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
```

```sql
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:

```python
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:

```
-- 1. Employee Satisfaction Score (using Job_Satisfaction as a proxy)
SELECT AVG(Job_Satisfaction) AS avg_employee_satisfaction
FROM fact_employee_performance;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

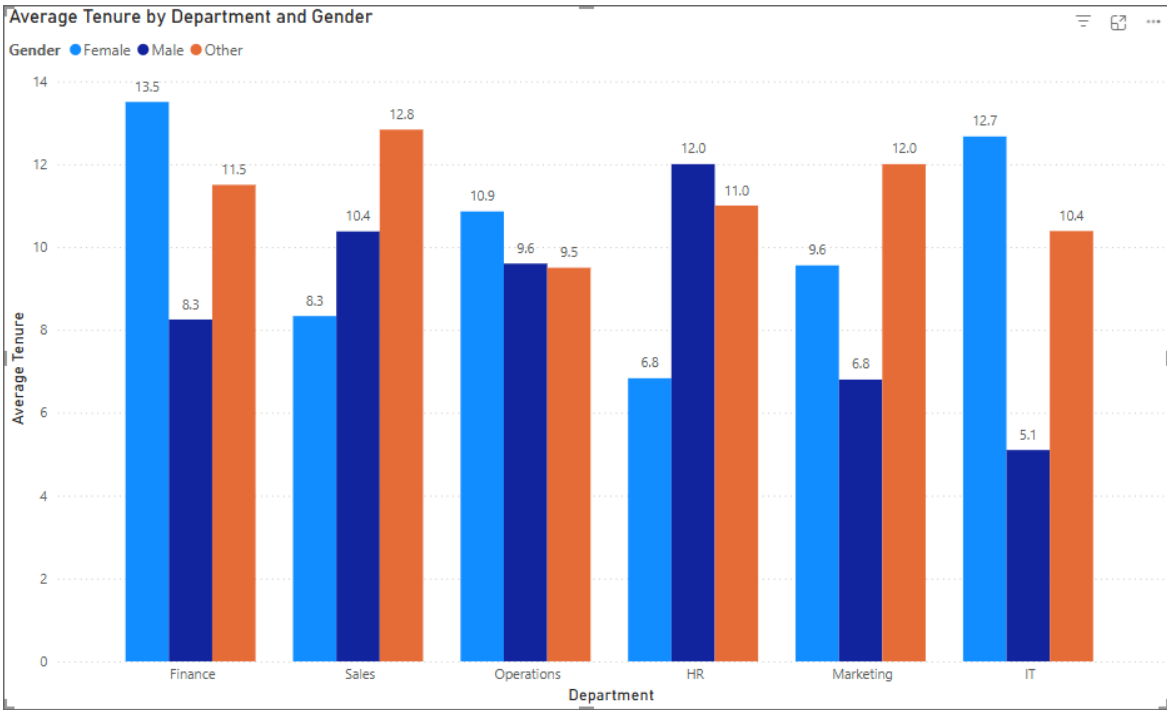
avg_employee_satisfaction
2.8346

- Average Tenure:

```sql
-- 2. Average Tenure (from the employee dimension)
SELECT AVG(Job_Tenure) AS avg_tenure
FROM dim_employee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

	avg_tenure
▶	9.8898

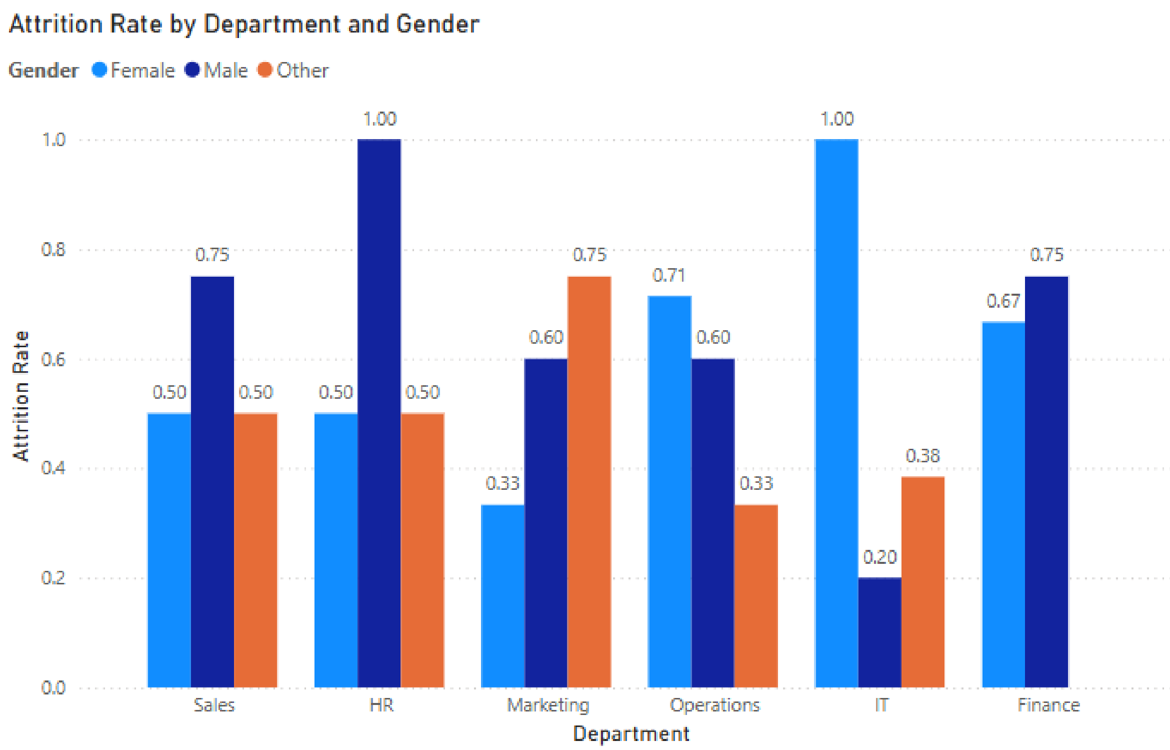


- Attrition Rate:

```sql
-- 3. Attrition Rate: Percentage of employees who left
SELECT
    (SUM(CASE WHEN attrition = TRUE THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS attrition_rate_percentage
FROM fact_employee_performance;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	attrition_rate_percentage
▶	48.0315

