



# HR Analytics Project

## Supervised by

## DR: Ahmed Samir

**Egypt (2024- 2025)**

# Team Members:

**Abeer Magdy Saad (Team Leader)**

**Malek Kamal El_Sayed**

**Rahma Maher Sayed**

# Introduction

Human Resource Analytics (**HR Analytics**) is a modern field that contributes to improving organizational performance and enabling data-driven decision-making. Today, companies rely on data analysis to gain valuable insights into employees, such as performance, turnover rates, and employee satisfaction, which helps enhance the work environment and increase productivity.

This project aims to study human resource data using various data analysis tools, including **Excel, Power BI, Tableau, SQL, and Python**, to extract valuable insights that enhance HR management efficiency. By utilizing these tools, we can effectively process data and conduct advanced analyses that help understand the factors influencing employee performance and enable smarter decision-making.

# Abstract

This project focuses on analyzing human resource data to explore the factors that impact job performance and support data-driven decision-making. Data is collected, cleaned, and analyzed using a diverse set of specialized tools, namely **Excel, Power BI, Tableau, SQL, and Python**, each offering a unique aspect of analysis, such as data processing, interactive reporting, and extracting key insights.

Various data analysis techniques are applied to derive insights that assist in improving recruitment processes, reducing employee turnover rates, and enhancing workforce efficiency. By combining statistical analysis with interactive visualizations, we provide data-driven recommendations to optimize organizational performance and improve the overall work environment.

# Table of contents

Table of contents:

# Chapter 1

## Introduction :-

In today's data-driven world, companies rely heavily on analytics to make informed decisions—especially in the Human Resources (HR) department, where understanding employee behavior, performance, and trends is essential.

This graduation project focuses on **HR Data Analysis**, with the aim of transforming raw HR data into meaningful insights that support effective decision-making processes.

The project utilizes **real-world HR data** and applies various data analysis techniques using a combination of professional tools including **Excel**, **Power BI**, **Tableau**, **SQL**, and **Python**. These tools were used collaboratively to explore workforce dynamics, clean and structure data, uncover hidden patterns, and visualize key HR metrics in an intuitive and professional way.

By leveraging these technologies, the project aims to deliver practical, interactive dashboards and evidence-based insights that can assist HR departments in improving employee management and organizational strategies.

## Problem Definition :-

Human Resources departments often face challenges in making strategic decisions due to the lack of actionable insights from raw data. While HR departments usually collect vast amounts of information about employees, such as attendance, salaries, performance, and job roles, this data often remains underutilized.

Without proper analysis, organizations may:

- Fail to detect high turnover risks.
- Overlook patterns in absenteeism or overtime.
- Struggle to allocate salaries fairly or optimize employee performance.
- Miss opportunities to improve employee satisfaction or productivity.

This project aims to bridge this gap by analyzing HR data to uncover valuable insights that can support better decision-making and improve overall HR operations.

## Project Objectives :-

This project aims to:

- Analyze employee data to detect trends in attendance, salary, and job satisfaction.
- Identify factors contributing to employee turnover.
- Discover patterns in overtime and absenteeism.
- Build interactive dashboards using Tableau and Power BI.
- Provide data-driven recommendations for HR decision-making.

## Significance of the Project :-

Human Resources departments often rely on assumptions when managing employee affairs. By transforming HR data into meaningful visual insights, this project helps shift decision-making from intuition to data-driven strategies.

It supports better planning, fairness in compensation, and enhanced employee satisfaction.

## Tools & Technologies Used :-

To ensure accurate, insightful, and professional HR data analysis, this project leverages a combination of industry-standard tools and technologies. Each tool played a key role in the data analysis pipeline, from data cleaning and querying to advanced visualization.

### ◆ Microsoft Excel

Excel was used for initial data exploration, cleaning, and formatting. It played a vital role in identifying missing values, correcting data types, and performing preliminary calculations to prepare the dataset for deeper analysis.

## ◆ SQL (Structured Query Language)

SQL was used to manage and manipulate structured datasets efficiently. It enabled the project team to extract specific records, filter large tables, perform joins between multiple sheets, and prepare meaningful subsets of data for analysis.

## ◆ Python

Python was used for advanced data manipulation and analysis using popular libraries such as:

- pandas – for data wrangling and transformation
- numpy – for numerical operations
- matplotlib and seaborn – for data visualization (when needed)
- Custom scripts – for automating repetitive tasks and generating insights

Python helped in handling complex logic, creating derived fields, and validating trends statistically.

## ◆ Power BI

Power BI was used to build dynamic and interactive dashboards. Its features allowed the creation of slicers, filters, and multi-page reports, enabling users to explore the data with ease and extract meaningful information on demand.

## ◆ Tableau

Tableau was used to design visually compelling dashboards and data stories. Its drag-and-drop interface and advanced visualization capabilities made it possible to represent HR metrics in a clear, attractive, and decision-oriented manner.

# Employee Dataset Overview (23 columns|1470 row)

| Column Name | Description | Sample Values / Notes |
|---|---|---|
| EmployeeID | Unique identifier for each employee | E.g., 3012-1A41 |
| FirstName | First name of the employee | Alix, Annabela |
| LastName | Last name of the employee | Simco, Berrie |
| Gender | Employee gender | Male, Female |
| Age | Employee's age | E.g., 25, 35, 42 |
| BusinessTravel | Frequency of business travel | Some Travel,Non-Travel |
| Department | Department the employee belongs to | HR, IT, Sales |
| DistanceFromHome | Distance from employee's home to work (in KM) | E.g., 5, 15, 30 |
| State | State where the employee resides | CA, NY, IL |
| Ethnicity | Ethnic background of the employee | E.g., Arab, African, Asian |
| Education | Education level (coded as numbers or labels) | 1 to 5 or e.g., High School, Master's |
| EducationField | The employee's field of study | Marketing, Computer Science |
| JobRole | The position or role the employee holds | Data Scientist, Manager, Recruiter |
| MaritalStatus | Marital status of the employee | Single, Married, Divorced |
| Salary | Monthly salary of the employee | E.g., 8000, 15000 |
| StockOptionLevel | Level of stock options offered | 0, 1, 2, 3 |
| OverTime | Whether the employee works overtime | Yes, No |
| HireDate | Date the employee was hired | E.g., 2020-01-01 |
| Attrition | Whether the employee left the company | Yes, No |

| | | |
|---|---|---|
| YearsAtCompany | Total number of years at the company | E.g., 3, 10, 15 |
| YearsInMostRecentRole | Years spent in the most recent job role | E.g., 1, 5, 7 |
| YearsSinceLastPromotion | Time since the last promotion | E.g., 0, 2, 6 |
| YearsWithCurrManager | Years under current manager | E.g., 2, 4, 8 |

## Performance Dataset Overview (11 columns|6710 row)

| Column Name | Description | Sample Values / Notes |
|---|---|---|
| PerformanceID | Unique identifier for each performance review | E.g., PR01, PR02 |
| EmployeeID | Unique identifier for the employee | E.g., B61E-0F26 |
| ReviewDate | Date when the performance review was conducted | E.g., 2020-05-01 |
| EnvironmentSatisfaction | Employee's satisfaction with the work environment | 1,2,3,4,5 |
| JobSatisfaction | Employee's satisfaction with the job role | 1,2,3,4,5 |
| RelationshipSatisfaction | Employee's satisfaction with work relationships | 1,2,3,4,5 |
| TrainingOpportunitiesWithinYear | Whether the employee had training opportunities within the year | Yes, No |
| TrainingOpportunitiesTaken | Whether the employee took the training opportunities | Yes, No |
| WorkLifeBalance | Employee's balance between work and personal life | 1,2,3,4,5 |
| SelfRating | Employee's self-rated performance | E.g., 1 to 5 |
| ManagerRating | Manager's rating of the employee's performance | E.g., 1 to 5 |

## Education Dataset Overview (2 columns|6 row)

| Column Name | Description | Sample Values / Notes |
|---|---|---|
| EducationLevelID | A unique identifier for each education level. This column contains numbers or unique values used to represent each education level individually. | E.g., 1,2,3,4,5 |
| EducationLevel | Represents the actual education level of the employee. This column contains textual descriptions specifying the type of education attained by the employee. | E.g., High School, Bachelor's Degree, Master's Degree |

## Satisfaction Dataset Overview (2 columns|6 row)

| Column Name | Description | Sample Values / Notes |
|---|---|---|
| SatisfactionID | A unique identifier for each satisfaction level. This column contains numeric or unique values used to represent each satisfaction level individually. | E.g., 1,2,3,4,5 |
| SatisfactionLevel | Represents the satisfaction level of the employee. This column contains a textual description specifying the employee's satisfaction level regarding their job, work environment, or other relevant factors. | E.g., Dissatisfied, Neutral, Satisfiedو Very Satisfied |

## RatingLevel Dataset Overview(2 columns|6 row)

| Column Name | Description | Sample Values / Notes |
|---|---|---|
| RatingID | A unique identifier for each rating level. This column contains numeric or unique values used to represent each rating level individually. | E.g., 1,2,3,4,5 |
| RatingLevel | Represents the rating level given to the employee. This column contains a textual description that specifies the level of the rating, such as performance rating or evaluation score. | E.g., Meets Expectation, Exceeds Expectation , Unacceptable |

# Chapter 2

## Data Cleaning & Transformation

## Introduction :-

In this chapter, the focus will be on the essential steps of Data Cleaning and Transformation. Data from various sources often contains inconsistencies, missing values, and irrelevant information. Cleaning and transforming this data ensures the integrity of the analysis and enables better insights. This chapter will detail the steps taken to clean and transform the dataset, addressing issues such as missing values, duplicates, and formatting errors.

### 2.1. Handling Missing Values:-

Before proceeding with any data analysis, it was essential to verify whether the dataset contained any missing values that could impact the accuracy of results.

After carefully inspecting the dataset using tools like Excel, SQL, and Python (Pandas), **no missing values were found** in any of the columns. This ensured that the data was complete and ready for further analysis without the need for imputation or deletion.

#### Checking for missing values using SQL

```sql
-- Step 1: Identify Missing Data
-- Count missing values in each table
SELECT 'Employee' AS TableName, COUNT(*) AS MissingCount
FROM Employee
WHERE EmployeeID IS NULL OR JobRole IS NULL OR Gender IS NULL;

SELECT 'PerformanceRating' AS TableName, COUNT(*) AS MissingCount
FROM PerformanceRating
WHERE PerformanceID IS NULL OR EmployeeID IS NULL;

SELECT 'EducationLevel' AS TableName, COUNT(*) AS MissingCount
FROM EducationLevel
WHERE EducationLevelID IS NULL OR EducationLevel IS NULL;

SELECT 'RatingLevel' AS TableName, COUNT(*) AS MissingCount
FROM RatingLevel
WHERE RatingID IS NULL OR RatingLevel IS NULL;

SELECT 'SatisfiedLevel' AS TableName, COUNT(*) AS MissingCount
FROM SatisfiedLevel
WHERE SatisfactionID IS NULL OR SatisfactionLevel IS NULL;
```

### Employee Table

| | TableName | MissingCount |
|---|---|---|
| 1 | Employee | 0 |

### PerformanceRating Table

| | TableName | MissingCount |
|---|---|---|
| 1 | PerformanceRating | 0 |

### EducationLevel Table

| | TableName | MissingCount |
|---|---|---|
| 1 | EducationLevel | 0 |

### RatingLevel Table

| | TableName | MissingCount |
|---|---|---|
| 1 | RatingLevel | 0 |

### SatisfiedLevel Table

| | TableName | MissingCount |
|---|---|---|
| 1 | SatisfiedLevel | 0 |

**The query shows that there are no missing values in the dataset.**

--_____--

## Checking for missing values using Pandas

```
: #Searching for Null values
print(Employee.isnull().sum())
print(PerformanceRating.isnull().sum())
print(RatingLevel.isnull().sum())
print(SatisfiedLevel.isnull().sum())
print(EducationLevel.isnull().sum())
```

### Employee Table

```
EmployeeID                0
FirstName                 0
LastName                  0
Gender                    0
Age                       0
BusinessTravel            0
Department                0
DistanceFromHome (KM)     0
State                     0
Ethnicity                 0
Education                 0
EducationField            0
JobRole                   0
MaritalStatus             0
Salary                    0
StockOptionLevel          0
OverTime                  0
HireDate                  0
Attrition                 0
YearsAtCompany            0
YearsInMostRecentRole     0
YearsSinceLastPromotion   0
YearsWithCurrManager      0
```

### PerformanceRating Table

```
PerformanceID                  0
EmployeeID                     0
ReviewDate                     0
EnvironmentSatisfaction        0
JobSatisfaction                0
RelationshipSatisfaction       0
TrainingOpportunitiesWithinYear 0
TrainingOpportunitiesTaken     0
WorkLifeBalance                0
SelfRating                     0
ManagerRating                  0
```

**RatingLevel Table**

| | |
|---|---|
| RatingID | 0 |
| RatingLevel | 0 |

**EducationLevel Table**

| | |
|---|---|
| EducationLevelID | 0 |
| EducationLevel | 0 |

**SatisfiedLevel Table**

| | |
|---|---|
| SatisfactionID | 0 |
| SatisfactionLevel | 0 |

**The output confirms that all columns have zero missing entries.**

--_____--

## 2.2 Handling Duplicate Records:-

Before proceeding with data analysis, it was essential to ensure that the dataset did not contain any duplicate records that might bias the results or lead to inaccurate conclusions.

Duplicate records occur when the same row appears more than once in the dataset, either entirely or based on specific key columns. These can result from repeated data entry, system issues, or merging datasets.

To handle this, I used both **SQL** and **Python (Pandas)** to check for and remove any duplicate records.

## Checking for duplicate EmployeeID values using SQL

```sql
-- Step 2: Identify Duplicates
-- Find duplicates by primary key in each table
SELECT EmployeeID, COUNT(*)
FROM Employee
GROUP BY EmployeeID
HAVING COUNT(*) > 1;

SELECT PerformanceID, EmployeeID, COUNT(*)
FROM PerformanceRating
GROUP BY PerformanceID, EmployeeID
HAVING COUNT(*) > 1;

SELECT EducationLevelID, COUNT(*)
FROM EducationLevel
GROUP BY EducationLevelID
HAVING COUNT(*) > 1;

SELECT RatingID, COUNT(*)
FROM RatingLevel
GROUP BY RatingID
HAVING COUNT(*) > 1;

SELECT SatisfactionID, COUNT(*)
FROM SatisfiedLevel
GROUP BY SatisfactionID
HAVING COUNT(*) > 1;
```

### Employee Table

| | EmployeeID | (No column nam… |
|---|---|---|

### PerformanceRating Table

| | PerformanceID | EmployeeID | (No column nam… |
|---|---|---|---|

### EducationLevel Table

| | EducationLevel… | (No column nam… |
|---|---|---|

### RatingLevel Table

| | RatingID | (No column nam… |
|---|---|---|

### SatisfiedLevel Table

| | SatisfactionID | (No column nam… |
|---|---|---|

**The result showed that no duplicate records were found.**

**Checking for full duplicate rows using Pandas**

```
In [12]: #Searching for duplicated values
         print(Employee.duplicated().sum())
         print(PerformanceRating.duplicated().sum())
         print(RatingLevel.duplicated().sum())
         print(SatisfiedLevel.duplicated().sum())
         print(EducationLevel.duplicated().sum())
```

```
0
0
0
0
0
```

**The result indicated that no completely duplicated rows exist in the dataset.**

--_____--

## 2.3 Data Type Corrections:-

As part of the data cleaning process, ensuring that each column has the correct data type is essential for accurate analysis and proper visualization.

In this project, all columns were reviewed to confirm their data types. Most columns had appropriate types such as integers for numerical values and strings for categorical data.

However, the **ReviewDate** column was initially in string (text) format, which would prevent any date-based calculations or filtering. Therefore, it was converted to a **Date/Datetime** format using Python and Excel to allow for accurate analysis, such as calculating the employee's tenure or filtering by hiring year.

## Converting ReviewDate to datetime using Python

```
In [14]: # Convert reviewdate to datetime
         PerformanceRating['ReviewDate'] = pd.to_datetime(PerformanceRating['ReviewDate'], errors='coerce')

         # Confirm the change
         print(PerformanceRating.dtypes)
```

```
PerformanceID                          object
EmployeeID                             object
ReviewDate                     datetime64[ns]
EnvironmentSatisfaction                 int64
JobSatisfaction                         int64
RelationshipSatisfaction                int64
TrainingOpportunitiesWithinYear         int64
TrainingOpportunitiesTaken              int64
WorkLifeBalance                         int64
SelfRating                              int64
ManagerRating                           int64
dtype: object
```

**This conversion allowed for accurate time-based analysis, such as determining the number of years an employee has been with the company.**

## 2.4 Merging Datasets:-

To perform comprehensive HR analysis, data from multiple sources needed to be combined into a single dataset. This merging process helps in creating a unified view of each employee by integrating various aspects such as personal details, performance reviews, satisfaction levels, and education background.

The following tables were merged:

- **Employee Table** – contains personal and job-related information.
- **Performance Table** – includes satisfaction scores and manager/self-ratings.
- **Education Level Table** – maps education level IDs to their respective labels.
- **Satisfaction Level Table** – provides descriptive levels for satisfaction scores.
- **Rating Table** – defines the meaning of different rating levels.

