HR DATA ANALYSIS

DEPI Final Project Documentation

1- Introduction:

This project provides an in-depth analysis of HR data to assess key aspects such as employee performance, satisfaction, experience, and attrition. Through interactive dashboards and SQL queries, we explore various metrics, including employee count, average satisfaction, ratings, and salary trends over time. The analysis also covers employee demographics such as age, gender, ethnicity, and business travel frequency.

The project focuses on identifying relationships between employee experience (e.g., years at the company) and performance ratings, as well as the impact of training opportunities on top performers. Additionally, attrition analysis by department and job role helps pinpoint areas with high turnover, allowing HR managers to implement targeted retention strategies.

By combining data visualization and SQL-driven insights, this project equips HR managers with actionable information to enhance employee retention, optimize performance evaluations, and improve overall job satisfaction.

- ## Key goals of this analysis include:

1- **Tracking Workforce Demographics and Diversity:** Gaining insights into workforce composition by gender, ethnicity, age, and business travel frequency to support diversity and inclusion initiatives.

2- **Improving Job Satisfaction:** Assessing satisfaction metrics (e.g., environment, job, work-life balance) across various employee groups to address areas with lower satisfaction and increase engagement.

3- **Understanding Employee Performance Trends:** Analyzing self and manager ratings to identify discrepancies, trends, and areas for improvement across different departments, job roles, and demographics.

4- **Optimizing Training and Development:** Evaluating the relationship between training opportunities and performance to ensure top performers are receiving adequate professional development.

5- **Enhancing Employee Retention:** Identifying factors contributing to employee attrition, such as department, job role, or experience, to develop targeted strategies for reducing turnover.

2- Tools and Technologies Used

1. Jupiter Notebook (Python):

- **Purpose:**

  To prepare and clean the HR datasets for analysis, ensuring data consistency and readiness for exploring employee performance, satisfaction, and attrition trends.

- **Actions:**

  Loaded and merged multiple datasets (employee, performance, education, satisfaction).

  Cleaned data by handling missing values, removing extra spaces, and formatting columns.

  Calculated key metrics like attrition year and filtered out-of-range performance reviews.

  Verified and standardized data types across all datasets for accurate analysis.

2. SQL:

- **Purpose:**

To connect to SQL Server, upload the cleaned data, and perform advanced data analysis using SQL queries directly from the Jupyter notebook for more efficient querying and insight extraction.

- **Actions:**

Established connection to SQL Server using SQLAlchemy and ODBC.

Uploaded preprocessed dataframes as SQL tables.

Executed SQL queries to extract insights on employee performance, satisfaction, and attrition directly from the database.

3. Power BI:

- **Purpose:**

  To visualize and analyze key HR metrics using Power BI dashboards, enabling HR managers to gain insights into employee performance, satisfaction, demographics, and attrition trends. These dashboards provide an interactive way to explore the data and track trends over time.

- **Actions:**

- Created interactive dashboards to visualize employee metrics such as satisfaction levels, performance ratings, and salary trends.
- Included filters for department, job role, year, state, education level, and other key demographics to allow for deeper analysis.
- Tracked and visualized trends in attrition, performance, and satisfaction over time to provide actionable insights for HR decision-making.

• Let's go in depth in steps in each tool:

i. Python:

In this project, we used Python to handle the preprocessing of multiple HR-related datasets. The preprocessing steps are crucial to ensure the data is clean and ready for analysis. The following steps were performed:

Importing Required Libraries

In this section, we import all the necessary libraries that will be used throughout preprocessing. These libraries provide various functions for data manipulation.

```
In [4]: import pandas as pd
        import numpy as np
        import pyodbc
        from sqlalchemy import create_engine
```

Data Loading and Initial Exploration¶

In this section, we load the various datasets required for the analysis, including education levels, employee information, performance ratings, and satisfaction levels. After loading the data, we conduct an initial exploration to inspect the structure of each dataset and identify any missing values or unique entries within key columns.

In [5]:
```
# Load the datasets
education_df = pd.read_csv('C:/Users/SANAD/Final/EducationLevel.csv')
employee_df = pd.read_csv('C:/Users/SANAD/Final/Employee.csv')
performance_df = pd.read_csv('C:/Users/SANAD/Final/PerformanceRating.csv')
rating_df = pd.read_csv('C:/Users/SANAD/Final/RatingLevel.csv')
satisfaction_df = pd.read_csv('C:/Users/SANAD/Final/SatisfiedLevel.csv')

# Displaying the first few rows of each dataset to inspect the structure
education_df_head = education_df.head()
employee_df_head = employee_df.head()
performance_df_head = performance_df.head()
rating_df_head = rating_df.head()
satisfaction_df_head = satisfaction_df.head()

education_df_head, employee_df_head, performance_df_head, rating_df_head, satisfaction_df_head
```

Checking for Missing Values:

We checked each dataset for missing values using isnull().sum(). Fortunately, there were no missing values in any of the datasets, ensuring data integrity and reducing the need for imputation.

In [6]:
```
# Check for missing values in all datasets
missing_education = education_df.isnull().sum()
missing_employee = employee_df.isnull().sum()
missing_performance = performance_df.isnull().sum()
missing_rating = rating_df.isnull().sum()
missing_satisfaction = satisfaction_df.isnull().sum()

missing_education, missing_employee, missing_performance, missing_rating, missing_satisfaction
```

Identifying Unique Values:

To understand the variety of data, we calculated the number of unique values for each column in the employee and performance rating datasets. This step helps identify categorical variables and spot potential anomalies in the data.

In [7]:
```
# Calculate the number of unique values for each column in the employee and performance_rating dataframes
unique_values_employee = employee_df.nunique()
unique_values_performance = performance_df.nunique()

# Display the results
print("Unique Values in Employee Dataset:")
print(unique_values_employee)

print("\nUnique Values in Performance Rating Dataset:")
print(unique_values_performance)
```

Preprocessing Steps

1. **Removing Spaces from the 'EducationField' Column**
   - **Objective**: To ensure that any leading or trailing spaces in the 'EducationField' column are removed for accurate analysis and comparisons
2. **Verifying Unique Values**

- **Objective**: To check the unique values in the 'EducationField' column after removing spaces, ensuring that the preprocessing step was successful.

```
In [9]: # Remove spaces from the 'EducationField' column
        employee_df['EducationField'] = employee_df['EducationField'].str.strip()

        # Check the unique values again to ensure the spaces have been removed
        unique_education_field = employee_df['EducationField'].unique()

        print(unique_education_field)
```

```
['Marketing' 'Computer Science' 'Technical Degree' 'Information Systems'
 'Other' 'Economics' 'Human Resources' 'Business Studies']
```

Summary

- This preprocessing step is crucial for maintaining data integrity, especially when performing further analysis or when categorizing employees based on their education fields.

3. **Calculating Attrition Year**

- **Objective**: To calculate the year in which an employee is expected to leave the company based on their hire date and the number of years they have been with the company.

```
In [10]: # Add a new column 'AttritionYear' by adding the year of the 'HireDate' to 'YearsAtCompany'
         employee_df['HireYear'] = pd.to_datetime(employee_df['HireDate']).dt.year
         employee_df['AttritionYear'] = employee_df['HireYear'] + employee_df['YearsAtCompany']
```

4. **Dropping Outdated Performance Ratings**

- **Objective**: To ensure that performance ratings are relevant by removing any ratings that fall outside the expected date range based on the calculated 'AttritionYear'.
- **Note**: If applicable, you can add code here for how you dropped the outdated performance ratings.

Summary

- This step is essential for accurately forecasting employee attrition and ensuring that performance evaluations are based on relevant time frames

5. **Converting Review Date to Datetime**

- **Objective**: To convert the 'ReviewDate' column in the performance rating dataframe to a datetime format, facilitating date comparisons.

6. **Merging Employee and Performance Data**

- **Objective**: To merge the employee dataframe with the performance dataframe, incorporating hire and attrition years for each employee

7. **Filtering Performance Reviews by Date Range**

- **Objective**: To filter out performance reviews that fall outside the employee's hire and attrition year range, ensuring data relevance.

8. **Dropping Unnecessary Columns**

- **Objective**: To remove extra columns ('HireYear' and 'AttritionYear') from the filtered performance dataframe, simplifying the dataset.

In [11]:
```
# Convert 'ReviewDate' in performance rating dataframe to datetime for comparison
performance_df['ReviewDate'] = pd.to_datetime(performance_df['ReviewDate'])

# Merge the employee_df with performance_df to get the hire and attrition year for each employee
performance_merged = pd.merge(performance_df, employee_df[['EmployeeID', 'HireYear', 'AttritionYear']], on='EmployeeID', how='le

# Filter out rows where the review date is out of the employee's hire and attrition year range
performance_df_filtered = performance_merged[
    (performance_merged['ReviewDate'].dt.year >= performance_merged['HireYear']) &
    (performance_merged['ReviewDate'].dt.year <= performance_merged['AttritionYear'])
]

# Drop the extra columns ('HireYear' and 'AttritionYear') from the filtered dataframe
performance_df_filtered = performance_df_filtered.drop(columns=['HireYear', 'AttritionYear'])

# Assign the filtered dataframe back to performance_df
performance_df = performance_df_filtered
```

Output:

```
In [12]: print("\nUnique Values in Performance Rating Dataset before filtering:")
         print(unique_values_performance)
         unique_values_performance = performance_df.nunique()
         print("\nUnique Values in Performance Rating Dataset after filtering:")
         print(unique_values_performance)
```

```
Unique Values in Performance Rating Dataset before filtering:
PerformanceID                     6709
EmployeeID                        1280
ReviewDate                        2771
EnvironmentSatisfaction              5
JobSatisfaction                      5
RelationshipSatisfaction             5
TrainingOpportunitiesWithinYear      3
TrainingOpportunitiesTaken           4
WorkLifeBalance                      5
SelfRating                           3
ManagerRating                        4
dtype: int64

Unique Values in Performance Rating Dataset after filtering:
PerformanceID                     5136
EmployeeID                        1235
ReviewDate                        2403
EnvironmentSatisfaction              5
JobSatisfaction                      5
RelationshipSatisfaction             5
TrainingOpportunitiesWithinYear      3
```

ii. SQL Server:

Connecting to SQL Server and Uploading Data

Establishing a Connection to SQL Server

- **Objective**: To create a connection to the SQL Server database for storing and analyzing HR data.

```
In [15]: # Define connection string
         conn_str = (
             r'DRIVER={ODBC Driver 17 for SQL Server};'
             r'SERVER=HADY;'
             r'DATABASE=HR final Project;'
             r'Trusted_Connection=yes;'
         )

         # Create a connection using SQLAlchemy
         engine = create_engine(f"mssql+pyodbc:///?odbc_connect={conn_str}")
```