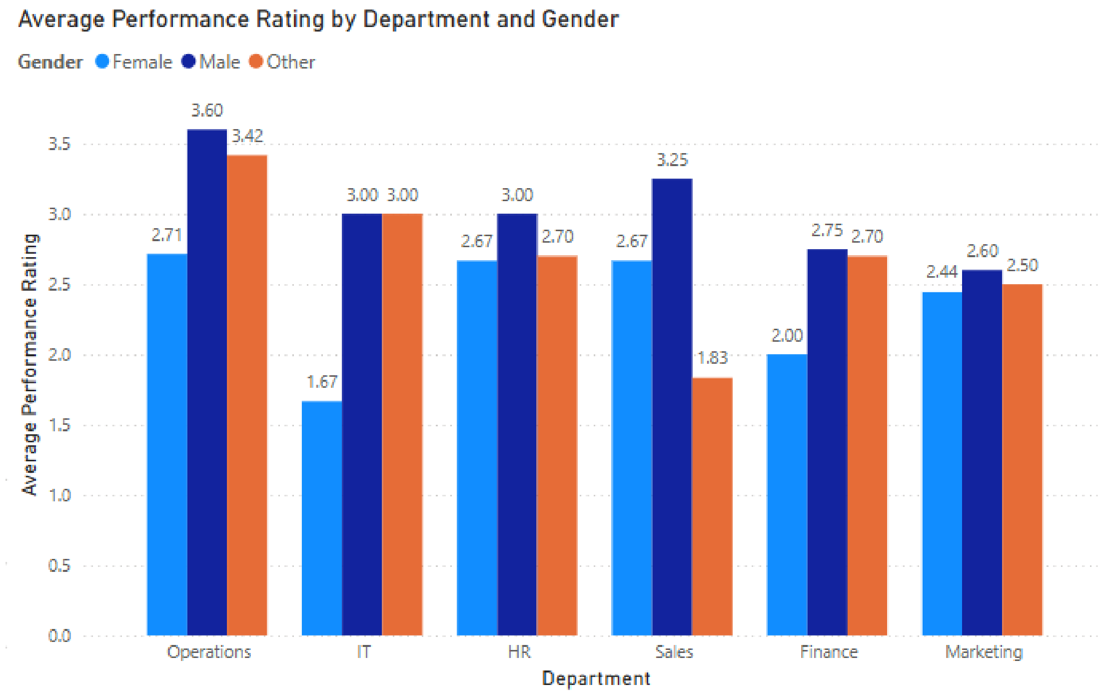


- Performance Rating Distribution:

```sql
-- 4. Performance Rating Distribution: Count employees by performance rating
SELECT
    Performance_Rating,
    COUNT(*) AS employee_count
FROM fact_employee_performance
GROUP BY Performance_Rating
ORDER BY Performance_Rating;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Performance_Rating	employee_count
1	28
2	31
3	27
4	24
5	17



- Department-wise Attrition Trends:

```sql
-- 5. Department-wise Attrition Trends: Attrition rate per department
SELECT
    d.Department,
    COUNT(*) AS total_employees,
    SUM(CASE WHEN f.attrition = TRUE THEN 1 ELSE 0 END) AS total_attritions,
    (SUM(CASE WHEN f.attrition = TRUE THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS attrition_rate_percentage
FROM fact_employee_performance f