## Merging Employee and Performance Data using Python

```python
# Merge Employee with PerformanceRating (Primary tables)
merged_df = pd.merge(Employee, PerformanceRating, on='EmployeeID', how='left')

# Merge with Satisfaction Levels (EnvironmentSatisfaction)
merged_df = pd.merge(merged_df, SatisfiedLevel, left_on='EnvironmentSatisfaction', right_on='SatisfactionID', how='left')

# Merge with Rating Levels (ManagerRating)
merged_df = pd.merge( merged_df, RatingLevel, left_on='ManagerRating', right_on='RatingID', how='left')

# Merge with Education Levels (Education)
merged_df = pd.merge( merged_df, EducationLevel, left_on='Education', right_on='EducationLevelID', how='left')

# Drop duplicate key columns
merged_df.drop(columns=['SatisfactionID', 'RatingID', 'EducationLevelID'], inplace=True)

# Check merged dataset structure
print(merged_df.info())

# Save the cleaned and merged dataset
merged_df.to_csv("Cleaned_Data.csv", index=False)

#Download the cleansed Data
merged_df.to_csv ("/Users/rahmasaadawy/Downloads/Cleaned_HR_Data.csv")
merged_df.to_excel("/Users/rahmasaadawy/Downloads/Cleaned_HR_Data.xlsx", index=False)

print(f"✅ Cleaned dataset saved successfully to: {"/Users/rahmasaadawy/Downloads/Cleaned_HR_Data.csv"}")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6899 entries, 0 to 6898
Data columns (total 36 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   EmployeeID                      6899 non-null   object
 1   FirstName                       6899 non-null   object
 2   LastName                        6899 non-null   object
 3   Gender                          6899 non-null   object
 4   Age                             6899 non-null   int64
 5   BusinessTravel                  6899 non-null   object
 6   Department                      6899 non-null   object
 7   DistanceFromHome (KM)           6899 non-null   int64
 8   State                           6899 non-null   object
 9   Ethnicity                       6899 non-null   object
 10  Education                       6899 non-null   int64
 11  EducationField                  6899 non-null   object
 12  JobRole                         6899 non-null   object
 13  MaritalStatus                   6899 non-null   object
 14  Salary                          6899 non-null   int64
 15  StockOptionLevel                6899 non-null   int64
 16  OverTime                        6899 non-null   object
 17  HireDate                        6899 non-null   object
 18  Attrition                       6899 non-null   object
 19  YearsAtCompany                  6899 non-null   int64
 20  YearsInMostRecentRole           6899 non-null   int64
 21  YearsSinceLastPromotion         6899 non-null   int64
 22  YearsWithCurrManager            6899 non-null   int64
 23  PerformanceID                   6709 non-null   object
 24  ReviewDate                      6709 non-null   datetime64[ns]
 25  EnvironmentSatisfaction         6709 non-null   float64
 26  JobSatisfaction                 6709 non-null   float64
 27  RelationshipSatisfaction        6709 non-null   float64
 28  TrainingOpportunitiesWithinYear 6709 non-null   float64
 29  TrainingOpportunitiesTaken      6709 non-null   float64
 30  WorkLifeBalance                 6709 non-null   float64
 31  SelfRating                      6709 non-null   float64
 32  ManagerRating                   6709 non-null   float64
 33  SatisfactionLevel               6709 non-null   object
 34  RatingLevel                     6709 non-null   object
 35  EducationLevel                  6899 non-null   object
dtypes: datetime64[ns](1), float64(8), int64(9), object(18)
memory usage: 1.9+ MB
None
✅ Cleaned dataset saved successfully to: /Users/rahmasaadawy/Downloads/Cleaned_HR_Data.csv
```

# Chapter 3

## Analysis Questions

### 1:- Calculate and display the total number of employees

**Solution using Python:-**

```
total_employees = merged_df["EmployeeID"].nunique()
print(f"Total Number of Employees: {total_employees}")
```

**Output**

```
Total Number of Employees: 1470
```

**Solution using SQL:-**

```
-- 1.1 Calculate and display the total number of employees
SELECT COUNT(DISTINCT EmployeeID) AS TotalEmployees
FROM Employee;
```

**Output**

| | TotalEmployees |
|---|---|
| 1 | 1470 |

## 2:- Count employees by gender

### Solution using Python:-

```
unique_gender_counts = merged_df.groupby("EmployeeID")["Gender"].first().value_counts()

print("Unique Number of Male and Female Employees:")
print(unique_gender_counts)
```

### Output

```
Unique Number of Male and Female Employees:
Gender
Female                675
Male                  651
Non-Binary            124
Prefer Not To Say      20
Name: count, dtype: int64
```

### Solution using SQL:-

```
SELECT
    Gender,
    COUNT(DISTINCT EmployeeID) AS UniqueEmployeeCount
FROM Employee
GROUP BY Gender;
```

### Output

| | Gender | UniqueEmployeeCount |
|---|---|---|
| 1 | Non-Binary | 124 |
| 2 | Prefer Not To Say | 20 |
| 3 | Male | 651 |
| 4 | Female | 675 |

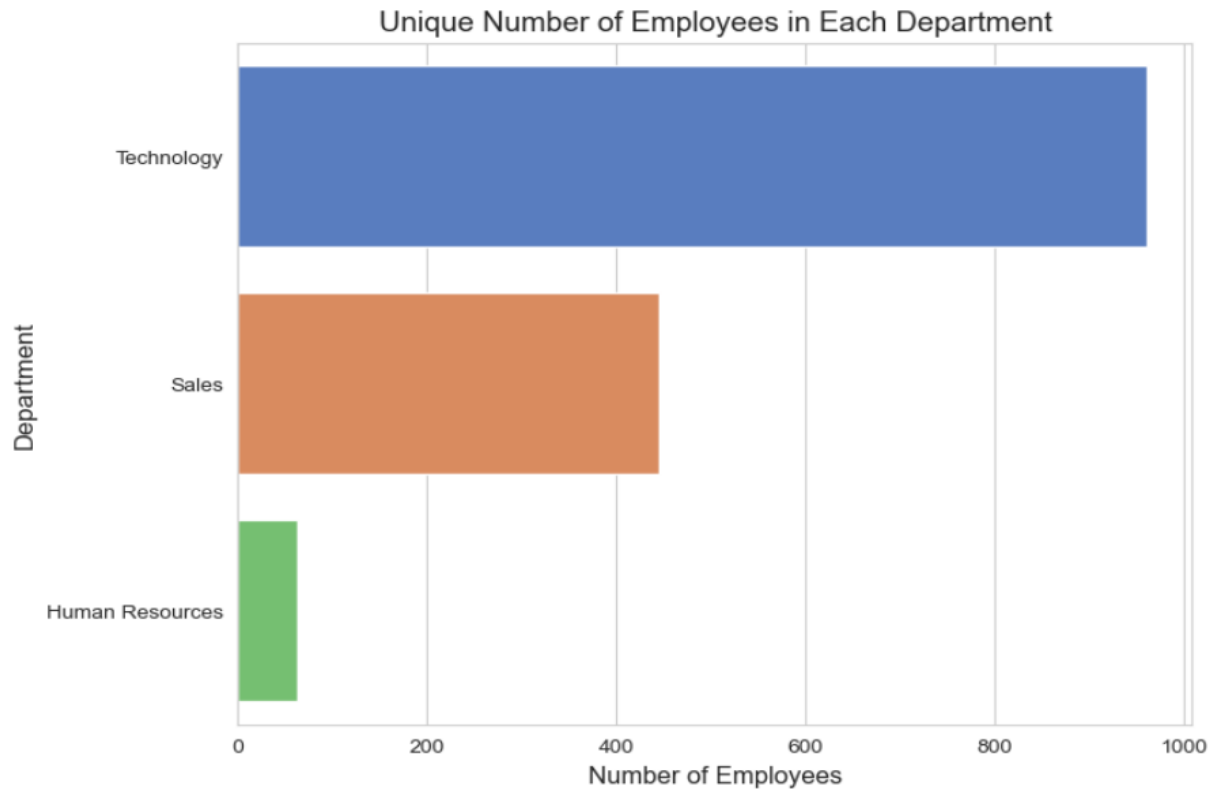
From the analysis, it appears that the number of **males** (651) is slightly lower than the number of **females** (675). As for **Non-Binary** individuals, there are 124, and 20 individuals prefer not to disclose their gender. This data may indicate greater diversity in gender identity within the organization, and it could be useful to explore this further in order to create a more inclusive work environment for everyone.

## 3:- Count employees by department

### Solution using Python:-

```
unique_department_counts = merged_df.groupby("EmployeeID")["Department"].first().value_counts()

print("Unique Number of Employees in Each Department:")
print(unique_department_counts)

# Visualize the unique number of employees in each department
# Set the plot style
sns.set_style("whitegrid")

# Create a bar plot for the unique employee count by department
plt.figure(figsize=(8, 6))
sns.barplot(x=unique_department_counts.values, y=unique_department_counts.index, palette="muted")

# Add labels and title
plt.title("Unique Number of Employees in Each Department", fontsize=14)
plt.xlabel("Number of Employees", fontsize=12)
plt.ylabel("Department", fontsize=12)

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

### Output



**Solution using SQL:-**

```
SELECT
    Department,
    COUNT(DISTINCT EmployeeID) AS UniqueEmployeeCount
FROM Employee
GROUP BY Department;
```

**Output**

| | Department | UniqueEmployeeCount |
|---|---|---|
| 1 | Sales | 446 |
| 2 | Human Resources | 63 |
| 3 | Technology | 961 |

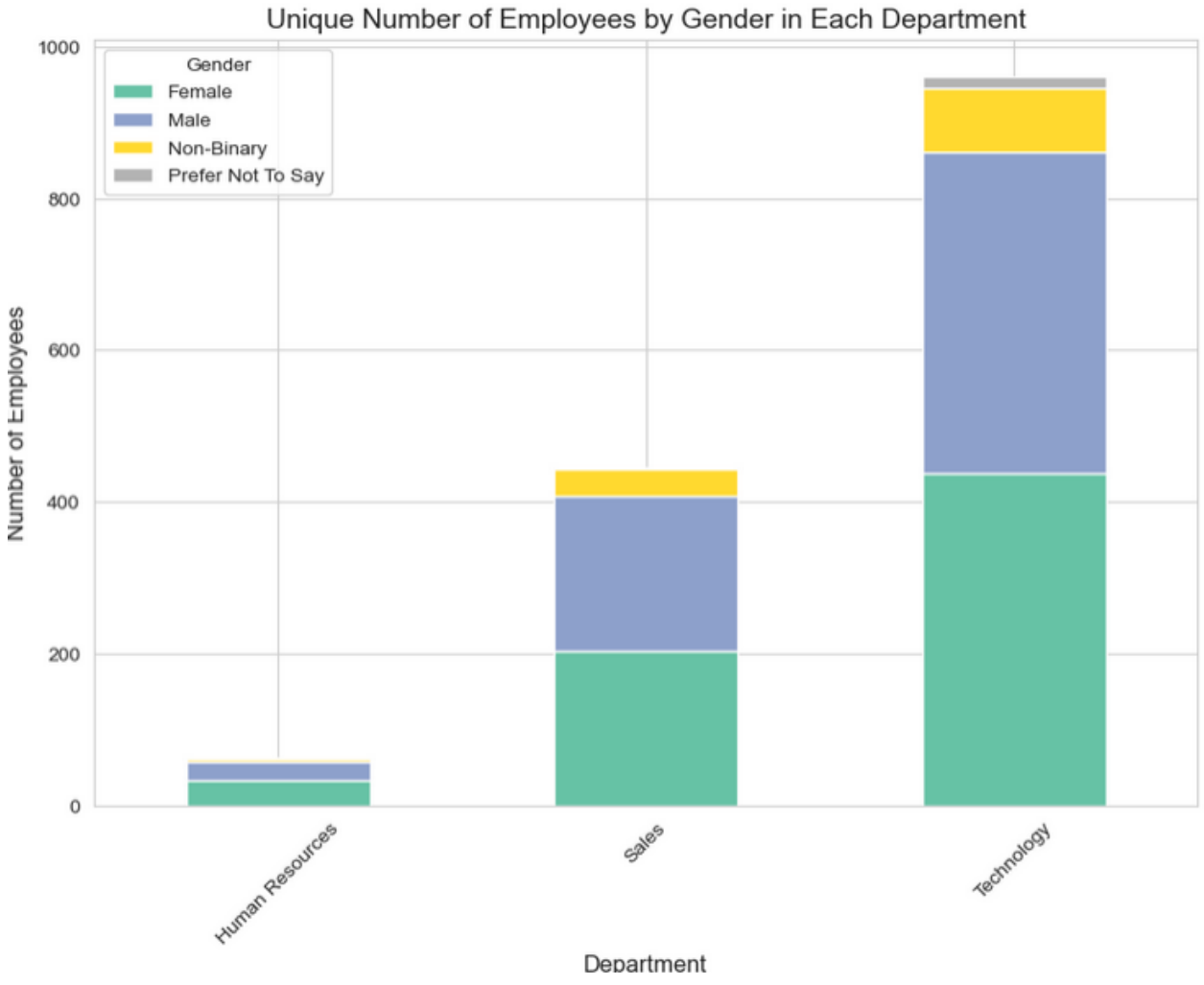
From the analysis, it is evident that the largest number of employees is in the **Technology** department (961 employees), followed by the **Sales** department with 464 employees, while the **HR** department has the smallest number of employees (63 employees). These distributions reflect a larger focus on resources in the technology and sales areas, which may indicate a need for larger teams in these departments to support growth and expansion.

## 4:- Count employees by gender within each department

### Solution using Python:-

```
unique_gender_department_counts = merged_df.groupby(["Department", "EmployeeID"])["Gender"].first().reset_index()
gender_by_department = unique_gender_department_counts.groupby(["Department", "Gender"]).size().unstack(fill_value=0)

print("Unique Number of Employees by Gender in Each Department:")
print(gender_by_department)

# Visualize gender distribution within each department
plt.figure(figsize=(10, 7))
gender_by_department.plot(kind="bar", stacked=True, colormap="Set2", figsize=(10, 7))

plt.title("Unique Number of Employees by Gender in Each Department", fontsize=14)
plt.xlabel("Department", fontsize=12)
plt.ylabel("Number of Employees", fontsize=12)
plt.xticks(rotation=45)
plt.legend(title="Gender")

plt.show()
```

### Output



## Solution using SQL:-

```sql
SELECT
    Department,
    Gender,
    COUNT(DISTINCT EmployeeID) AS UniqueEmployeeCount
FROM Employee
GROUP BY Department, Gender
ORDER BY Department, Gender;
```

## Output

| | Department | Gender | UniqueEmployeeCount |
|---|---|---|---|
| 1 | Human Resources | Female | 33 |
| 2 | Human Resources | Male | 24 |
| 3 | Human Resources | Non-Binary | 5 |
| 4 | Human Resources | Prefer Not To Say | 1 |
| 5 | Sales | Female | 204 |
| 6 | Sales | Male | 204 |
| 7 | Sales | Non-Binary | 35 |
| 8 | Sales | Prefer Not To Say | 3 |
| 9 | Technology | Female | 438 |
| 10 | Technology | Male | 423 |
| 11 | Technology | Non-Binary | 84 |
| 12 | Technology | Prefer Not To Say | 16 |

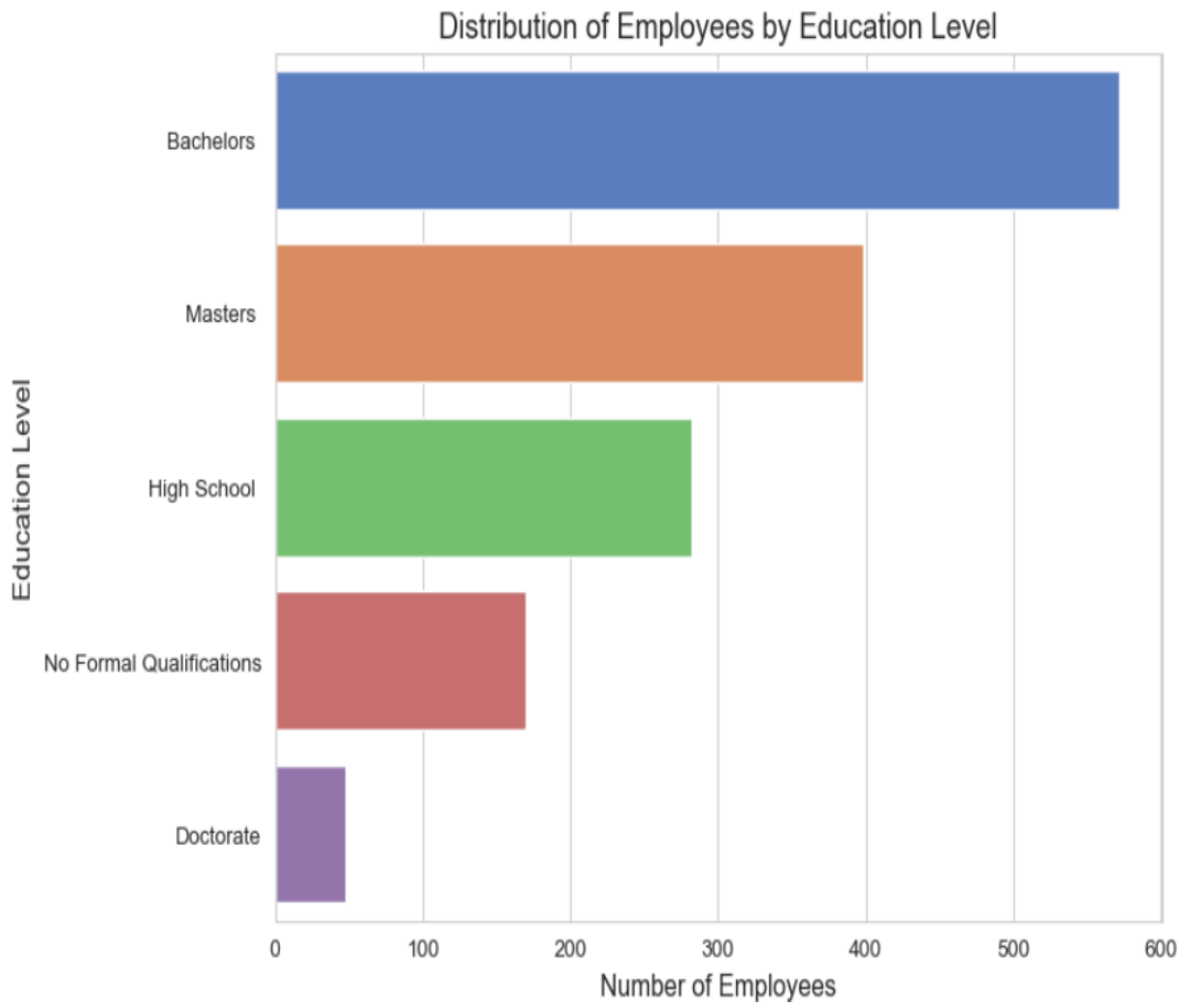
Through the analysis, the number of unique employees by gender within each department was calculated. The results show a diverse distribution of genders across different departments. For example, some departments may have a more balanced gender ratio, while others may be dominated by a specific gender. This data can provide insights into the diversity across departments and help in making strategic decisions to promote diversity and inclusion within the organization.

## 5:- Count employees by education level

### Solution using Python:-

```
unique_education_counts = merged_df.groupby("EmployeeID")["EducationLevel"].first().value_counts()

print("Unique Number of Employees by Education Level:")
print(unique_education_counts)