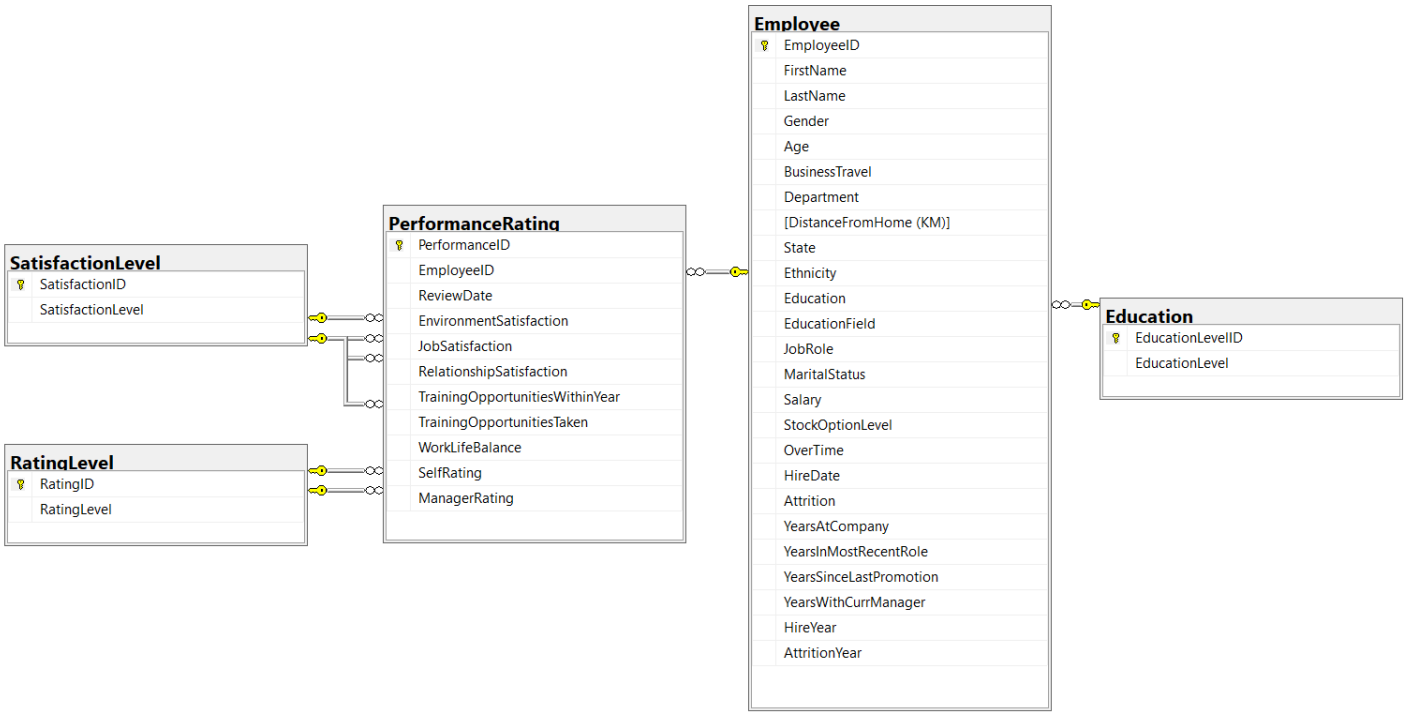
Uploading DataFrames to SQL Server

- **Objective**: To upload the processed DataFrames to SQL Server tables for further analysis and insights.

```
In [16]: # Upload DataFrames to SQL Server
         education_df.to_sql('Education', con=engine, if_exists='replace', index=False)
         employee_df.to_sql('Employee', con=engine, if_exists='replace', index=False)
         performance_df.to_sql('PerformanceRating', con=engine, if_exists='replace', index=False)
         rating_df.to_sql('RatingLevel', con=engine, if_exists='replace', index=False)
         satisfaction_df.to_sql('SatisfactionLevel', con=engine, if_exists='replace', index=False)
```

Summary

- This final step establishes a connection to the SQL Server and uploads the relevant Data Frames, ensuring the data is available for further analysis. Each Data Frame is uploaded to its corresponding table, with existing data replaced to maintain consistency.



Extracting Insights Using SQL Queries

After successfully uploading the data, we can start executing SQL queries within the notebook to extract insights from the data. The SQL queries can now be run directly against the SQL Server database, taking advantage of SQL's powerful querying capabilities to generate insights for further analysis.

This integration provides the flexibility to use both Python and SQL in a seamless workflow, enhancing data analysis efficiency.

Salary Insights

In this section, we use SQL queries to extract insights related to employee salaries. These insights include average salary by department and job role, salary breakdown by age group and gender, and an analysis of the correlation between salary and performance ratings. The results of these queries are displayed in a scrollable format for better readability.

1. Average Salary by Department and Job Role

We begin by calculating the average salary across all departments and job roles. The query computes the average salary for each department as well as the breakdown by individual job roles within each department.

- **SQL Query**:
  - First, we calculate the average salary for each department.
  - Then, we compute the average salary for each combination of department and job role.
  - Results are ordered by department and job role for easier comparison.

```
query = """
SELECT
    Department,
    'ALL' AS JobRole,
    AVG(Salary) AS AvgSalary
FROM Employee
GROUP BY Department

UNION ALL

SELECT
    Department,
    JobRole,
    AVG(Salary) AS AvgSalary
FROM Employee
GROUP BY JobRole, Department
ORDER BY Department,JobRole;

"""
```

Department	JobRole	AvgSalary
Human Resources	ALL	119698
Human Resources	HR Business Partner	314002
Human Resources	HR Executive	94362
Human Resources	HR Manager	449330
Human Resources	Recruiter	37647
Sales	ALL	119117
Sales	Manager	317531
Sales	Sales Executive	116574
Sales	Sales Representative	40656
Technology	ALL	109655
Technology	Analytics Manager	346484
Technology	Data Scientist	56079

2. Salary and Experience by Age Group and Gender

Next, we investigate how salary, along with various experience-related metrics, differs by age group and gender for employees who have not left the company (i.e., **Attrition = 'No'**). The query provides average years at the company, years in the most recent role, years since last promotion, years with the current manager, and average salary for each age group and gender.

- **SQL Query**:
  - The data is grouped by age and gender.
  - The analysis includes employees who have not left the company (Attrition = 'No').

```
query = """
SELECT
    age AS AgeGroup,
    gender AS Gender,
    AVG(YearsAtCompany) AS AVGYearsAtCompany,
    AVG(YearsInMostRecentRole) AS AVGYearsInMostRecentRole,
    AVG(YearsSinceLastPromotion) AS AVGYearsSinceLastPromotion,
    AVG(YearsWithCurrManager) AS AVGYearsWithCurrManager,
    COUNT(*) AS EmployeeCount,
    AVG(Salary) AS AVGSalary
FROM Employee
WHERE Attrition = 'No'
GROUP BY age, gender
ORDER BY age ASC, gender ASC;
"""
```

AgeGroup	Gender	AVGYearsAtCompany	AVGYearsInMostRecentRole	AVGYearsSinceLastPromotion	AVGYearsWithCurrManager	EmployeeCount	AVGSalary
18	Female	0	0	0	0	10	42111
18	Male	0	0	0	0	9	47324
18	Non-Binary	0	0	0	0	2	67238
18	Prefer Not To Say	0	0	0	0	1	74515
19	Female	0	0	0	0	21	40017
19	Male	0	0	0	0	19	50056
19	Non-Binary	0	0	0	0	6	50444
19	Prefer Not To Say	0	0	0	0	1	39722
20	Female	0	0	0	0	13	36741
20	Male	1	0	0	0	17	29959
20	Non-Binary	0	0	0	0	2	38931

3. Average Salary by Department, Job Role, Education Level, and Business Travel

We also analyze how salary varies across different departments, job roles, education levels, and business travel frequency for employees who are still active (Attrition = 'No'). This query helps us understand the impact of education and travel requirements on compensation within each role and department.

- **SQL Query**:
  - The data is grouped by department, job role, education level, and business travel frequency.
  - Employees with **Attrition = 'No'** are included in the analysis.

```
query = """
SELECT
    e.Department,
    e.JobRole,
    ed.EducationLevel,
    e.BusinessTravel,
    COUNT(*) AS EmployeeCount,
    AVG(salary) AS AVGEmployeeSalary
FROM Employee e
JOIN Education ed ON e.Education = ed.EducationLevelID
WHERE e.Attrition = 'No'
GROUP BY  e.BusinessTravel, e.Department, e.JobRole, ed.EducationLevel
ORDER BY e.JobRole ,ed.EducationLevel, AVGEmployeeSalary ASC;
"""
```

Department	JobRole	EducationLevel	BusinessTravel	EmployeeCount	AVGEmployeeSalary
Technology	Analytics Manager	Bachelors	Some Travel	15	302810
Technology	Analytics Manager	Bachelors	Frequent Traveller	5	438020
Technology	Analytics Manager	Bachelors	No Travel	1	542695
Technology	Analytics Manager	Doctorate	Frequent Traveller	1	362540
Technology	Analytics Manager	Doctorate	Some Travel	1	482510
Technology	Analytics Manager	High School	Some Travel	8	311921
Technology	Analytics Manager	High School	Frequent Traveller	1	546549
Technology	Analytics Manager	Masters	Frequent Traveller	2	268578
Technology	Analytics Manager	Masters	Some Travel	8	331242
Technology	Analytics Manager	Masters	No Travel	2	368587
Technology	Analytics Manager	No Formal Qualifications	Some Travel	2	272668
Technology	Analytics Manager	No Formal Qualifications	No Travel	2	373197

4. Correlation Between Salary and Manager Rating

Finally, we examine whether there is any correlation between salary and manager rating. This query retrieves active employees' average manager ratings, their job roles, and salaries, and orders the results by job role and salary to see if higher-performing employees tend to have higher salaries.

- **SQL Query**:
  - We calculate the average manager rating for each employee.
  - The results are grouped by job role and ordered by salary in descending order to analyze potential correlations.

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.ManagerRating) AS AVGManagerRating
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    CONCAT(FirstName, ' ', LastName) AS FullName,
    e.JobRole,
    ea.AVGManagerRating,
    e.Salary
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.FirstName, e.LastName, e.JobRole, e.Salary, ea.AVGManagerRating
ORDER BY e.JobRole, e.Salary DESC;

"""
```

FullName	JobRole	AVGManagerRating	Salary
Norbie Mosdill	Analytics Manager	4	547204
Roda Costin	Analytics Manager	3	546549
Leonidas Clarke-Williams	Analytics Manager	3	542695
Angelia Letrange	Analytics Manager	3	517695
Stearne Axelbee	Analytics Manager	3	513608
Adelina Bittlestone	Analytics Manager	2	513325
Jamill Woolger	Analytics Manager	5	495977
Lebbie Poure	Analytics Manager	3	482510
Niko Purvess	Analytics Manager	3	465219
Kaleena Ellsbury	Analytics Manager	4	462687
Clywd Burril	Analytics Manager	3	456418
Dulci Berthot	Analytics Manager	3	454993

Employee Satisfaction Insights

In this section, we utilize SQL queries to analyze employee satisfaction across various dimensions such as department, job role, age group, gender, and ethnicity. The goal is to identify trends in employee satisfaction and uncover areas where improvement may be needed. We also explore the relationship between employee satisfaction and performance ratings.

1. High Satisfaction by Department and Job Role (Average >= 3)

This query groups employees by department and job role, displaying those with average satisfaction scores (environment, job, relationship, and work-life balance) greater than or equal to 3. Employees with high satisfaction are important to identify for retaining top talent.

- **Metrics Calculated**:
  - Average Environment Satisfaction
  - Average Job Satisfaction
  - Average Relationship Satisfaction
  - Average Work-Life Balance
  - Employee Count per group