JOIN dim_department d ON f.Department_ID = d.Department_ID
GROUP BY d.Department;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Department	total_employees	total_attritions	attrition_rate_percentage
IT	26	10	38.4615
Marketing	18	9	50.0000
Finance	20	7	35.0000
HR	19	11	57.8947
Operations	24	12	50.0000
Sales	20	12	60.0000

- 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.

```sql
SELECT
    d.Department,
    AVG(f.Exit_Interview_Score) AS avg_exit_interview_score
FROM
    fact_employee_performance f
JOIN
    dim_department d
    ON f.Department_ID = d.Department_ID
GROUP BY
    d.Department;

```

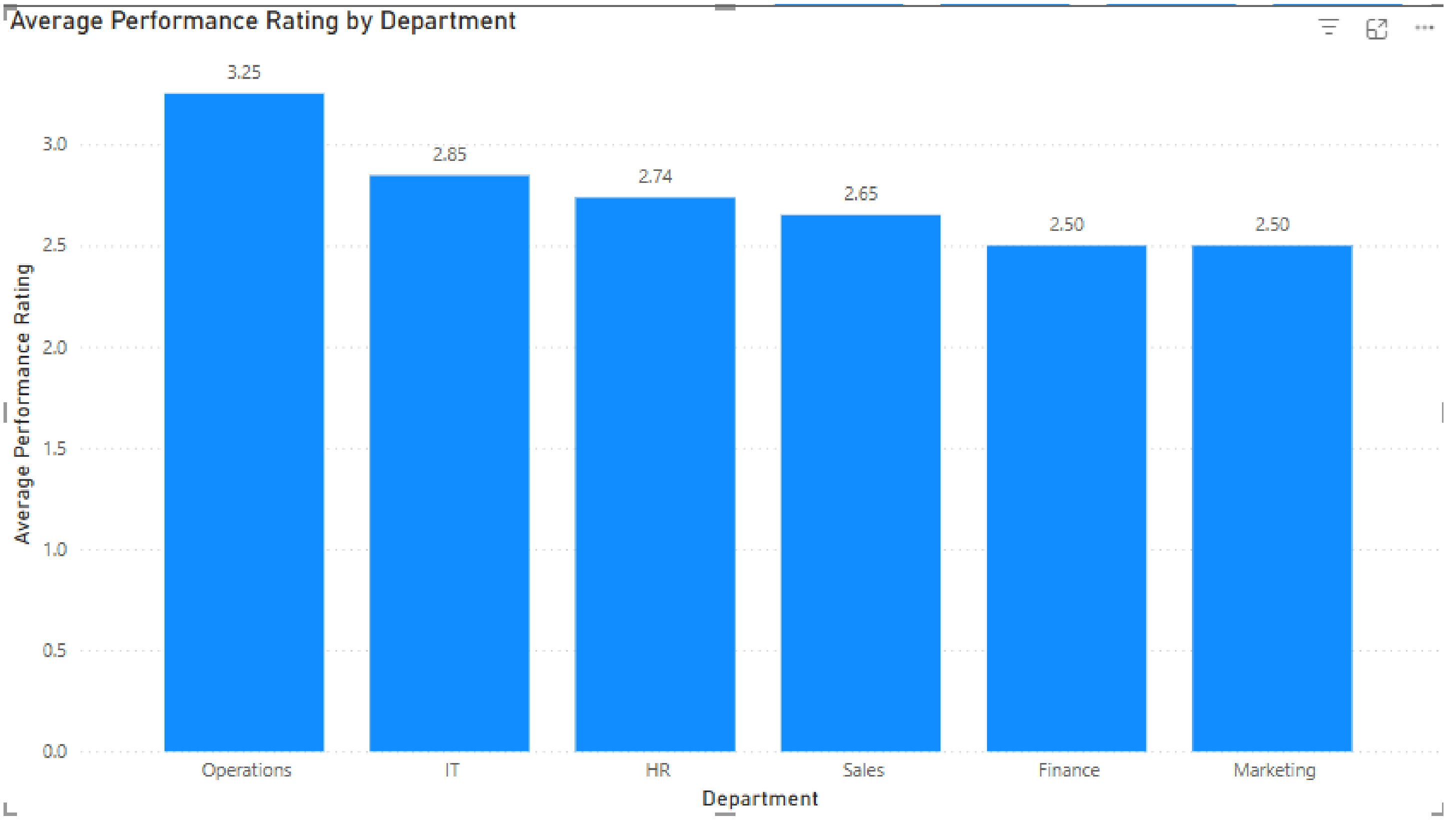
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Department	avg_exit_interview_score
IT	3.1153846153846154
Marketing	3.388888888888889
Finance	2.9
HR	3.0526315789473686
Operations	3.375
Sales	3.15