# Visualize the distribution of employees by education level
plt.figure(figsize=(8, 6))
sns.barplot(x=unique_education_counts.values, y=unique_education_counts.index, palette="muted")

plt.title("Distribution of Employees by Education Level", fontsize=14)
plt.xlabel("Number of Employees", fontsize=12)
plt.ylabel("Education Level", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```
SELECT
    el.EducationLevel,
    COUNT(DISTINCT e.EmployeeID) AS UniqueEmployeeCount
FROM Employee e
JOIN EducationLevel el ON e.Education = el.EducationLevelID
GROUP BY el.EducationLevel
ORDER BY el.EducationLevel;
```

## Output

| | EducationLevel | UniqueEmployeeCount |
|---|---|---|
| 1 | Bachelors | 572 |
| 2 | Doctorate | 48 |
| 3 | High School | 282 |
| 4 | Masters | 398 |
| 5 | No Formal Qualifications | 170 |

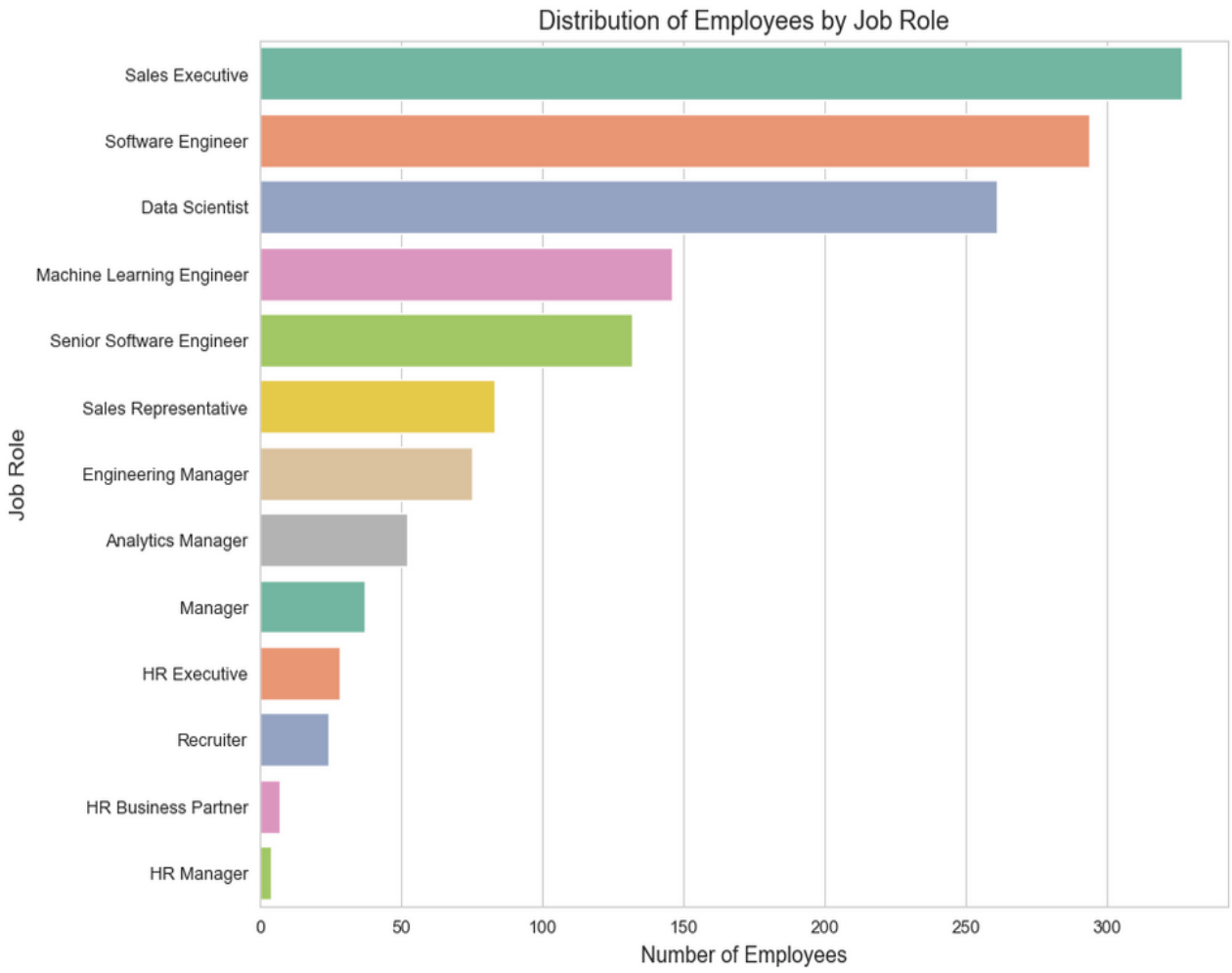
**Through the analysis, the number of unique employees was counted based on their education level. The results show that the largest group of employees holds a** Bachelor's degree **(572 employees), followed by those with a** High School diploma **(280 employees), then** Master's degree **holders (398 employees), and finally** Doctorates **(48 employees). Additionally, there are 170 employees with** Non-formal education**. This data indicates that most employees have a university education, while the smaller number with doctorates reflects a relatively low concentration of employees in advanced academic roles.**

## 6:- Count employees by job role

### Solution using Python:-

```
unique_jobrole_counts = merged_df.groupby("EmployeeID")["JobRole"].first().value_counts()

print("Unique Number of Employees by Job Role:")
print(unique_jobrole_counts)

# Visualize the distribution of employees by job role
plt.figure(figsize=(10, 8))
sns.barplot(x=unique_jobrole_counts.values, y=unique_jobrole_counts.index, palette="Set2")

plt.title("Distribution of Employees by Job Role", fontsize=14)
plt.xlabel("Number of Employees", fontsize=12)
plt.ylabel("Job Role", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```sql
SELECT
    JobRole,
    COUNT(DISTINCT EmployeeID) AS UniqueEmployeeCount
FROM Employee
GROUP BY JobRole
ORDER BY JobRole;
```

## Output

| | JobRole | UniqueEmployeeCount |
|---|---|---|
| 1 | Analytics Manager | 52 |
| 2 | Data Scientist | 261 |
| 3 | Engineering Manager | 75 |
| 4 | HR Business Partner | 7 |
| 5 | HR Executive | 28 |
| 6 | HR Manager | 4 |
| 7 | Machine Learning Engineer | 146 |
| 8 | Manager | 37 |
| 9 | Recruiter | 24 |
| 10 | Sales Executive | 327 |
| 11 | Sales Representative | 83 |
| 12 | Senior Software Engineer | 132 |
| 13 | Software Engineer | 294 |

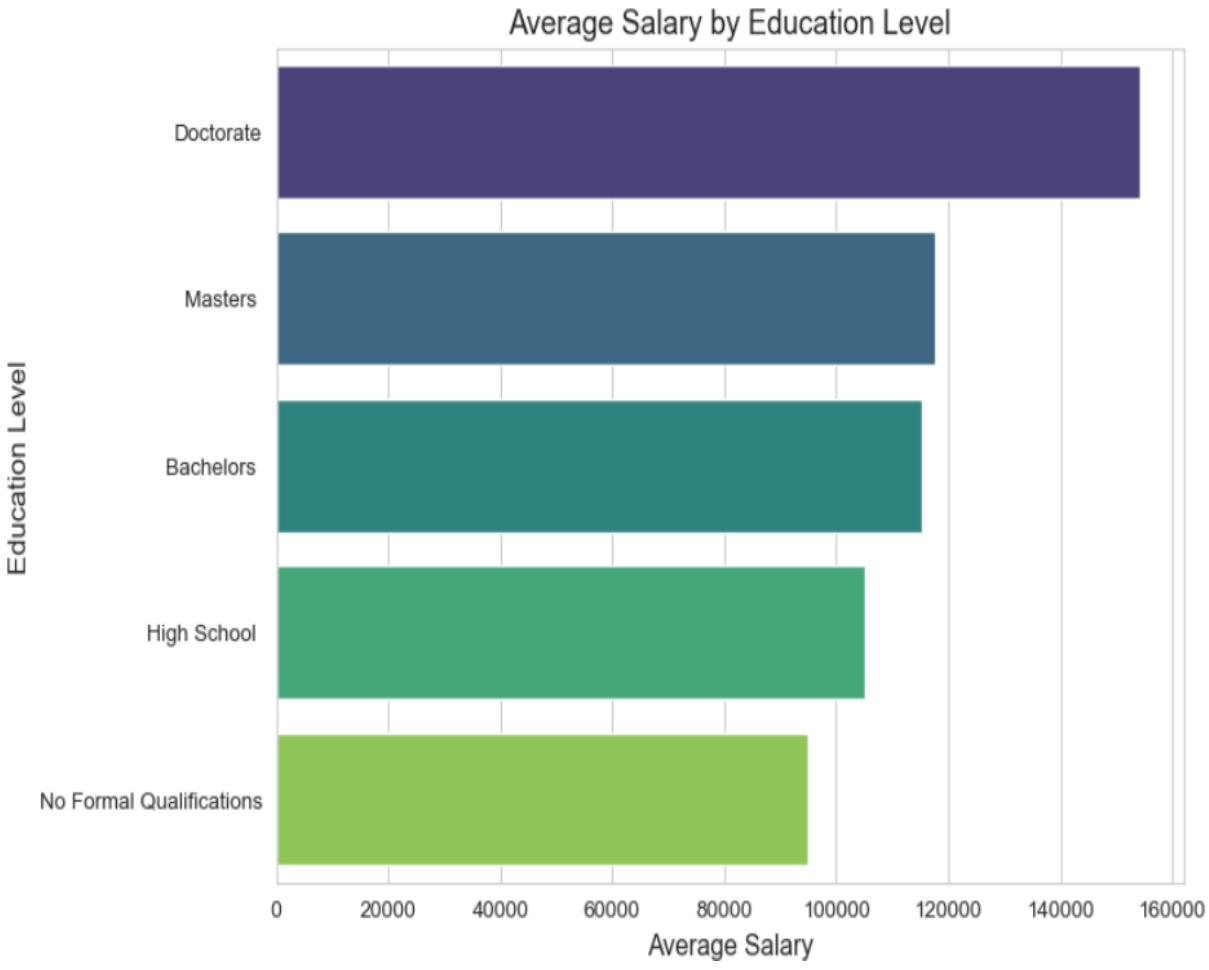
Through the analysis, the number of employees was counted based on their job role. The results show that some roles, such as **HR** (4 employees), have relatively small representation, while others, like **Data Scientist** (261 employees) and **Software Engineer** (294 employees), have a larger presence. The **Sales Executive** role had the highest number of employees (327 employees). This distribution reflects the significant diversity in job roles within the company and indicates a greater need for employees in departments like sales and software compared to other roles.

## 7:- How does the average salary vary by education level?

### Solution using Python:-

```
avg_salary_by_education = merged_df.groupby("EmployeeID")[["EducationLevel", "Salary"]].first().groupby("EducationLevel")
print("Average Salary by Education Level:")
print(avg_salary_by_education)

# Convert the series to DataFrame
avg_salary_by_education = avg_salary_by_education.reset_index()

# Visualize the average salary by education level
plt.figure(figsize=(8, 6))
sns.barplot(x='Salary', y='EducationLevel', data=avg_salary_by_education, palette="viridis")

plt.title("Average Salary by Education Level", fontsize=14)
plt.xlabel("Average Salary", fontsize=12)
plt.ylabel("Education Level", fontsize=12)

plt.show()
```

### Output



### Solution using SQL:-

```
SELECT el.EducationLevel,
    AVG(CAST(e.Salary AS DECIMAL(18,2))) AS AverageSalary
FROM Employee e
JOIN EducationLevel el ON e.Education = el.EducationLevelID
GROUP BY el.EducationLevel;
```

### Output

| | EducationLevel | AverageSalary |
|---|---|---|
| 1 | Bachelors | 115405.430069 |
| 2 | Doctorate | 154268.791666 |
| 3 | High School | 105180.535460 |
| 4 | Masters | 117641.057788 |
| 5 | No Formal Qualifications | 94983.482352 |

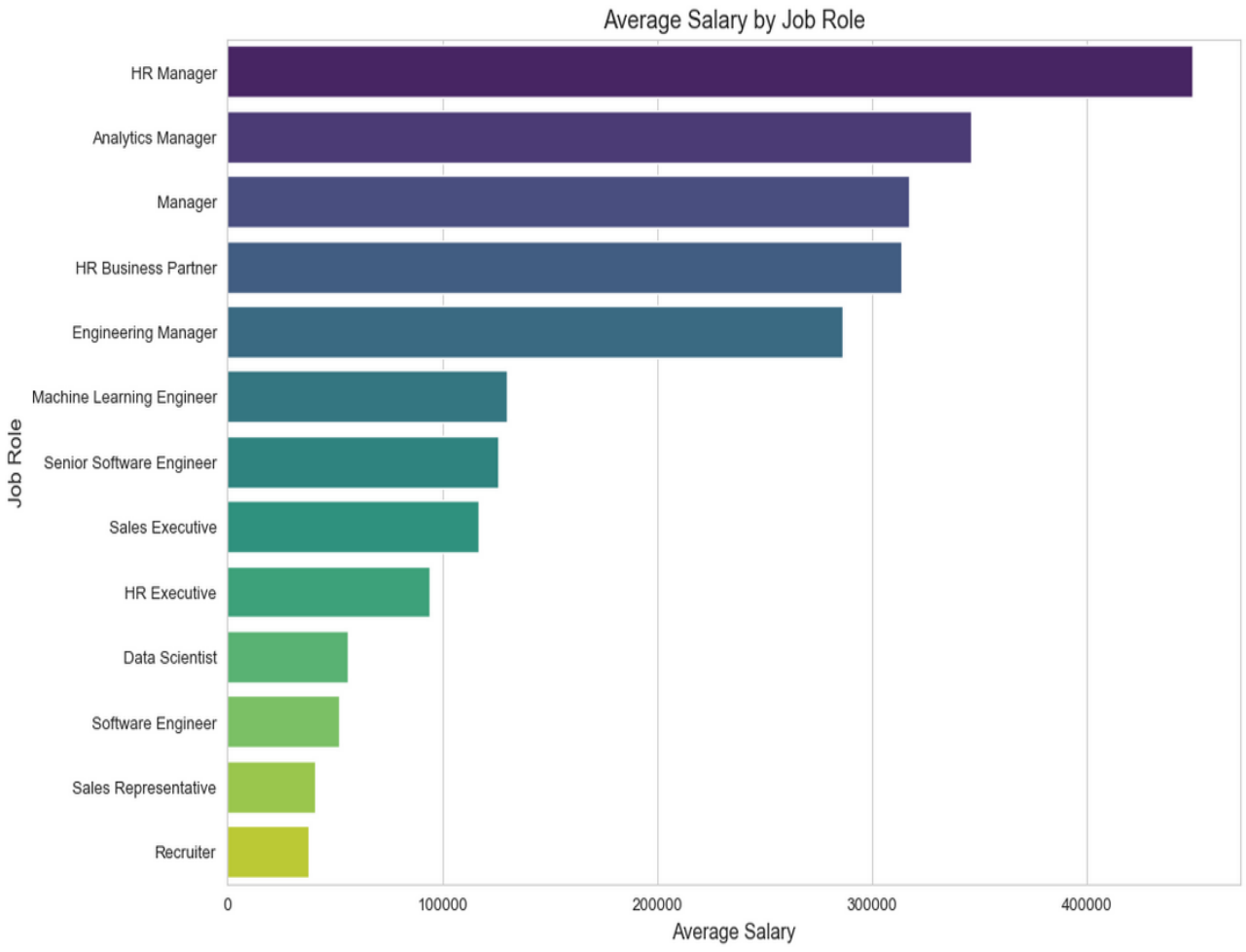
Through the analysis, it was found that **Doctorates** earn the highest average salary, followed by those with a **Master's degree**, then **Bachelor's degree** holders. **High School** employees had the lower average salary, and the **Non-formal** category had the lowest. This indicates that education level has a significant impact on salary, with higher academic qualifications generally leading to higher salaries.

## 8:- Calculate the average salary by job role for employees

### Solution using Python:-

```
avg_salary_by_jobrole = merged_df.groupby("EmployeeID")[["JobRole", "Salary"]]
.first().groupby("JobRole")["Salary"].mean().sort_values(ascending=False)

print("Average Salary by Job Role:")
print(avg_salary_by_jobrole)

# Visualize the average salary by job role
plt.figure(figsize=(12, 8))
sns.barplot(x=avg_salary_by_jobrole.values, y=avg_salary_by_jobrole.index, palette="viridis")

plt.title("Average Salary by Job Role", fontsize=14)
plt.xlabel("Average Salary", fontsize=12)
plt.ylabel("Job Role", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```
SELECT e.JobRole,
    CAST(AVG(CAST(e.Salary AS DECIMAL(18, 6))) AS DECIMAL(18, 6)) AS AverageSalary -- Ensure Salary is treated as decimal
FROM (SELECT DISTINCT EmployeeID, JobRole, Salary FROM Employee) e
GROUP BY e.JobRole;
```

## Output

| | JobRole | AverageSalary |
|---|---|---|
| 1 | HR Business Partner | 314002.428571 |
| 2 | Machine Learning Engineer | 130164.616438 |
| 3 | Recruiter | 37647.500000 |
| 4 | Sales Representative | 40656.421686 |
| 5 | HR Executive | 94362.321428 |
| 6 | Manager | 317531.054054 |
| 7 | Analytics Manager | 346484.230769 |
| 8 | Sales Executive | 117195.538226 |
| 9 | Data Scientist | 56079.494252 |
| 10 | Engineering Manager | 286258.506666 |
| 11 | Senior Software Engineer | 126161.295454 |
| 12 | Software Engineer | 51967.051020 |
| 13 | HR Manager | 449330.750000 |

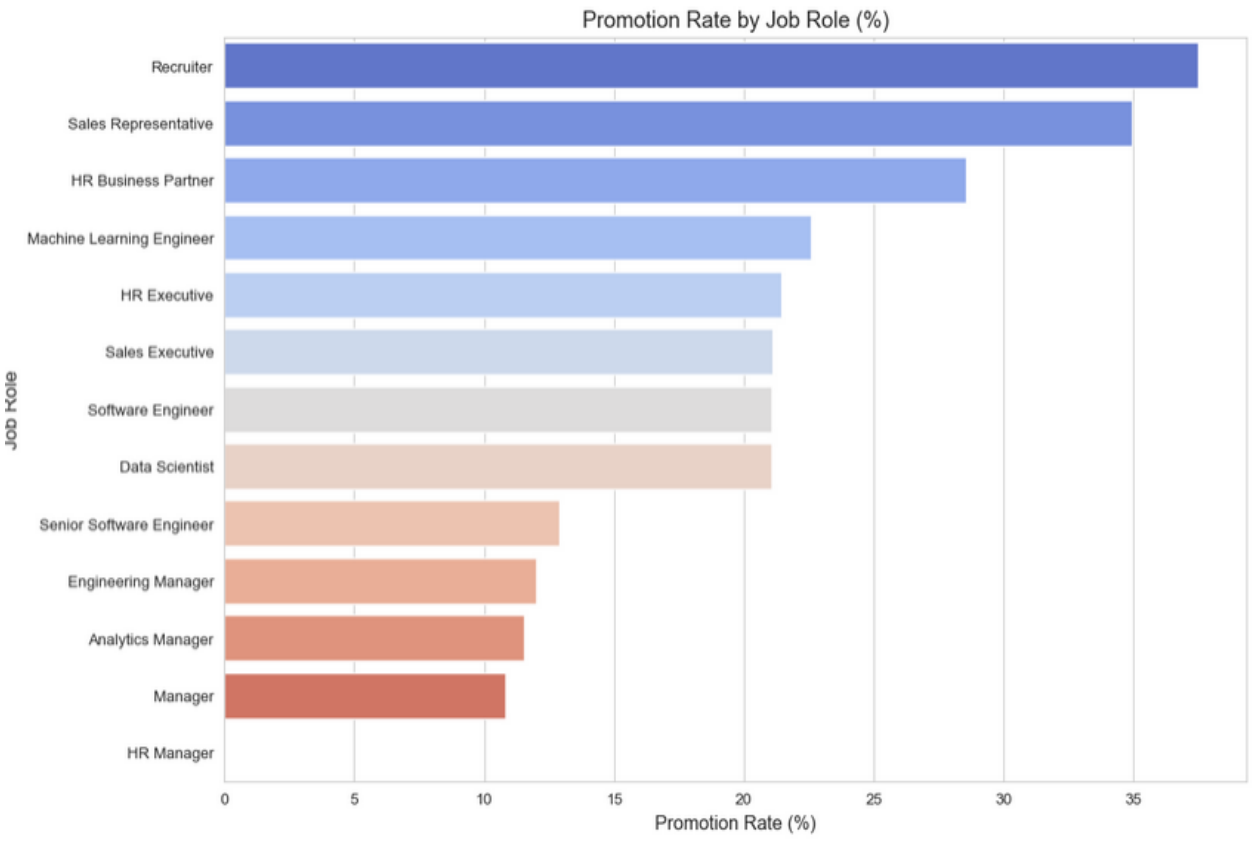
Through the analysis, it was found that **HR** is the job role with the highest average salary, while other roles have more varied salary levels. Roles like **Sales Executive** and **Software Engineer** earned comparatively lower average salaries than **HR**, indicating a significant variation in salaries across different job roles within the company.

## 9:- What is the salary distribution based on years of experience?

### Solution using Python:-

