance Rating', fontsize=10)

# Set overall title
plt.suptitle('Tenure vs Performance Analysis (Improved)', fontsize=14, y=1.02)
plt.tight_layout()
plt.show()

```

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

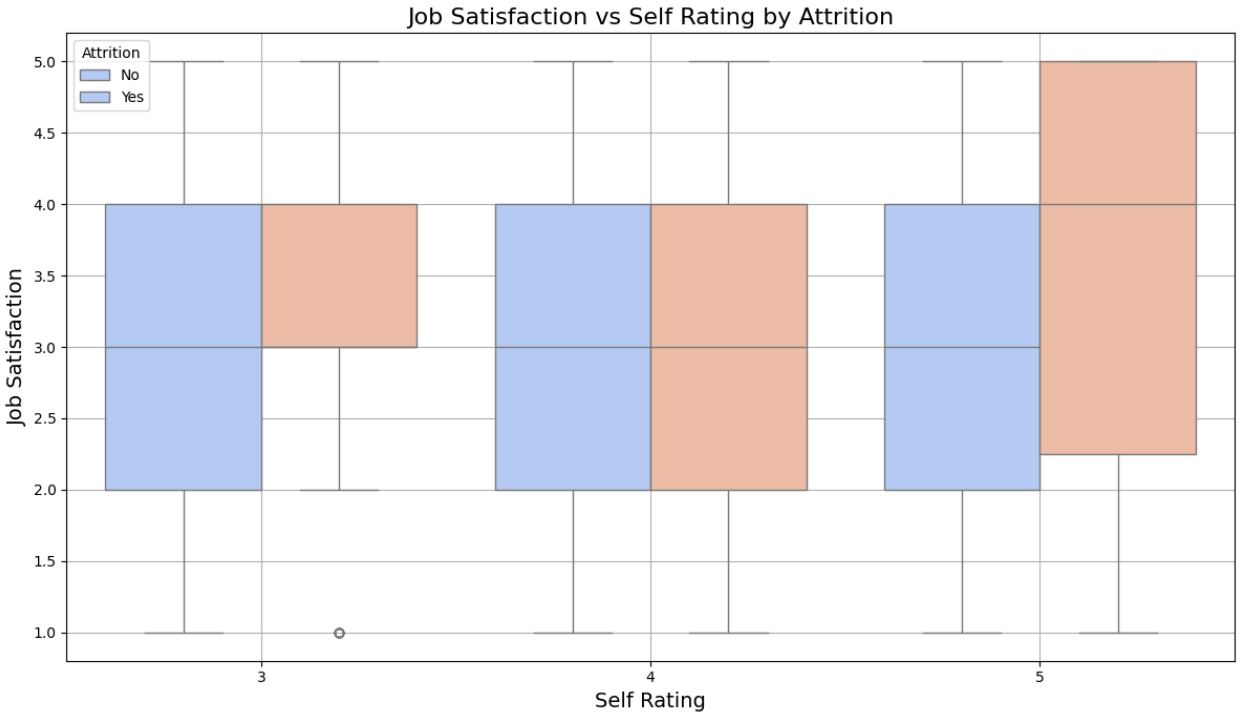


31.How does performance rating correlate with job satisfaction?

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.

```python

# Calculate the average job satisfaction based on self-rating
avg_job_satisfaction = df.groupby('SelfRating')['JobSatisfaction'].mean()
print("Average Job Satisfaction by Self Rating:")
print(avg_job_satisfaction)

# Create a boxplot for job satisfaction based on self-rating
plt.figure(figsize=(12, 7))
ax = sns.boxplot(
    x='SelfRating',
    y='JobSatisfaction',
    hue='Attrition',  # Add attrition information to the plot
    data=df,
    palette='coolwarm'
)

# Customize the appearance of the boxplot using matplotlib
for patch in ax.artists:
    patch.set_facecolor('lightblue')  # Change the box color
    patch.set_edgecolor('black')  # Box edge color
    patch.set_linewidth(1.5)  # Edge line width

plt.title('Job Satisfaction vs Self Rating by Attrition', fontsize=16)
plt.xlabel('Self Rating', fontsize=14)
plt.ylabel('Job Satisfaction', fontsize=14)
plt.grid(True)  # Add gridlines for better visualization

