HR Attrition Analytics: Unlocking Insights on Employee Satisfaction & Performance

DEPI Initiative - Data Analyst Specialist Track

Group: ALX3_DAT1_G2

Under the Supervision of: Eng./Dina Ezzat

رواد مصر الرقمية

Team Members (Group 3)

- Moataz Tammam Abdelaziz —> Data Cleaning & Analysis
- Yasmeen Farouq Saleh —> Data Cleaning & Analysis
- Muhammed Nasseim El-Saied —> Data Cleaning & Analysis
- Sara Ayed Wahba —> Data Visualization & Presentation Preparation
- Malak Tarek Muhammed —> Data Visualization & Presentation Preparation

Project Overview

This project focuses on uncovering the underlying factors that lead to ***Employee Attrition*** within the company.Our analysis examines three key evaluation dimensions:

- **Employees' Satisfaction and Engagement.**
- **Manager Assessments of Employees.**
- **Employees' Self-evaluation.**

These insights are derived from HR dataset across **three** core departments: **Sales**, **Technology**, and **Human Resources.**

Data Analysis Process

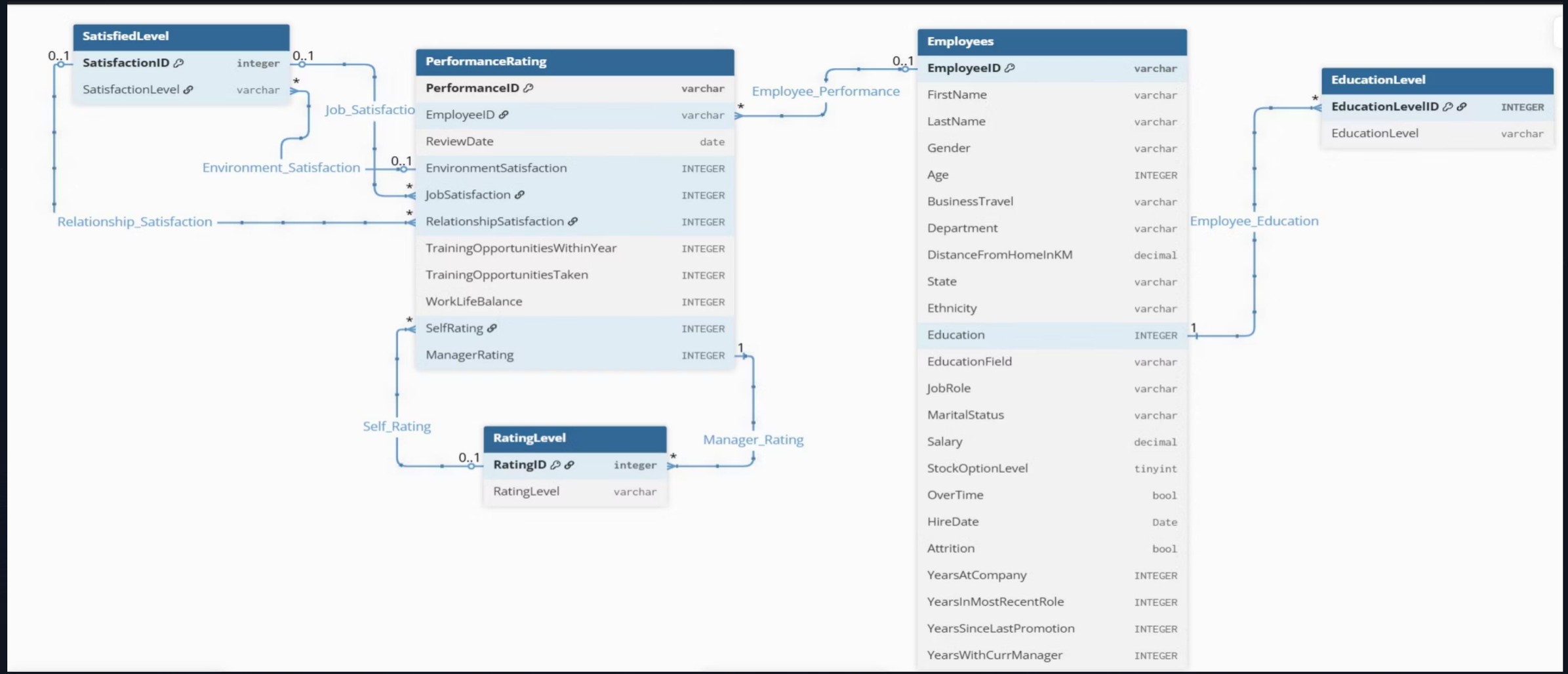
1. Data Exploration
2. Data Cleaning & Preparation
3. Data Analysis (EDA)
4. Data Visualization
5. Data Insights & Reporting

Understanding Data

Our HR Dataset is built on a relational database consisting of **five interconnected tables:**

1. Employee table —> Master employee table (Demographics & Employment data, Employee Attrition & tenure fields). **Primary key: EmployeeID**
2. Performance Rating —> Employee Performance & Satisfaction scores per review (ReviewDate) and training participations. **Primary Key: PerformanceID, Foreign Key: EmployeeID.**
3. Education Level —> Employees' Educational Background.
   - Lookup table: mapping **EducationLevelID "PK"** to Employees table **"FK: Education"**
4. Rating Level —> Categorical description for each rating.
   - Lookup table: **RatingID "PK"** → **Manager & Self-Rating "FK"** in PerformanceRating table.
5. Satisfied Level —> Categorical description for each Satisfaction Criteria.
   - Lookup table: **SatisfactionID "PK"** → **Environment, Job & Relationship Sats. "FK"** in PerformanceRating table.

Database Schema



Data Loading

```sql
-- Data Exploration PerformanceRating
SELECT * FROM PerformanceRating;
-- Data Exploration Employee
SELECT * FROM Employees;
-- Data Exploration SatisfiedLevel
SELECT * FROM SatisfiedLevel;
-- Data Exploration RatingLevel
SELECT * FROM RatingLevel;
-- Data Exploration EducationLevel
SELECT * FROM EducationLevel;
```

Data Cleaning & Preparation

- Identifying Duplicates
- Handling Missing Data
- Standardize/ Normalize Formats
- Validate Results/Check Integrity and Consistency

Handling Duplicates

```sql
-- Checking Duplicates in Employees table
SELECT EmployeeID, COUNT (*) FROM Employees
GROUP BY EmployeeID
HAVING COUNT(*)> 1;
-- Checking Duplicates in PerPerformanceRating table
SELECT PerformanceID, COUNT (*) FROM PerformanceRating
GROUP BY PerformanceID
HAVING COUNT(*)> 1;
```

It appears that there are no duplicates in our dataset.

Handling Null Values

```sql
-- Searching for Missing Values & Removing Trailing Whitespaces at key Fields in Employees table
SELECT
    COUNT(IF(EmployeeID IS NULL OR TRIM(EmployeeID) = '', 1, NULL)) AS Missing_EmpID,
    COUNT(IF(Gender IS NULL OR TRIM(Gender) = '', 1, NULL)) AS Missing_Gender,
    COUNT(IF(Age IS NULL OR TRIM(Age) = '', 1, NULL)) AS Missing_Age,
    COUNT(IF(Department IS NULL OR TRIM(Department) = '', 1, NULL)) AS Missing_Dept,
    COUNT(IF(Salary IS NULL OR TRIM(Salary) = '', 1, NULL)) AS Missing_Salary,
    COUNT(IF(YearsAtCompany IS NULL OR TRIM(YearsAtCompany) = '', 1, NULL)) AS Missing_Exp_Yrs
FROM Employees
-- Searching for Missing Values  & Removing Trailing Whitespaces at key Fields in PerformanceRating table
SELECT
    COUNT(IF(PerformanceID IS NULL OR TRIM(PerformanceID) = '', 1, NULL)) AS Missing_PerfID,
    COUNT(IF(EmployeeID IS NULL OR TRIM(EmployeeID) = '', 1, NULL)) AS Missing_EmpID,
    COUNT(IF(SelfRating IS NULL OR TRIM(SelfRating) = '', 1, NULL)) AS Missing_Self_Rate,
    COUNT(IF(ManagerRating IS NULL OR TRIM(ManagerRating) = '', 1, NULL)) AS Missing_Mangager_Rate,
    COUNT(IF(JobSatisfaction IS NULL OR TRIM(JobSatisfaction) = '', 1, NULL)) AS Missing_Job_Sat,
    COUNT(IF(TrainingOpportunitiesWithinYear IS NULL OR TRIM(TrainingOpportunitiesWithinYear) = '', 1, NULL)) AS Miss_TrOppWithinYear,
    COUNT(IF(RelationshipSatisfaction IS NULL OR TRIM(RelationshipSatisfaction) = '', 1, NULL)) AS Missing_Rel_Sat,
    COUNT(IF(TrainingOpportunitiesTaken IS NULL OR TRIM(TrainingOpportunitiesTaken) = '', 1, NULL)) AS Missing_TrOppTaken
FROM PerformanceRating;
```

- It appears that there are no Null Values in our dataset.