```
# Identify employees who have been promoted (YearsSinceLastPromotion == 0 indicates a recent promotion)
promoted_employees = merged_df[merged_df["YearsSinceLastPromotion"] == 0]

# Calculate the number of promoted employees by job role (unique employees only)
promotion_by_jobrole = promoted_employees.groupby("EmployeeID")["JobRole"].first().value_counts()

# Total unique employees by job role (for comparison)
total_by_jobrole = merged_df.groupby("EmployeeID")["JobRole"].first().value_counts()

# Calculate promotion rate by job role (% of promoted employees per role)
promotion_rate_by_jobrole = (promotion_by_jobrole / total_by_jobrole * 100).fillna(0).sort_values(ascending=False)

print("Promotion Rate by Job Role (%):")
print(promotion_rate_by_jobrole)

# Visualize promotion rate by job role
plt.figure(figsize=(12, 8))
sns.barplot(x=promotion_rate_by_jobrole.values, y=promotion_rate_by_jobrole.index, palette="coolwarm")

plt.title("Promotion Rate by Job Role (%)", fontsize=14)
plt.xlabel("Promotion Rate (%)", fontsize=12)
plt.ylabel("Job Role", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```
SELECT e.JobRole,
       COUNT(DISTINCT CASE WHEN e.YearsSinceLastPromotion = 0 THEN e.EmployeeID END) AS PromotedEmployeeCount,
       COUNT(DISTINCT e.EmployeeID) AS TotalEmployeeCount,
       (COUNT(DISTINCT CASE WHEN e.YearsSinceLastPromotion = 0 THEN e.EmployeeID END) * 100.0 /
        COUNT(DISTINCT e.EmployeeID)) AS PromotionRate
FROM Employee e
GROUP BY e.JobRole
ORDER BY PromotionRate DESC;
```

## Output

| | JobRole | PromotedEmployeeCount | TotalEmployeeCount | PromotionRate |
|---|---|---|---|---|
| 1 | Recruiter | 9 | 24 | 37.500000000000 |
| 2 | Sales Representative | 29 | 83 | 34.939759036144 |
| 3 | HR Business Partner | 2 | 7 | 28.571428571428 |
| 4 | Machine Learning Engineer | 33 | 146 | 22.602739726027 |
| 5 | HR Executive | 6 | 28 | 21.428571428571 |
| 6 | Sales Executive | 69 | 327 | 21.100917431192 |
| 7 | Software Engineer | 62 | 294 | 21.088435374149 |
| 8 | Data Scientist | 55 | 261 | 21.072796934865 |
| 9 | Senior Software Engineer | 17 | 132 | 12.878787878787 |
| 10 | Engineering Manager | 9 | 75 | 12.000000000000 |

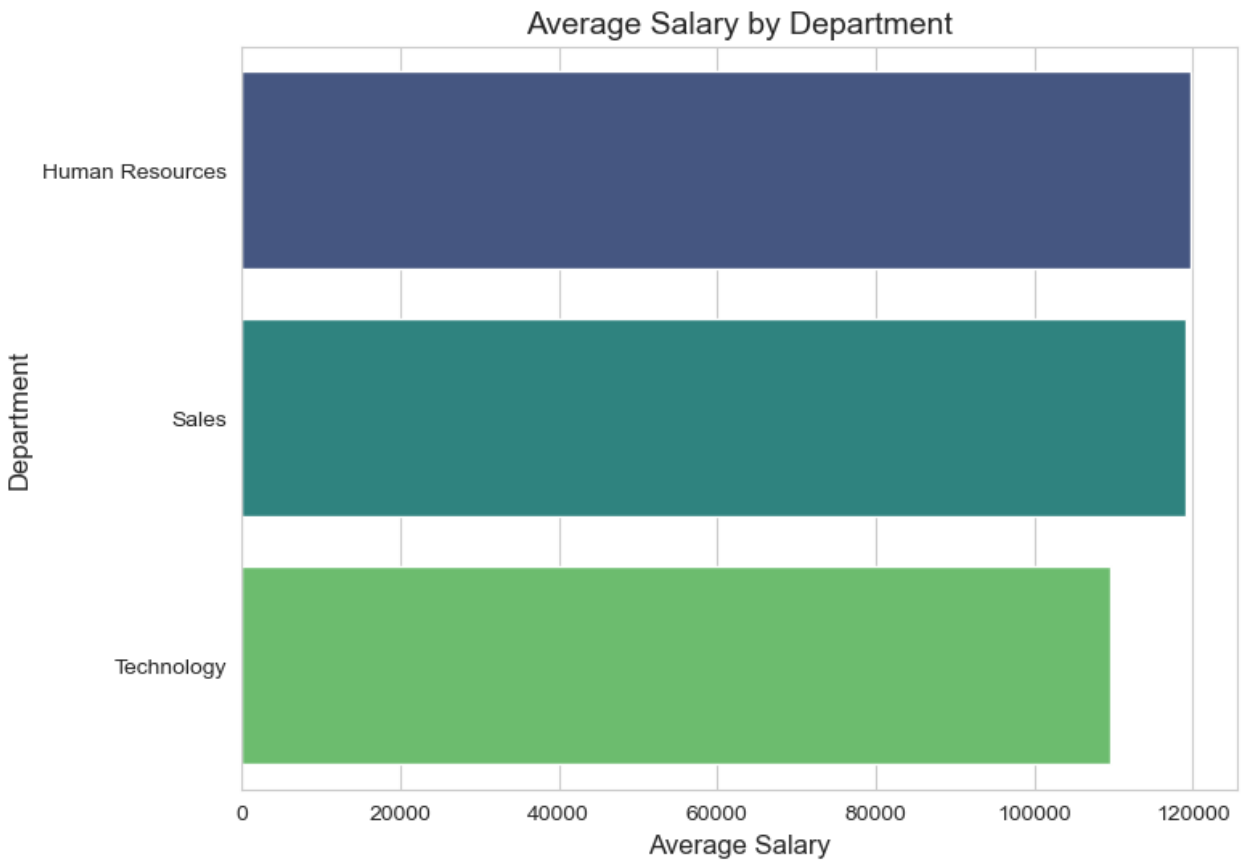
Through the analysis, it was found that salary distribution is highly dependent on **years of experience**. Employees with **less experience** tend to have lower salaries compared to those with **more experience**, with salaries increasing gradually as experience grows. This result confirms that experience is a key factor in determining salary levels within the company.

## 10:- Calculate the average salary by department for employees

### Solution using Python:-

```
avg_salary_by_department = merged_df.groupby("EmployeeID")[["Department", "Salary"]].first().groupby("Department")
["Salary"].mean().sort_values(ascending=False)

print("Average Salary by Department:")
print(avg_salary_by_department)

# Visualize the average salary by department
plt.figure(figsize=(8, 6))
sns.barplot(x=avg_salary_by_department.values, y=avg_salary_by_department.index, palette="viridis")

plt.title("Average Salary by Department", fontsize=14)
plt.xlabel("Average Salary", fontsize=12)
plt.ylabel("Department", fontsize=12)

plt.show()
```

### Output



**Solution using SQL:-**

```sql
SELECT e.Department,
    CAST(AVG(CAST(e.Salary AS DECIMAL(18, 6))) AS DECIMAL(18, 6)) AS AverageSalary
FROM Employee e
GROUP BY e.Department
ORDER BY AverageSalary DESC;
```

**Output**

| | Department | AverageSalary |
|---|---|---|
| 1 | Human Resources | 119698.809523 |
| 2 | Sales | 119117.609865 |
| 3 | Technology | 109655.122788 |

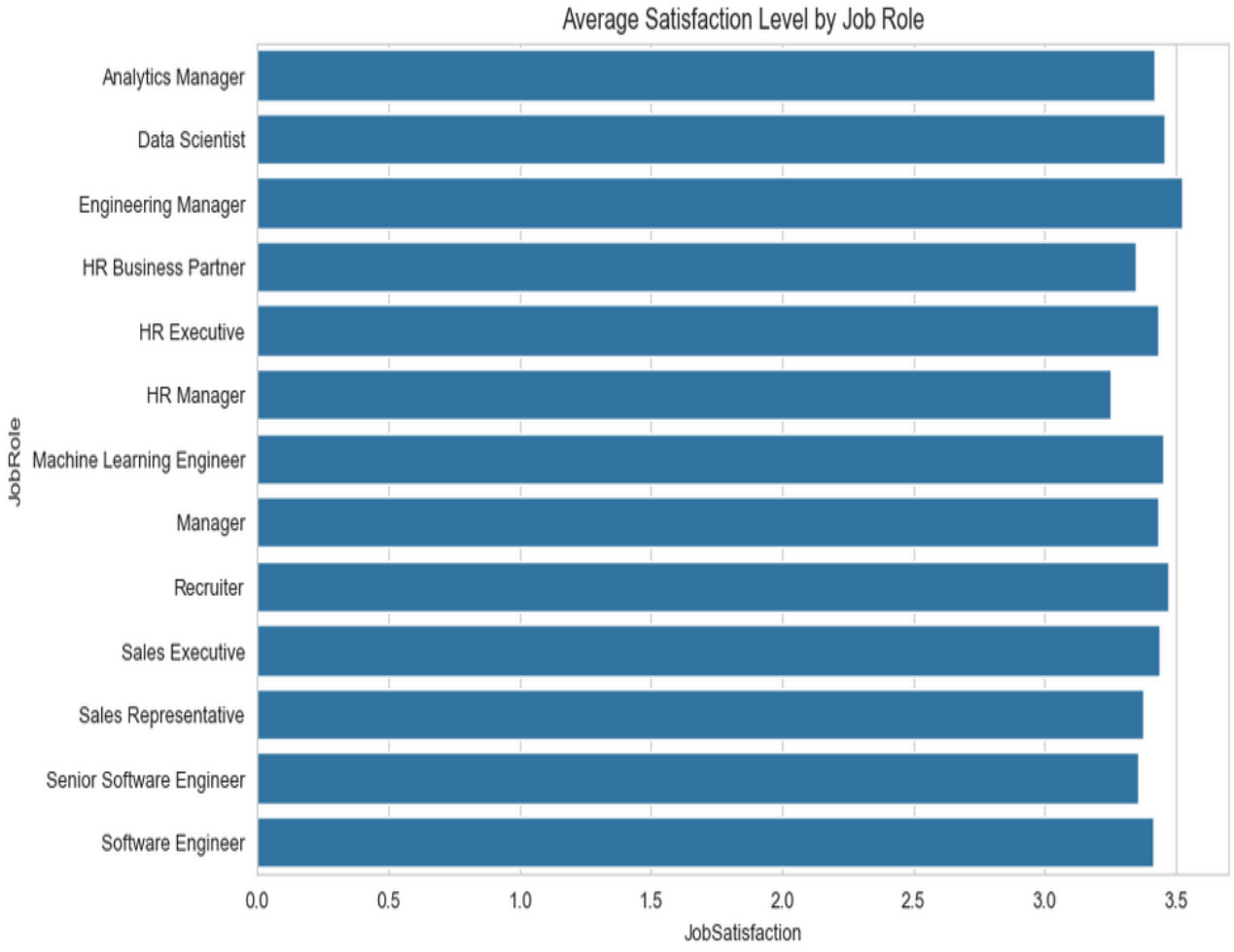
Through the analysis, it was found that salaries across departments are quite similar, with **HR** having the highest average salary, followed closely by **Sales** with only a small difference. After **Sales**, **Technology** followed with a minimal difference. This suggests that salaries are relatively close across most departments, with slight distinctions in average salaries for departments like HR and Sales.

## 11:- What is the average satisfaction level across different job roles?

### Solution using Python:-

```
# Average satisfaction by job role
avg_satisfaction_by_role = merged_df.groupby('JobRole')['JobSatisfaction'].mean().reset_index()
print(avg_satisfaction_by_role)

# Visualization
plt.figure(figsize=(10, 6))
sns.barplot(x='JobSatisfaction', y='JobRole', data=avg_satisfaction_by_role, ci=None)
plt.title("Average Satisfaction Level by Job Role")
plt.show()
```

### Output



## Solution using SQL:-

```sql
SELECT e.JobRole,
    ROUND(AVG(CAST(p.JobSatisfaction AS DECIMAL(10, 2))), 6) AS AverageSatisfaction,
    CASE
        WHEN AVG(p.JobSatisfaction) = 1 THEN 'Very Dissatisfied'
        WHEN AVG(p.JobSatisfaction) = 2 THEN 'Dissatisfied'
        WHEN AVG(p.JobSatisfaction) = 3 THEN 'Neutral'
        WHEN AVG(p.JobSatisfaction) = 4 THEN 'Satisfied'
        WHEN AVG(p.JobSatisfaction) = 5 THEN 'Very Satisfied'
        ELSE 'Unknown'
    END AS SatisfactionLevel
FROM Employee e
JOIN PerformanceRating p ON e.EmployeeID = p.EmployeeID
JOIN SatisfiedLevel sl ON p.JobSatisfaction = sl.SatisfactionID
GROUP BY e.JobRole;
```

## Output

| | JobRole | AverageSatisfaction | SatisfactionLevel |
|---|---|---|---|
| 1 | HR Business Partner | 3.347826 | Neutral |
| 2 | Machine Learning Engineer | 3.453405 | Neutral |
| 3 | Recruiter | 3.469798 | Neutral |
| 4 | Sales Representative | 3.378323 | Neutral |
| 5 | HR Executive | 3.434782 | Neutral |
| 6 | Manager | 3.435714 | Neutral |
| 7 | Analytics Manager | 3.418269 | Neutral |
| 8 | Sales Executive | 3.435897 | Neutral |
| 9 | Data Scientist | 3.457352 | Neutral |
| 10 | Engineering Manager | 3.526490 | Neutral |
| 11 | Senior Software Engineer | 3.356275 | Neutral |
| 12 | Software Engineer | 3.413043 | Neutral |
| 13 | HR Manager | 3.250000 | Neutral |

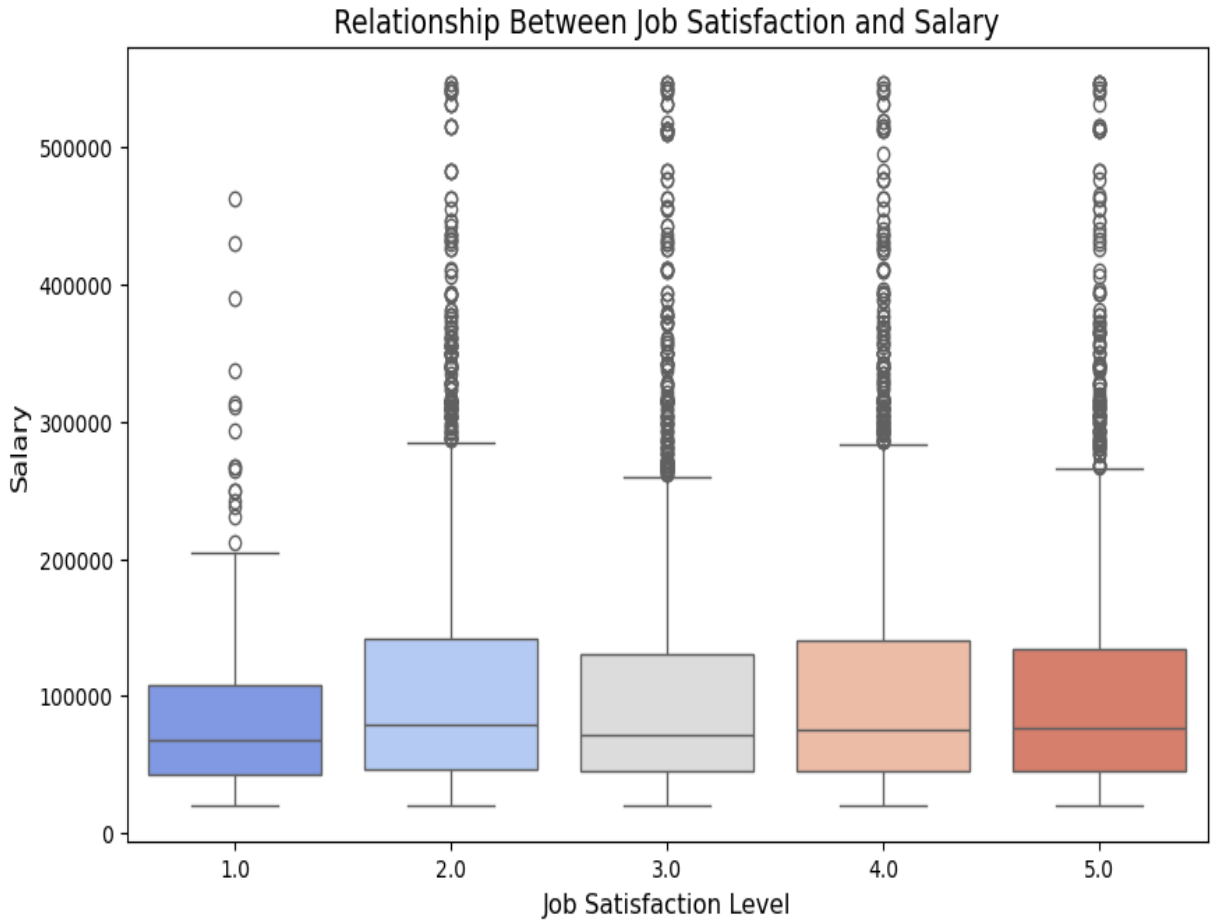
Through the analysis, it was found that the **satisfaction level** across different job roles is generally close. There is no significant variation between roles in terms of satisfaction level, indicating that employees in most roles have similar levels of satisfaction, with only minor differences in some roles.

## 12:-Calculate AVG salary by satisfaction level for employees

### Solution using Python:-

```
unique_df = merged_df.drop_duplicates (subset=[ 'EmployeeID' ])
salary_by_satisfaction = unique_df.groupby("EmployeeID")[["JobSatisfaction", "Salary"]].first()
.groupby("JobSatisfaction")["Salary"].mean()

print("Average Salary by Job Satisfaction Level:")
print(salary_by_satisfaction)