# Add a legend to improve understanding
plt.legend(title='Attrition', loc='upper left', labels=['No', 'Yes'])
plt.tight_layout()  # Improve layout
plt.show()
```

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

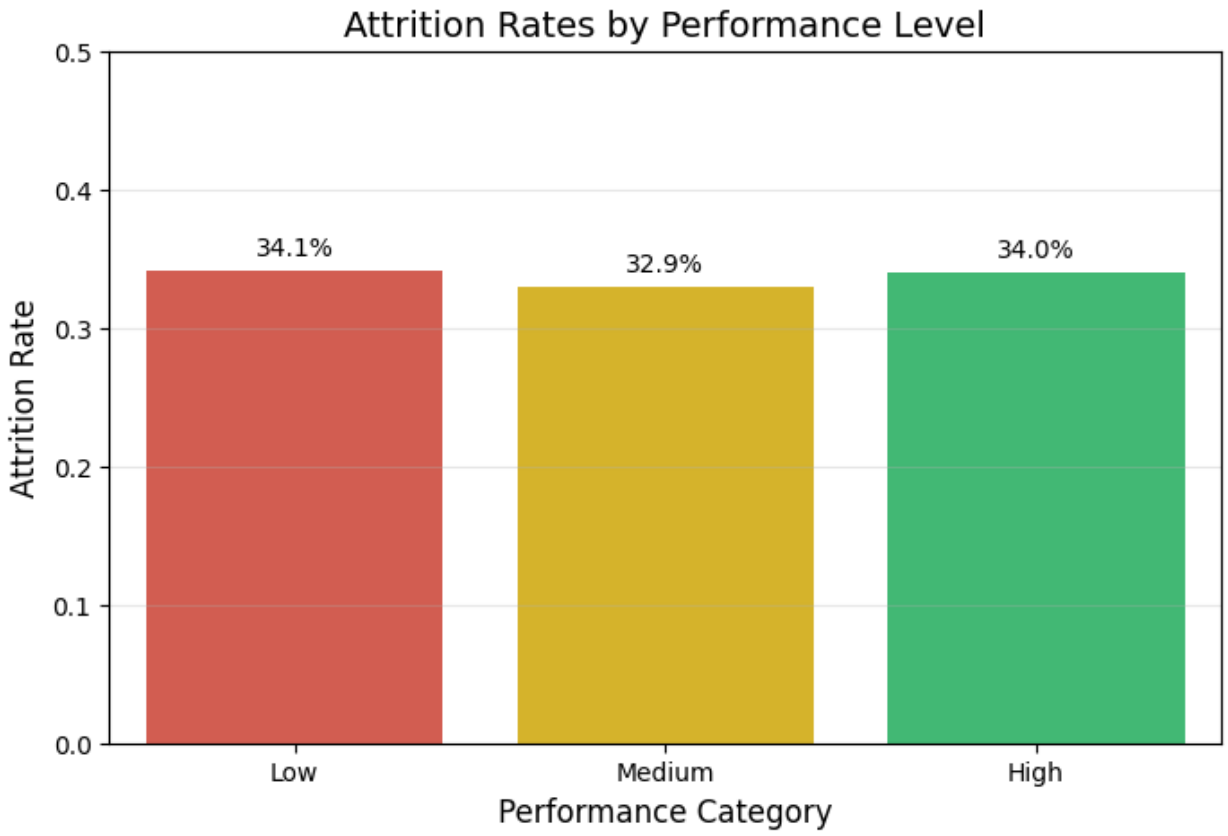


32.What is the attrition rate for employees with different performance ratings?

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

```python
# Statistics
low = df[df['Performance'] == 'Low']['Attrition']
high = df[df['Performance'] == 'High']['Attrition']
t_stat, p_value = stats.ttest_ind(low, high)
print(f'''
[Attrition by Performance]
T-statistic: {t_stat:.2f}
P-value: {p_value:.4f}
Significant difference: {'Yes' if p_value < 0.05 else 'No'}
''')


# Create performance categories
df['Performance'] = pd.qcut(
    df['ManagerRating'],
    q=3,
    labels=['Low', 'Medium', 'High']
)