```sql
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.EnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
        AVG(p.JobSatisfaction) AS AVGJobSatisfaction,
        AVG(p.RelationshipSatisfaction) AS AVGRelationshipSatisfaction,
        AVG(p.WorkLifeBalance) AS AVGWorkLifeBalance
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Department,
    e.JobRole,
    AVG(ea.AVGEnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
    AVG(ea.AVGJobSatisfaction) AS AVGJobSatisfaction,
    AVG(ea.AVGRelationshipSatisfaction) AS AVGRelationshipSatisfaction,
    AVG(ea.AVGWorkLifeBalance) AS AVGWorkLifeBalance,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition ='No'
GROUP BY e.Department, e.JobRole
HAVING AVG(ea.AVGEnvironmentSatisfaction) >= 3
    AND AVG(ea.AVGJobSatisfaction) >= 3
    AND AVG(ea.AVGRelationshipSatisfaction) >= 3
    AND AVG(ea.AVGWorkLifeBalance) >= 3
ORDER BY e.Department, e.JobRole;
"""
```

Department	JobRole	AVGEnvironmentSatisfaction	AVGJobSatisfaction	AVGRelationshipSatisfaction	AVGWorkLifeBalance	EmployeesCount
Human Resources	HR Manager	3	3	3	3	3
Sales	Sales Executive	3	3	3	3	213
Sales	Sales Representative	3	3	3	3	35
Technology	Analytics Manager	3	3	3	3	43
Technology	Data Scientist	3	3	3	3	165
Technology	Engineering Manager	3	3	3	3	66
Technology	Machine Learning Engineer	3	3	3	3	106
Technology	Sales Executive	3	5	4	5	1
Technology	Senior Software Engineer	3	3	3	3	102
Technology	Software Engineer	3	3	3	3	197

2. Low Satisfaction by Department and Job Role (Average < 3)

This query is similar to the one above, but focuses on employees whose satisfaction scores are below 3. Identifying areas with low satisfaction helps pinpoint departments and roles where interventions may be necessary to improve employee engagement and retention.

- **Metrics Calculated**:

- Average Environment Satisfaction
- Average Job Satisfaction
- Average Relationship Satisfaction
- Average Work-Life Balance
- Employee Count per group

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.EnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
        AVG(p.JobSatisfaction) AS AVGJobSatisfaction,
        AVG(p.RelationshipSatisfaction) AS AVGRelationshipSatisfaction,
        AVG(p.WorkLifeBalance) AS AVGWorkLifeBalance
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Department,
    e.JobRole,
    AVG(ea.AVGEnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
    AVG(ea.AVGJobSatisfaction) AS AVGJobSatisfaction,
    AVG(ea.AVGRelationshipSatisfaction) AS AVGRelationshipSatisfaction,
    AVG(ea.AVGWorkLifeBalance) AS AVGWorkLifeBalance,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition ='No'
GROUP BY e.Department, e.JobRole
HAVING AVG(ea.AVGEnvironmentSatisfaction) < 3
    OR AVG(ea.AVGJobSatisfaction) < 3
    OR AVG(ea.AVGRelationshipSatisfaction) < 3
    OR AVG(ea.AVGWorkLifeBalance) < 3
ORDER BY e.Department, e.JobRole;
"""
```

Department	JobRole	AVGEnvironmentSatisfaction	AVGJobSatisfaction	AVGRelationshipSatisfaction	AVGWorkLifeBalance	EmployeesCount
Human Resources	HR Business Partner	3	2	3	3	5
Human Resources	HR Executive	3	2	2	2	21
Human Resources	Recruiter	3	3	2	3	12
Sales	Manager	3	3	2	3	29

3. High Satisfaction by Age Group, Gender, and Ethnicity (Average >= 3)

In this query, employees are grouped by age, gender, and ethnicity, and the average satisfaction scores are calculated for each group. This helps reveal any demographic patterns in employee satisfaction, ensuring that all groups are receiving a positive work experience.

- **Metrics Calculated**:

- Average Environment Satisfaction
- Average Job Satisfaction
- Average Relationship Satisfaction
- Average Work-Life Balance
- Employee Count per group

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.EnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
        AVG(p.JobSatisfaction) AS AVGJobSatisfaction,
        AVG(p.RelationshipSatisfaction) AS AVGRelationshipSatisfaction,
        AVG(p.WorkLifeBalance) AS AVGWorkLifeBalance
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Age AS AgeGroup,
    e.Gender,
    e.Ethnicity,
    AVG(ea.AVGEnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
    AVG(ea.AVGJobSatisfaction) AS AVGJobSatisfaction,
    AVG(ea.AVGRelationshipSatisfaction) AS AVGRelationshipSatisfaction,
    AVG(ea.AVGWorkLifeBalance) AS AVGWorkLifeBalance,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.Age, e.Gender, e.Ethnicity
HAVING AVG(ea.AVGEnvironmentSatisfaction) >= 3
    AND AVG(ea.AVGJobSatisfaction) >= 3
    AND AVG(ea.AVGRelationshipSatisfaction) >= 3
    AND AVG(ea.AVGWorkLifeBalance) >= 3
ORDER BY e.Age , e.Gender, e.Ethnicity;
"""
```

AgeGroup	Gender	Ethnicity	AVGEnvironmentSatisfaction	AVGJobSatisfaction	AVGRelationshipSatisfaction	AVGWorkLifeBalance	EmployeesCount
20	Female	White	3	4	3	3	4
20	Male	White	3	3	3	4	7
21	Male	Asian or Asian American	4	4	4	3	1
21	Male	White	3	3	3	3	14
22	Female	American Indian or Alaska Native	3	4	3	4	1
22	Female	White	3	3	3	3	16
22	Male	American Indian or Alaska Native	4	4	3	3	1
22	Male	Asian or Asian American	4	3	4	4	1
22	Male	White	3	3	3	3	11
		Black or African					

4. Low Satisfaction by Age Group, Gender, and Ethnicity (Average < 3)

This query identifies age, gender, and ethnicity groups with low satisfaction scores. It helps in uncovering demographic groups that may require attention to improve their work experience and engagement levels.

- **Metrics Calculated**:
  - Average Environment Satisfaction
  - Average Job Satisfaction
  - Average Relationship Satisfaction
  - Average Work-Life Balance
  - Employee Count per group

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.EnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
        AVG(p.JobSatisfaction) AS AVGJobSatisfaction,
        AVG(p.RelationshipSatisfaction) AS AVGRelationshipSatisfaction,
        AVG(p.WorkLifeBalance) AS AVGWorkLifeBalance
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Age AS AgeGroup,
    e.Gender,
    e.Ethnicity,
    AVG(ea.AVGEnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
    AVG(ea.AVGJobSatisfaction) AS AVGJobSatisfaction,
    AVG(ea.AVGRelationshipSatisfaction) AS AVGRelationshipSatisfaction,
    AVG(ea.AVGWorkLifeBalance) AS AVGWorkLifeBalance,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.Age, e.Gender, e.Ethnicity
HAVING AVG(ea.AVGEnvironmentSatisfaction) < 3
    OR AVG(ea.AVGJobSatisfaction) < 3
    OR AVG(ea.AVGRelationshipSatisfaction) < 3
    OR AVG(ea.AVGWorkLifeBalance) < 3
ORDER BY e.Age, e.Gender, e.Ethnicity;
"""
```

AgeGroup	Gender	Ethnicity	AVGEnvironmentSatisfaction	AVGJobSatisfaction	AVGRelationshipSatisfaction	AVGWorkLifeBalance	EmployeesCount
19	Female	White	4	2	3	3	6
19	Male	White	3	2	4	2	2
19	Non-Binary	White	5	5	5	2	1
21	Female	Black or African American	3	3	4	2	2
21	Female	White	3	3	3	2	12
21	Male	American Indian or Alaska Native	4	3	2	3	2
21	Male	Black or African American	4	2	3	3	3
21	Non-Binary	Asian or Asian American	3	3	3	2	1
22	Female	Asian or Asian American	3	2	3	3	2

5. Highly Rated Employees with Low Satisfaction

This query focuses on employees who receive high performance ratings from their managers (average rating >= 4), but report low satisfaction in terms of environment, job, relationships, or work-life balance (average satisfaction < 3 in any category). These employees are critical to identify, as they may be at risk of attrition despite being high performers.

- **Metrics Calculated**:
  - Average Environment Satisfaction
  - Average Job Satisfaction
  - Average Relationship Satisfaction
  - Average Work-Life Balance
  - Average Manager Rating

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.EnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
        AVG(p.JobSatisfaction) AS AVGJobSatisfaction,
        AVG(p.RelationshipSatisfaction) AS AVGRelationshipSatisfaction,
        AVG(p.WorkLifeBalance) AS AVGWorkLifeBalance,
        AVG(p.ManagerRating) AS AVGManagerRating
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    CONCAT(FirstName, ' ', LastName) AS FullName,
    e.JobRole,
    AVG(ea.AVGEnvironmentSatisfaction) AS AVGEnvironmentSatisfaction,
    AVG(ea.AVGJobSatisfaction) AS AVGJobSatisfaction,
    AVG(ea.AVGRelationshipSatisfaction) AS AVGRelationshipSatisfaction,
    AVG(ea.AVGWorkLifeBalance) AS AVGWorkLifeBalance,
    AVG(ea.AVGManagerRating) AS AVGManagerRating
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.FirstName,e.LastName,e.JobRole
HAVING AVG(ea.AVGManagerRating)>=4
    AND(AVG(ea.AVGEnvironmentSatisfaction) < 3
    OR AVG(ea.AVGJobSatisfaction) < 3
    OR AVG(ea.AVGRelationshipSatisfaction) < 3
    OR AVG(ea.AVGWorkLifeBalance) < 3)
ORDER BY JobRole


"""
```

FullName	JobRole	AVGEnvironmentSatisfaction	AVGJobSatisfaction	AVGRelationshipSatisfaction	AVGWorkLifeBalance	AVGManagerRating
Carmita Kenion	Analytics Manager	4	2	3	4	4
Josh Tomczykowski	Analytics Manager	3	3	3	2	4
Kaleena Ellsbury	Analytics Manager	2	1	2	2	4
Norbie Mosdill	Analytics Manager	4	4	2	3	4
Ashlee Mc Combe	Data Scientist	3	3	2	2	4
Dehlia Fullerton	Data Scientist	3	4	2	2	5
Donavon Hallet	Data Scientist	3	1	2	2	4
Franz Brighouse	Data Scientist	3	4	2	3	4
Jonah Chatwood	Data Scientist	4	3	3	2	4
Marlin Upcraft	Data Scientist	1	1	4	3	4
Oswell McCloid	Data Scientist	4	2	3	4	4
Othelia Tumber	Data Scientist	4	2	2	4	4

Employee Rating Insights

In this section, we use SQL queries to gain insights into employee ratings, which include both **self-ratings** and **manager ratings**. The goal is to understand how employee ratings vary across different departments, job roles, age groups, genders, and ethnicities. We also identify areas where employee ratings may be below expectations and need attention.

1. High Ratings by Department and Job Role (Average >= 3)

This query groups employees by department and job role and calculates the average self-rating and manager rating for each group. Only those groups with average ratings greater than or equal to 3 are displayed. This helps identify departments and roles with employees who consistently rate their own performance positively and receive high ratings from their managers.

- **Metrics Calculated**:
  - Average Self Rating
  - Average Manager Rating
  - Employee Count per group

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.SelfRating) AS AVGSelfRating,
        AVG(p.ManagerRating) AS AVGManagerRating
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Department,
    e.JobRole,
    AVG(ea.AVGSelfRating) AS AVGSelfRating,
    AVG(ea.AVGManagerRating) AS AVGManagerRating,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition ='No'
GROUP BY e.Department, e.JobRole
HAVING AVG(ea.AVGSelfRating) >= 3
    AND AVG(ea.AVGManagerRating) >= 3
ORDER BY e.Department, e.JobRole;
"""
```

Department	JobRole	AVGSelfRating	AVGManagerRating	EmployeesCount
Human Resources	HR Executive	3	3	21
Human Resources	Recruiter	4	3	12
Sales	Manager	3	3	29
Sales	Sales Executive	3	3	213
Sales	Sales Representative	3	3	35
Technology	Analytics Manager	3	3	43
Technology	Data Scientist	3	3	165
Technology	Engineering Manager	3	3	66
Technology	Machine Learning Engineer	3	3	106
Technology	Sales Executive	4	4	1
Technology	Senior Software Engineer	3	3	102
Technology	Software Engineer	3	3	197

2. Low Ratings by Department and Job Role (Average < 3)

This query focuses on employees whose average self-rating or manager rating is below 3. Identifying areas with low ratings helps pinpoint where performance may need improvement, or where additional training and support could be provided.