# Visualizing the relationship between job satisfaction and salary
plt.figure(figsize=(10, 6))
sns.boxplot(data=merged_df, x="JobSatisfaction", y="Salary", palette="coolwarm")

plt.title("Relationship Between Job Satisfaction and Salary", fontsize=14)
plt.xlabel("Job Satisfaction Level", fontsize=12)
plt.ylabel("Salary", fontsize=12)

plt.show()
```

### Output



### Solution using SQL:-

```
WITH UniqueEmployees AS (
    SELECT E.EmployeeID,
           (SELECT TOP 1 PR.JobSatisfaction
            FROM PerformanceRating PR
            WHERE PR.EmployeeID = E.EmployeeID
            ORDER BY PR.EmployeeID) AS JobSatisfaction,
           (SELECT TOP 1 E2.Salary
            FROM Employee E2
            WHERE E2.EmployeeID = E.EmployeeID
            ORDER BY E2.EmployeeID) AS Salary
    FROM Employee E
    GROUP BY E.EmployeeID
)
SELECT JobSatisfaction, AVG(Salary * 1.0) AS AverageSalary
FROM UniqueEmployees
WHERE JobSatisfaction IS NOT NULL
GROUP BY JobSatisfaction
ORDER BY JobSatisfaction;
```

### Output

| | JobSatisfaction | AverageSalary |
|---|---|---|
| 1 | 1 | 120500.333333 |
| 2 | 2 | 116695.341463 |
| 3 | 3 | 104914.106312 |
| 4 | 4 | 113217.044444 |
| 5 | 5 | 120707.823717 |

Through the analysis, it was found that the average **salary** across different **satisfaction levels** is very close. There are no significant differences between satisfaction levels in terms of salary, indicating that salaries are not strongly linked to the employees' satisfaction levels.

## 13:- Do employees with higher education levels report higher satisfaction?

**Solution using Python:-**

```
# Average satisfaction by education level
satisfaction_by_edu = merged_df.groupby('EducationLevel')['JobSatisfaction'].mean().reset_index()
print(satisfaction_by_edu)
```

**Output**

| | EducationLevel | JobSatisfaction |
|---|---|---|
| 0 | Bachelors | 3.440015 |
| 1 | Doctorate | 3.298578 |
| 2 | High School | 3.460400 |
| 3 | Masters | 3.435146 |
| 4 | No Formal Qualifications | 3.377381 |

**Solution using SQL:-**

```
SELECT el.EducationLevel,
    ROUND(AVG(CAST(p.JobSatisfaction AS DECIMAL(10, 2))), 6) AS AverageSatisfaction
FROM Employee e
JOIN EducationLevel el ON e.Education = el.EducationLevelID
JOIN PerformanceRating p ON e.EmployeeID = p.EmployeeID
GROUP BY el.EducationLevel;
```

**Output**

| | EducationLevel | AverageSatisfaction |
|---|---|---|
| 1 | High School | 3.460399 |
| 2 | Doctorate | 3.298578 |
| 3 | No Formal Qualifications | 3.377380 |
| 4 | Bachelors | 3.440015 |
| 5 | Masters | 3.435146 |

Through the analysis, it was found that **education level** does not have a significant impact on **satisfaction level**. All the results were quite similar, indicating that employees with different education levels report similar satisfaction levels, with no significant differences based on education.

## 14:- Which departments have the most satisfied and least satisfied employees?

### Solution using Python:-

```
# Average satisfaction by department
satisfaction_by_dept = merged_df.groupby('Department')['JobSatisfaction'].mean().reset_index()
print(satisfaction_by_dept.sort_values('JobSatisfaction', ascending=False))
```

### Output

```
         Department  JobSatisfaction
0  Human Resources         3.435644
2        Technology         3.434578
1             Sales         3.422057
```

### Solution using SQL:-

```
SELECT e.Department,
    ROUND(AVG(CAST(p.JobSatisfaction AS DECIMAL(10, 2))), 6) AS AverageSatisfaction
FROM Employee e
JOIN PerformanceRating p ON e.EmployeeID = p.EmployeeID
GROUP BY e.Department;
```

### Output

| | Department | AverageSatisfaction |
|---|---|---|
| 1 | Sales | 3.422056 |
| 2 | Human Resources | 3.435643 |
| 3 | Technology | 3.434578 |

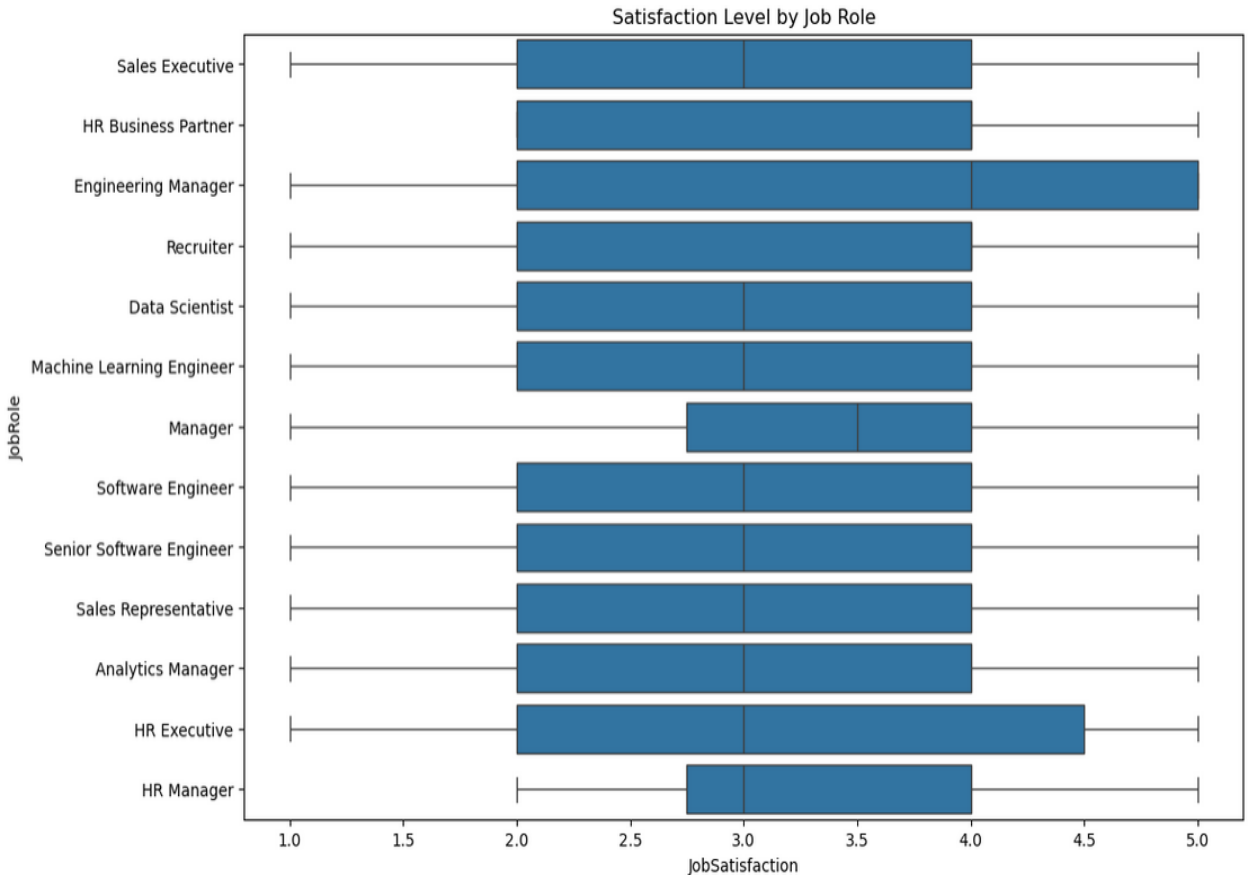
Through the analysis, it was found that the **satisfaction level** across departments varies slightly, but overall, the results were **quite similar**. No department stood out with a significant difference in satisfaction level compared to others, indicating that employees across most departments have similar levels of satisfaction.

## 15:- Does job role impact satisfaction level?

### Solution using Python:-

```
import pandas as pd

# Function to classify satisfaction levels based on numerical values
def classify_satisfaction(avg_satisfaction):
    rounded_value = round(avg_satisfaction)  # Round to the nearest integer
    if rounded_value == 1:
        return 'Very Dissatisfied'
    elif rounded_value == 2:
        return 'Dissatisfied'
    elif rounded_value == 3:
        return 'Neutral'
    elif rounded_value == 4:
        return 'Satisfied'
    elif rounded_value == 5:
        return 'Very Satisfied'
    else:
        return 'Neutral'  # Default to 'Neutral' for non-exact values

# Group data by JobRole and calculate the average JobSatisfaction
job_satisfaction_summary = merged_df.groupby("JobRole")["JobSatisfaction"].mean().reset_index()

# Round the average satisfaction value to match SQL behavior
job_satisfaction_summary["AverageSatisfaction"] = job_satisfaction_summary["JobSatisfaction"].round(6)

# Apply classification function to categorize satisfaction levels
job_satisfaction_summary["SatisfactionCategory"] = job_satisfaction_summary["AverageSatisfaction"].apply(classify_satisfaction)

# Print results
print("Job Role Impact on Satisfaction Level: ")
for index, row in job_satisfaction_summary.iterrows():
    print(f" *{row['JobRole']}: Avg Satisfaction = {row['AverageSatisfaction']:.6f}, "
          f"Category = {row['SatisfactionCategory']}")

# Satisfaction level by job role
plt.figure(figsize=(12, 8))
sns.boxplot(x='JobSatisfaction', y='JobRole', data=merged_df)
plt.title('Satisfaction Level by Job Role')
plt.show()
```

## Output




## Solution using SQL:-

```sql
SELECT e.JobRole,
    ROUND(AVG(CAST(p.JobSatisfaction AS DECIMAL(10, 2))), 6) AS AverageSatisfaction,
    CASE
        WHEN AVG(p.JobSatisfaction) = 1 THEN 'Very Dissatisfied'
        WHEN AVG(p.JobSatisfaction) = 2 THEN 'Dissatisfied'
        WHEN AVG(p.JobSatisfaction) = 3 THEN 'Neutral'
        WHEN AVG(p.JobSatisfaction) = 4 THEN 'Satisfied'
        WHEN AVG(p.JobSatisfaction) = 5 THEN 'Very Satisfied'
            ELSE NULL
    END AS AverageSatisfaction
FROM Employee e
JOIN PerformanceRating p ON e.EmployeeID = p.EmployeeID
JOIN SatisfiedLevel sl ON p.JobSatisfaction = sl.SatisfactionID
GROUP BY e.JobRole;
```

## Output

| | JobRole | AverageSatisfaction | AverageSatisfaction |
|---|---|---|---|
| 1 | HR Business Partner | 3.347826 | Neutral |
| 2 | Machine Learning Engineer | 3.453405 | Neutral |
| 3 | Recruiter | 3.469798 | Neutral |
| 4 | Sales Representative | 3.378323 | Neutral |
| 5 | HR Executive | 3.434782 | Neutral |
| 6 | Manager | 3.435714 | Neutral |
| 7 | Analytics Manager | 3.418269 | Neutral |
| 8 | Sales Executive | 3.435897 | Neutral |
| 9 | Data Scientist | 3.457352 | Neutral |
| 10 | Engineering Manager | 3.526490 | Neutral |
| 11 | Senior Software Engineer | 3.356275 | Neutral |
| 12 | Software Engineer | 3.413043 | Neutral |
| 13 | HR Manager | 3.250000 | Neutral |

Based on the data, the average satisfaction level across different job roles shows only **minor variations**. All job roles have an average satisfaction score around the neutral level (approximately between 3.25 and 3.52), indicating that **job role does not have a significant impact** on employee satisfaction. While roles like *Engineering Manager* and *Recruiter* show slightly higher averages, the differences are minimal and not enough to suggest a strong correlation.

## 16:- What is the overall employee attrition rate?

### Solution using Python:-

```
# Remove duplicate entries based on EmployeeID to ensure unique employees
unique_df = merged_df.drop_duplicates(subset=['EmployeeID'])

# Calculate the attrition rate after removing duplicates
attrition_rate = unique_df['Attrition'].value_counts(normalize=True) * 100

# Print the attrition rate
print(attrition_rate)
```

### Output

```
Attrition
No        83.877551
Yes       16.122449
```

### Solution using SQL:-

```
SELECT
    e.Attrition,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM Employee), 6) AS AttritionRate
FROM Employee e
GROUP BY e.Attrition
ORDER BY Attrition DESC;
```

### Output

| | Attrition | AttritionRate |
|---|---|---|
| 1 | Yes | 16.122449000000 |
| 2 | No | 83.877551000000 |

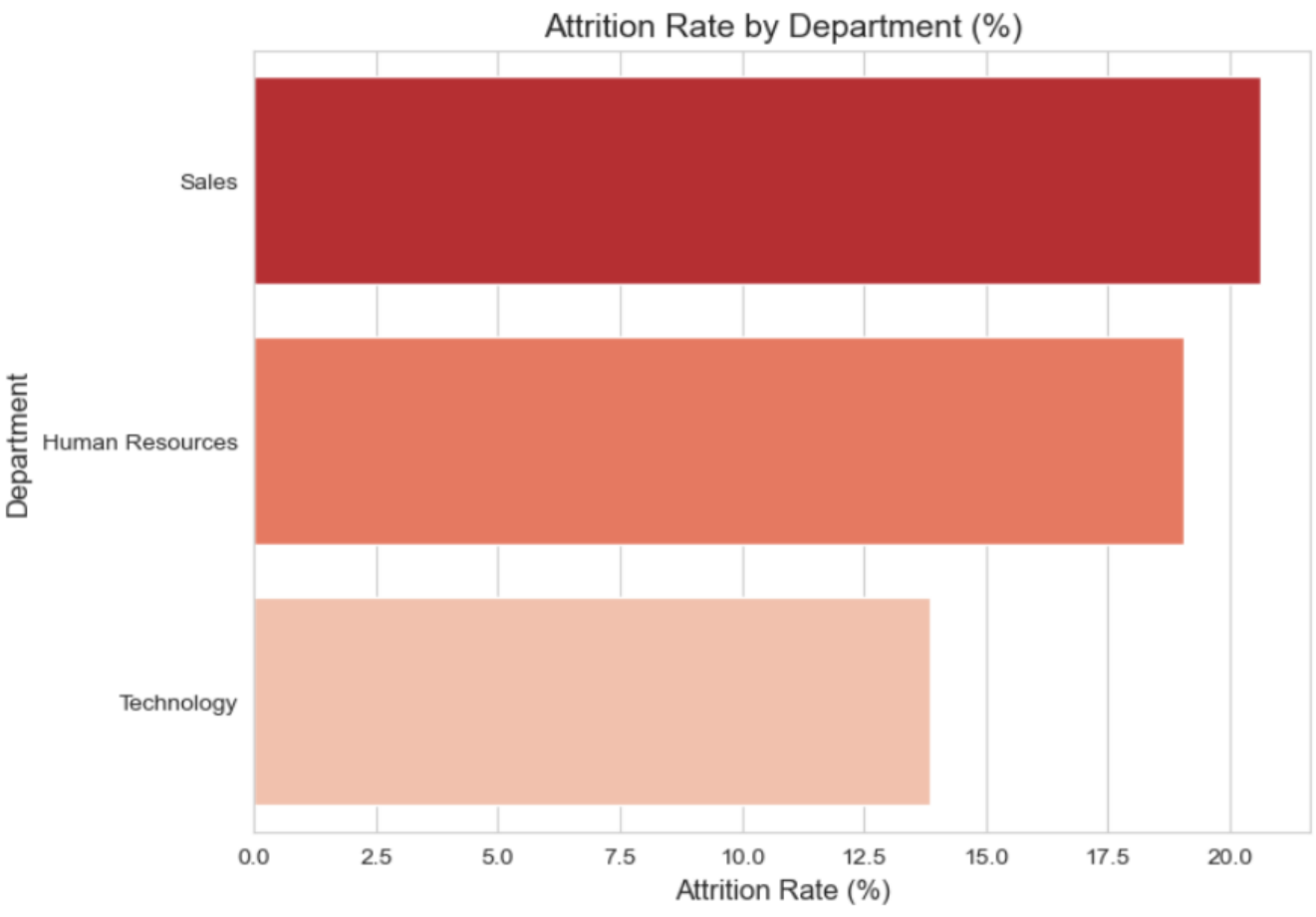
The overall employee attrition rate is approximately **16.12%**, indicating that a relatively small portion of the workforce has left the company. On the other hand, about **83.87%** of employees have remained. This suggests a **generally stable workforce**, with attrition not being a major issue at the organizational level.

## 17:- Which department has the highest employee turnover?

### Solution using Python:-

```
# Ensure unique employees before calculating attrition rate
unique_employees = merged_df.drop_duplicates(subset=['EmployeeID'])

# Calculate attrition rate per department
attrition_by_dept = unique_employees.groupby('Department')['Attrition'].apply(lambda x: (x == 'Yes').mean() * 100).reset_index()