# Visualization
plt.figure(figsize=(8, 5))
ax = sns.barplot(
    x='Performance',
    y='Attrition',
    data=df,
    order=['Low', 'Medium', 'High'],
    palette=['#e74c3c', '#f1c40f', '#2ecc71'],
    errorbar=None
)

# Add percentages
for p in ax.patches:
    ax.annotate(
        f'{p.get_height():.1%}',
        (p.get_x() + p.get_width()/2., p.get_height()),
        ha='center',
        va='center',
        xytext=(0, 9),
        textcoords='offset points'
    )

plt.title('Attrition Rates by Performance Level', fontsize=14)
plt.xlabel('Performance Category', fontsize=12)
plt.ylabel('Attrition Rate', fontsize=12)
plt.ylim(0, 0.5)
plt.grid(axis='y', alpha=0.3)
plt.show()

```

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

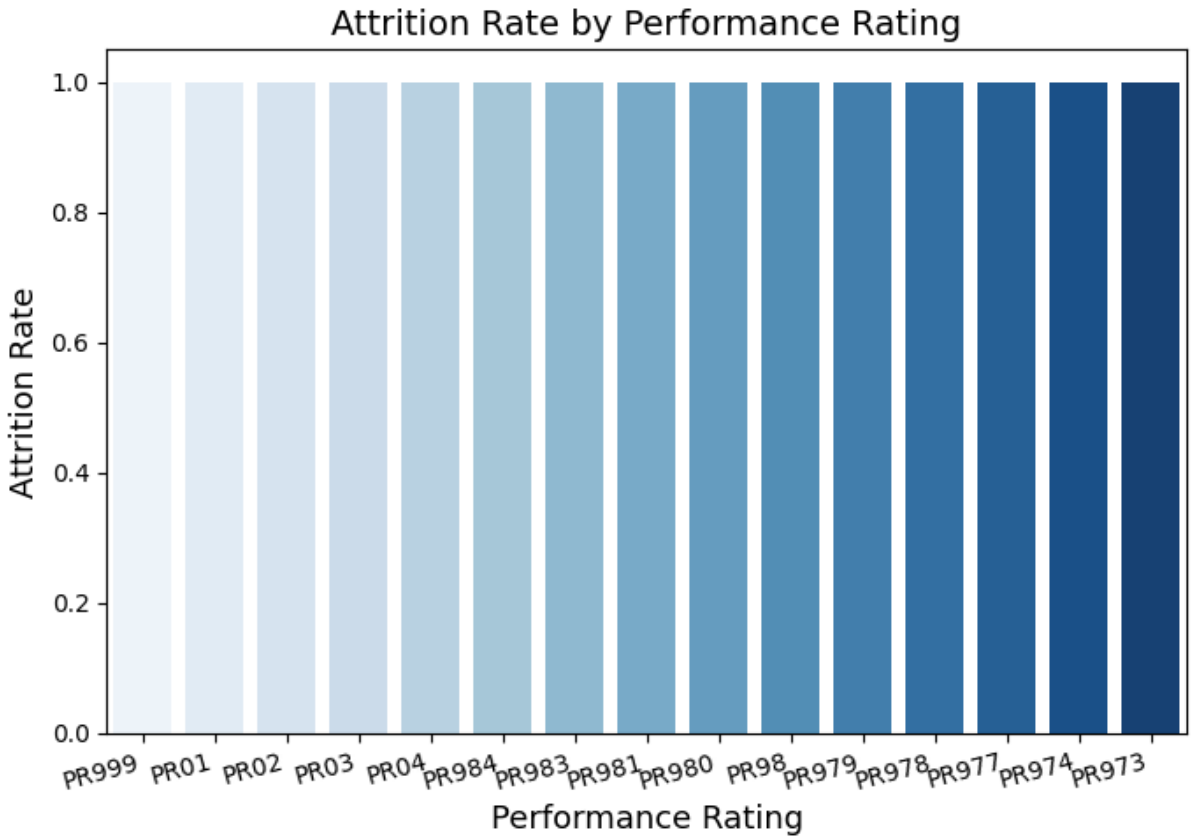


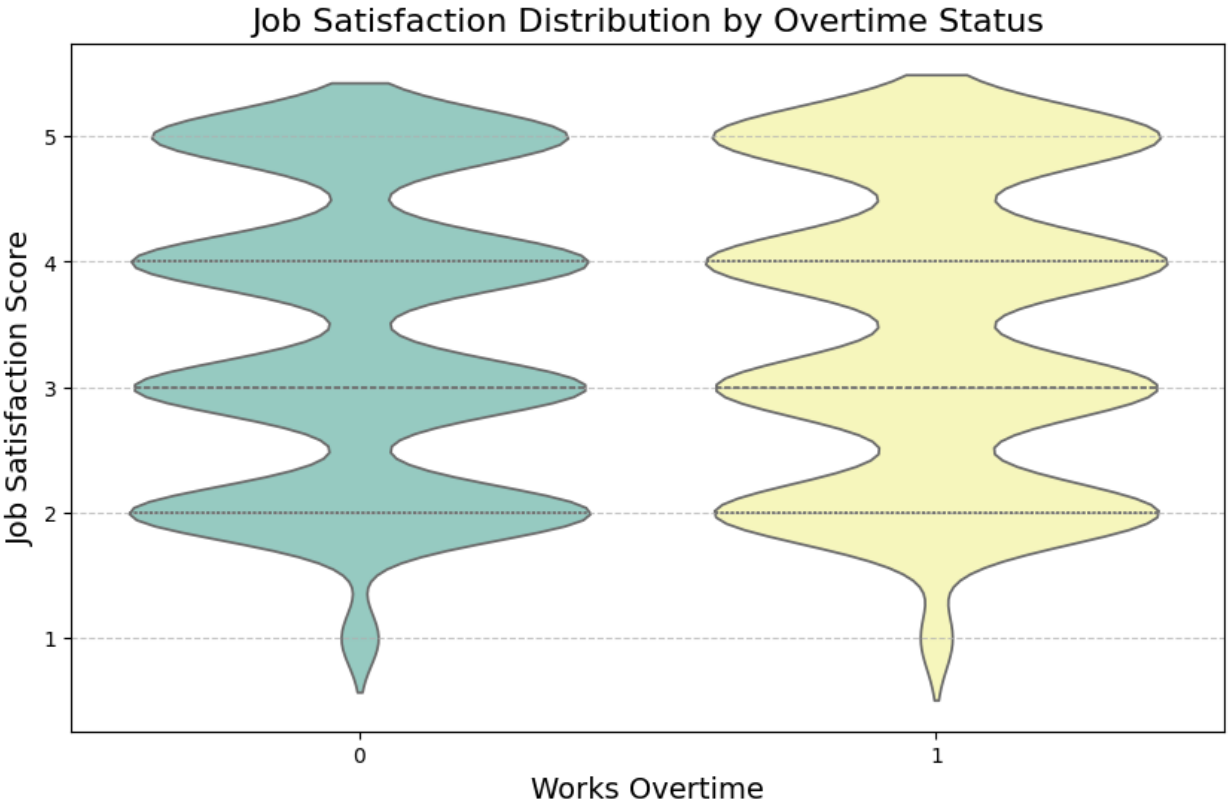
33.What is the impact of overtime on job satisfaction?

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.

