# Employee Performance and Attrition Report

- Sayan Das

## 1. Data Extraction: Loading CSV Files into Pandas

I began by extracting the data from the provided CSV files. The project involved three datasets: - **Employee Data:** Contains personal and job-related details. - **Attrition Data:** Contains attrition flags and exit interview scores. - **Employee Performance Data:** Contains performance ratings, training hours, and other performance metrics.

I loaded each CSV into a Pandas DataFrame and ensured that the lower-case gender column was removed so that only the Gender column was used. Here is the code I used:

# Read CSV files

employee\_df = pd.read\_csv('employee\_data 1.csv')

attrition\_df = pd.read\_csv('Attrition 1.csv')

performance\_df = pd.read\_csv('employee\_performance\_data 1.csv')

# Display initial shapes

print("Employee Data Shape:", employee\_df.shape)

print("Attrition Data Shape:", attrition\_df.shape)

print("Performance Data Shape:", performance\_df.shape)

# Drop lower-case 'gender' column if it exists; keep only the 'Gender' column.

if 'gender' in employee\_df.columns:

    employee\_df = employee\_df.drop(columns=['gender'])

print("Columns in employee\_df after dropping 'gender':", employee\_df.columns.tolist())

## 2. Data Transformation & Cleaning

After loading the data, I validated the uniqueness of the primary key (Employee\_ID) in each dataset to ensure data integrity.

# Check uniqueness in employee data

if employee\_df['Employee\_ID'].is\_unique:

    print("Employee\_ID is unique in employee\_data.")

else:

    print("Employee\_ID has duplicates in employee\_data.")

# Standardize column name in attrition data and check uniqueness

attrition\_df.rename(columns={'employee\_ID': 'Employee\_ID'}, inplace=True)

if attrition\_df['Employee\_ID'].is\_unique:

    print("Employee\_ID is unique in attrition\_data.")

else:

    print("Employee\_ID has duplicates in attrition\_data.")

# Check uniqueness in performance data

if performance\_df['Employee\_ID'].is\_unique:

    print("Employee\_ID is unique in performance\_data.")

else:

    print("Employee\_ID has duplicates in performance\_data.")

I then merged the datasets: - I performed an inner join between employee\_df and performance\_df on Employee\_ID to capture only the employees with available performance data. - I then left-joined the resulting DataFrame with attrition\_df on Employee\_ID.

**Since the project required only complete records, I removed any rows that had missing values in either the attrition or Exit\_Interview\_Score columns**.

**Just because I don't have the data do not mean that those employee havn’t left the company.**

# Merge employee and performance data on Employee\_ID (inner join)

emp\_perf\_df = pd.merge(employee\_df, performance\_df, on='Employee\_ID', how='inner')

print("Shape after merging employee and performance data:", emp\_perf\_df.shape)

# Merge with attrition data (left join)

full\_df = pd.merge(emp\_perf\_df, attrition\_df, on='Employee\_ID', how='left')

print("Shape before dropping incomplete records:", full\_df.shape)

# Drop rows with missing values for 'attrition' or 'Exit\_Interview\_Score'

full\_df = full\_df.dropna(subset=['attrition', 'Exit\_Interview\_Score'])

print("Shape after dropping rows with missing attrition or exit interview score:", full\_df.shape)

I also merged the first\_name and last\_name columns into a single name column in the employee dimension later in the process.

## 3. Creating Fact and Dimension Tables (Star Schema)

I then transformed the cleaned DataFrame into a star schema by creating one fact table and several dimension tables.

### Fact Table: fact\_employee\_performance

This table captures performance metrics along with attrition and exit interview scores. It also includes surrogate key references for departments and job roles.

fact\_table = full\_df[['Employee\_ID', 'Performance\_Rating', 'Last\_Promotion\_Year',

                        'Training\_Hours', 'Work\_Life\_Balance', 'Job\_Satisfaction',

                        'attrition', 'Exit\_Interview\_Score']].copy()

### Dimension Tables

#### Employee Dimension (dim\_employee)

I excluded Department and Job\_Role from this dimension, and I merged first\_name and last\_name into a new name column.