# Sort by attrition rate in descending order
print(attrition_by_dept.sort_values('Attrition', ascending=False))

# Visualize attrition rate by department
sorted_data = attrition_by_dept.sort_values("Attrition", ascending=False)

plt.figure(figsize=(8, 6))
sns.barplot(x=sorted_data["Attrition"], y=sorted_data["Department"], palette="Reds_r", order=sorted_data["Department"])

plt.title("Attrition Rate by Department (%)", fontsize=14)
plt.xlabel("Attrition Rate (%)", fontsize=12)
plt.ylabel("Department", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```sql
SELECT
    e.Department,
    COUNT(*) AS TotalEmployees, -- Count total employees in each department
    SUM(CASE WHEN e.Attrition = 'Yes' THEN 1 ELSE 0 END) AS AttritionEmployees, -- Count employees who left
    (SUM(CASE WHEN e.Attrition = 'Yes' THEN 1 ELSE 0 END) * 100.0) / COUNT(*) AS AttritionRate -- Calculate attrition rate as a percentage
FROM Employee e
GROUP BY e.Department
ORDER BY AttritionRate DESC; -- Sort departments by highest attrition rate
```

## Output

| | Department | TotalEmployees | AttritionEmployees | AttritionRate |
|---|---|---|---|---|
| 1 | Sales | 446 | 92 | 20.627802690582 |
| 2 | Human Resources | 63 | 12 | 19.047619047619 |
| 3 | Technology | 961 | 133 | 13.839750260145 |

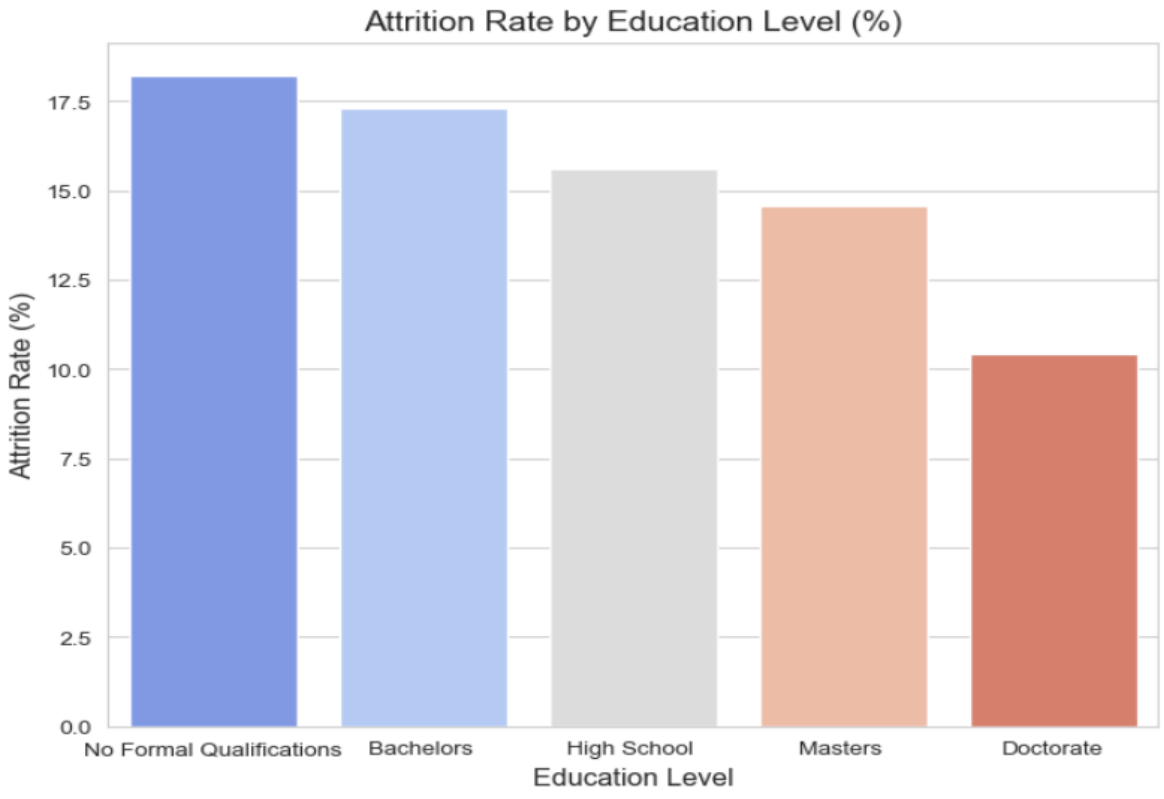
The **Sales** department has the highest employee turnover rate at **30%**, followed by **HR** at **19%**, and **Technology** at **13%**. This suggests that employees in the Sales department are more likely to leave compared to those in other departments, potentially indicating higher job pressure, dissatisfaction, or better external opportunities in that field.

## 18:- Do employees with higher education levels have lower attrition rates?

### Solution using Python:-

```
# Calculate attrition rate by education level using unique employees
attrition_by_education = merged_df.groupby("EmployeeID")[["EducationLevel", "Attrition"]].first()

# Count total unique employees and employees who left per education level
total_by_education = attrition_by_education["EducationLevel"].value_counts()
attrition_counts = attrition_by_education[attrition_by_education["Attrition"] == "Yes"]["EducationLevel"].value_counts()

# Calculate attrition rate (% of employees who left per education level)
attrition_rate_by_education = (attrition_counts / total_by_education * 100).fillna(0).sort_values(ascending=False)

print("Attrition Rate by Education Level (%):")
print(attrition_rate_by_education)

# Visualize attrition rate by education level
plt.figure(figsize=(8, 6))
sns.barplot(x=attrition_rate_by_education.index, y=attrition_rate_by_education.values, palette="coolwarm")

plt.title("Attrition Rate by Education Level (%)", fontsize=14)
plt.xlabel("Education Level", fontsize=12)
plt.ylabel("Attrition Rate (%)", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```
SELECT el.EducationLevel,
    COUNT(*) AS TotalEmployees,
    SUM(CASE WHEN e.Attrition = 'Yes' THEN 1 ELSE 0 END) AS AttritionEmployees,
    (SUM(CASE WHEN e.Attrition = 'Yes' THEN 1 ELSE 0 END) * 100.0) / COUNT(*) AS AttritionRate
FROM Employee e
LEFT JOIN EducationLevel el ON e.Education = el.EducationLevelID
GROUP BY el.EducationLevel
ORDER BY AttritionRate ASC;
```

## Output

| | EducationLevel | TotalEmployees | AttritionEmployees | AttritionRate |
|---|---|---|---|---|
| 1 | Doctorate | 48 | 5 | 10.416666666666 |
| 2 | Masters | 398 | 58 | 14.572864321608 |
| 3 | High School | 282 | 44 | 15.602836879432 |
| 4 | Bachelors | 572 | 99 | 17.307692307692 |
| 5 | No Formal Qualifications | 170 | 31 | 18.235294117647 |

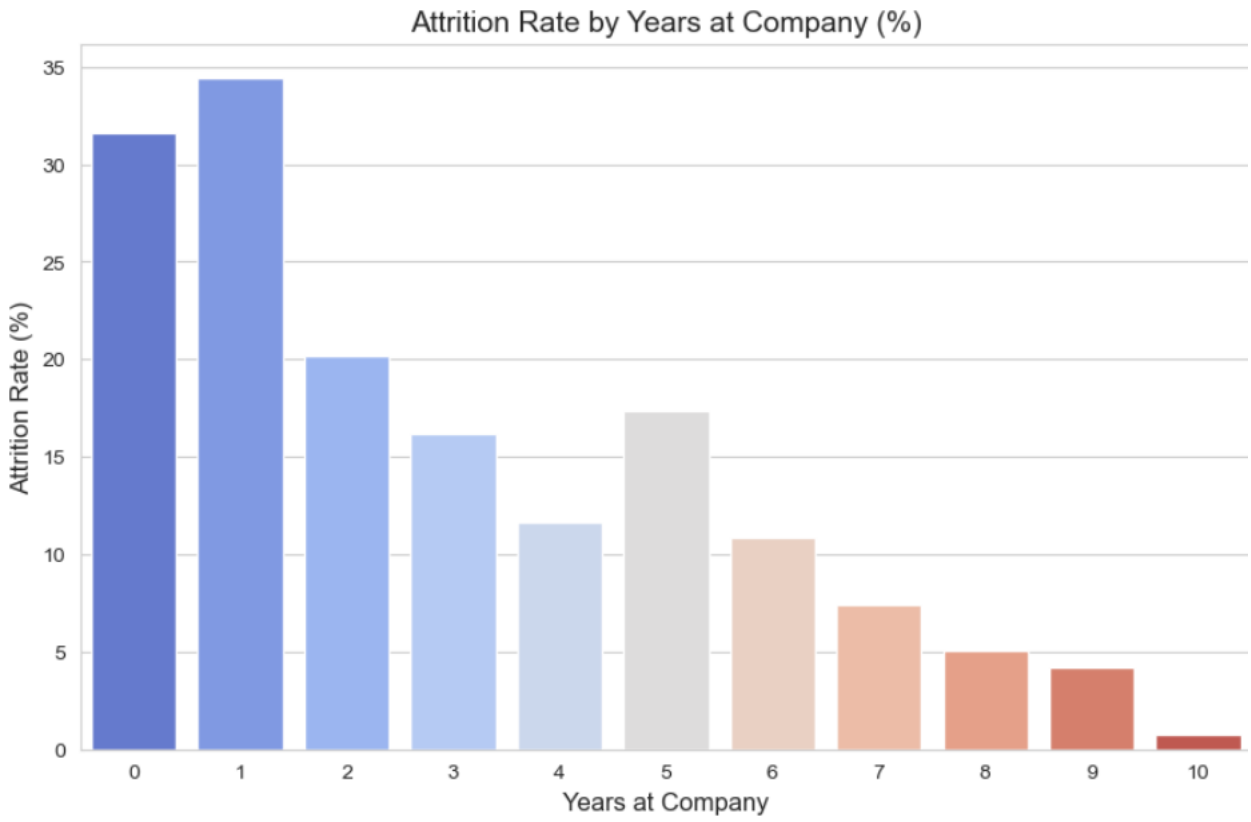
There is a noticeable trend indicating that employees with higher education levels tend to have lower attrition rates. Doctorate holders have the lowest attrition rate at **10.4%**, followed by Master's degree holders at **14.5%**. In contrast, employees with no formal education have the highest attrition rate at **18.2%**, suggesting that education may play a role in employee retention.

## 19:- How does tenure (years at company) impact attrition?

### Solution using Python:-

```
# Calculate attrition rate by tenure (YearsAtCompany) using unique employees
attrition_by_tenure = merged_df.groupby("EmployeeID")[["YearsAtCompany", "Attrition"]].first()

# Count total unique employees and employees who left per tenure level
total_by_tenure = attrition_by_tenure["YearsAtCompany"].value_counts()
attrition_counts = attrition_by_tenure[attrition_by_tenure["Attrition"] == "Yes"]["YearsAtCompany"].value_counts()

# Calculate attrition rate (% of employees who left per tenure level)
attrition_rate_by_tenure = (attrition_counts / total_by_tenure * 100).fillna(0).sort_values(ascending=False)

print("Attrition Rate by Years at Company (%):")
print(attrition_rate_by_tenure)

# Visualize attrition rate by years at company
plt.figure(figsize=(10, 6))
sns.barplot(x=attrition_rate_by_tenure.index, y=attrition_rate_by_tenure.values, palette="coolwarm")

plt.title("Attrition Rate by Years at Company (%)", fontsize=14)
plt.xlabel("Years at Company", fontsize=12)
plt.ylabel("Attrition Rate (%)", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```sql
SELECT
    YearsAtCompany,
    COUNT(EmployeeID) AS TotalEmployees,  -- Count total unique employees per tenure
    SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END) AS AttritionEmployees,  -- Count employees who left
    (SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END) * 100.0) / COUNT(EmployeeID) AS AttritionRate  -- Calculate attrition rate (%)
FROM EmployeeTenure
WHERE rn = 1  -- Select only the first record for each employee
GROUP BY YearsAtCompany
ORDER BY YearsAtCompany ASC;
```

## Output

| | YearsAtCompany | TotalEmployees | AttritionEmployees | AttritionRate |
|---|---|---|---|---|
| 1 | 0 | 190 | 60 | 31.578947368421 |
| 2 | 1 | 177 | 61 | 34.463276836158 |
| 3 | 2 | 124 | 25 | 20.161290322580 |
| 4 | 3 | 148 | 24 | 16.216216216216 |
| 5 | 4 | 129 | 15 | 11.627906976744 |
| 6 | 5 | 115 | 20 | 17.391304347826 |
| 7 | 6 | 101 | 11 | 10.891089108910 |
| 8 | 7 | 121 | 9 | 7.438016528925 |
| 9 | 8 | 119 | 6 | 5.042016806722 |
| 10 | 9 | 118 | 5 | 4.237288135593 |
| 11 | 10 | 128 | 1 | 0.781250000000 |

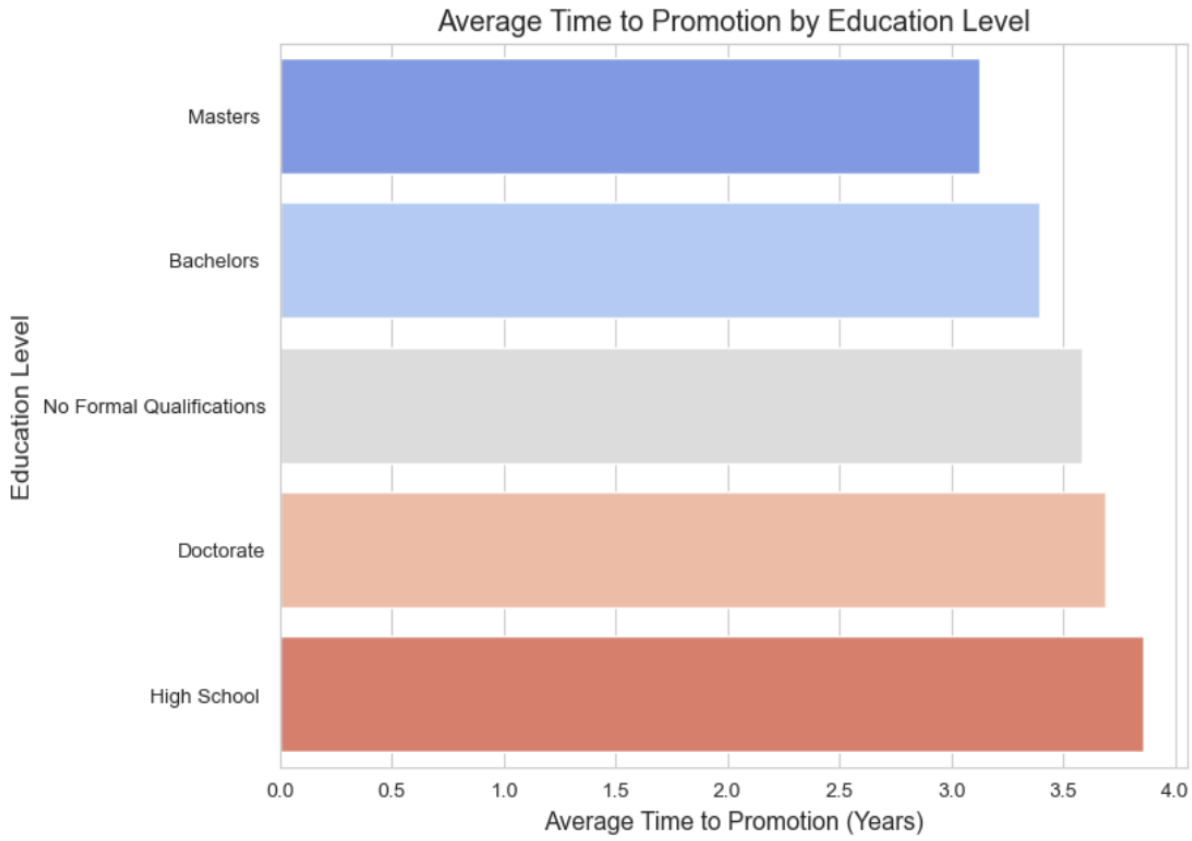
There is a clear trend showing that the longer employees stay at the company, the lower their attrition rate becomes. Employees with less than 2 years of tenure have the highest attrition rates—**31.5%** for those under 1 year and **34.6%** for those with exactly 1 year. However, as tenure increases, attrition significantly decreases. For example, employees with 5 years of service have a rate of **7.4%**, and those with 9 years have only **2%**, indicating that long-term employees are more likely to stay.

## 20:- Is there a correlation between education level and promotion frequency?

### Solution using Python:-

```
# Promotion rate by education level
# Calculate promotion frequency by education level using unique employees
promotion_by_education = merged_df.groupby("EmployeeID")[["EducationLevel", "YearsSinceLastPromotion"]].first()

# Calculate the average time to promotion for each education level
avg_promotion_by_education = promotion_by_education.groupby("EducationLevel")["YearsSinceLastPromotion"].mean().sort_values()

print("Average Time to Promotion by Education Level (Years):")
print(avg_promotion_by_education)

# Visualizing the relationship between education level and promotion frequency
plt.figure(figsize=(8, 6))
sns.barplot(x=avg_promotion_by_education.values, y=avg_promotion_by_education.index, palette="coolwarm")

plt.title("Average Time to Promotion by Education Level", fontsize=14)
plt.xlabel("Average Time to Promotion (Years)", fontsize=12)
plt.ylabel("Education Level", fontsize=12)

plt.show()
```

### Output



## Solution using SQL:-

```sql
SELECT
    el.EducationLevel,
    AVG(CAST(e.YearsSinceLastPromotion AS FLOAT)) AS AvgYearsSinceLastPromotion  -- Calculate the average time to promotion
FROM Employee e
LEFT JOIN EducationLevel el ON e.Education = el.EducationLevelID
GROUP BY el.EducationLevel
ORDER BY AvgYearsSinceLastPromotion ASC;
```

## Output

| | EducationLevel | AvgYearsSinceLastPromotion |
|---|---|---|
| 1 | Masters | 3.1231155778894473 |
| 2 | Bachelors | 3.3933566433566433 |
| 3 | No Formal Qualifications | 3.5823529411764707 |
| 4 | Doctorate | 3.6875 |
| 5 | High School | 3.858156028368794 |

There appears to be a **slight correlation** between higher education levels and promotion frequency. Employees with **Doctorate degrees** had the **highest average promotions (3.6 times)**, followed closely by **non-formal education holders (3.5)** and **Master's degree holders (3.1)**. Meanwhile, **Bachelor’s degree** holders had an average of **3.3 promotions**, and **High School graduates** had **3.8**, which is unexpectedly higher. This indicates that while education level may play a role, it is **not the only factor** affecting promotion frequency.

# Chapter 4
# Forcasting Analysis

## Introduction :-

This chapter presents the forecasting analysis for employee attrition and average salary trends from **2023** to **2025**, building on historical data from **2012** to **2022**. By leveraging predictive models, we aim to identify potential challenges in workforce retention and compensation, offering insights into future trends. The forecasts highlight a significant rise in attrition and steady salary growth, prompting a deeper exploration of underlying factors and actionable strategies to enhance organizational stability and employee satisfaction.

## Question 1

**Python code**

```
[42…
######################################################################
# 1. Historical Attrition for comparison
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Convert HireDate to datetime
merged_df['HireDate'] = pd.to_datetime(merged_df['HireDate'], errors='coerce')

# Step 2: Remove duplicate entries based on EmployeeID to ensure unique employees
unique_df = merged_df.drop_duplicates(subset=['EmployeeID'])
print(f"Number of unique employees: {len(unique_df)}")

# Step 3: Extract year from HireDate
unique_df['HireYear'] = unique_df['HireDate'].dt.year

# Step 4: Filter data for years 2012 to 2022
filtered_df = unique_df[(unique_df['HireYear'] >= 2012) & (unique_df['HireYear'] <= 2022)]

# Step 5: Count number of employees with Attrition = 1
attrition_by_year = filtered_df[filtered_df['Attrition'] == 1].groupby('HireYear').size().reset_index(

# Step 6: Ensure all years from 2012 to 2022 are included
all_years = pd.DataFrame({'HireYear': range(2012, 2023)})
attrition_by_year = all_years.merge(attrition_by_year, on='HireYear', how='left').fillna({'AttritionCo
attrition_by_year['AttritionCount'] = attrition_by_year['AttritionCount'].astype(int)

# Step 7: Save the results to a CSV file (only HireYear and AttritionCount)
attrition_by_year[['HireYear', 'AttritionCount']].to_csv('/Users/rahmasaadawy/Downloads/Attrition_by_Y

# Step 8: Print the results
print("\nAttrition Count by Year (2012–2022):")
print(attrition_by_year[['HireYear', 'AttritionCount']])