```python
# Group by PerformanceID and calculate attrition rate
performance_attrition_rate = df.groupby('PerformanceID')['Attrition'].mean().sort_values(ascending=False).head(15)


# Display all performance ratings
print(f"Attrition Rate by Performance Rating:\n{performance_attrition_rate}")

# Visualization for attrition rate by performance rating
plt.figure(figsize=(7, 5))  # Adjust figure size
sns.barplot(x=performance_attrition_rate.index, y=performance_attrition_rate.values, palette='Blues')

plt.title('Attrition Rate by Performance Rating', fontsize=14)
plt.xlabel('Performance Rating', fontsize=13)
plt.ylabel('Attrition Rate', fontsize=13)

# Rotate x-axis labels slightly for better readability
plt.xticks(rotation=15, ha='right')

# Adjust layout to ensure labels fit
plt.tight_layout()

plt.show()

# Violin plot for Job Satisfaction by Overtime Status
plt.figure(figsize=(10, 6))
sns.violinplot(
    x='OverTime',
    y='JobSatisfaction',
    data=df,
    palette='Set3',
    inner='quartile'  # Display quartiles within the violin plot
)
plt.title('Job Satisfaction Distribution by Overtime Status', fontsize=16)
plt.xlabel('Works Overtime', fontsize=14)
plt.ylabel('Job Satisfaction Score', fontsize=14)
plt.xticks(rotation=0)  # Rotate x-axis labels for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add grid for better readability
plt.show()

# Statistical test
overtime_yes = df[df['OverTime'] == 'Yes']['JobSatisfaction']
overtime_no = df[df['OverTime'] == 'No']['JobSatisfaction']
t_stat, p_value = stats.ttest_ind(overtime_yes, overtime_no, nan_policy='omit')
print(f"Overtime Satisfaction Difference: t={t_stat:.2f}, p={p_value:.4f}")

```

```
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
```


Attrition Rate by Performance Rating
Attrition Rate
1.0
0.8
0.6
0.4
0.2
0.0
PR999
PR01
PR02
PR03
PR04
PR984
PR983
PR981
PR980
PR98
PR979
PR978
PR977
PR974
PR973
Performance Rating

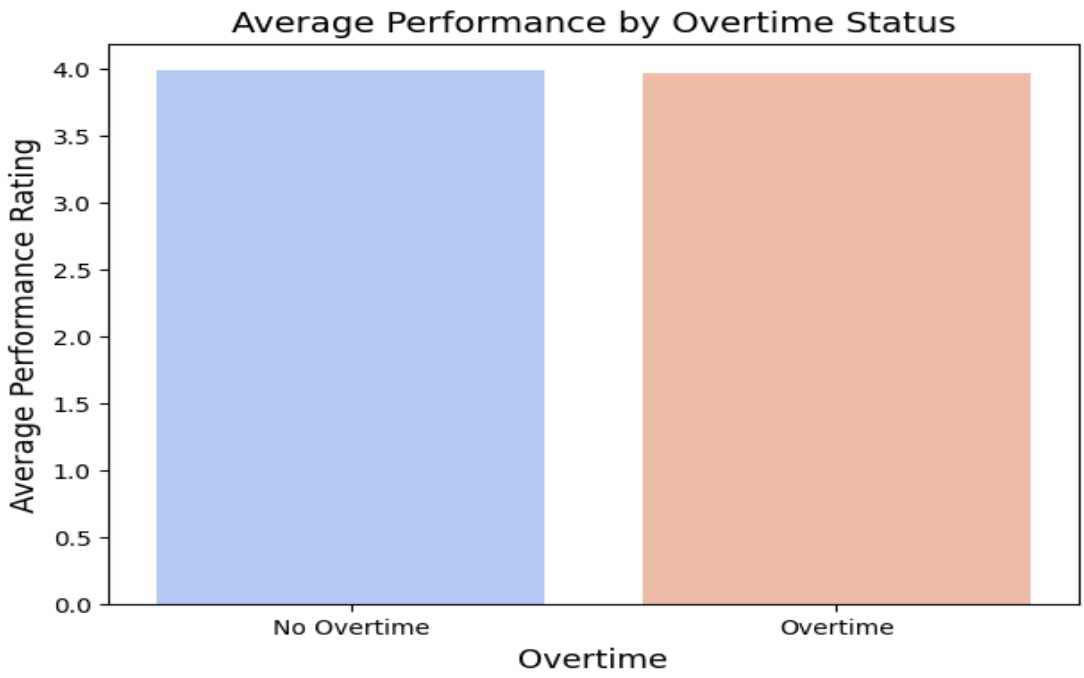

Job Satisfaction Distribution by Overtime Status
Job Satisfaction Score
5
4
3
2
1
0
1
Works Overtime

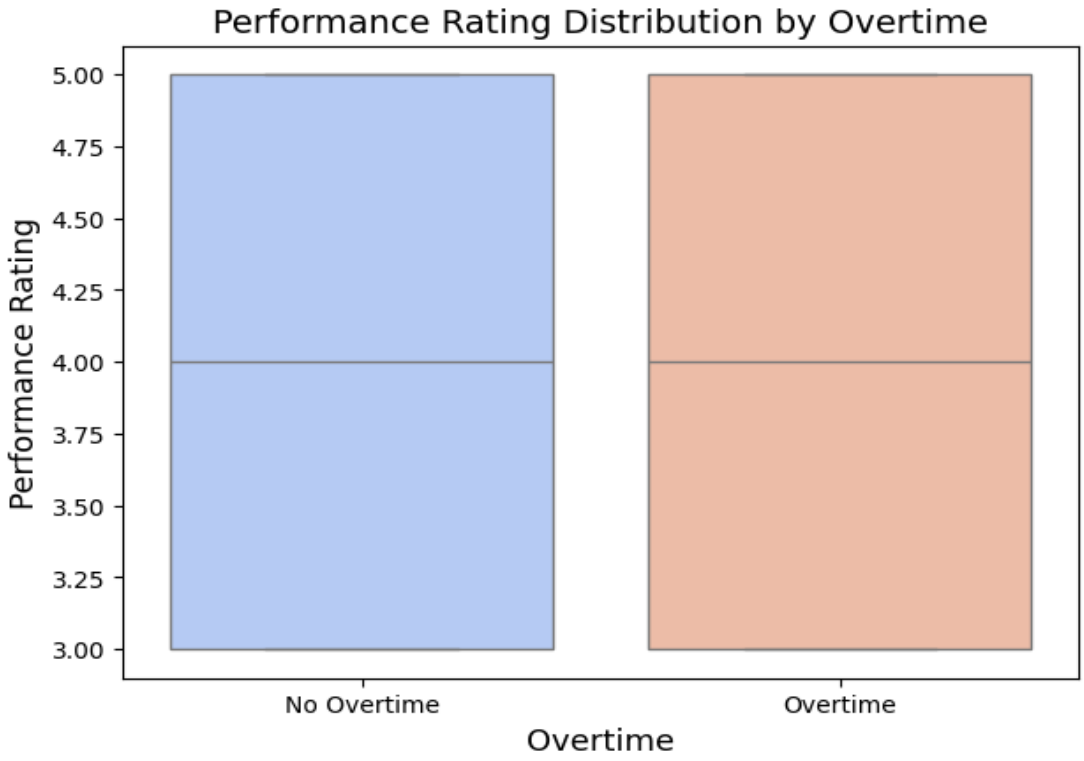
34.Is there a relationship between overtime and performance?

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