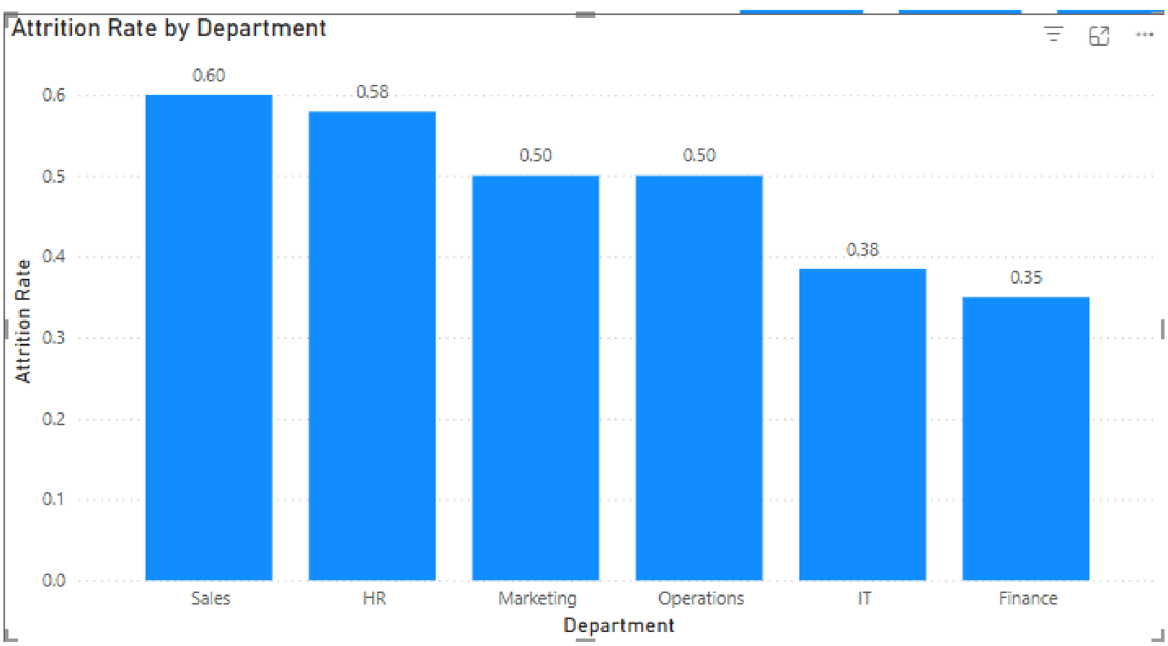
- Attrition Rate, Retention Rate, Average Tenure, Average Performance Rating, Average Performance Rating

2.83 Average Employee Satisfaction	**2.77** Average Performance Rating	**0.52** Retention Rate
3.17 Average Exit Interview Score	**9.89** Average Tenure	**0.48** Attrition Rate