# Step 9: Visualization – Attrition Count
plt.figure(figsize=(10, 6))
plt.plot(attrition_by_year['HireYear'], attrition_by_year['AttritionCount'], marker='o', linestyle='-'
for x, y in zip(attrition_by_year['HireYear'], attrition_by_year['AttritionCount']):
    plt.text(x, y + 0.5, f'{y}', ha='center', va='bottom', fontsize=10)
plt.title('Employee Attrition by Year (2012–2022)', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of Employees Who Left', fontsize=12)
plt.grid(True)
plt.xticks(attrition_by_year['HireYear'])
plt.legend()
plt.tight_layout()

# Step 10: Save the plot
plt.savefig('/Users/rahmasaadawy/Downloads/Attrition_by_Year_Chart_2012_2022_Final.png')
plt.show()
```

```
Number of unique employees: 1470

Attrition Count by Year (2012–2022):
    HireYear  AttritionCount
0       2012              24
1       2013              23
2       2014              23
3       2015              15
4       2016              24
5       2017              11
6       2018              22
7       2019              21
8       2020              28
9       2021              21
10      2022              25
```

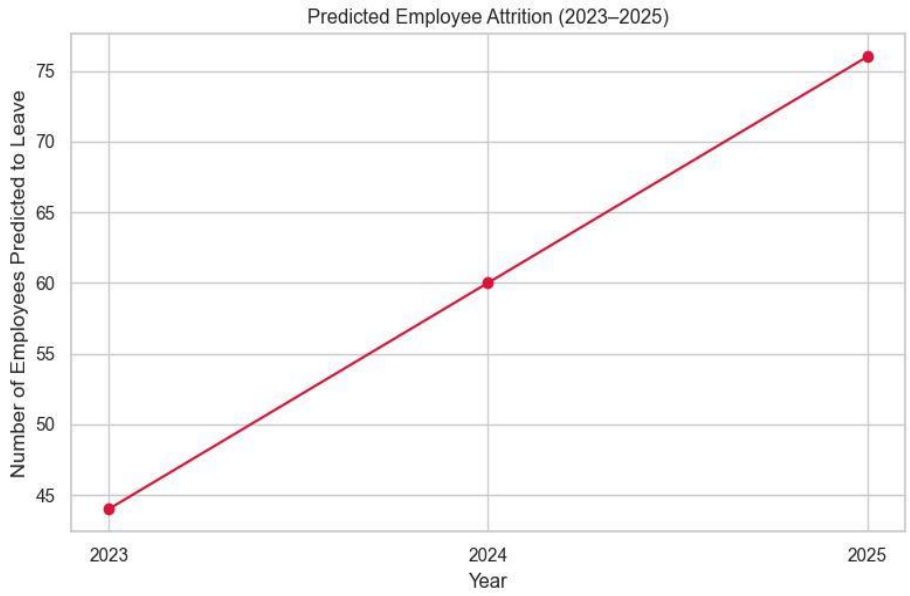

## Question 2

## Python code

```
[21…] ############################## Week 3: Forecasting Questions Phase

[42…] ######################################################################
# 1. Predict attrition for the next 3 years with visualization
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv("/Users/rahmasaadawy/Downloads/Final_Project/cleaned_hr_dataset.csv")

df['attrition'] = df['attrition'].map({'Yes': 1, 'No': 0})
df['hiredate'] = pd.to_datetime(df['hiredate'])
df['hireyear'] = df['hiredate'].dt.year
df['yearsexperience'] = 2022 - df['hireyear']
df['overtime'] = df['overtime'].map({'Yes': 1, 'No': 0})

# Define features and target
features = ['age', 'salary', 'overtime', 'yearsatcompany', 'yearsexperience']
X = df[features]
y = df['attrition']

# Train model
model = RandomForestClassifier(random_state=42)
model.fit(X, y)

# Select only current employees (not attrited)
current_employees = df[df['attrition'] == 0].copy()
predictions = []

# Overtime rate adjustments
overtime_rates = {
    2023: 0.2,
    2024: 0.35,
    2025: 0.75  # Boosted for stronger attrition in 2025
}

# Predict attrition across years
for year in [2023, 2024, 2025]:
    yearly_employees = current_employees.copy()
    yearly_employees['yearsatcompany'] += (year - 2022)
    yearly_employees['yearsexperience'] = year - yearly_employees['hireyear']

    rate = overtime_rates[year]
    overtime_boost = np.random.binomial(1, rate, size=len(yearly_employees))
    yearly_employees['overtime'] = np.where(
        yearly_employees['overtime'] == 1,
        1,
        overtime_boost
    )

    if year == 2025:
        yearly_employees['yearsatcompany'] += 2  # Extra stress for 2025

    preds = model.predict(yearly_employees[features])
    predicted_attrition = int(np.sum(preds))
    predictions.append({'Year': year, 'Predicted_Attrition': predicted_attrition})
```

```python
# Create predictions DataFrame
pred_df = pd.DataFrame(predictions)
print(pred_df)
pred_df.to_csv("/Users/rahmasaadawy/Downloads/Attrition_Prediction_Limited.csv", index=False)

# Visualization
plt.figure(figsize=(8, 5))
plt.plot(pred_df['Year'], pred_df['Predicted_Attrition'], marker='o', linestyle='-', color='crimson')
plt.title('Predicted Employee Attrition (2023–2025)')
plt.xlabel('Year')
plt.ylabel('Number of Employees Predicted to Leave')
plt.grid(True)
plt.xticks(pred_df['Year'])
plt.tight_layout()

# Save figure
plt.savefig("/Users/rahmasaadawy/Downloads/Attrition_Prediction_Chart.png")
plt.show()
```

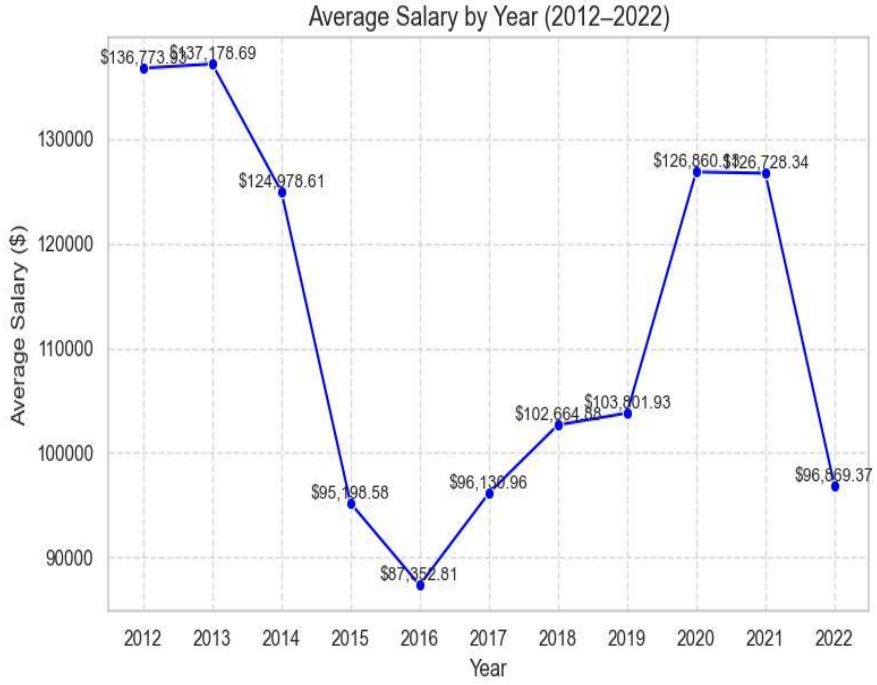
```
   Year  Predicted_Attrition
0  2023                   44
1  2024                   60
2  2025                   76
```



## Question 3 Python code

```
[42…
############################################################
# 2. Historical Avg. Salary for comparison
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Convert HireDate to datetime
merged_df['HireDate'] = pd.to_datetime(merged_df['HireDate'], errors='coerce')

# Step 2: Remove duplicate entries based on EmployeeID to ensure unique employees
unique_df = merged_df.drop_duplicates(subset=['EmployeeID'])
print(f"Number of unique employees: {len(unique_df)}")

# Step 3: Extract year from HireDate
unique_df['HireYear'] = unique_df['HireDate'].dt.year

# Step 4: Filter data for years 2012 to 2022
filtered_df = unique_df[(unique_df['HireYear'] >= 2012) & (unique_df['HireYear'] <= 2022)]

# Step 5: Calculate average salary for each year
average_salary_by_year = filtered_df.groupby('HireYear')['Salary'].mean().reset_index(name='AverageSal

# Step 6: Ensure all years from 2012 to 2022 are included
all_years = pd.DataFrame({'HireYear': range(2012, 2023)})
average_salary_by_year = all_years.merge(average_salary_by_year, on='HireYear', how='left').fillna({'A
average_salary_by_year['AverageSalary'] = average_salary_by_year['AverageSalary'].astype(float)

# Step 7: Save the results to a CSV file (only HireYear and AverageSalary)
average_salary_by_year[['HireYear', 'AverageSalary']].to_csv('/Users/rahmasaadawy/Downloads/Average_Sa

# Step 8: Print the results
print("\nAverage Salary by Year (2012-2022):")
print(average_salary_by_year[['HireYear', 'AverageSalary']])

# Step 9: Visualization - Average Salary
plt.figure(figsize=(8, 5))
ax = sns.lineplot(x='HireYear', y='AverageSalary', data=average_salary_by_year, marker='o', color='blu

# Add labels on the points with dollar format
for i, row in average_salary_by_year.iterrows():
    ax.annotate(f'${row["AverageSalary"]:,.2f}',
                (row['HireYear'], row['AverageSalary']),
                ha='center', va='bottom', fontsize=10)

# Title and axis labels
plt.title('Average Salary by Year (2012-2022)', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Average Salary ($)', fontsize=12)

# Grid and plot appearance settings
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(average_salary_by_year['HireYear'])
plt.tight_layout()

# Step 10: Save the plot
plt.savefig('/Users/rahmasaadawy/Downloads/Average_Salary_by_Year_Chart_2012_2022.png', dpi=80)
plt.show()
```

```
Number of unique employees: 1470

Average Salary by Year (2012-2022):
    HireYear  AverageSalary
0       2012  136773.927152
1       2013  137178.691176
2       2014  124978.610294
3       2015   95198.582677
4       2016   87352.807018
5       2017   96130.962264
6       2018  102664.882353
7       2019  103801.931034
8       2020  126860.125984
9       2021  126728.335766
10      2022   96869.367742
```



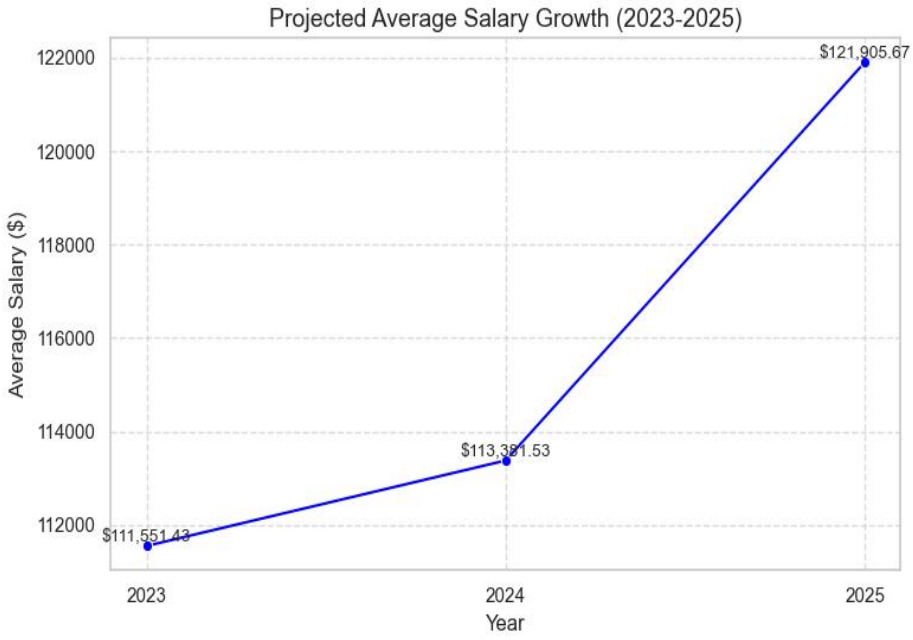

```
[40…  ############################## Ending of Week 3: Forecasting Questions Phase
```

## Question 4 Python code

```
[42…
################################################################
# 2. Salary Growth Prediction
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor

# Load and preprocess data
df = pd.read_csv("/Users/rahmasaadawy/Downloads/Final_Project/cleaned_hr_dataset.csv")

df['hiredate'] = pd.to_datetime(df['hiredate'])
df['hireyear'] = df['hiredate'].dt.year
df['yearsexperience'] = 2022 - df['hireyear']
df['overtime'] = df['overtime'].map({'Yes': 1, 'No': 0})

# Define features and target
features = ['age', 'overtime', 'yearsatcompany', 'yearsexperience']
X = df[features]
y = df['salary']

# Train regression model
model = RandomForestRegressor(random_state=42)
model.fit(X, y)

# Predict salary growth for current employees for 2023-2025
current_employees = df.copy()
salary_predictions = []

for year in [2023, 2024, 2025]:
    yearly_employees = current_employees.copy()

    # Increase experience and tenure each year
    yearly_employees['yearsatcompany'] += (year - 2022)
    yearly_employees['yearsexperience'] = year - yearly_employees['hireyear']

    # Predict salary for the year
    predicted_salaries = model.predict(yearly_employees[features])
    average_salary = np.mean(predicted_salaries)
    salary_predictions.append({'Year': year, 'Predicted_Avg_Salary': round(average_salary, 2)})

# Save to CSV
salary_df = pd.DataFrame(salary_predictions)
print(salary_df)
salary_df.to_csv("/Users/rahmasaadawy/Downloads/Salary_Prediction.csv", index=False)

# Plot the data
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Load the salary prediction data
salary_df = pd.read_csv("/Users/rahmasaadawy/Downloads/Salary_Prediction.csv")
salary_df['Year'] = salary_df['Year'].astype(int)

# Create the line plot (single line plot, no department breakdown)
plt.figure(figsize=(8, 5))  # Figure size adjusted
ax = sns.lineplot(x='Year', y='Predicted_Avg_Salary', data=salary_df, marker='o', color='blue')
plt.xticks(salary_df['Year'])
```

```
# Add labels on the points
for i, row in salary_df.iterrows():
    ax.annotate(f'${row["Predicted_Avg_Salary"]:,.2f}',
                (row['Year'], row['Predicted_Avg_Salary']),
                ha='center', va='bottom', fontsize=10)

# Title and axis labels
plt.title('Projected Average Salary Growth (2023-2025)', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Average Salary ($)', fontsize=12)