```python
# Group by 'OverTime' and calculate mean performance
overtime_performance = df.groupby('OverTime')['SelfRating'].mean()
print(f"Average Performance by Overtime Status:\n{overtime_performance}")

# Statistical test: Is there a significant difference in performance?
overtime_yes = df[df['OverTime'] == 1]['SelfRating']
overtime_no = df[df['OverTime'] == 0]['SelfRating']
t_stat, p_value = ttest_ind(overtime_yes, overtime_no, equal_var=False)
print(f"T-test Results: t-statistic = {t_stat:.4f}, p-value = {p_value:.4f}")

# Visualization: Bar Plot
plt.figure(figsize=(7, 5))
sns.barplot(x=overtime_performance.index, y=overtime_performance.values, palette='coolwarm')
plt.xticks([0, 1], ['No Overtime', 'Overtime'])
plt.title('Average Performance by Overtime Status', fontsize=14)
plt.xlabel('Overtime', fontsize=13)
plt.ylabel('Average Performance Rating', fontsize=13)
plt.show()

# Visualization: Box Plot
plt.figure(figsize=(7, 5))
sns.boxplot(x=df['OverTime'], y=df['SelfRating'], palette='coolwarm')
plt.xticks([0, 1], ['No Overtime', 'Overtime'])
plt.title('Performance Rating Distribution by Overtime', fontsize=14)
plt.xlabel('Overtime', fontsize=13)
plt.ylabel('Performance Rating', fontsize=13)
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
Average Performance Rating
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
No Overtime
Overtime
Overtime


Performance Rating Distribution by Overtime
Performance Rating
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
No Overtime
Overtime
Overtime