- **Metrics Calculated**:
  - Average Self Rating
  - Average Manager Rating
  - Employee Count per group

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.SelfRating) AS AVGSelfRating,
        AVG(p.ManagerRating) AS AVGManagerRating
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Department,
    e.JobRole,
    AVG(ea.AVGSelfRating) AS AVGSelfRating,
    AVG(ea.AVGManagerRating) AS AVGManagerRating,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition ='No'
GROUP BY e.Department, e.JobRole
HAVING AVG(ea.AVGSelfRating) < 3
    OR AVG(ea.AVGManagerRating) < 3
ORDER BY e.Department, e.JobRole;
"""
```

Department	JobRole	AVGSelfRating	AVGManagerRating	EmployeesCount
Human Resources	HR Business Partner	3	2	5
Human Resources	HR Manager	3	2	3

3. High Ratings by Age Group, Gender, and Ethnicity (Average >= 3)

In this query, employees are grouped by age, gender, and ethnicity, and the average self-rating and manager rating are calculated for each demographic group. This helps reveal trends in employee performance ratings across different demographics and ensures that there is no bias in performance evaluations.

- **Metrics Calculated**:
  - Average Self Rating
  - Average Manager Rating
  - Employee Count per group

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.SelfRating) AS AVGSelfRating,
        AVG(p.ManagerRating) AS AVGManagerRating
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Age AS AgeGroup,
    e.Gender,
    e.Ethnicity,
    AVG(ea.AVGSelfRating) AS AVGSelfRating,
    AVG(ea.AVGManagerRating) AS AVGManagerRating,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.Age , e.Gender, e.Ethnicity
HAVING AVG(ea.AVGSelfRating) >= 3
    AND AVG(ea.AVGManagerRating) >= 3
ORDER BY e.Age, e.Gender, e.Ethnicity;
"""
```

AgeGroup	Gender	Ethnicity	AVGSelfRating	AVGManagerRating	EmployeesCount
19	Female	White	4	4	6
19	Male	White	4	4	2
19	Non-Binary	White	5	4	1
20	Male	White	3	3	7
21	Female	Black or African American	4	3	2
21	Female	White	4	3	12
21	Male	American Indian or Alaska Native	4	3	2
21	Male	Black or African American	3	3	3
21	Male	White	4	3	14
22	Female	American Indian or Alaska Native	4	3	1
22	Female	Asian or Asian American	4	3	2
22	Female	Black or African American	4	3	3

4. Low Ratings by Age Group, Gender, and Ethnicity (Average < 3)

This query identifies age, gender, and ethnicity groups with low self or manager ratings. This helps in uncovering demographic groups that may require more attention, training, or support to improve their performance.

- **Metrics Calculated**:
  - Average Self Rating
  - Average Manager Rating
  - Employee Count per group

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.SelfRating) AS AVGSelfRating,
        AVG(p.ManagerRating) AS AVGManagerRating
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Age AS AgeGroup,
    e.Gender,
    e.Ethnicity,
    AVG(ea.AVGSelfRating) AS AVGSelfRating,
    AVG(ea.AVGManagerRating) AS AVGManagerRating,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.Age, e.Gender, e.Ethnicity
HAVING AVG(ea.AVGSelfRating) < 3
    OR AVG(ea.AVGManagerRating) < 3
ORDER BY e.Age, e.Gender, e.Ethnicity;
"""
```

AgeGroup	Gender	Ethnicity	AVGSelfRating	AVGManagerRating	EmployeesCount
20	Female	White	3	2	4
21	Male	Asian or Asian American	3	2	1
21	Non-Binary	Asian or Asian American	3	2	1
22	Non-Binary	White	3	2	4
23	Male	Black or African American	3	2	2
23	Non-Binary	White	3	2	4
24	Female	Native Hawaiian	3	2	2
24	Male	White	3	2	20
24	Non-Binary	Black or African American	3	2	4
24	Non-Binary	Mixed or multiple ethnic groups	3	2	2
25	Female	Mixed or multiple ethnic groups	3	2	14
25	Female	White	3	2	17

Experience, Training, and Attrition Analysis

This section covers analysis of employee experience, training opportunities, and attrition rates. Through SQL queries, we explore the relationships between experience and performance, identify employees in long-standing roles, assess training opportunities, and calculate attrition rates by department and job role.

1. Correlation Between Experience and Performance

This query explores the relationship between the number of years employees have been with the company (and in their most recent role) and their average manager ratings. It helps identify whether more experienced employees tend to perform better or if there is any observable correlation.

- **Metrics Calculated**:
  - Job Role
  - Years at Company
  - Years in Most Recent Role
  - Average Manager Rating

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.ManagerRating) AS AVGManagerRating
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    CONCAT(FirstName, ' ', LastName) AS FullName,
    e.JobRole,
    e.YearsAtCompany,
    e.YearsInMostRecentRole,
    ea.AVGManagerRating
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.FirstName, e.LastName, e.JobRole, e.YearsAtCompany, e.YearsInMostRecentRole, ea.AVGManagerRating
ORDER BY e.JobRole , ea.AVGManagerRating DESC;
"""
```

FullName	JobRole	YearsAtCompany	YearsInMostRecentRole	AVGManagerRating
Jamill Woolger	Analytics Manager	2	2	5
Carmita Kenion	Analytics Manager	6	4	4
Josh Tomczykowski	Analytics Manager	3	1	4
Kaleena Ellsbury	Analytics Manager	2	2	4
Kali Jeppe	Analytics Manager	6	1	4
Norbie Mosdill	Analytics Manager	8	7	4
Otha Lehrmann	Analytics Manager	2	0	4
Amelia Izard	Analytics Manager	4	2	3
Angelia Letrange	Analytics Manager	3	2	3
April Carstairs	Analytics Manager	6	3	3
Bell Di Biasi	Analytics Manager	7	1	3
Clywd Burril	Analytics Manager	2	0	3

2. Employees in the Same Role for Extended Periods

This query identifies employees who have stayed in the same job role for more than four years. This insight is valuable for understanding whether employees who remain in one role for a long time are being considered for promotions or other growth opportunities.

- **Metrics Calculated**:
  - Full Name
  - Job Role
  - Years in Most Recent Role

```
query = """
SELECT CONCAT(FirstName, ' ', LastName) AS FullName, JobRole, YearsInMostRecentRole
FROM Employee
WHERE YearsInMostRecentRole >= 4 AND Attrition = 'No'
ORDER BY YearsInMostRecentRole DESC;
"""
```

FullName	JobRole	YearsInMostRecentRole
Eleanora Thornbarrow	Senior Software Engineer	10
Roderic Daddow	Sales Representative	10
Rolf Cunah	Software Engineer	10
Ermentrude Berrie	Engineering Manager	10
Forbes Toretta	Senior Software Engineer	10
Karlen Gulston	Data Scientist	10
Ignacius Streeter	Machine Learning Engineer	10
Gifford Poynser	Machine Learning Engineer	10
Nichols Baty	Senior Software Engineer	10
Munmro Ledamun	Machine Learning Engineer	10
Elvira Ianelli	Recruiter	10
Hagen Worge	Sales Executive	10

3. Training Opportunities for Top Performers

Here, we examine how many training opportunities have been taken by top-rated employees (those with a manager rating of 4 or higher). Understanding whether top performers are actively engaging in training can provide insights into employee development efforts.

- **Metrics Calculated**:
  - Full Name
  - Job Role
  - Number of Training Opportunities Taken

```
query = """
SELECT
    CONCAT(FirstName, ' ', LastName) AS FullName,
    JobRole,
    COUNT(p.TrainingOpportunitiesTaken) AS TrainingOpportunitiesForTopPerformers
FROM Employee AS e
JOIN PerformanceRating AS p ON e.EmployeeID = p.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.EmployeeID, e.FirstName, e.LastName,JobRole
HAVING AVG(p.ManagerRating) >= 4
ORDER BY TrainingOpportunitiesForTopPerformers DESC;

"""
```

FullName	JobRole	TrainingOpportunitiesForTopPerformers
Uta Melmar	Engineering Manager	9
Burnaby Guillet	Software Engineer	8
Camila Hernik	Sales Executive	8
Jayne Mickleburgh	HR Executive	8
Jaclin Chadburn	Software Engineer	8
Munmro Ledamun	Machine Learning Engineer	8
Curcio Franek	Recruiter	8
Jessa Glasscoo	Sales Representative	8
Gustavo Chatel	Sales Executive	8
Zondra Duigenan	Senior Software Engineer	8
Cassandre Joubert	Senior Software Engineer	8
Karina Weddup	Software Engineer	8

4. Training Opportunities and Training Taken by Department and Job Role

This query provides an overview of the average number of training opportunities offered and the number of training opportunities taken by employees, grouped by department and job role. This helps assess the effectiveness of training programs and identify departments or roles with lower training engagement.

- **Metrics Calculated**:
  - Department
  - Job Role
  - Average Training Opportunities Within Year
  - Average Training Opportunities Taken
  - Employee Count

```
query = """
WITH EmployeeAverages AS (
    SELECT
        p.EmployeeID,
        AVG(p.TrainingOpportunitiesWithinYear) AS AVGTrainingOpportunitiesWithinYear,
        AVG(p.TrainingOpportunitiesTaken) AS AVGTrainingOpportunitiesTaken
    FROM PerformanceRating p
    GROUP BY p.EmployeeID
)
SELECT
    e.Department,
    e.JobRole,
    AVG(ea.AVGTrainingOpportunitiesWithinYear) AS AVGTrainingOpportunitiesWithinYear,
    AVG(ea.AVGTrainingOpportunitiesTaken) AS AVGTrainingOpportunitiesTaken,
    COUNT(e.EmployeeID) AS EmployeesCount
FROM EmployeeAverages ea
JOIN Employee e ON ea.EmployeeID = e.EmployeeID
WHERE e.Attrition = 'No'
GROUP BY e.Department, e.JobRole
ORDER BY e.Department , e.JobRole;
"""
```

Department	JobRole	AVGTrainingOpportunitiesWithinYear	AVGTrainingOpportunitiesTaken	EmployeesCount
Human Resources	HR Business Partner	1	0	5
Human Resources	HR Executive	1	0	21
Human Resources	HR Manager	1	0	3
Human Resources	Recruiter	1	0	12
Sales	Manager	1	0	29
Sales	Sales Executive	1	0	213
Sales	Sales Representative	1	0	35
Technology	Analytics Manager	1	0	43
Technology	Data Scientist	1	0	165
Technology	Engineering Manager	1	0	66
Technology	Machine Learning Engineer	1	0	106
Technology	Sales Executive	1	3	1

5. Attrition Rate by Department and Job Role

This query calculates the attrition rate by department and job role. The attrition rate is the percentage of employees who have left the company out of the total number of employees in each department and role. This insight helps identify high-turnover areas, which may require further investigation.