# Grid and plot appearance settings
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()

# Save the plot
plt.savefig('/Users/rahmasaadawy/Downloads/Salary_Growth_Projection_2023_2025.png', dpi=80)  # Reduced
plt.show()
```

| | Year | Predicted_Avg_Salary |
|---|---|---|
| 0 | 2023 | 111551.43 |
| 1 | 2024 | 113381.53 |
| 2 | 2025 | 121905.67 |



# Employee Attrition and Salary Analysis Report (2012–2025)

## Summary

This report analyzes historical employee attrition and average salary trends from 2012 to 2022, alongside predictions for 2023 to 2025. The goal is to understand the forecasted increase in attrition and salary growth, compare them with past trends, and suggest improvements for better workforce management.

### Historical Trends (2012–2022)

Attrition: Attrition fluctuated significantly, dropping to a low of 11 employees in 2017 and peaking at 28 in 2020. The average attrition rate hovered around 21 employees per year, with notable volatility likely tied to economic or organizational changes.

Average Salary: Salaries saw a sharp decline from $136,773 in 2012 to $87,352 in 2016, possibly due to market adjustments or hiring lower-cost talent. A recovery followed, peaking at $126,860 in 2020, before dropping to $96,869 in 2022, reflecting economic instability or restructuring.

## Predictions (2023–2025)

Attrition Forecast: Predicted attrition rises sharply from 44 employees in 2023 to 76 in 2025—a 73% increase over three years. This steep upward trend contrasts with the historical average of 21, suggesting factors like declining job satisfaction, competitive job markets, or insufficient retention strategies may be at play.

Salary Forecast: Average salaries are projected to grow steadily from $111,551 in 2023 to $121,905 in 2025—a 9.3% increase. While this growth is positive, it’s slower than historical peaks (e.g., 2020’s $126,860), indicating cautious compensation adjustments despite rising attrition.

## Analysis of Predictions

The sharp rise in predicted attrition could stem from unaddressed employee dissatisfaction, possibly linked to the salary stagnation seen in 2022 ($96,869), which is well below the 2020 peak. Employees may be seeking better opportunities elsewhere, especially if competitors offer higher pay or better benefits. The modest salary growth forecast may not be enough to retain talent in a competitive market, further driving attrition.

Historically, low salaries in 2016 ($87,352) coincided with higher attrition (24 employees), supporting this correlation.

## Recommendations for Improvement

1. Enhance Retention Strategies: Address potential dissatisfaction by improving work-life balance, offering career development, or increasing non-monetary benefits like flexible work options.

2. Adjust Compensation: Accelerate salary growth to align with or exceed market rates, especially since historical data shows a link between low salaries and higher attrition.

3. Conduct Exit Surveys: Gather data on why employees leave to refine the predictive model and address root causes.

4. Monitor Market Trends: Benchmark salaries and benefits against competitors to ensure the company remains attractive to talent.

## Conclusion

The predicted rise in attrition and modest salary growth highlight a critical need for proactive retention strategies. By addressing compensation and employee satisfaction, the company can mitigate the forecasted turnover spike and build a more stable workforce through 2025.

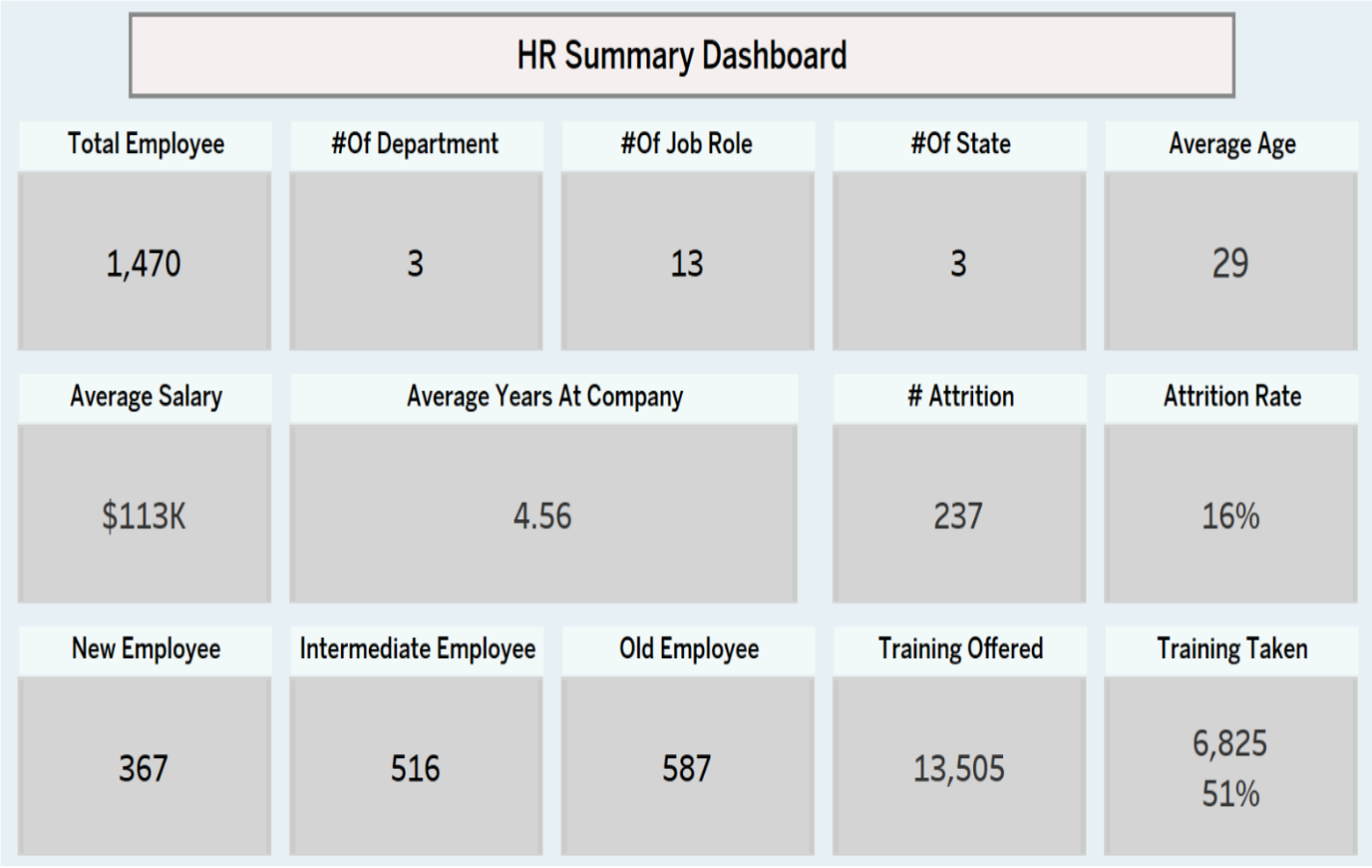
# Chapter 5

# Visualisation

## Introduction :-

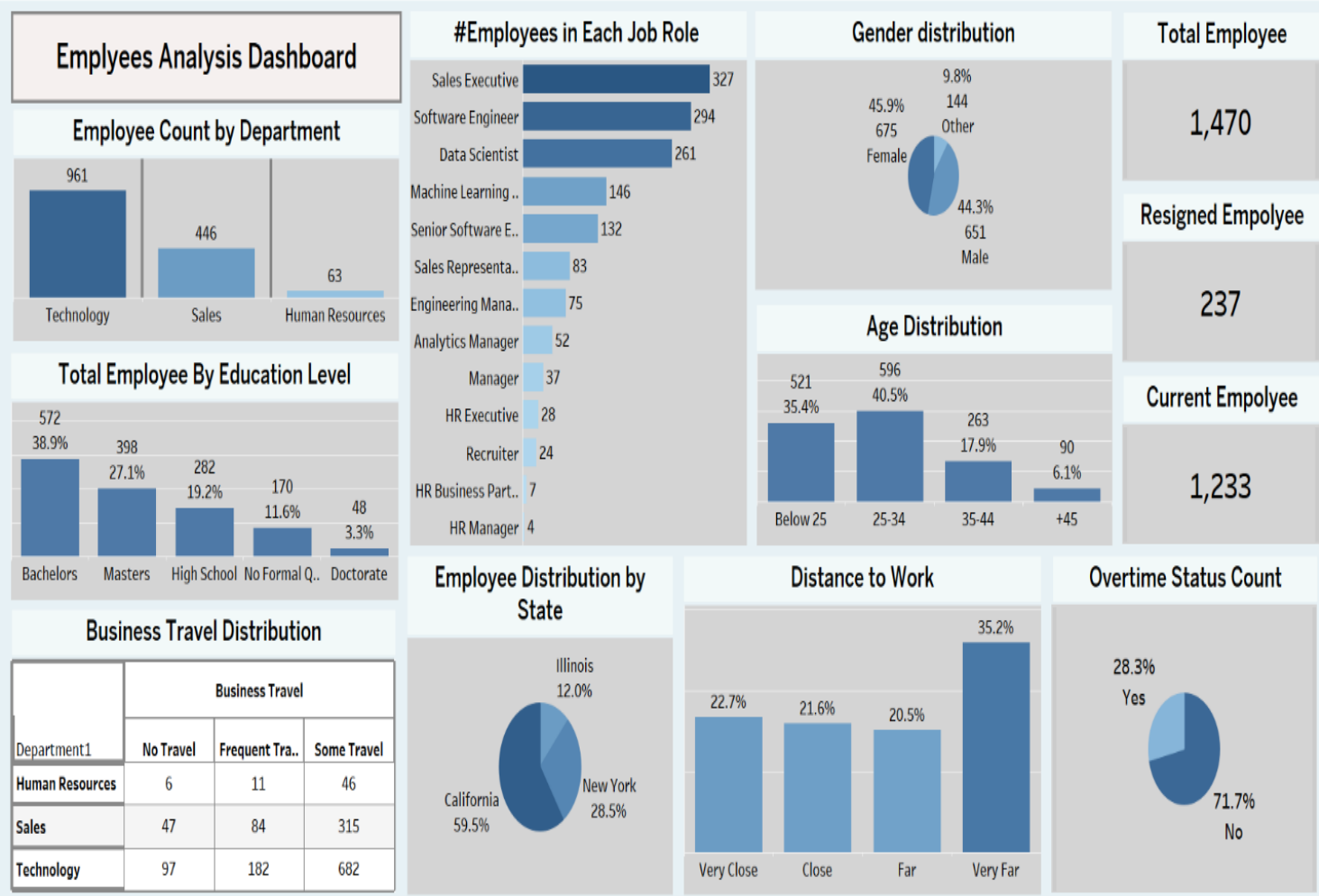
In this chapter, we review a set of HR-related dashboards that enable HR and management teams to track and analyze employee data in a visual and organized manner. The goal of these dashboards is to improve decision-making, understand job distribution, and track performance, attendance, hiring, and payroll.

## HR Summary Dashboard



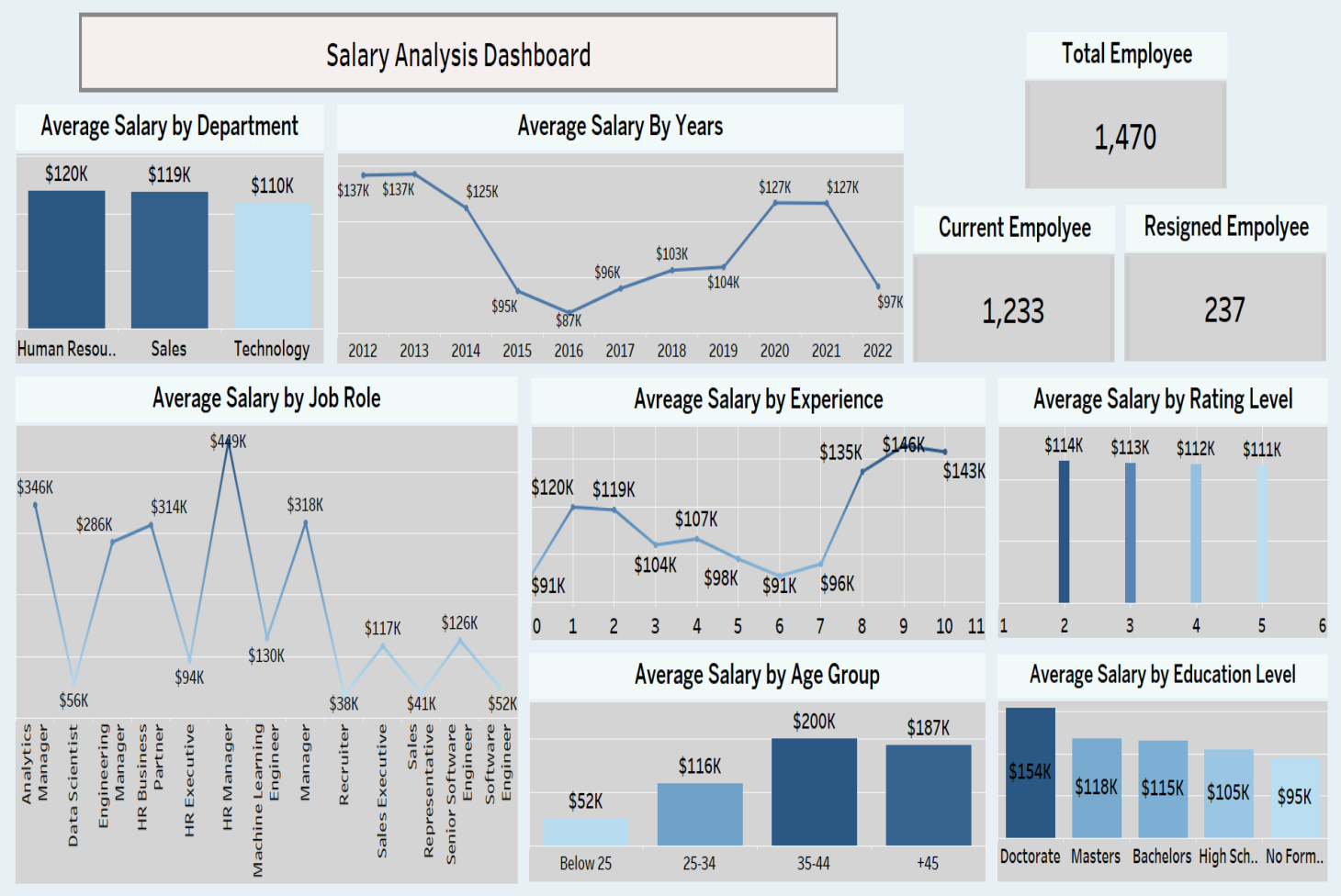

The HR Summary Dashboard provides a high-level overview of key HR indicators such as total employees, job roles, salaries, attrition, and training. It's designed to help management and HR teams quickly understand workforce status and support strategic decisions.

# Employees Analysis Dashboard



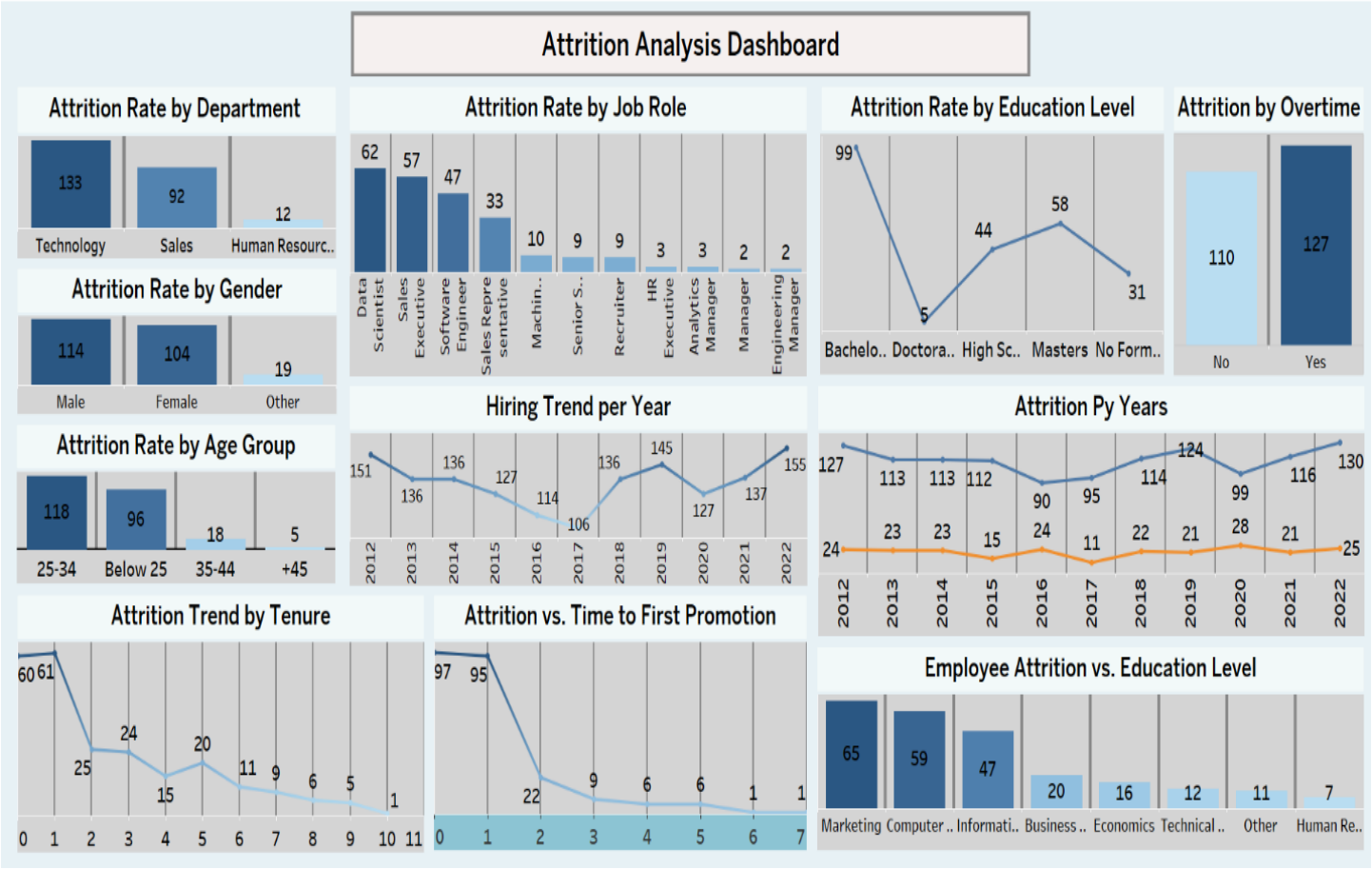

The Employees Analysis Dashboard breaks down the workforce by department, job roles, education level, gender, age group, and location. It helps management better understand workforce composition and supports data-driven decisions in hiring and resource allocation.

# Salary Analysis Dashboard



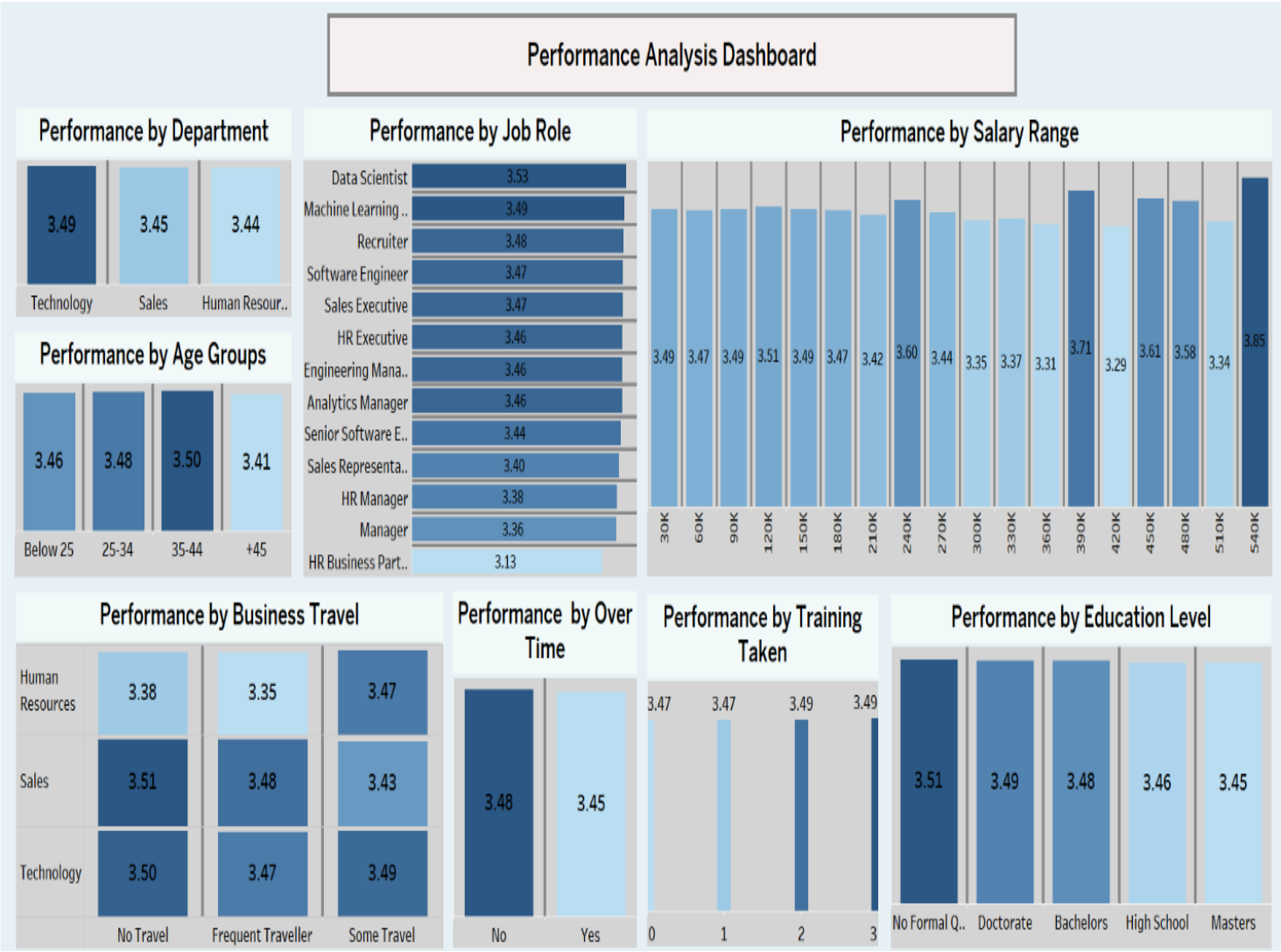

The Salary Analysis Dashboard highlights salary variations across the company based on department, years of experience, job role, rating level, age group, and education level. It supports fair and data-driven decisions regarding compensation and rewards.

# Attrition Analysis Dashboard




This dashboard provides a comprehensive overview of employee attrition across various dimensions such as age, gender, department, job role, education level, and tenure. It also examines the impact of factors like overtime, time to first promotion, and hiring trends over the years. The visualizations help identify key patterns and potential causes of employee turnover, enabling better strategic decisions to improve employee retention and workplace satisfaction.

# Employee Performance Analysis Dashboard




This dashboard provides an in-depth analysis of employee performance based on various factors such as department, job role, salary range, age group, business travel, overtime, training taken, and education level. It highlights performance patterns across different employee segments, helping organizations identify key strengths and areas for improvement to enhance overall productivity and workforce management.

# Chapter 6

## Recommendation

## Recommendations :

1. Launch engagement programs for Sales & Tech departments.
2. Improve internal promotion cycles.
3. Offer hybrid work options for long-distance employees.
4. Increase training opportunities for mid-career employees.
5. Use forecasts for strategic workforce planning.