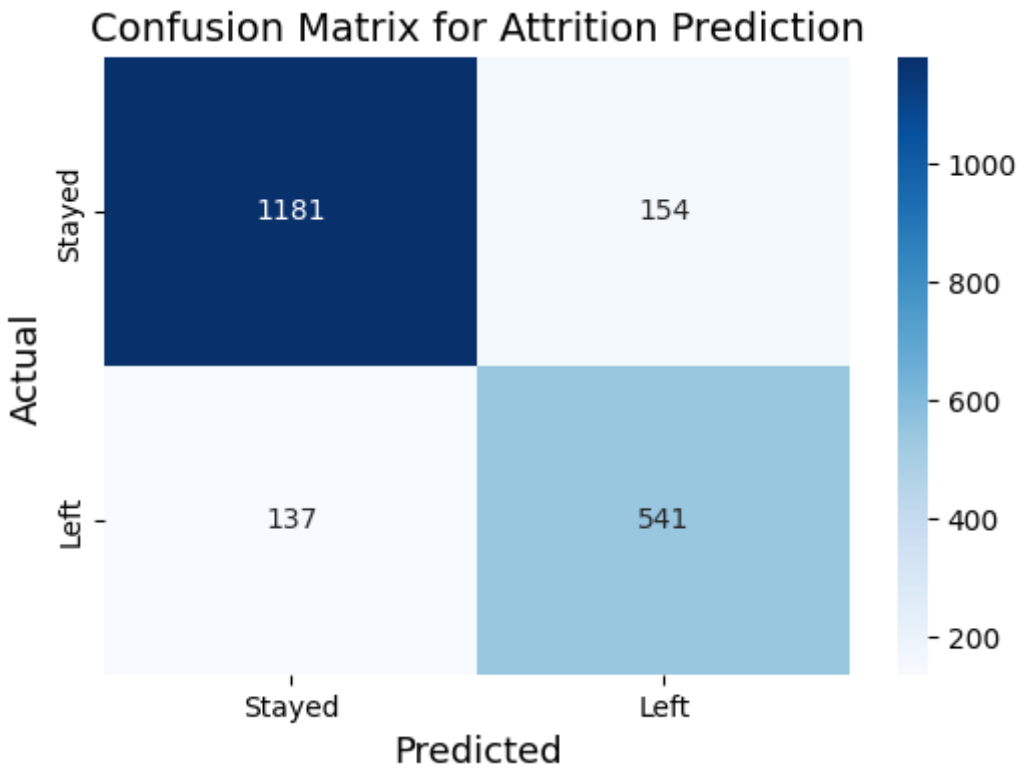
35.Can we predict the likelihood of attrition based on employee attributes?

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.

```python
# Select features and target
X = df[['YearsAtCompany', 'OverTime', 'SelfRating']]
y = df['Attrition']

# Handle missing values (fill with median for numerical columns)
X.fillna(X.median(), inplace=True)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print(f"Classification Report:\n{classification_report(y_test, y_pred)}")

# Confusion Matrix
plt.figure(figsize=(6, 4))
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Stayed', 'Left'], yticklabels=['Stayed', 'Left'])
plt.title('Confusion Matrix for Attrition Prediction', fontsize=14)
plt.xlabel('Predicted', fontsize=13)
plt.ylabel('Actual', fontsize=13)
plt.show()
```

```
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
```