Checking Invalid Data (Ensure Data Quality)

```sql
-- Checking Data Quality if there are any Invalid Values in Employees Table (Age & Salary)
SELECT COUNT ( * ) FROM Employee
WHERE Age < 0;
SELECT COUNT ( * ) FROM Employee
WHERE salary < 0;
-- Checking invalid data in PerformanceRating Table
SELECT COUNT(*) AS InvalidRatings
FROM PerformanceRating
WHERE SelfRating NOT BETWEEN 1 AND 5
      OR ManagerRating NOT BETWEEN 1 AND 5
      OR EnvironmentSatisfaction NOT BETWEEN 1 AND 5
      OR JobSatisfaction NOT BETWEEN 1 AND 5
      OR RelationshipSatisfaction NOT BETWEEN 1 AND 5
      OR WorkLifeBalance NOT BETWEEN 1 AND 5;
-- Identifying Outliers in Employee Table
SELECT EmployeeID, Age
FROM Employees
WHERE Age < 18 OR Age > 60;
```

Exploratory Data Analysis (EDA)

- To examine & investigate datasets to discover patterns, trends & spot anomalies.

```sql
-- Summary Statistics
-- Salary & Employee Tenure by their Education Level
SELECT ed.EducationLevel,
       COUNT(e.EmployeeID) AS cnt,
       ROUND(AVG(COALESCE(e.Salary,0)), 2) AS Avg_Salary,
       ROUND(AVG(COALESCE(e.YearsAtCompany,0)), 2) AS Avg_Years_At_Company
FROM Employees e
LEFT JOIN EducationLevel ed
     ON e.Education = ed.EducationLevelID
GROUP BY ed.EducationLevel
ORDER BY Avg_Salary DESC;
-- Salary Distribution Statistics
SELECT MIN(Salary) AS MinSalary,
       MAX(Salary) AS MaxSalary,
       AVG(Salary) AS AvgSalary
FROM Employees;
-- Age Statistics
SELECT MIN(Age) AS MinAge,
       MAX(Age) AS MaxAge,
       AVG(Age) AS AvgAge
FROM Employees;
```

Focus Areas of Analysis

1. Factors influencing Employee Satisfaction and Attrition
2. Overtime Impact on Employee WLB & Performance
3. Correlation between Salary and Satisfaction or Performance
4. Training Impact on Performance & Promotion Potential
5. Common Characteristics of Attritors

Performance Analysis (Performance Correlation with Age)

```sql
-- Correlation
-- Performance analysis and Age / Years At Company.
-- Employee Performance & Age
WITH perf AS (
    SELECT
        EmployeeID,
        AVG(JobSatisfaction) AS Avg_JobSatisfaction,
        AVG(SelfRating) AS Avg_SelfRating,
        AVG(ManagerRating) AS Avg_ManagerRating
    FROM PerformanceRating
    GROUP BY EmployeeID
)
SELECT
    CASE
        WHEN e.Age < 25 THEN '<25'
        WHEN e.Age BETWEEN 25 AND 35 THEN '25-35'
        WHEN e.Age BETWEEN 36 AND 45 THEN '36-45'
        ELSE '>45'
    END AS Age_Range,
    ROUND(AVG(p.Avg_JobSatisfaction),2) AS Avg_JobSatisfaction,
    ROUND(AVG(p.Avg_SelfRating),2) AS Avg_SelfRating,
    ROUND(AVG(p.Avg_ManagerRating),2) AS Avg_ManagerRating,
    COUNT(*) AS Employee_Count
FROM Employee e
LEFT JOIN perf p
    ON e.EmployeeID = p.EmployeeID
GROUP BY Age_Range
ORDER BY Age_Range;
```

Performance Analysis (Performance Correlation with tenure)

```sql
--Employee Performance & Years At Company
WITH perf AS (
    SELECT
        EmployeeID,
        AVG(JobSatisfaction) AS JobSatisfaction,
        AVG(SelfRating) AS SelfRating,
        AVG(ManagerRating) AS ManagerRating
    FROM PerformanceRating
    GROUP BY EmployeeID
)
SELECT
    CASE
        WHEN YearsAtCompany < 2 THEN '<2'
        WHEN YearsAtCompany BETWEEN 2 AND 5 THEN '2-5'
        WHEN YearsAtCompany BETWEEN 6 AND 10 THEN '6-10'
        ELSE '>10'
    END AS YearsAtCompany_Range,
    AVG(p.JobSatisfaction) AS Avg_JobSatisfaction,
    AVG(p.SelfRating) AS Avg_SelfRating,
    AVG(p.ManagerRating) AS Avg_ManagerRating,
    COUNT(*) AS Employee_Count
FROM Employee e
JOIN perf p ON e.EmployeeID = p.EmployeeID
GROUP BY YearsAtCompany_Range
ORDER BY YearsAtCompany_Range;
```

Factors influencing Employee Satisfaction and Attrition

```sql
-- Identify the factors most associated with Low Employee Satisfaction or High Attrition
SELECT
    e.Department,
    ROUND(AVG(p.EnvironmentSatisfaction),2) AS Avg_Env_Satisfaction,
    ROUND(AVG(p.JobSatisfaction),2) AS Avg_Job_Satisfaction,
    ROUND(AVG(p.RelationshipSatisfaction),2) AS Avg_Relationship_Satisfaction,
    ROUND(AVG(p.WorkLifeBalance),2) AS Avg_WorkLifeBalance,
    ROUND(AVG(p.SelfRating),2) AS Avg_SelfRating,
    ROUND(AVG(p.TrainingOpportunitiesWithinYear),2) AS Avg_Training_Opportunities
FROM Employees AS e
LEFT JOIN PerformanceRating AS p
    ON e.EmployeeID = p.EmployeeID
GROUP BY e.Department
ORDER BY Avg_Job_Satisfaction ASC;
```

Overtime Impact on Work-Life Balance and Employee Performance:

```sql
-- Determine the Impact of Overtime on Work-Life balance and Overall Performance
SELECT
    e.OverTime,
    COUNT(DISTINCT(e.EmployeeID)) AS Num_Employees,
    ROUND(AVG(p.WorkLifeBalance),2) AS Avg_WorkLife_Balance,
    ROUND(AVG(p.JobSatisfaction),2) AS Avg_Job_Satisfaction,
    ROUND(ROUND(AVG(p.ManagerRating),2) AS Avg_Manager_Rating,
    ROUND(AVG(p.SelfRating),2) AS Avg_Self_Rating
FROM Employees AS e
LEFT JOIN PerformanceRating AS p
     ON e.EmployeeID = p.EmployeeID
GROUP BY e.OverTime
ORDER BY Avg_WorkLife_Balance ASC;
```

Overtime Impact on Employee Performance

Overtime and Job Satisfaction

Overtime had a noticeable effect on job satisfaction. Employees who **worked extra hours** reported **higher satisfaction** compared to those who did not (3.51 vs 3.38).

Overtime and Performance

Employees who **did not work overtime** have **higher manager rating** average of 3.47 compared to 3.45 for those who do.