# Create Employee Dimension (exclude department and job role)

dim\_employee = full\_df[['Employee\_ID', 'Age', 'first\_name', 'last\_name', 'Gender',

                          'Education\_Level', 'Marital\_Status', 'Job\_Tenure', 'Distance\_From\_Home']].drop\_duplicates()

# Merge first and last names into a single column 'name'

dim\_employee['Name'] = dim\_employee['first\_name'] + ' ' + dim\_employee['last\_name']

dim\_employee = dim\_employee.drop(columns=['first\_name', 'last\_name'])

#### Department Dimension (dim\_department)

I created a table containing unique departments and added a surrogate key:

dim\_department = full\_df[['Department']].drop\_duplicates().reset\_index(drop=True)

dim\_department['Department\_ID'] = dim\_department.index + 1

print("Department Dimension Shape:", dim\_department.shape)

#### Role Dimension (dim\_role)

Similarly, I created a role dimension table:

dim\_role = full\_df[['Job\_Role']].drop\_duplicates().reset\_index(drop=True)

dim\_role['Role\_ID'] = dim\_role.index + 1

Next, I merged the department and role information into the fact table to reference their surrogate keys:

# First, add the original department and job role columns to fact table for the lookup.

fact\_table = pd.merge(fact\_table, full\_df[['Employee\_ID', 'Department', 'Job\_Role']], on='Employee\_ID', how='left')

# Merge department ID from dim\_department

fact\_table = pd.merge(fact\_table, dim\_department, on='Department', how='left')

# Merge role ID from dim\_role

fact\_table = pd.merge(fact\_table, dim\_role, on='Job\_Role', how='left')

# Remove redundant text columns (Department and Job\_Role) after merging IDs

fact\_table.drop(columns=['Department', 'Job\_Role'], inplace=True)

print("Fact Table Shape:", fact\_table.shape)

## 4. Removing Duplicate Employee\_ID Records

Although I ensured data integrity during transformation, I also implemented a method to remove duplicate Employee\_ID records directly from the fact\_employee\_performance table.

### In SQL:

After loading the transformed data into MySQL, I used the following SQL code to remove any duplicate records from the fact\_employee\_performance table. To work around MySQL safe update mode, I disabled safe updates for the session:

-- Disable safe update mode for this session  
SET SQL\_SAFE\_UPDATES = 0;  
  
-- Add a temporary auto-increment primary key column  
ALTER TABLE fact\_employee\_performance   
ADD COLUMN temp\_id INT AUTO\_INCREMENT PRIMARY KEY;  
  
-- Delete duplicate rows, keeping the row with the smallest temp\_id for each Employee\_ID  
DELETE f1   
FROM fact\_employee\_performance f1  
INNER JOIN fact\_employee\_performance f2   
 ON f1.Employee\_ID = f2.Employee\_ID   
 AND f1.temp\_id > f2.temp\_id;  
  
-- Remove the temporary column  
ALTER TABLE fact\_employee\_performance   
DROP COLUMN temp\_id;

This SQL code ensures that only one record per Employee\_ID remains in the fact table.

## 5. Loading Transformed Data into MySQL

I used SQLAlchemy to connect to the MySQL database and loaded the fact and dimension tables into their respective tables. Here is the code snippet:

**from sqlalchemy import create\_engine**  
# MySQL connection details  
username = 'root'

password = '12345'

host = 'localhost'

port = '3306'

database = 'case3'

engine = create\_engine(f'mysql+pymysql://{username}:{password}@{host}:{port}/{database}')  
  
# Load tables into MySQL

fact\_table.to\_sql('fact\_employee\_performance', con=engine, if\_exists='replace', index=False)

dim\_employee.to\_sql('dim\_employee', con=engine, if\_exists='replace', index=False)

dim\_department.to\_sql('dim\_department', con=engine, if\_exists='replace', index=False)

dim\_role.to\_sql('dim\_role', con=engine, if\_exists='replace', index=False)

print("Data loaded to MySQL successfully.")

## 6. KPI Tracking & Monitoring

* **DAX measures**
  + **Attrition Rate:**

Attrition Rate =

DIVIDE(

CALCULATE(COUNTROWS(fact\_employee\_performance), fact\_employee\_performance[attrition] = TRUE),

COUNTROWS(fact\_employee\_performance)

)

* + **Retention Rate:**

Retention Rate = 1 - [Attrition Rate]

* + **Average Tenure:**

Average Tenure = AVERAGE(dim\_employee[Job\_Tenure])

* + **Department-wise Employee Score:**

Average Employee Satisfaction = AVERAGE(fact\_employee\_performance[Job\_Satisfaction])

* + **Average Performance Rating:**

Average Performance Rating = AVERAGE(fact\_employee\_performance[Performance\_Rating])