- **Metrics Calculated**:
  - Department
  - Job Role
  - Total Employees
  - Employees Who Left
  - Current Employees

- Attrition Rate (percentage)

```
query = """
SELECT
    E.Department,
    E.JobRole,
    COUNT(*) AS TotalEmployees,
    SUM(CASE WHEN E.Attrition = 'Yes' THEN 1 ELSE 0 END) AS EmployeesLeft,
    SUM(CASE WHEN E.Attrition = 'No' THEN 1 ELSE 0 END) AS CurrentEmployees,
    CAST(ROUND((SUM(CASE WHEN E.Attrition = 'Yes' THEN 1 ELSE 0 END) * 100.0 / COUNT(*)), 2) AS DECIMAL(10,2)) AS AttritionRate
FROM Employee E
GROUP BY E.Department, E.JobRole
ORDER BY AttritionRate DESC;


"""
```

Department	JobRole	TotalEmployees	EmployeesLeft	CurrentEmployees	AttritionRate
Sales	Sales Representative	83	33	50	39.76
Human Resources	Recruiter	24	9	15	37.50
Technology	Data Scientist	261	62	199	23.75
Sales	Sales Executive	326	57	269	17.48
Technology	Software Engineer	294	47	247	15.99
Human Resources	HR Executive	28	3	25	10.71
Technology	Machine Learning Engineer	146	10	136	6.85
Technology	Senior Software Engineer	132	9	123	6.82
Technology	Analytics Manager	52	3	49	5.77
Sales	Manager	37	2	35	5.41
Technology	Engineering Manager	75	2	73	2.67
Human Resources	HR Business Partner	7	0	7	0.00

Displaying Results

For each query, the results are loaded into a Pandas DataFrame and displayed in a scrollable format, allowing for easier inspection of large datasets within the notebook environment.

```
# Execute the query and load the result into a Pandas DataFrame
df_complex_query = pd.read_sql_query(query, engine)

# Display a scrollable dataframe
def display_scrollable_dataframe(df, max_height=400):
    display(HTML(f'''
        <div style="height:{max_height}px; overflow:auto; border:1px solid lightgray;">
            {df.to_html(index=False)}
        </div>
    '''))

# Display the the result
display_scrollable_dataframe(df_complex_query)
```

iii. Power BI:

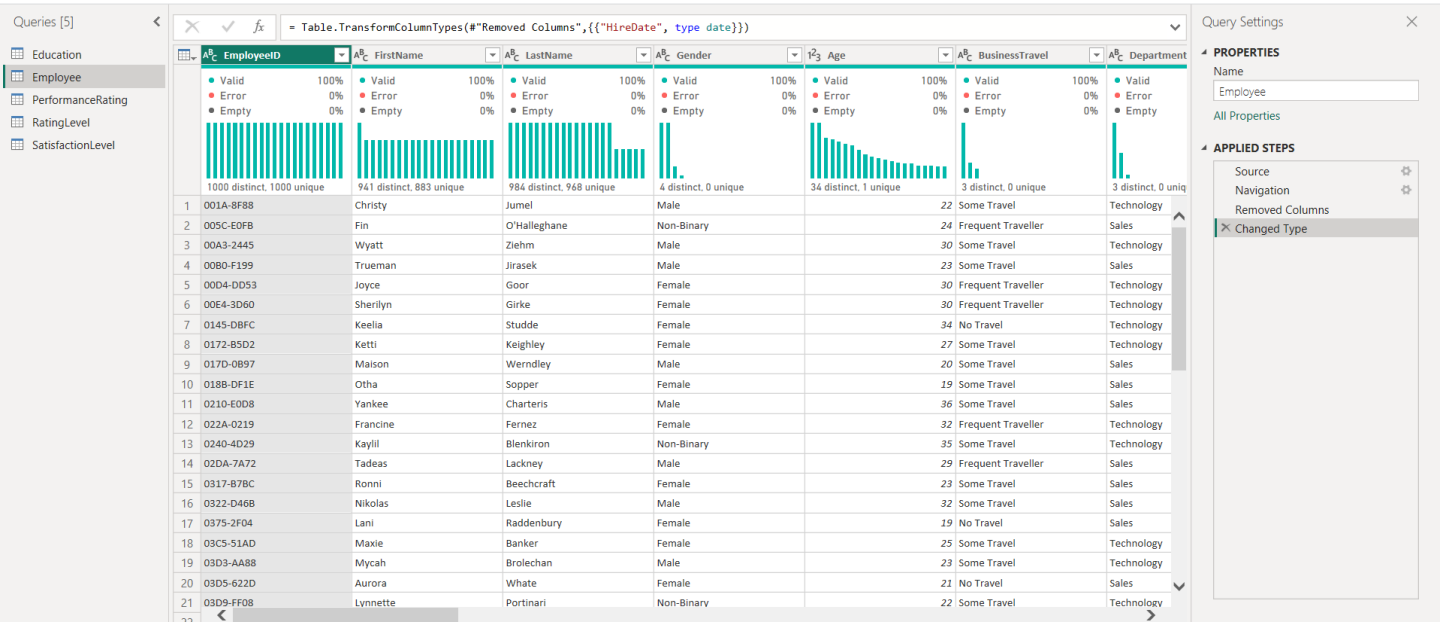
Step 1: Connecting Power BI to SQL Server and Data Transformation

1. Connecting Power BI to SQL Server

The initial step in this project involves connecting Power BI to a SQL Server database to load the HR datasets. By establishing this connection, we can pull the necessary data directly from SQL Server and ensure that the data is updated in real time when needed.

2. Data Transformation with Power Query

Once the data is loaded, it's essential to prepare it for analysis. Using **Power Query** in Power BI, we clean and transform the data to ensure consistency and accuracy in the final analysis.



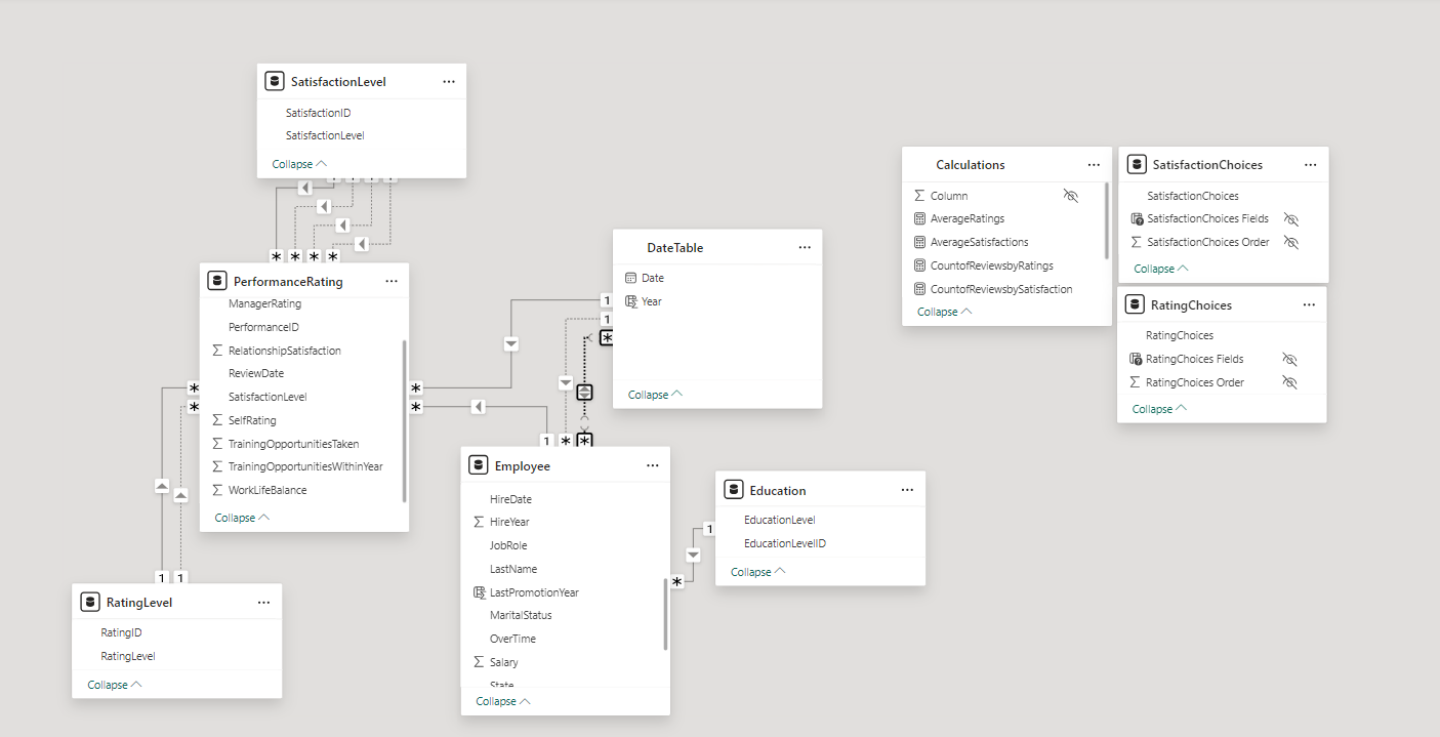
	EmployeeID	FirstName	LastName	Gender	Age	BusinessTravel	Department
1	001A-8F88	Christy	Jumel	Male	22	Some Travel	Technology
2	005C-E0FB	Fin	O'Halleghane	Non-Binary	24	Frequent Traveller	Sales
3	00A3-2445	Wyatt	Ziehm	Male	30	Some Travel	Technology
4	00B0-F199	Trueman	Jirasek	Male	23	Some Travel	Sales
5	00D4-DD53	Joyce	Goor	Female	30	Frequent Traveller	Technology
6	00E4-3D60	Sherilyn	Girke	Female	30	Frequent Traveller	Technology
7	0145-DBFC	Keelia	Studde	Female	34	No Travel	Technology
8	0172-B5D2	Ketti	Keighley	Female	27	Some Travel	Technology
9	017D-0B97	Maison	Werndley	Male	20	Some Travel	Sales
10	018B-DF1E	Otha	Sopper	Female	19	Some Travel	Sales
11	0210-E0D8	Yankee	Charteris	Male	36	Some Travel	Sales
12	022A-0219	Francine	Fernez	Female	32	Frequent Traveller	Technology
13	0240-4D29	Kaylil	Blenkiron	Non-Binary	35	Some Travel	Technology
14	02DA-7A72	Tadeas	Lackney	Male	29	Frequent Traveller	Sales
15	0317-B7BC	Ronni	Beechcraft	Female	23	Some Travel	Sales
16	0322-D46B	Nikolas	Leslie	Male	32	Some Travel	Sales
17	0375-2F04	Lani	Raddenbury	Female	19	No Travel	Sales
18	03C5-51AD	Maxie	Banker	Female	25	Some Travel	Technology
19	03D3-AA88	Mycah	Brolechan	Male	23	Some Travel	Technology
20	03D5-622D	Aurora	Whate	Female	21	No Travel	Sales
21	03D9-FF08	Lynnette	Portinari	Non-Binary	22	Some Travel	Technology

Step 2: Data Modeling and Calculations in Power BI

After loading and transforming the data, the next step involves creating an efficient **data model** by connecting the relevant tables and establishing relationships between them. Additionally, we create specialized tables for handling dates and calculated measures, as well as parameters for slicers to enhance interactivity in the dashboard.

1. Creating the Data Model and Connecting Tables

To ensure smooth analysis and accurate visualizations, it's important to establish relationships between different tables based on key identifiers such as EmployeeID. This step allows for seamless data retrieval and interaction across different datasets, such as employee information, satisfaction levels, and performance ratings.



2. Creating a Date Table

A **Date Table** is crucial for time-based analysis, enabling insights into trends over time (e.g., performance ratings or satisfaction scores over the years). It also helps with time intelligence calculations.

Date Table Relationships

The **Date Table** in this model plays a crucial role in handling time-based analysis for various employee-related events, including performance reviews, hire dates, and promotions. The table has the following relationships:

1. **Active One-to-Many Relationship with ReviewDate in PerformanceRating Table**:
    - This relationship is active and connects the Date column in the **Date Table** with the ReviewDate column in the PerformanceRating table.
    - **Purpose**: It allows for time-based analysis of performance reviews, enabling filtering and reporting by review year, month, or quarter.
2. **Inactive One-to-Many Relationship with HireDate in Employee Table**:
    - This relationship is inactive by default, connecting the Date column in the **Date Table** with the HireDate column in the Employee table.
    - **Purpose**: This relationship is used when analyzing employees based on their hire dates. It can be activated as needed using USERELATIONSHIP in DAX calculations to ensure proper filtering for hire-related analyses.
3. **Inactive Many-to-Many Relationship with LastPromotionYear in Employee Table**:
    - This is an inactive many-to-many relationship between the Date column in the **Date Table** and the LastPromotionYear in the Employee table.