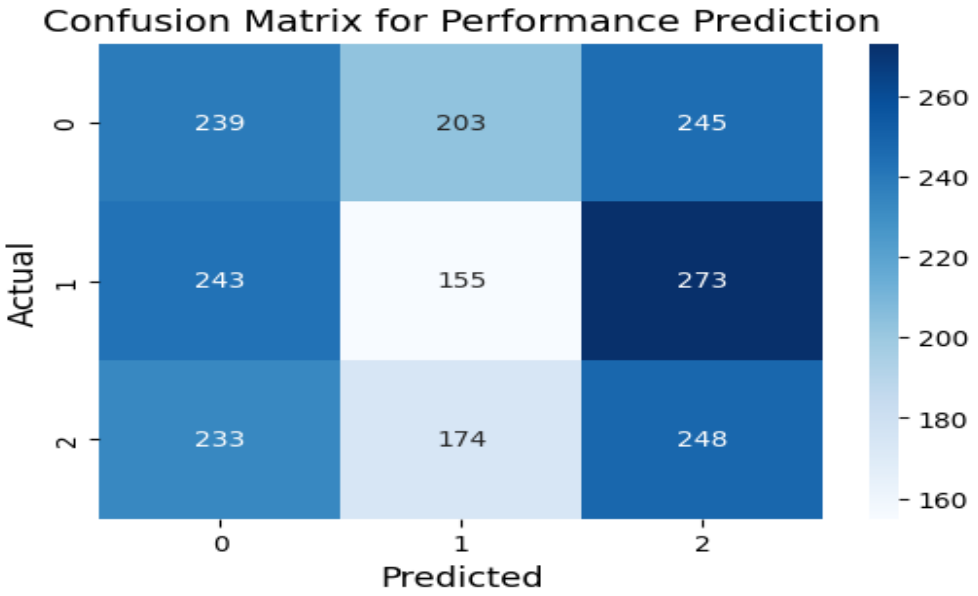


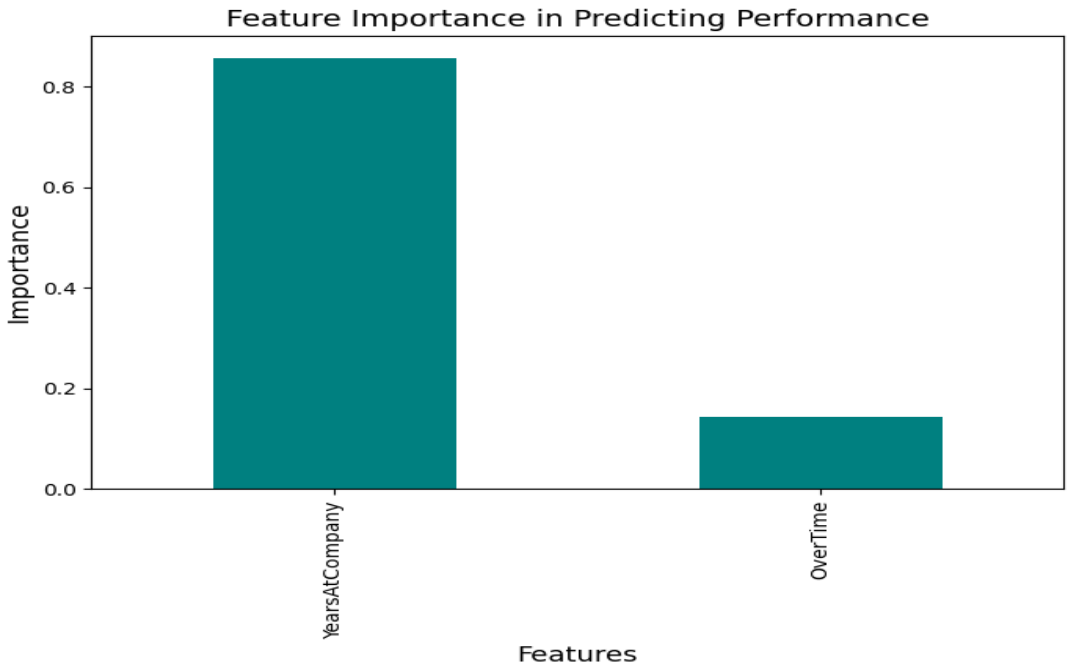
36.Are there any leading indicators of high-performing employees?

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

```python
# Select features and target (predicting performance)
X = df[['YearsAtCompany', 'OverTime']]
y = df['SelfRating']

# Handle missing values
X.fillna(X.median(), inplace=True)
y.fillna(y.median(), inplace=True)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print(f"Classification Report:\n{classification_report(y_test, y_pred)}")

# Confusion Matrix
plt.figure(figsize=(6, 4))
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix for Performance Prediction', fontsize=14)
plt.xlabel('Predicted', fontsize=13)
plt.ylabel('Actual', fontsize=13)
plt.show()

# Feature Importance
feature_importance = pd.Series(model.feature_importances_, index=['YearsAtCompany', 'OverTime'])
feature_importance.sort_values(ascending=False).plot(kind='bar', color='teal', figsize=(8, 5))
plt.title('Feature Importance in Predicting Performance', fontsize=14)
plt.xlabel('Features', fontsize=13)
plt.ylabel('Importance', fontsize=13)
plt.show()
```

```
  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
```





Data Dynamos