Relation between Salary and Employee Satisfaction & Performance

```sql
-- Relation between Salary and Employee Satisfaction & Performance
SELECT
    e.Department,
    ROUND(AVG(e.Salary), 2) AS Avg_Salary,
    ROUND(AVG(p.ManagerRating), 2) AS Avg_Manager_Rating,
    ROUND(AVG(p.SelfRating), 2) AS Avg_Self_Rating,
    ROUND(AVG(p.JobSatisfaction), 2) AS Avg_Job_Satisfaction
FROM Employees AS e
LEFT JOIN PerformanceRating AS p
     ON e.EmployeeID = p.EmployeeID
GROUP BY e.Department
ORDER BY Avg_Salary DESC;
SELECT
    CASE
        WHEN Salary < 80000 THEN '< 80k'
        WHEN Salary BETWEEN 80000 AND 120000 THEN '80k-120k'
        WHEN Salary BETWEEN 120001 AND 180000 THEN '120k-180k'
        WHEN Salary BETWEEN 180001 AND 300000 THEN '180k-300k'
        ELSE '> 300k'
    END AS Salary_Band,
    COUNT(*) AS Employee_Count
FROM Employees
GROUP BY Salary_Band
ORDER BY Employee_Count DESC;
```

Training Opportunities Impact on Performance & Promotion Potential

```sql
-- The Impact of Training Opportunities on Performance and Promotion Rate
SELECT
    CASE
        WHEN p.TrainingOpportunitiesWithinYear > 0 THEN 'Received Training'
        ELSE 'No Training'
    END AS Training_Status,
    COUNT(DISTINCT e.EmployeeID) AS Num_Employees,
    ROUND(AVG(p.ManagerRating), 2) AS Avg_Manager-Rating,
    ROUND(AVG(p.SelfRating), 2) AS Avg_Self_Rating,
    ROUND(AVG(e.YearsAtCompany), 2) AS Avg_Years_At_Company
FROM Employees e
LEFT JOIN PerformanceRating p
     ON e.EmployeeID = p.EmployeeID
GROUP BY Training_Status
ORDER BY Avg_Years_At_Company DESC;
SELECT
    e.Department,
    ROUND(AVG(p.TrainingOpportunitiesWithinYear), 2) AS Avg_Training_Opportunities_WithinYear,
    ROUND(AVG(p.TrainingOpportunitiesTaken), 2) AS Avg_Training_Opportunities_Taken,
    ROUND(AVG(p.ManagerRating), 2) AS Avg_ManagerRating,
    ROUND(AVG(p.SelfRating), 2) AS Avg_SelfRating,
    ROUND(AVG(e.YearsAtCompany), 2) AS Avg_Years_At_Company,
    ROUND(AVG((ManagerRating + SelfRating) / 2.0),2) AS PromotionPotentialIndex
FROM Employees AS e
LEFT JOIN PerformanceRating AS p
     ON e.EmployeeID = p.EmployeeID
GROUP BY e.Department
ORDER BY Avg_Training_Opportunities_Taken DESC;
```

- **HR Department** has the **lowest training opportunities** (Average: 0.96), but the **highest Performance "Manager Rating"** (Average: 3.49).
- **Sales Department** has **slightly higher training opportunities** (Average: 0.99), but the **lowest Performance** (Average: 3.43).
- **Technology Department** has the **highest training opportunities** (Average: 1.04) and **Manager rating of (Average: 3.47).**

Common Characteristics of Employees who Left

```sql
-- Common Characteristics of Employees who left (Attrition Rate = Yes)
SELECT
  UPPER(TRIM(e.Department)) AS Department,
  ROUND(AVG(e.Age), 2) AS Avg_Age,
  ROUND(AVG(e.Salary), 2) AS Avg_Salary,
  ROUND(AVG(e.YearsAtCompany), 2) AS Avg_YearsAtCompany,
  ROUND(AVG(p.JobSatisfaction), 2) AS Avg_JobSatisfaction,
  ROUND(AVG(p.ManagerRating), 2) AS Avg_ManagerRating,
  ROUND(AVG(p.WorkLifeBalance), 2) AS Avg_WorkLifeBalance,
  ROUND(AVG(p.EnvironmentSatisfaction), 2) AS Avg_EnvironmentSatisfaction,
  ROUND(AVG(p.RelationshipSatisfaction), 2) AS Avg_RelationshipSatisfaction,
  COUNT(DISTINCT e.EmployeeID) AS Attrition_Count,
  ROUND(100.0 * COUNT(DISTINCT e.EmployeeID) /
    (SELECT COUNT(DISTINCT EmployeeID) FROM Employees WHERE LOWER(TRIM(Attrition)) = 'yes'), 2) AS Attrition_Rate
FROM Employees e
LEFT JOIN PerformanceRating p ON e.EmployeeID = p.EmployeeID
WHERE LOWER(TRIM(e.Attrition)) = 'yes'
GROUP BY e.Department
ORDER BY Attrition_Count DESC;
```

Employees who are **younger, less incentivized,** and have **shorter tenure** tend to **leave at higher rates.**

HR Project Jupyter Notebook "Python"

HR Analytics: Employees' Performance & Satisfaction

Project Overview:

This project explores employees' data to uncover insights into workforce performance & satisfaction by analyzing key factors such as demographics, education levels & other factors. This project aims to support HR Decision-making and provide recommendations to improve employee engagement and overall organizational performance.

Importing libraries & Loading our Dataset

1. Importing Libraries

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Loading Datasets

[36]:
```
# Loading datasets
Employees = pd.read_csv ('Employees.csv')
Educational_Level = pd.read_csv('EducationLevel.csv')
Performance = pd.read_csv('PerformanceRating.csv')
Rating_Level = pd.read_csv('RatingLevel.csv')
Satisfied_Level = pd.read_csv('SatisfiedLevel.csv')

# Merging tables
# Merge employees with educational level (EducationID)
Employees = Employees.merge(Educational_Level, left_on="Education", right_on="EducationLevelID", how="left")

# Merge employees with performance rating (EmployeeID)
Employees = Employees.merge(Performance, on="EmployeeID", how="left")


# Merge rating level with both (Manager & Self-Rating)
Employees = Employees.merge(Rating_Level, left_on="ManagerRating", right_on="RatingID", how="left"
                            ).rename(columns={"RatingLevel": "Mananger_RatingLevel"}).drop(columns=["RatingID"])

Employees = Employees.merge(Rating_Level, left_on="SelfRating", right_on="RatingID", how="left"
                            ).rename(columns={"RatingLevel": "Self_RatingLevel"}).drop(columns=["RatingID"])

# Merge Satisfied level with (Job, Environment, Relationship Satisfaction & Work-Life Balance)
# Environment Satisfaction
Employees = Employees.merge(Satisfied_Level, left_on="EnvironmentSatisfaction", right_on="SatisfactionID", how="left"
                            ).rename(columns={"SatisfactionLevel": "Environment_SatisfactionLevel"}).drop(columns=["SatisfactionID"])

# Job Satisfaction
Employees = Employees.merge(Satisfied_Level, left_on="JobSatisfaction", right_on="SatisfactionID", how="left"
                            ).rename(columns={"SatisfactionLevel": "Job_SatisfactionLevel"}).drop(columns=["SatisfactionID"])

# Relationship Satisfaction
Employees = Employees.merge(Satisfied_Level, left_on="RelationshipSatisfaction", right_on="SatisfactionID", how="left"
                            ).rename(columns={"SatisfactionLevel": "Relationship_SatisfactionLevel"}).drop(columns=["SatisfactionID"])

# Work-Life Balance
Employees = Employees.merge(Satisfied_Level, left_on="WorkLifeBalance", right_on="SatisfactionID", how="left"
                            ).rename(columns={"SatisfactionLevel": "WLB_Level"}).drop(columns=["SatisfactionID"])
```

[37]:
```
#Showing first few rows
Employees.head(10)
```

[37]:

	EmployeeID	FirstName	LastName	Gender	Age	BusinessTravel	Department	DistanceFromHomeInKM	State	Ethnicity	...	TrainingOpportunitiesTaken	WorkLifeBa
0	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	0.0	
1	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	1.0	
2	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	0.0	
3	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	1.0	
4	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	0.0	
5	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	0.0	
6	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	0.0	
7	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	0.0	
8	3012-1A41	Leonelle	Simco	Female	30	Some Travel	Sales	27	IL	White	...	1.0	
9	CBCB-9C9D	Leonerd	Aland	Male	38	Some Travel	Sales	23	CA	White	...	0.0	

10 rows × 41 columns

Exploring our Dataset (EDA)

3. Exploratory Data Analysis

In this section, we will explore the datasets to understand its structure and get Summary Statistics on it before performing data analysis

[38]:
```
# Checking shape of the data
Employees.shape
```

[38]:
```
(6899, 41)
```

[39]:
```
# Checking info of the data
Employees.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6899 entries, 0 to 6898
Data columns (total 41 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   DistanceFromHomeInKM            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  EducationLevelID                6899 non-null   int64
 24  EducationLevel                  6899 non-null   object
 25  PerformanceID                   6709 non-null   object
 26  ReviewDate                      6709 non-null   object
 27  EnvironmentSatisfaction         6709 non-null   float64
 28  JobSatisfaction                 6709 non-null   float64
 29  RelationshipSatisfaction        6709 non-null   float64
 30  TrainingOpportunitiesWithinYear 6709 non-null   float64
 31  TrainingOpportunitiesTaken      6709 non-null   float64
 32  WorkLifeBalance                 6709 non-null   float64
 33  SelfRating                      6709 non-null   float64
 34  ManagerRating                   6709 non-null   float64
```

Descriptive/ Summary Statistics

```
[40]: #Showing Summary Statistics
      Employees.describe()
```

[40]:

	Age	DistanceFromHomeInKM	Education	Salary	StockOptionLevel	YearsAtCompany	YearsInMostRecentRole	YearsSinceLastPromotion	YearsW
count	6899.000000	6899.000000	6899.000000	6899.000000	6899.000000	6899.000000	6899.000000	6899.000000	
mean	30.604146	22.327874	2.866647	110898.374112	0.725467	5.578055	2.778953	4.143934	
std	7.986542	12.899799	1.028396	98427.862382	0.839724	3.410087	2.810170	3.203770	
min	18.000000	1.000000	1.000000	20387.000000	0.000000	0.000000	0.000000	0.000000	
25%	25.000000	12.000000	2.000000	44646.000000	0.000000	3.000000	0.000000	1.000000	
50%	28.000000	22.000000	3.000000	74458.000000	1.000000	6.000000	2.000000	4.000000	
75%	36.000000	33.000000	4.000000	137219.500000	1.000000	9.000000	5.000000	7.000000	
max	51.000000	45.000000	5.000000	547204.000000	3.000000	10.000000	10.000000	10.000000	

Data Cleaning (Checking Nulls & Duplicates)

4. Data Cleaning

Before analyzing the dataset, it is important to perform data cleaning to ensure accuracy, consistency, and reliability of results. In this process, we can handle (Missing values, Duplicate Records, Inconsistencies and any errors/ Outliers detected in the datasets).

This process helps us improve data quality and make it ready for exploration, visualization, and modeling which ensures that insights and conclusions drawn from the analysis are trustworthy.

A. Checking Duplicates

```
[41]: Employees.duplicated().sum()
```

```
[41]: np.int64(0)
```

It appears that there are no duplicate values within this dataset.

B. Checking Null Values

```
[42]: Employees.isna().sum()
```

```
[42]: EmployeeID                  0
      FirstName                   0
      LastName                    0
      Gender                      0
      Age                         0
      BusinessTravel              0
      Department                  0
      DistanceFromHomeInKM        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
      EducationLevelID            0
      EducationLevel              0
      PerformanceID             190
      ReviewDate                190
      EnvironmentSatisfaction   190
      JobSatisfaction           190
      RelationshipSatisfaction  190
```

Data Cleaning (Checking Nulls & Duplicates)

```
TrainingOpportunitiesWithinYear     190
TrainingOpportunitiesTaken          190
WorkLifeBalance                     190
SelfRating                          190
ManagerRating                       190
Mananger_RatingLevel                190
Self_RatingLevel                    190
Environment_SatisfactionLevel       190
Job_SatisfactionLevel               190
Relationship_SatisfactionLevel      190
WLB_Level                           190
dtype: int64
```

43]:
```
missing_rows = Employees[Employees.isnull().any(axis=1)]
print(missing_rows)
```

```
     EmployeeID FirstName      LastName       Gender  Age       BusinessTravel  \
6421  3BAB-C57E     Hashim          Payn         Male   19          Some Travel
6422  BC20-368E       Flem       Simenon         Male   39          Some Travel
6423  F3AE-DEF2       Vern         Foord         Male   20  Frequent Traveller
6424  F5B1-A266       Arly      Fleetham       Female   22          Some Travel
6425  2EB9-9D31    Cybill         Mitkin       Female   27           No Travel
...         ...        ...           ...          ...  ...                 ...
6894  467E-977A        Jud     Melanaphy         Male   20          Some Travel
6895  6FB9-A624       Marc         Calver  Non-Binary   27          Some Travel
6896  EBF4-5928    Rudolph   MacDearmont         Male   21          Some Travel
6897  60E6-B1D9     Merill           Agg         Male   21          Some Travel
6898  84D4-D4C3      Naoma       Hebbard       Female   20           No Travel

            Department  DistanceFromHomeInKM State                    Ethnicity  \
6421        Technology                    28    CA                        White
6422             Sales                    37    NY                        White
6423        Technology                    39    CA                        White
6424        Technology                    34    CA                        White
6425  Human Resources                     4    CA                        White
...               ...                   ...   ...                       ...
6894        Technology                    28    CA  Black or African American
6895        Technology                     8    CA  Black or African American
6896             Sales                     4    CA  Black or African American
6897        Technology                     7    CA  Black or African American
6898        Technology                    28    CA  Black or African American

      ...  TrainingOpportunitiesTaken WorkLifeBalance SelfRating  \
6421  ...                         NaN             NaN        NaN
6422  ...                         NaN             NaN        NaN
6423  ...                         NaN             NaN        NaN
6424  ...                         NaN             NaN        NaN
6425  ...                         NaN             NaN        NaN
...   ...                         ...             ...        ...
6894  ...                         NaN             NaN        NaN
6895  ...                         NaN             NaN        NaN
6896  ...                         NaN             NaN        NaN
6897  ...                         NaN             NaN        NaN
6898  ...                         NaN             NaN        NaN

[190 rows x 41 columns]
```

It appears that there are 190 Employees that have no Performance Records

Checking Unique Values

[54]:
```
# Checking Unique Values
for col in Employees:
    print('-'*30)
    print(col)
    print(Employees[col].unique())
```

```
------------------------------
EmployeeID
['3012-1A41' 'CBCB-9C9D' '95D7-1CE9' ... 'EBF4-5928' '60E6-B1D9'
 '84D4-D4C3']
------------------------------
FirstName
['Leonelle' 'Leonerd' 'Ahmed' ... 'Rudolph' 'Merill' 'Naoma']
------------------------------
LastName
['Simco' 'Aland' 'Sykes' ... 'MacDearmont' 'Agg' 'Hebbard']
------------------------------
Gender
['Female' 'Male' 'Non-Binary' 'Prefer Not To Say']
------------------------------
Age
[30 38 43 39 29 34 42 40 31 32 35 37 33 45 36 48 47 44 28 41 46 27 26 25
 24 23 22 21 49 50 20 19 51 18]
------------------------------
BusinessTravel
['Some Travel' 'No Travel ' 'Frequent Traveller']
------------------------------
Department
['Sales' 'Human Resources' 'Technology']
------------------------------
DistanceFromHomeInKM
[27 23 29 12 30 45  3 20  4 42  8 35 21 34 19  1 17 36 37 41 25 44 22 40
 14 15  7  5 31  9 13 38 16 39 28 32  2  6 11 33 26 43 10 18 24]
------------------------------
State
['IL' 'CA' 'NY']
------------------------------
Ethnicity
['White' 'Asian or Asian American' 'Mixed or multiple ethnic groups'
 'Black or African American' 'Native Hawaiian ' 'Other '
 'American Indian or Alaska Native']
------------------------------
Education
['Doctorate' 'Masters ' 'Bachelors ' 'High School '
 'No Formal Qualifications']
------------------------------
EducationField
['Marketing' 'Marketing ' 'Computer Science' 'Technical Degree'
 'Information Systems' 'Other' 'Economics' 'Human Resources'
 'Business Studies']
```