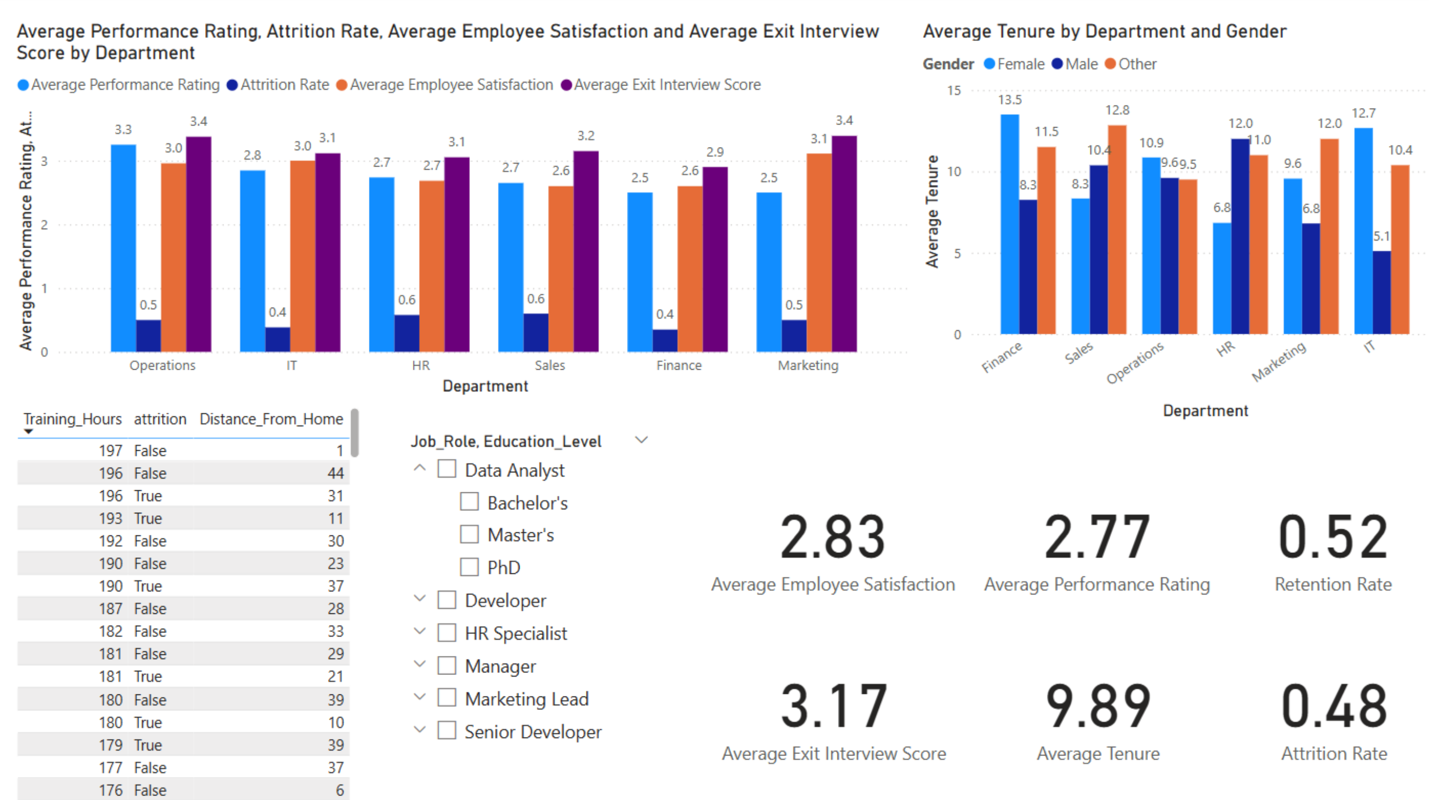
- Department-wise Employee Score – I assumed performance rating is same as employee code.



- Department wise Attrition Rate



- Overall



Training_Hours	attrition	Distance_From_Home
197	False	1
196	False	44
196	True	31
193	True	11
192	False	30
190	False	23
190	True	37
187	False	28
182	False	33
181	False	29
181	True	21
180	False	39
180	True	10
179	True	39
177	False	37
176	False	6

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.