- **Purpose**: This relationship is used to track employees based on the year they were last promoted. Since the relationship is many-to-many, it handles scenarios where multiple employees share the same promotion year, and is activated through DAX expressions when needed for promotion-based analysis.

3. Creating a Calculations Table for Measures

A **Calculations Table** is used to store all custom measures and calculations in a single place, keeping the data model organized. These measures are critical for performing more advanced analysis.

```
EmpCumulativeCount =
IF(
    ISFILTERED(DateTable[Year]),
    // If a year is selected, apply the filters
    CALCULATE(
        DISTINCTCOUNT(Employee[EmployeeID]),
        USERELATIONSHIP(DateTable[Date], Employee[HireDate]),
        DateTable[Date] <= MAX(DateTable[Date]),
        (
            Employee[Attrition] = "No"
            ||
            YEAR(Employee[HireDate]) + Employee[YearsAtCompany] > MAX(DateTable[Year])
        )
    ),
    // If no year is selected, return the distinct count of employees without any filters
    DISTINCTCOUNT(Employee[EmployeeID])
)
```

The EmpCumulativeCount measure is used to calculate the cumulative count of employees based on whether a specific year or range of years is selected from the DateTable. This measure ensures that only employees who were active in the selected year(s) or who left after that period are included in the count.

1. **When One or More Years are Selected**:
    - The measure uses CALCULATE to apply filtering logic based on the selected year(s) from the DateTable.
    - It uses USERELATIONSHIP to activate the relationship between DateTable[Date] and Employee[HireDate] temporarily, ensuring that employee records are filtered based on their hire date.
    - The DateTable[Date] <= MAX(DateTable[Date]) condition ensures that only employees hired on or before the selected date are counted.
    - The Attrition condition ensures that only employees who are still with the company (Attrition = "No") or employees who left the company after the selected year (determined by adding YearsAtCompany to HireYear) are included in the count.
2. **When No Year is Selected**:

- If no year is selected, the measure defaults to returning the total distinct count of employees without applying any date filters or attrition logic.

This measure allows for dynamic filtering when a specific year or range of years is selected, providing cumulative employee counts for those who were active during the selected period. If no year is selected, it defaults to showing the total number of employees.

```
ReviewCount =
IF(
    ISFILTERED(DateTable[Year]),
    // If one or more years are selected, apply the filters
    CALCULATE(
        COALESCE(DISTINCTCOUNT(PerformanceRating[PerformanceID]), 0),
        PerformanceRating[ReviewDate] IN VALUES(DateTable[Date]),  // Consider reviews only within the selected date range
        // Only include reviews for employees who are still active as of the selected year
        (
            Employee[Attrition] = "No"
            ||
            YEAR(Employee[HireDate]) + Employee[YearsAtCompany] >= MIN(DateTable[Year])
        )
    ),
    // If no year is selected, return the distinct count of all reviews without any filters
    COALESCE(DISTINCTCOUNT(PerformanceRating[PerformanceID]), 0)
)
```

The ReviewCount measure is designed to calculate the distinct count of performance reviews based on whether a year or range of years is selected in the DateTable. This measure includes logic to filter reviews for employees who were still active during the selected year(s) or left the company after that period.

1. **When One or More Years are Selected**:
    - The measure uses CALCULATE to filter performance reviews based on the selected year(s).
    - It checks if the ReviewDate of each review falls within the selected date range.
    - Additionally, it ensures that only reviews for employees who were still active during the selected year(s) or who left after that year are included.
        - Active employees are determined by checking whether their Attrition status is "No" or if their calculated attrition year (hire year + years at the company) is greater than or equal to the minimum selected year.
2. **When No Year is Selected**:
    - If no specific year is selected, the measure defaults to counting all distinct performance reviews without applying any date filters or employee status conditions.

This measure is flexible, accounting for both single and multiple year selections, while ensuring that only relevant reviews from active employees (or those who left after the selected years) are included in the final count. If no year is selected, the measure returns the total distinct count of all reviews.

```
SalarySinceLastPromotion =
CALCULATE(
    AVERAGEX(
        Employee,
        IF(
            Employee[LastPromotionYear] <= MAX(DateTable[Year])
            && (
                Employee[Attrition] = "No"
                || YEAR(Employee[HireDate]) + Employee[YearsAtCompany] > MAX(DateTable[Year])
            ),
            Employee[Salary],
            BLANK()
        )
    ),
    USERELATIONSHIP(Employee[LastPromotionYear], DateTable[Year]),
    USERELATIONSHIP(Employee[HireDate], DateTable[Date]),
    DateTable[Date] <= MAX(DateTable[Date])
)
```

The SalarySinceLastPromotion measure calculates the **average salary of employees** based on their promotion year and employment status, considering the selected year from the DateTable. This measure takes into account employees who are either still with the company or who left after the selected year, ensuring that only relevant employees are included in the calculation. The measure uses a combination of conditional logic and dynamic relationships to filter and calculate the average salary.

```
AverageSatisfactions = COALESCE(AVERAGEX(PerformanceRating,(PerformanceRating[EnvironmentSatisfaction]+PerformanceRating[JobSatisfaction]
+PerformanceRating[RelationshipSatisfaction]+PerformanceRating[WorkLifeBalance])/4),0)
```

The AverageSatisfactions measure calculates the average of four different satisfaction metrics for employees:

1. **Environment Satisfaction**
2. **Job Satisfaction**
3. **Relationship Satisfaction**
4. **Work-Life Balance**

- **AVERAGEX**: Iterates through each row in the PerformanceRating table, calculating the average of the four satisfaction scores.
- **COALESCE**: Ensures that if no satisfaction data is available, the measure returns 0 instead of NULL.

This measure provides a single, comprehensive satisfaction score by averaging the key satisfaction metrics for each employee.

```
AverageRatings = COALESCE(AVERAGEX(PerformanceRating,(PerformanceRating[ManagerRating]+PerformanceRating[SelfRating])/2),0)
```

The AverageRatings measure calculates the average of the **Manager Rating** and **Self Rating** for each employee.

- **AVERAGEX**: Iterates through the PerformanceRating table, calculating the average of the ManagerRating and SelfRating for each row.
- **COALESCE**: Ensures that if there are no ratings available (resulting in a NULL), the measure returns 0 instead of NULL.

This measure provides an overall performance rating by combining both manager and self-assessments, offering a balanced view of employee performance.

```
CountofReviewsbySatisfaction =
VAR SelectedField = SELECTEDVALUE(SatisfactionChoices[SatisfactionChoices Order])
VAR SelectedLevels = VALUES(SatisfactionLevel[SatisfactionID])
VAR LevelsCount = COUNTROWS(SelectedLevels)
RETURN
IF(
    LevelsCount = 0,
    0, -- Return zero if no satisfaction levels are selected
    SWITCH(
        TRUE(),
        SelectedField = 0, -- EnvironmentSatisfaction
            CALCULATE(

                COALESCE(COUNTROWS(PerformanceRating), 0),
                PerformanceRating[EnvironmentSatisfaction] IN SelectedLevels
                -- No need for USERELATIONSHIP since this is the active relationship
            ),
        SelectedField = 1, -- JobSatisfaction
            CALCULATE(
                COALESCE(COUNTROWS(PerformanceRating), 0),
                USERELATIONSHIP(PerformanceRating[JobSatisfaction], SatisfactionLevel[SatisfactionID]),
                PerformanceRating[JobSatisfaction] IN SelectedLevels
            ),
        SelectedField = 2, -- RelationshipSatisfaction
            CALCULATE(
                COALESCE(COUNTROWS(PerformanceRating), 0),
                USERELATIONSHIP(PerformanceRating[RelationshipSatisfaction], SatisfactionLevel[SatisfactionID]),
                PerformanceRating[RelationshipSatisfaction] IN SelectedLevels
            ),
        SelectedField = 3, -- WorkLifeBalance
```

```
        CALCULATE(
            COALESCE(COUNTROWS(PerformanceRating), 0),
            USERELATIONSHIP(PerformanceRating[WorkLifeBalance], SatisfactionLevel[SatisfactionID]),
            PerformanceRating[WorkLifeBalance] IN SelectedLevels
        ),
    BLANK()
)
```

The CountofReviewsbySatisfaction measure counts the number of performance reviews based on the selected satisfaction metric and levels. It dynamically adjusts based on the selected satisfaction category (Environment, Job, Relationship, or Work-Life Balance) and the satisfaction levels chosen by the user.

- **VAR SelectedField**: Captures the user's selected satisfaction type (from SatisfactionChoices).
- **VAR SelectedLevels**: Gets the list of selected satisfaction levels.
- **LevelsCount**: Checks if any satisfaction levels are selected. If none are selected, it returns 0.
- **SWITCH**: Based on the selected satisfaction category (SelectedField), the measure calculates the number of reviews for the chosen satisfaction level:
    - **Environment Satisfaction**: The active relationship is already defined, so no need for USERELATIONSHIP.
    - **Job, Relationship, and Work-Life Balance Satisfaction**: Uses USERELATIONSHIP to activate the correct relationship and filter based on the selected satisfaction levels.

This measure dynamically counts reviews based on the satisfaction metric chosen by the user, adjusting for the selected levels of satisfaction in real-time.

```
CountofReviewsbyRatings =
VAR SelectedField = SELECTEDVALUE(RatingChoices[RatingChoices Order])
VAR SelectedLevels = VALUES(RatingLevel[RatingID])
VAR LevelsCount = COUNTROWS(SelectedLevels)
RETURN
IF(
    LevelsCount = 0,
    0, -- Return zero if no rating levels are selected
    SWITCH(
        TRUE(),
        SelectedField = 0, -- Selfrating
            CALCULATE(
                COALESCE(COUNTROWS(PerformanceRating), 0),
                PerformanceRating[SelfRating] IN SelectedLevels
                -- No need for USERELATIONSHIP since this is the active relationship
            ),
        SelectedField = 1, -- Managerrating
            CALCULATE(
                COALESCE(COUNTROWS(PerformanceRating), 0),
                USERELATIONSHIP(PerformanceRating[ManagerRating], RatingLevel[RatingID]),
                PerformanceRating[ManagerRating] IN SelectedLevels
            ),

        BLANK()
    )
)
```

The CountofReviewsbyRatings measure dynamically counts the number of performance reviews based on the selected **rating type** (either self-rating or manager rating) and the chosen **rating levels**.

- **VAR SelectedField**: Captures the selected rating type (from RatingChoices), which determines whether the user is looking at self-ratings or manager ratings.
- **VAR SelectedLevels**: Retrieves the specific rating levels selected by the user.
- **LevelsCount**: If no rating levels are selected, the measure returns 0.
- **SWITCH**: Based on the selected rating type (SelectedField):
    - **SelfRating**: Counts reviews based on self-rating, no need for USERELATIONSHIP since the relationship is active by default.
    - **ManagerRating**: Uses USERELATIONSHIP to temporarily activate the relationship between ManagerRating and RatingLevel[RatingID], ensuring proper filtering.