Trimming Extra Spaces & Removing Inconsistencies

In Education Field: there are 'Marketing' and 'Marketing ' (with an empty space)

[62]:
```
# Replacing Marketing with white space with (Marketing)
Employees['EducationField'] = Employees['EducationField'].replace('Marketing ', 'Marketing')
```

[66]:
```
# Ensure all data in Marketing Department are standardized
Employees['EducationField'].unique()
```

[66]:
```
array(['Marketing', 'Computer Science', 'Technical Degree',
       'Information Systems', 'Other', 'Economics', 'Human Resources',
       'Business Studies'], dtype=object)
```

[60]:
```
# Converting the data type into datetime
Employees['HireDate'] = pd.to_datetime(Employees['HireDate'], errors='coerce')
Employees['ReviewDate'] = pd.to_datetime(Employees['ReviewDate'], errors='coerce')
```

[65]:
```
# Removing trailing white Extra Spaces from Columns and Text Fields

Employees.columns = Employees.columns.str.strip()
text_columns = Employees.select_dtypes(include='object').columns
Employees[text_columns] = Employees[text_columns].apply(lambda x: x.str.strip())
```

[67]:
```
# Removing Logically Inconsistent Records
Employees = Employees[
    (Employees['YearsWithCurrManager'] <= Employees['YearsAtCompany']) &
    (Employees['YearsSinceLastPromotion'] <= Employees['YearsAtCompany'])]

Employees = Employees[Employees['YearsAtCompany'] <= Employees['Age']]
```

[68]:
```
# Extracting Year from Review Date
Employees['ReviewYear'] = Employees['ReviewDate'].dt.year
```

Checking if there are any Outliers

[70]:
```
# Check for Outliers
numeric_cols = Employees.select_dtypes(include=['int64', 'float64']).columns


for col in numeric_cols:
    Q1 = Employees[col].quantile(0.25)
    Q3 = Employees[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = Employees[(Employees[col] < lower_bound) | (Employees[col] > upper_bound)]

    if len(outliers) > 0:
        print(f"\n🔹 Column: {col}")
        print(f"   Number of Outliers: {len(outliers)}")
    else :
        print("No Outliers")
```

```
No Outliers
No Outliers

🔹 Column: Salary
   Number of Outliers: 561

🔹 Column: StockOptionLevel
   Number of Outliers: 361
No Outliers
No Outliers
No Outliers
No Outliers
No Outliers
No Outliers
No Outliers
No Outliers
No Outliers
```

Experience Level in relation to Years in Recent Role

Define and Apply Experience Level Based on Years in Most Recent Role

[71]:
```python
def experience_level(x):
    if x < 2:
        return 'Junior'
    elif x < 5:
        return 'Mid'
    else:
        return 'Senior'

Employees['ExperienceLevel'] = Employees['YearsInMostRecentRole'].apply(experience_level)
```

[72]:
```python
# Detect and Count Salary Outliers Before Capping (Grouped by Year, Job Role, and Experience Level)

outliers_before = []

for (year, role, exp), group in Employees.groupby(['ReviewYear', 'JobRole', 'ExperienceLevel']):
    Q1 = group['Salary'].quantile(0.25)
    Q3 = group['Salary'].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    group_outliers = group[(group['Salary'] < lower) | (group['Salary'] > upper)]
    outliers_before.append(group_outliers)

outliers_before_df = pd.concat(outliers_before)
num_outliers_before = len(outliers_before_df)
print(f"Outliers_Num_Before_Preprocessing: {num_outliers_before}")
```

```
Outliers_Num_Before_Preprocessing: 277
```

[73]:
```python
# Apply Contextual Capping to Adjust Salary Outliers (Grouped by Year, Job Role, and Experience Level)

data_capped = Employees.copy()

for (year, role, exp), group in Employees.groupby(['ReviewYear', 'JobRole', 'ExperienceLevel']):
    Q1 = group['Salary'].quantile(0.25)
    Q3 = group['Salary'].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    data_capped.loc[Employees.index, 'New_Salary'] = Employees['Salary'].clip(lower, upper)
```

[74]:
```python
# Detect and Count Salary Outliers After Capping (Grouped by Year, Job Role, and Experience Level)

outliers_after = []

for (year, role, exp), group in data_capped.groupby(['ReviewYear', 'JobRole', 'ExperienceLevel']):
    Q1 = group['New_Salary'].quantile(0.25)
    Q3 = group['New_Salary'].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    group_outliers = group[(group['New_Salary'] < lower) | (group['New_Salary'] > upper)]
    outliers_after.append(group_outliers)

remaining_outliers = pd.concat(outliers_after)
num_outliers_after = len(remaining_outliers)
print(f"Outliers_Num_After_Preprocessing: {num_outliers_after}")
```

```
Outliers_Num_After_Preprocessing: 198
```

[77]:
```python
improvement = ((num_outliers_before - num_outliers_after) / num_outliers_before) * 100
print(f"Outliers_Num_Before_Preprocessing: {num_outliers_before}")
print(f"Outliers_Num_After_Preprocessing: {num_outliers_after}")
print(f"Improvement: {num_outliers_before - num_outliers_after} less")
print(f"Improvement_Rate: {improvement:.2f}%")
```

```
Outliers_Num_Before_Preprocessing: 277
Outliers_Num_After_Preprocessing: 198
Improvement: 79 less
Improvement_Rate: 28.52%
```

[76]:
```
remaining_outliers[['EmployeeID', 'ReviewYear', 'JobRole', 'ExperienceLevel', 'YearsInMostRecentRole', 'New_Salary']].sort_values('New_Salary', ascending
```

[76]:

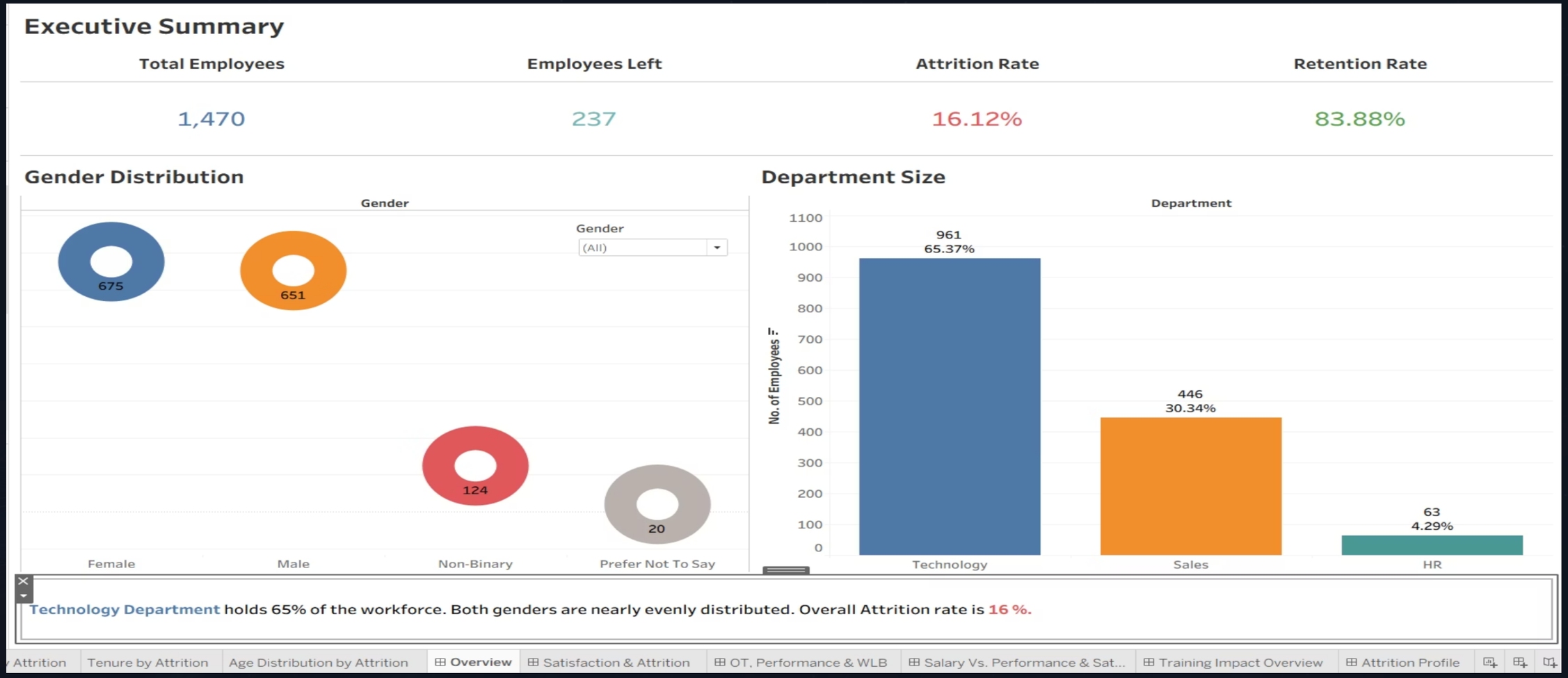
	EmployeeID	ReviewYear	JobRole	ExperienceLevel	YearsInMostRecentRole	New_Salary
2234	79F7-78EC	2013.0	Sales Executive	Senior	5	133668.0
1445	DF11-9C46	2016.0	Software Engineer	Mid	4	133668.0
156	ED73-F078	2016.0	Software Engineer	Mid	3	133668.0
1822	CD4D-7BBA	2016.0	Software Engineer	Junior	0	133668.0
163	ED73-F078	2015.0	Software Engineer	Mid	3	133668.0
1860	9811-F345	2016.0	Data Scientist	Senior	9	133668.0
1861	9811-F345	2017.0	Data Scientist	Senior	9	133668.0
160	ED73-F078	2020.0	Software Engineer	Mid	3	133668.0
1449	DF11-9C46	2020.0	Software Engineer	Mid	4	133668.0
3020	DEB9-1BBF	2020.0	Software Engineer	Mid	3	133668.0
580	4969-9A35	2020.0	Software Engineer	Senior	9	133668.0
920	3994-472A	2020.0	Software Engineer	Senior	10	133668.0
6367	AFC3-E23F	2021.0	Data Scientist	Junior	1	133668.0
5865	66D5-3142	2021.0	Data Scientist	Junior	0	133668.0
1858	819A-2C9C	2021.0	Data Scientist	Mid	2	133668.0
1865	9811-F345	2021.0	Data Scientist	Senior	9	133668.0
1827	CD4D-7BBA	2021.0	Software Engineer	Junior	0	133668.0
921	3994-472A	2021.0	Software Engineer	Senior	10	133668.0
3021	DEB9-1BBF	2021.0	Software Engineer	Mid	3	133668.0
581	4969-9A35	2021.0	Software Engineer	Senior	9	133668.0

[78]:
```
data_capped.drop(
    data_capped[
        (data_capped["EmployeeID"] == "1A7C-19DB") &
        (data_capped["ReviewYear"] == 2015)
    ].index,
    inplace=True
)
```

[79]:
```
data_capped.to_excel('HR_Cleaned.xlsx', index=False)
data_capped.to_csv('HR_Cleaned.csv', index=False)
```

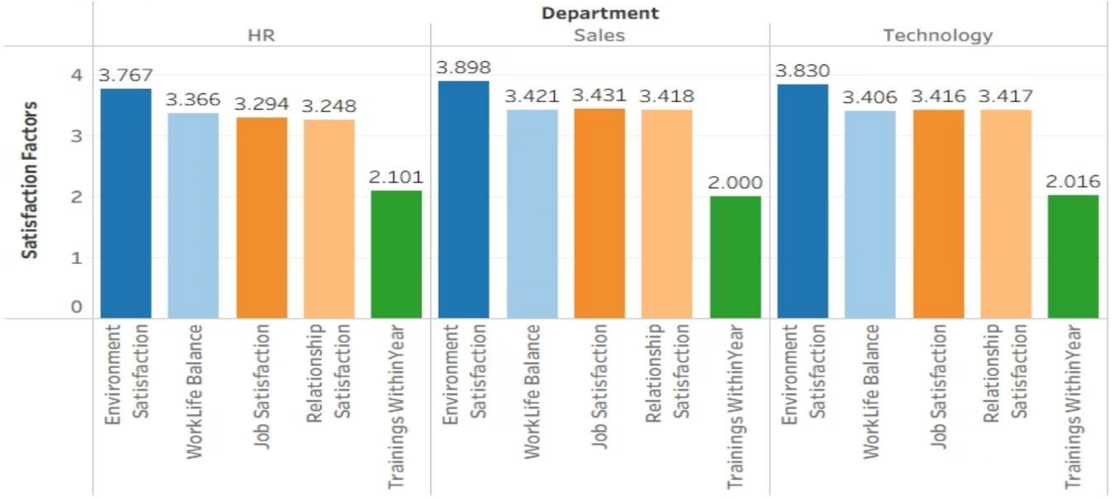
Data Visualization (Tableau Dashboard)

An Overview for HR Attrition Project

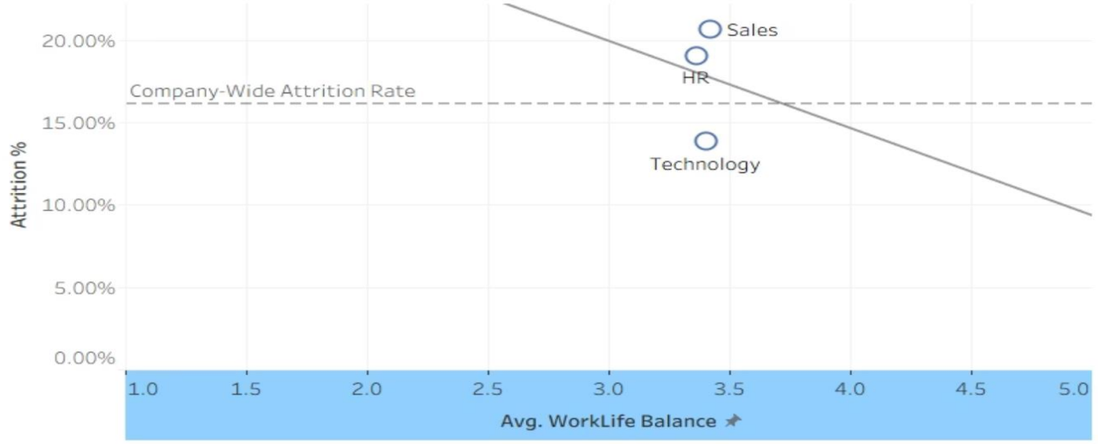


Satisfaction Drivers Dashboard

Employee Satisfaction & Attrition Drivers



WLB Vs. Attrition



Department Satisfaction Metrics

2.000 — 3.898

Department	Environment Satisfaction	Job Satisfaction	Relationship Satisfaction	WorkLife Balance	Trainings WithinYear
HR	3.767	3.294	3.248	3.366	2.101
Sales	3.898	3.431	3.418	3.421	2.000
Technology	3.830	3.416	3.417	3.406	2.016

Insights:

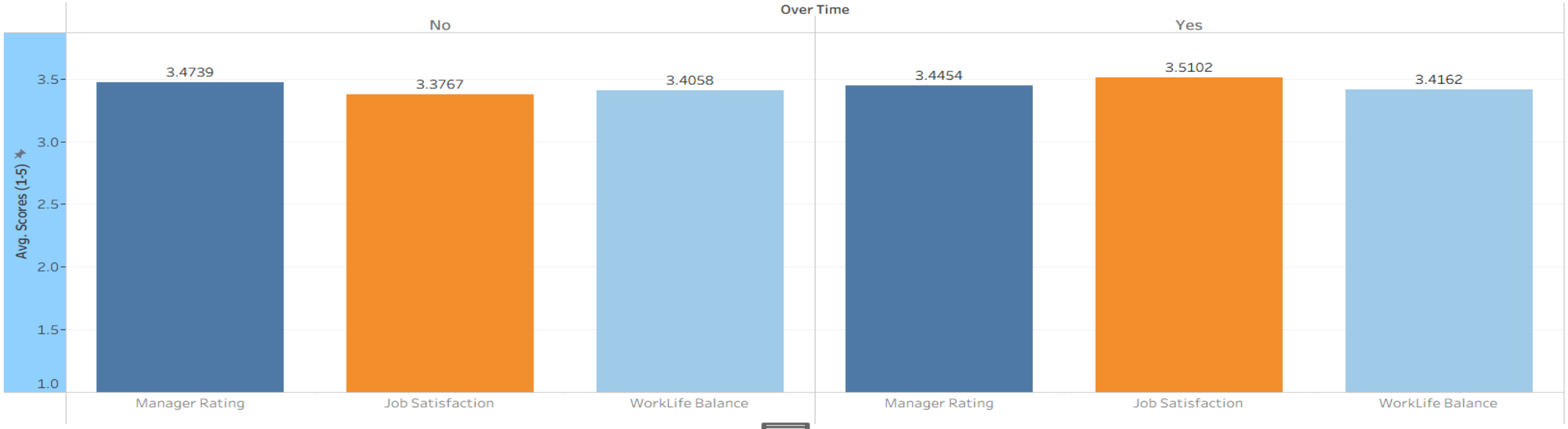
Satisfaction Scores across departments vary only **slightly**, indicating that Attrition is not driven by satisfaction alone. Other factors such as WLB, Salary, and Overtime appear more impactful.

Overtime Impact Dashboard

Impact of Overtime on Attrition & Performance

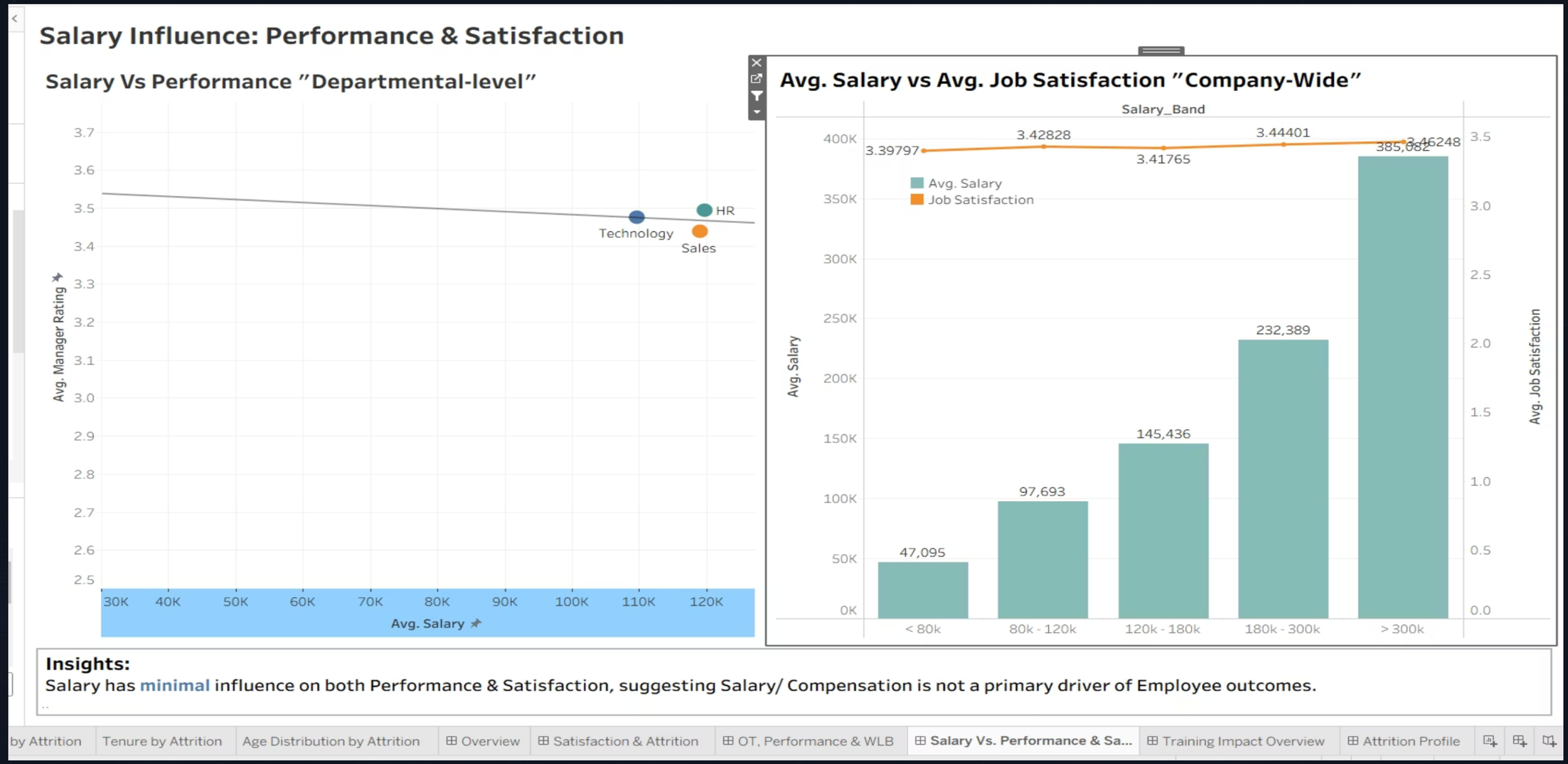
Total Employees	Employees Left	Attrition Rate	WorkLife Balance
1,470	237	16.12%	3