* + **Average Exit Interview Satisfaction Score:**

Average Exit Interview Score = AVERAGE(fact\_employee\_performance[Exit\_Interview\_Score])

* + **Department-wise Attrition Rate:**

Dept Attrition Rate =

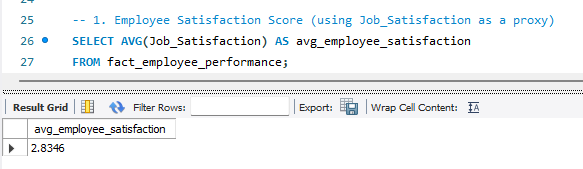
DIVIDE(

CALCULATE(COUNTROWS(fact\_employee\_performance), fact\_employee\_performance[attrition] = TRUE),

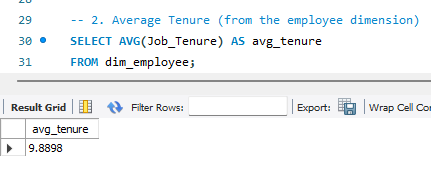
COUNTROWS(fact\_employee\_performance)

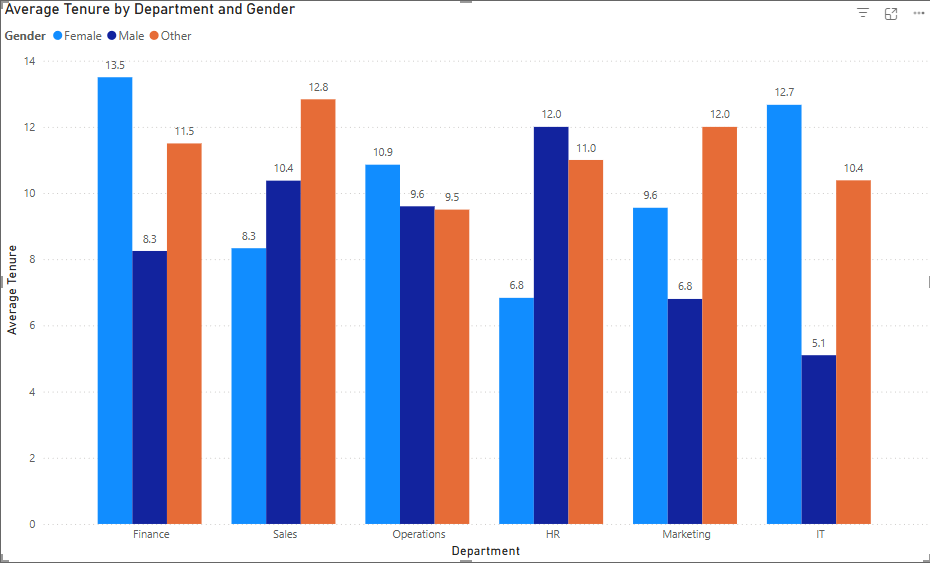
)

* Employee Satisfaction Score:

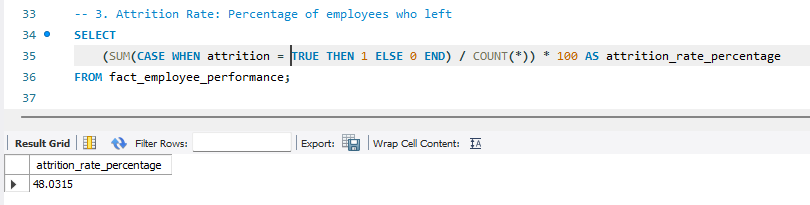


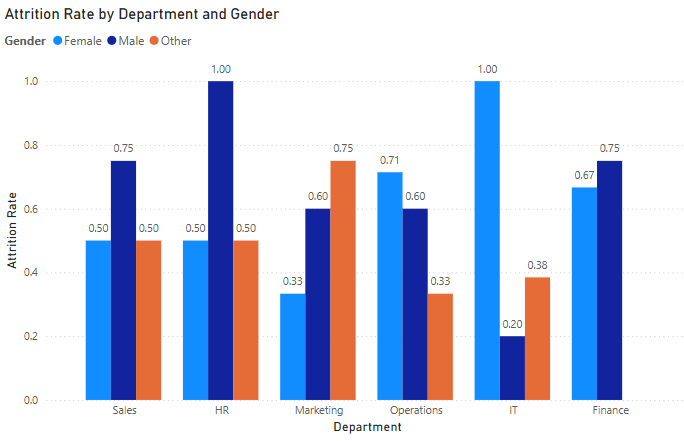
* Average Tenure:



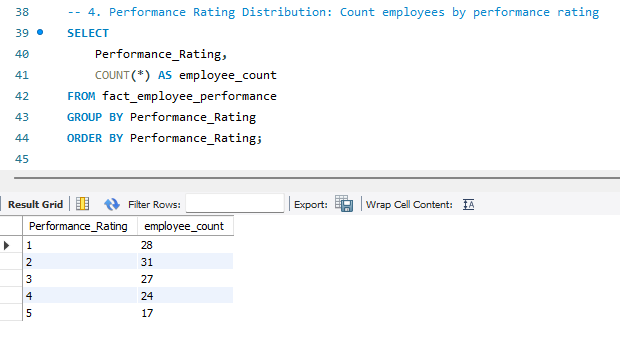


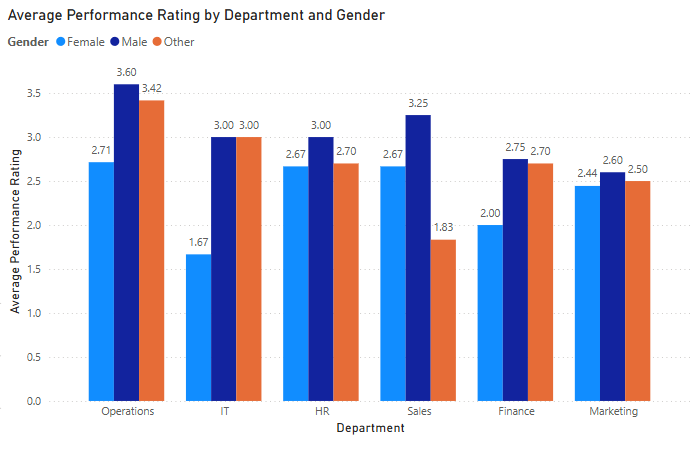
* Attrition Rate:



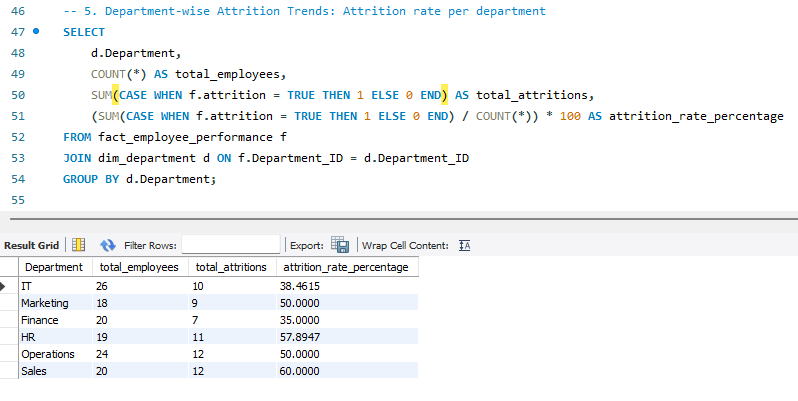


* Performance Rating Distribution:



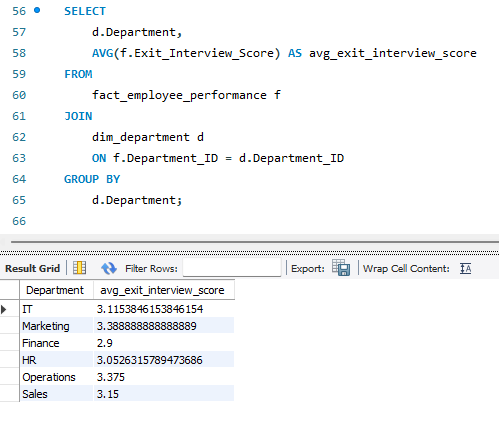


* Department-wise Attrition Trends:

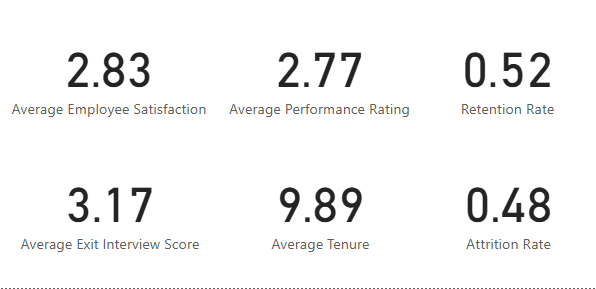


* Exit Interview Sentiment Analysis:

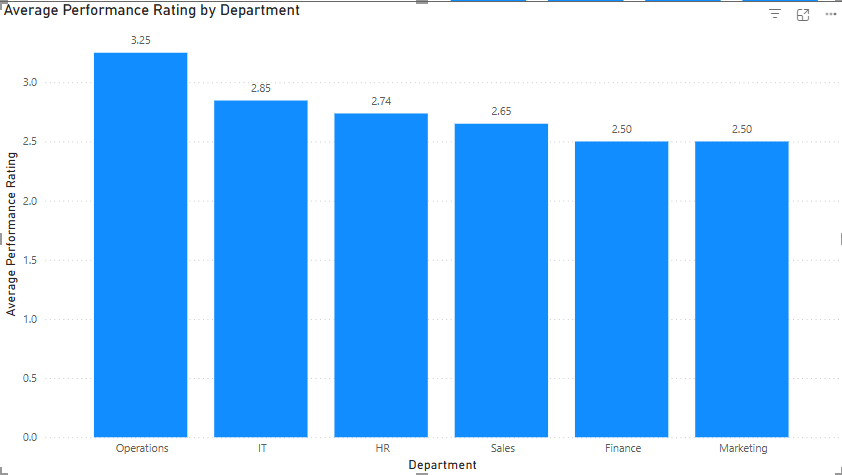
Given below are average Exit\_Interview\_Score per department. I cannot do Text analysis because there is no text/transcript to analyse.



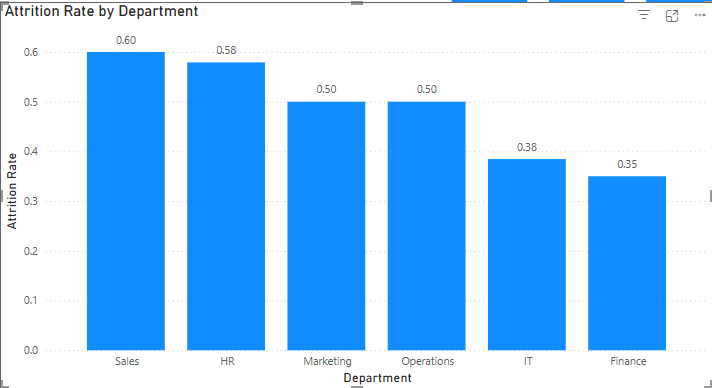
* Attrition Rate, Retention Rate, Average Tenure, Average Performance Rating, Average Performance Rating



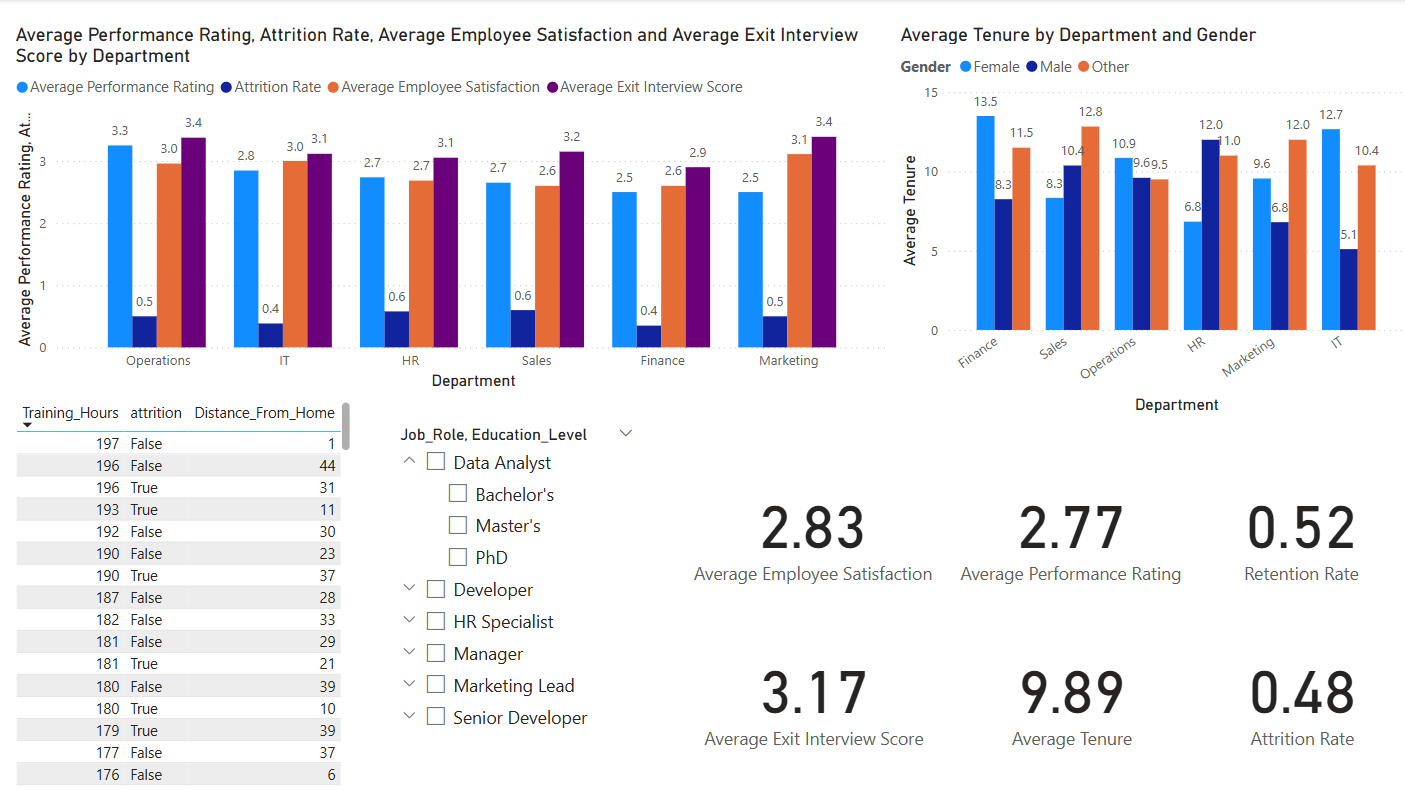
* Department-wise Employee Score – I assumed performance rating is same as employee code.