This measure allows users to filter and count reviews by different rating categories and levels, dynamically adjusting based on the user's selections.

```
AttritionCumulativeCount =
IF(
    // Case 1: Only one year is selected
    HASONEVALUE(DateTable[Year]),
    CALCULATE(
        COALESCE(DISTINCTCOUNT(Employee[EmployeeID]), 0),
        USERELATIONSHIP(DateTable[Date], Employee[HireDate]),
        DateTable[Date] <= MAX(DateTable[Date]),
        (
            Employee[Attrition] = "yes"
            && Employee[AttritionYear] = MAX(DateTable[Year])  // Exact match to the selected year
        )
    ),

    // Case 2: Multiple years selected
    IF(
        ISFILTERED(DateTable[Year]),
        CALCULATE(
            COALESCE(DISTINCTCOUNT(Employee[EmployeeID]), 0),
            USERELATIONSHIP(DateTable[Date], Employee[HireDate]),
            DateTable[Date] <= MAX(DateTable[Date]),
            (
                Employee[Attrition] = "yes"
                && Employee[AttritionYear] <= MAX(DateTable[Year])  // Up to the latest year selected
            )
        ),

        // Case 3: No year is selected (default case)
        CALCULATE(
            COALESCE(DISTINCTCOUNT(Employee[EmployeeID]), 0),
            USERELATIONSHIP(DateTable[Date], Employee[HireDate]),
            DateTable[Date] <= MAX(DateTable[Date]),
            (
                Employee[Attrition] = "yes"  // No specific year filtering applied
            )
        )
    )
)
```

The AttritionCumulativeCount measure is a dynamic calculation that adjusts to user input in Power BI. It calculates the cumulative count of employees who have left the company (those with Attrition = "yes") based on whether one year, multiple years, or no years are selected in the DateTable. The measure ensures that the correct context and relationships are applied, providing accurate results for employee attrition based on the selected time period.

4. Creating Parameters for Slicers (Satisfaction and Ratings)

To allow users to dynamically filter the data based on employee satisfaction type and different performance rating, we create **parameters** for these metrics and link them to slicers in the dashboard.

SatisfactionChoices	SatisfactionChoices Fields	SatisfactionChoices Order
Environment Satisfaction	'PerformanceRating'[EnvironmentSatisfaction]	0
Job Satisfaction	'PerformanceRating'[JobSatisfaction]	1
Relationship Satisfaction	'PerformanceRating'[RelationshipSatisfaction]	2
Work Life Balance	'PerformanceRating'[WorkLifeBalance]	3

RatingChoices	RatingChoices Fields	RatingChoices Order
Self Rating	'PerformanceRating'[SelfRating]	0
Manager Rating	'PerformanceRating'[ManagerRating]	1

Step 3: Creating an Interactive Dashboard with Multiple Views

In this final phase, we create an interactive Power BI dashboard that allows users to switch between different views—**Employee Overview**, **Satisfaction Overview**, and **Ratings Overview**—using buttons. The dashboard also includes reset buttons to clear filters and return to the default view, providing a smooth and user-friendly experience for exploring various HR metrics.

1. Creating Multiple Views

The dashboard features three distinct views that provide comprehensive insights into the HR data:

• **Employee Overview**:

- Displays key metrics such as total employee count, average salary, demographics (age, gender, ethnicity), and employment trends.
- Includes visuals like bar charts, pie charts, and line graphs for a clear breakdown of employee data.

• **Satisfaction Overview**:

- Focuses on employee satisfaction metrics, including environment, job, relationship, and work-life balance satisfaction levels.
- Allows users to filter satisfaction metrics by department, job role, and other key categories.

• **Ratings Overview**:

- Provides insights into performance ratings, including manager and self-ratings.
- Includes the ability to filter by rating levels, job role, and department, with charts illustrating ratings over time and across demographic categories.

• **Attrition Overview**:

- Highlights employee turnover trends by showing the total number of employees who have left, and attrition rates according to different aspects.
- Visualizes attrition data across dimensions such as demographic groups providing insights into which areas are most affected by employee turnover.

- Includes filters for analyzing attrition by key categories, such as department, job role, and year.

2. Switching Views Using Buttons

To enhance interactivity, we include buttons that allow users to seamlessly switch between the different dashboard views. Each button corresponds to one of the views:

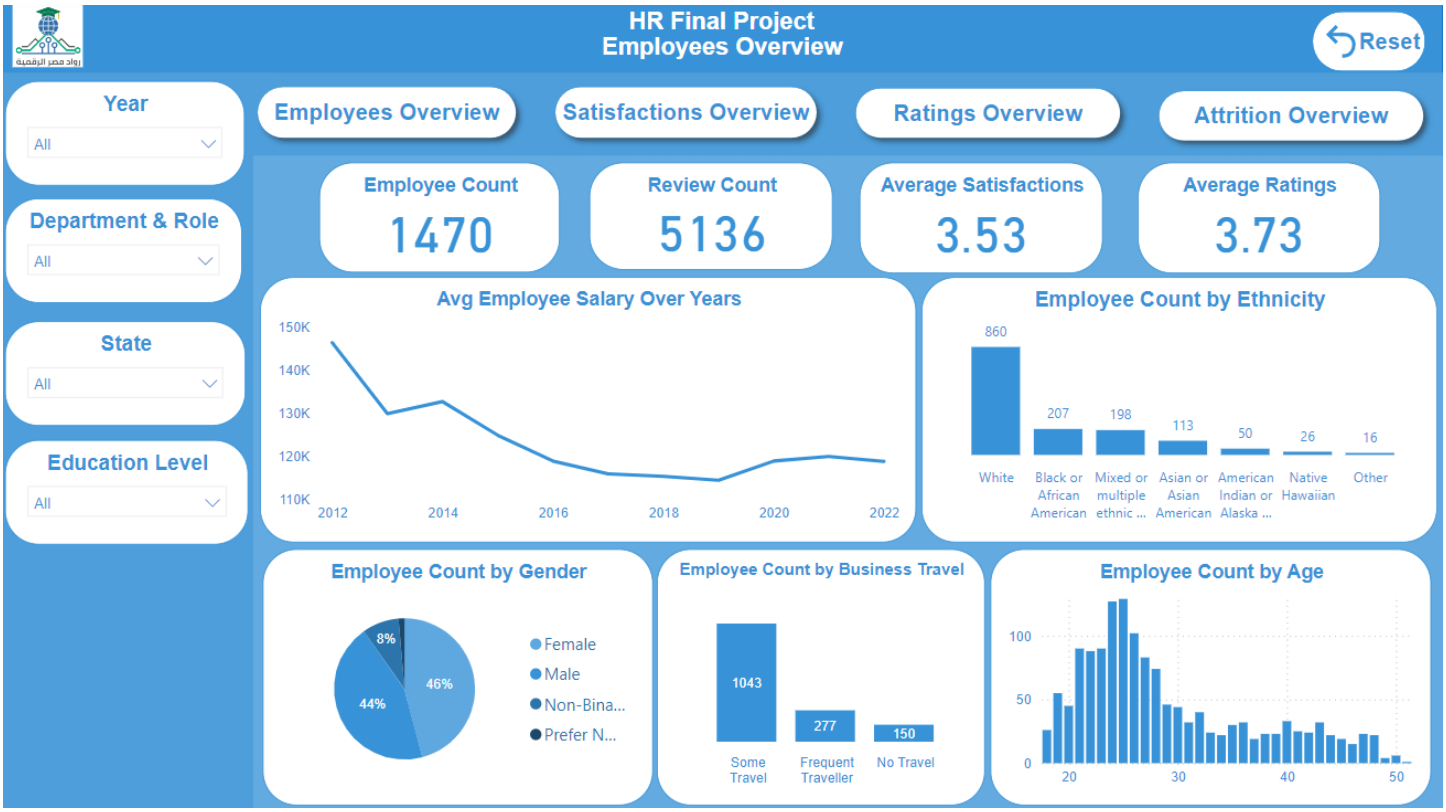
This functionality ensures users can easily navigate between the different views without overwhelming the dashboard with too many visuals at once.