Overtime Vs. WLB & Performance

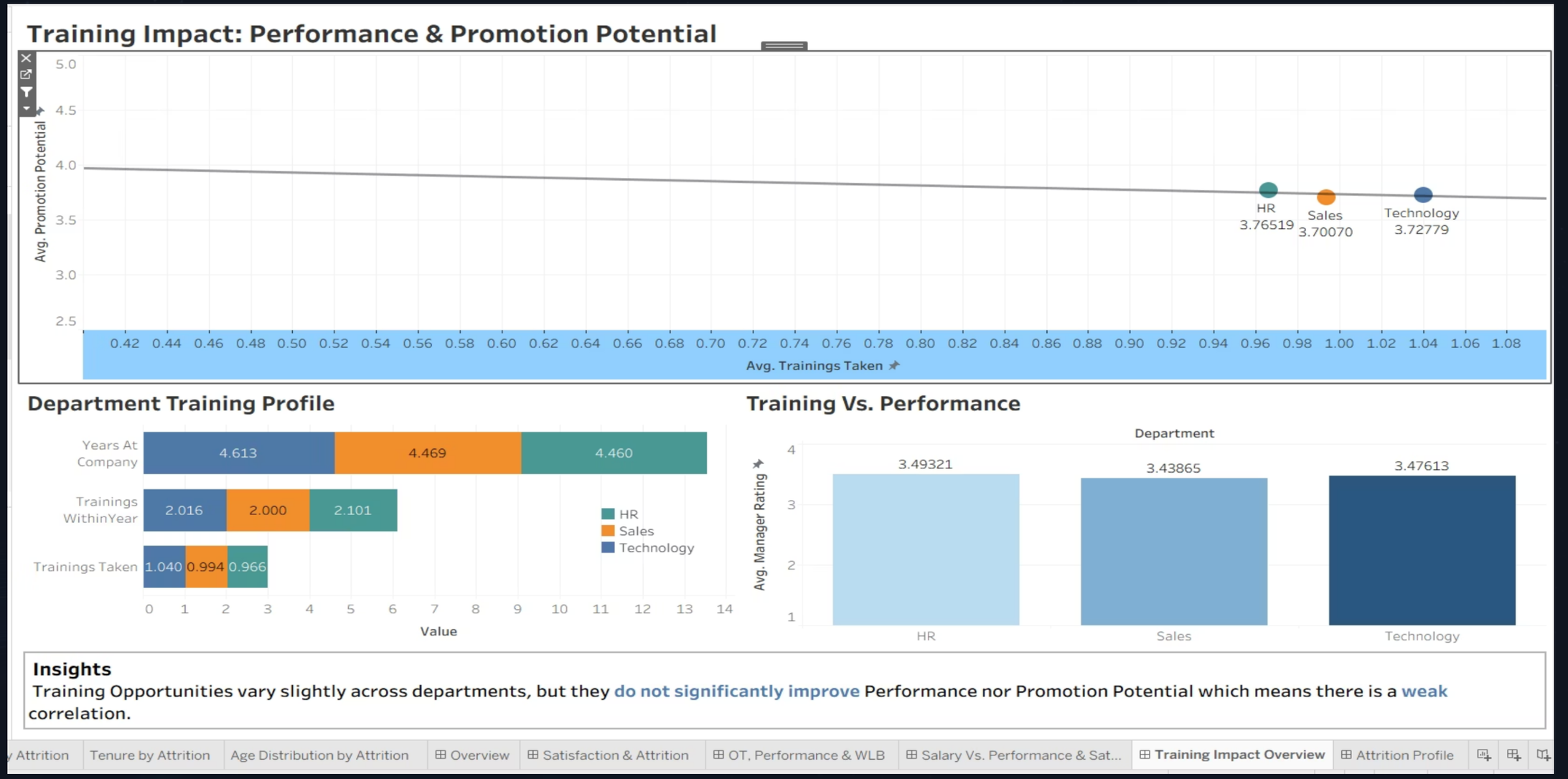


Insights:
Employees working overtime have higher Job Satisfaction, however, lower Performance indicating overtime affects employee performance.

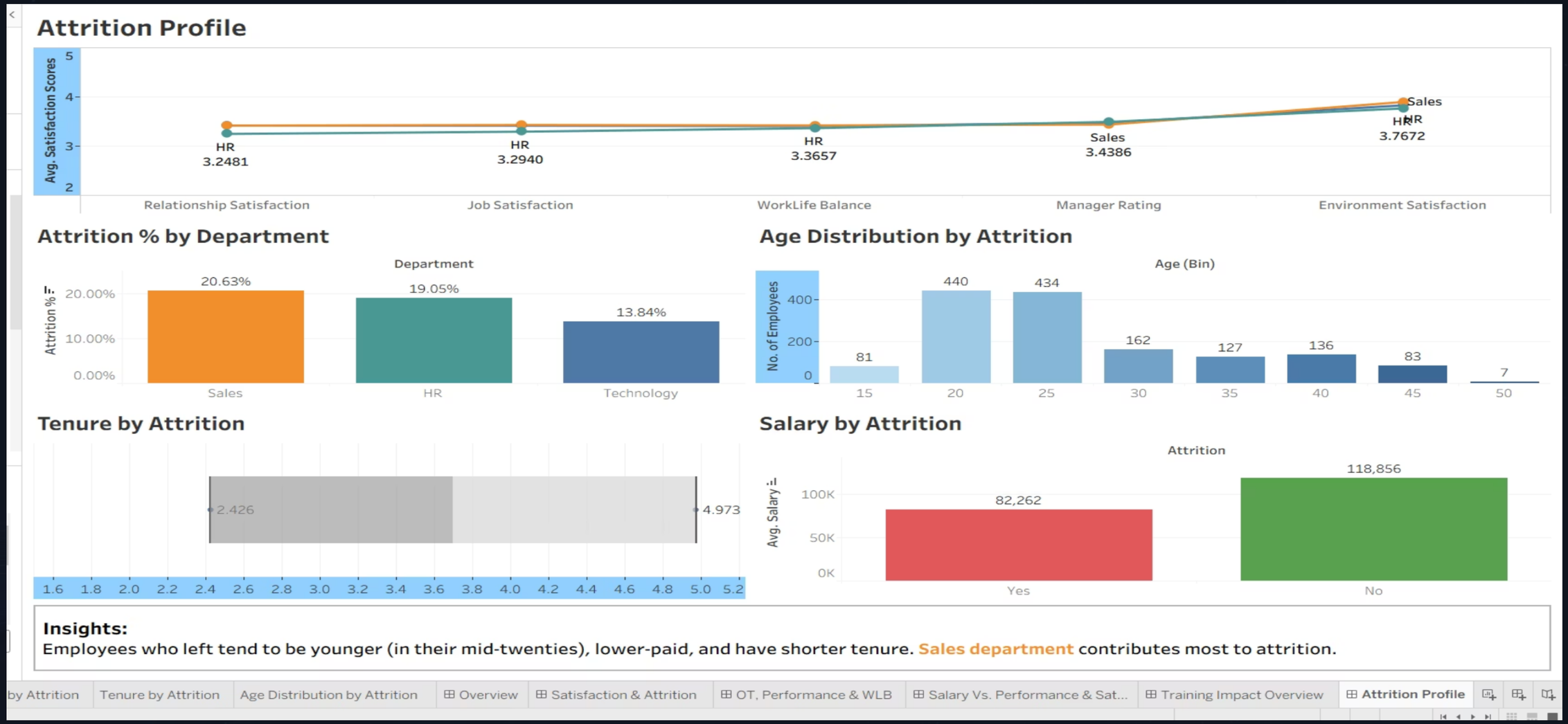
Salary Influence Dashboard



Training Impact Dashboard



Attrition Profile Dashboard



Conclusion

Our HR Analytics project revealed several key patterns shaping Employee Satisfaction and turnover:

- The **Technology department dominates the workforce (65%)**, and gender distribution is nearly **balanced** across the organization.
- Attrition is **moderately high at 16%**, but satisfaction scores across departments remain **relatively consistent**, suggesting that **Attrition is not driven by Satisfaction alone**, but rather by other operational and behavioral factors.
- **Salary level shows Limited influence** on Satisfaction or Performance, suggesting **compensation is not the main concern** for most employees.
- **Training opportunities show Weak impact** on Performance or Promotion readiness, indicating current programs **may lack** effectiveness or alignment with role needs.
- Employees who leave are generally **younger, earlier in their careers, lower-paid, and have shorter tenures**. The **Sales department** contributes **most** to the **overall attrition rate.**

Recommendations to effectively reduce Employee turnover & strengthen Employee Experience

Improve Early-Career Retention

- Create structured career paths, mentorship programs, and faster development cycles targeting younger, low-tenure employees who are most at risk.
- Create growth pathways to reduce early exits.

Enhance WLB Company-Wide

- Introduce flexible scheduling, remote-work options, or mandatory time-off initiatives.
- Regularly monitor WLB metrics for early warning signs.

Address High Attrition in Sales Department

- Review workload, targets, incentives, and managerial support practices.
- Conduct focused interviews or targeted pulse surveys to uncover department-specific stressors.

Recommendations to effectively reduce Employee turnover & strengthen Employee Experience

Reassess Training Strategies

- Shift from quantity to **quality and relevance**: design role-specific, skill-based, and progression-aligned development programs.
- Evaluate training effectiveness and ensure it contributes to skill growth and promotion readiness.

Maintain Transparent Compensation Strategy

- While salary is not a primary turnover driver, ensure fairness across levels and roles to prevent future dissatisfaction.

Adopt Predictive Analytics (Strategic Add-On)

- Build attrition prediction models to proactively identify high-risk employees and intervene early.

Thank You!

Feel free to ask any questions!