* Department wise Attrition Rate



* Overall



## 7. Conclusion

**Department-Level Observations:**

* **HR** appears to have the **highest performance rating** and **highest employee satisfaction** among departments. Correspondingly, it shows a **lower attrition rate** and relatively higher exit interview scores.
  + **Interpretation**: HR’s strong performance and job satisfaction likely contribute to reduced turnover.
* **Marketing** exhibits the **highest attrition rate** alongside the **lowest satisfaction** and **lowest exit interview scores**.
  + **Interpretation**: High turnover may be tied to lower satisfaction; it suggests a need for deeper investigation into work conditions, role expectations, or leadership in Marketing.
* **IT** and **Finance** fall somewhere in the middle, with moderate performance ratings and satisfaction. However, Finance’s attrition rate is somewhat high, indicating room for improvement.
* **Operations** has moderately high attrition but not as severe as Marketing, suggesting some departmental-specific issues.

**Average Tenure Differences:**

* **HR** employees tend to have **longer average tenure**, suggesting higher retention and possibly better internal mobility or more favorable work conditions.
* **Marketing** tends to show **shorter average tenure**, aligning with the higher attrition rate. This could point to burnout, role dissatisfaction, or a mismatch in job expectations.

**Overall Company Metrics:**

* **Average Employee Satisfaction** is around **2.83** (on the scale shown). This is neither very high nor extremely low, but it does leave room for improvement.
* **Average Performance Rating** is **2.77**, indicating that most employees are performing moderately. Departments like HR stand out with slightly higher averages, while Marketing and Finance might need targeted performance management interventions.
* **Average Exit Interview Score** is **2.52**, which suggests that employees who do leave have mixed feelings. Departments with particularly low scores (like Marketing) should investigate root causes—possibly leadership issues, workload concerns, or career development opportunities.
* **Attrition Rate** of **0.48** (48%) is notably high. This signals that nearly half of the workforce observed may be leaving in the measured period. Reducing attrition, especially in high-turnover departments, should be a top priority.
* **Average Tenure** is around **9.89 years**, which is reasonably long overall, but it likely skews higher in departments like HR and lower in Marketing.

**Actionable Insights & Recommendations:**

* **Focus on Marketing**: With the highest attrition rate and lowest satisfaction, it’s crucial to investigate whether employees have clear career paths, sufficient resources, and supportive management.
* **Boost Overall Satisfaction**: Since satisfaction correlates with lower attrition, HR interventions—like flexible schedules, better recognition, and clearer promotion paths—could improve retention.
* **Continue HR Best Practices**: The HR department’s relatively high satisfaction and lower attrition can be used as a case study for best practices—potentially replicating them in other departments.