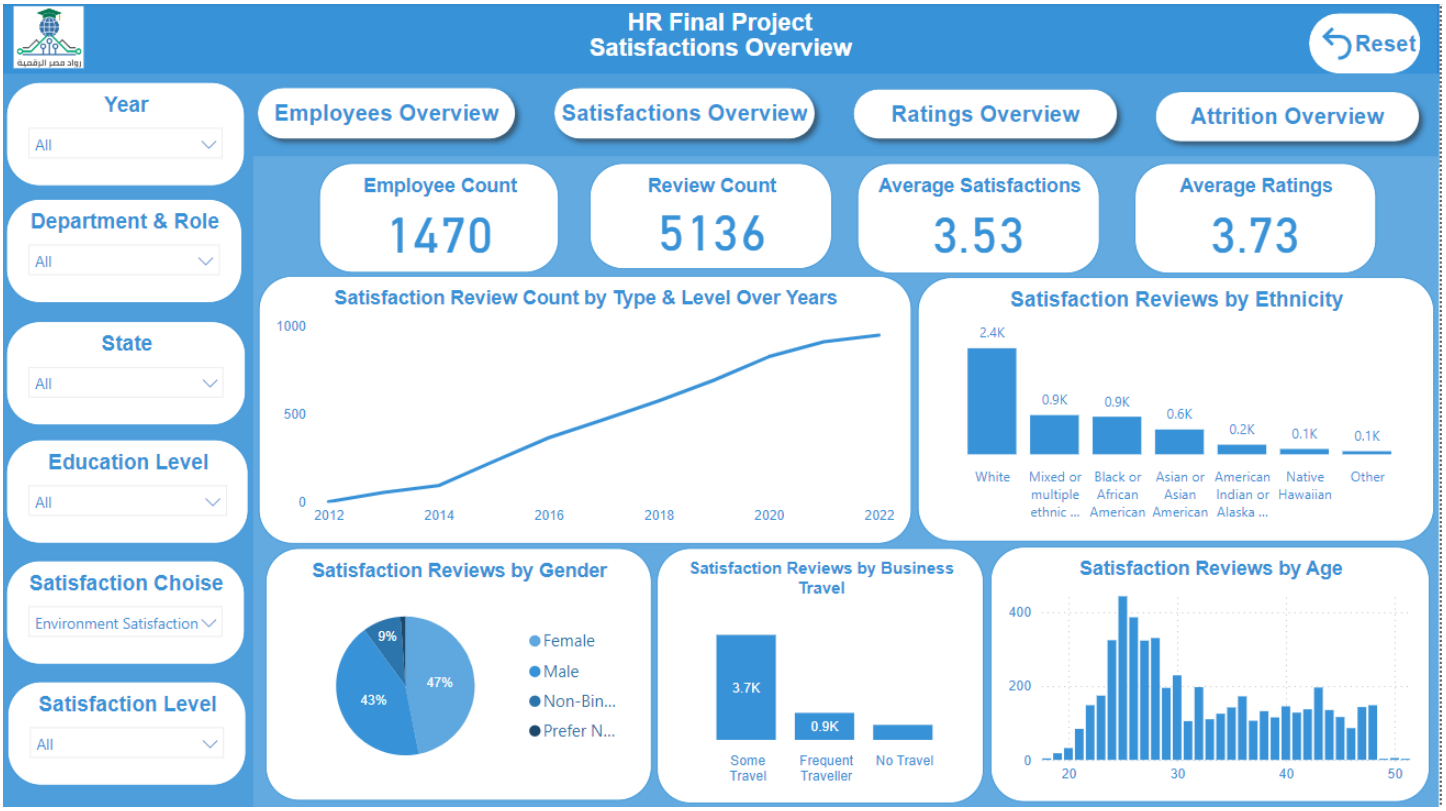
3. Adding Reset Buttons

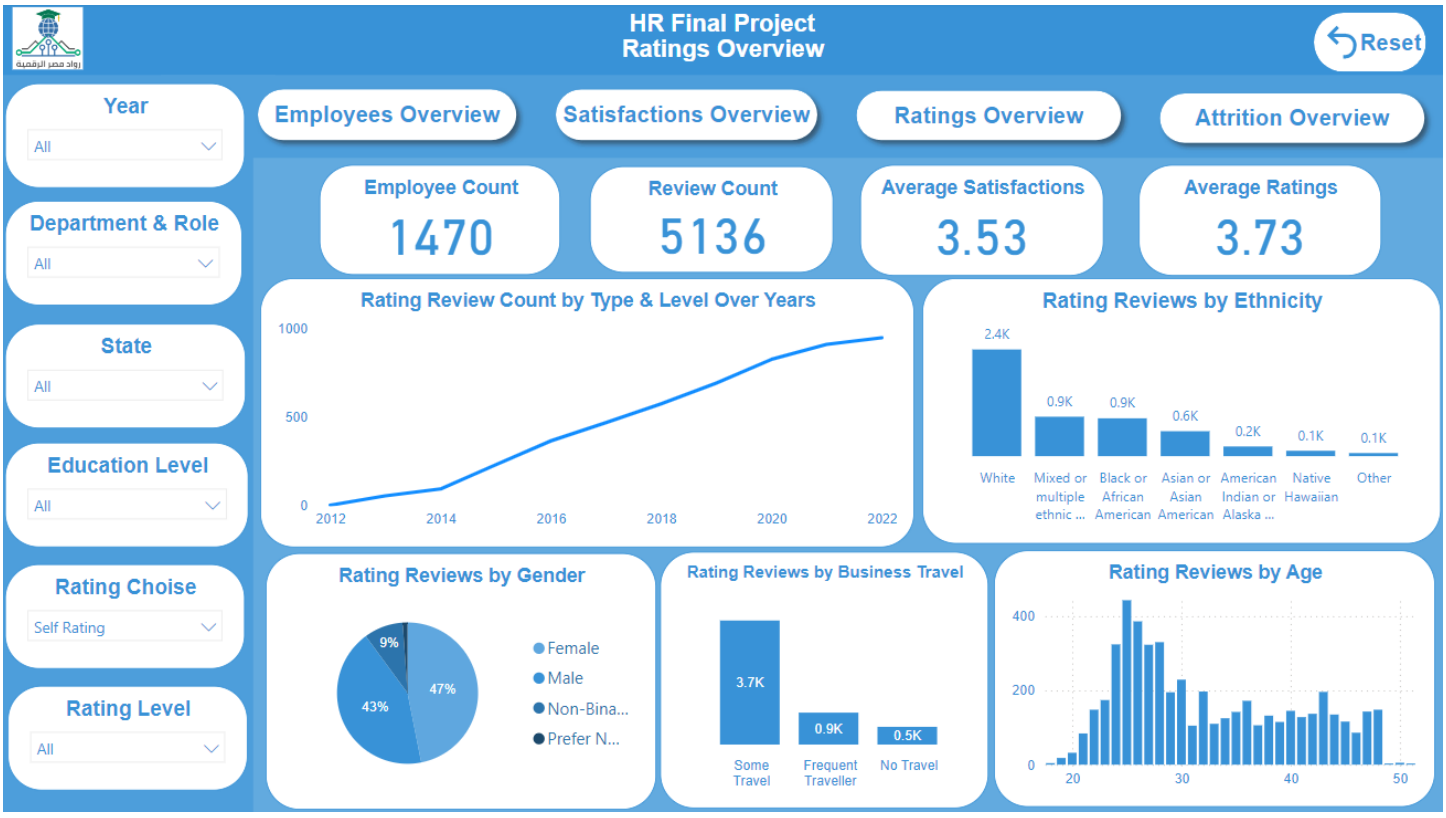
To further improve the user experience, we add **Reset Filters** buttons that allow users to clear any applied filters and return to the default state of the dashboard:

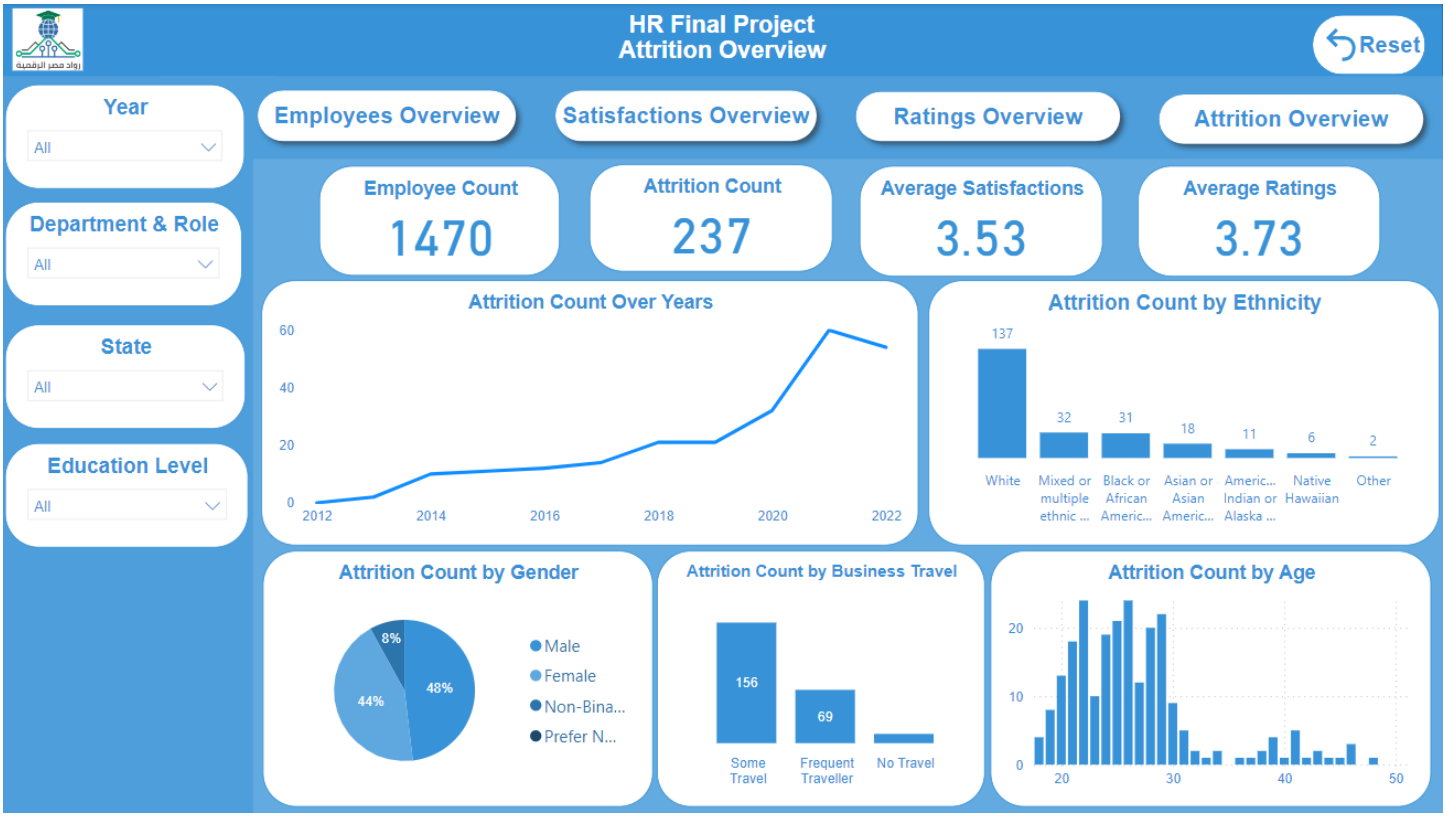
4. Final Dashboard Design

- The dashboard is designed with clear visuals and intuitive navigation. Each view provides a focused analysis, with buttons allowing users to explore different aspects of the HR data (employees, satisfaction, ratings, and attrition).
- Users can apply filters such as department, job role, and satisfaction/rating levels, then easily switch views or reset the dashboard for further exploration.


HR Final Project
Employees Overview
Reset
Year
All
Department & Role
All
State
All
Education Level
All
Employees Overview
Satisfactions Overview
Ratings Overview
Attrition Overview
Employee Count
1470
Review Count
5136
Average Satisfactions
3.53
Average Ratings
3.73
Avg Employee Salary Over Years
150K
140K
130K
120K
110K
2012
2014
2016
2018
2020
2022
Employee Count by Ethnicity
860
207
198
113
50
26
16
White
Black or African American
Mixed or multiple ethnic ...
Asian or Asian American
American Indian or Alaska ...
Native Hawaiian
Other
Employee Count by Gender
8%
46%
44%
Female
Male
Non-Bina...
Prefer N...
Employee Count by Business Travel
1043
277
150
Some Travel
Frequent Traveller
No Travel
Employee Count by Age
100
50
0
20
30
40
50


HR Final Project
Satisfactions Overview
Reset
Year
All
Department & Role
All
State
All
Education Level
All
Satisfaction Choise
Environment Satisfaction
Satisfaction Level
All
Employees Overview
Satisfactions Overview
Ratings Overview
Attrition Overview
Employee Count
1470
Review Count
5136
Average Satisfactions
3.53
Average Ratings
3.73
Satisfaction Review Count by Type & Level Over Years
1000
500
0
2012
2014
2016
2018
2020
2022
Satisfaction Reviews by Ethnicity
2.4K
0.9K
0.9K
0.6K
0.2K
0.1K
0.1K
White
Mixed or multiple ethnic ...
Black or African American
Asian or Asian American
American Indian or Alaska ...
Native Hawaiian
Other
Satisfaction Reviews by Gender
9%
47%
43%
Female
Male
Non-Bin...
Prefer N...
Satisfaction Reviews by Business Travel
3.7K
0.9K
Some Travel
Frequent Traveller
No Travel
Satisfaction Reviews by Age
400
200
0
20
30
40
50


HR Final Project
Ratings Overview
Reset
Year
All
Department & Role
All
State
All
Education Level
All
Rating Choise
Self Rating
Rating Level
All
Employees Overview
Satisfactions Overview
Ratings Overview
Attrition Overview
Employee Count
1470
Review Count
5136
Average Satisfactions
3.53
Average Ratings
3.73
Rating Review Count by Type & Level Over Years
1000
500
0
2012
2014
2016
2018
2020
2022
Rating Reviews by Ethnicity
2.4K
0.9K
0.9K
0.6K
0.2K
0.1K
0.1K
White
Mixed or multiple ethnic ...
Black or African American
Asian or Asian American
American Indian or Alaska ...
Native Hawaiian
Other
Rating Reviews by Gender
9%
47%
43%
Female
Male
Non-Bina...
Prefer N...
Rating Reviews by Business Travel
3.7K
0.9K
0.5K
Some Travel
Frequent Traveller
No Travel
Rating Reviews by Age
400
200
0
20
30
40
50


HR Final Project
Attrition Overview
Reset
Year
All
Department & Role
All
State
All
Education Level
All
Employees Overview
Satisfactions Overview
Ratings Overview
Attrition Overview
Employee Count
1470
Attrition Count
237
Average Satisfactions
3.53
Average Ratings
3.73
Attrition Count Over Years
60
40
20
0
2012
2014
2016
2018
2020
2022
Attrition Count by Ethnicity
137
32
31
18
11
6
2
White
Mixed or multiple ethnic ...
Black or African Americ...
Asian or Asian Americ...
Americ... Indian or Alaska ...
Native Hawaiian
Other
Attrition Count by Gender
8%
48%
44%
Male
Female
Non-Bina...
Prefer N...
Attrition Count by Business Travel
156
69
Some Travel
Frequent Traveller
No Travel
Attrition Count by Age
20
10
0
20
30
40
50