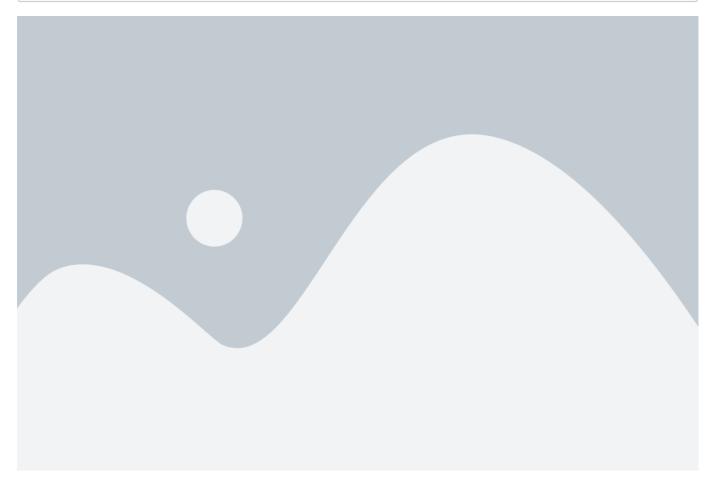
## **Employee Database Analysis**

#### Create the database

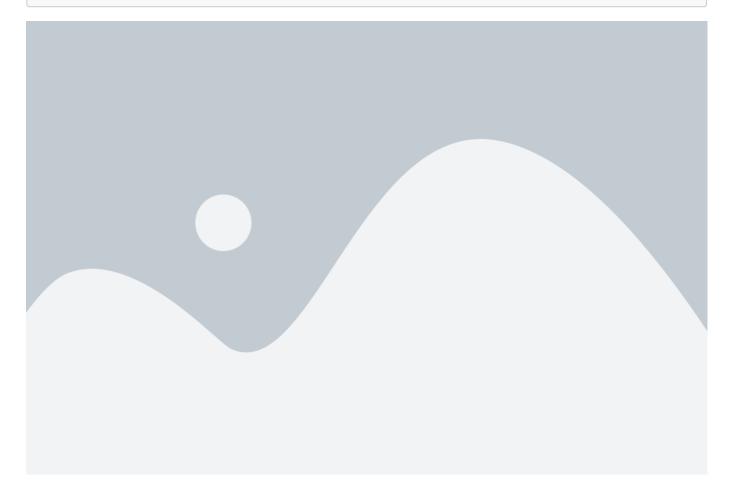
```
![screenshot](screenshots/placeholder.png)
CREATE DATABASE HREmployeeDB;
USE HREmployeeDB;
```



#### Create the EmployeeData table

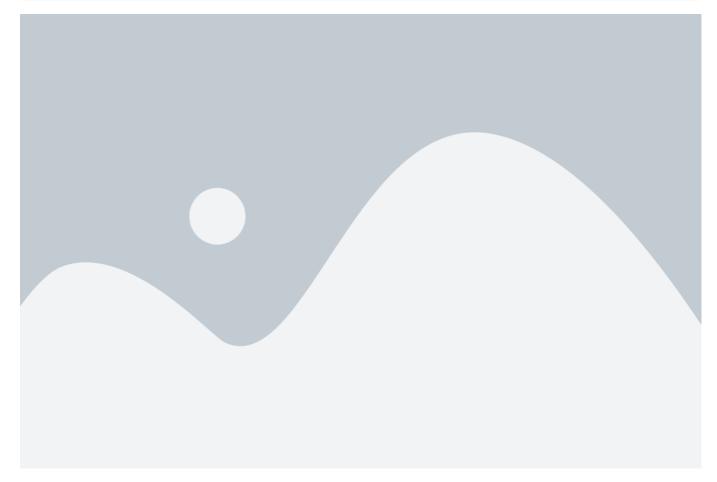
```
CREATE TABLE EmployeeData (
Attrition NVARCHAR(50),
BusinessTravel NVARCHAR(50),
CF_age_band NVARCHAR(50),
CF_attrition_label NVARCHAR(50),
Department NVARCHAR(50),
EducationField NVARCHAR(50),
emp_no NVARCHAR(50),
EmployeeNumber INT,
Gender NVARCHAR(50),
JobRole NVARCHAR(50),
MaritalStatus NVARCHAR(50),
OverTime NVARCHAR(50),
OverTime NVARCHAR(50),
```

```
TrainingTimesLastYear INT,
    Age INT,
    CF_current NVARCHAR(50),
   DailyRate INT,
   DistanceFromHome INT,
    Education NVARCHAR(50),
    EmployeeCount INT,
    EnvironmentSatisfaction INT,
   HourlyRate INT,
   JobInvolvement INT,
   JobLevel INT,
    JobSatisfaction INT,
   MonthlyIncome INT,
   MonthlyRate INT,
   NumCompaniesWorked INT,
    PercentSalaryHike INT,
   PerformanceRating INT,
    RelationshipSatisfaction INT,
    StandardHours INT,
    StockOptionLevel INT,
   TotalWorkingYears INT,
   WorkLifeBalance INT,
   YearsAtCompany INT,
   YearsInCurrentRole INT,
   YearsSinceLastPromotion INT,
   YearsWithCurrManager INT
);
```



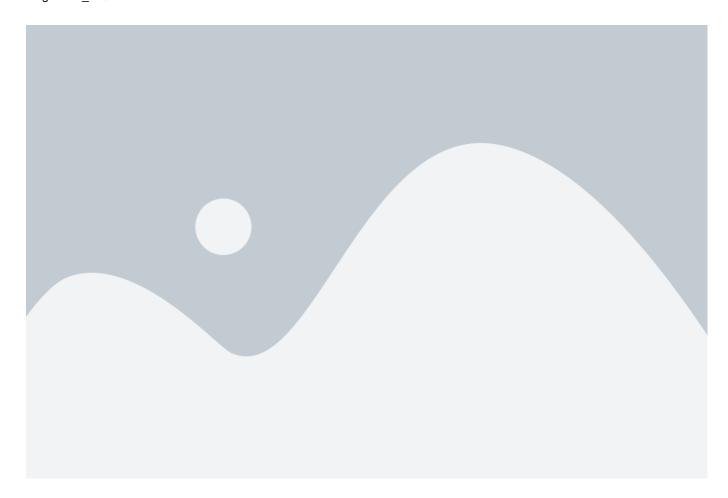
#### Bulk insert data into the table

```
BULK INSERT EmployeeData
FROM 'C:\Users\Administrator\Downloads\HR_Employee1.csv'
WITH
(
    FIELDTERMINATOR = ',', ### Delimiter for fields
    ROWTERMINATOR = '0x0a', ### End of each row
    FIRSTROW = 2 ### Skip header row
);
```



#### a. Return the shape of the table

```
SELECT
  (SELECT COUNT(*) FROM EmployeeData) AS row_bo,
   (SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME =
   'EmployeeData') AS no_columns;
```



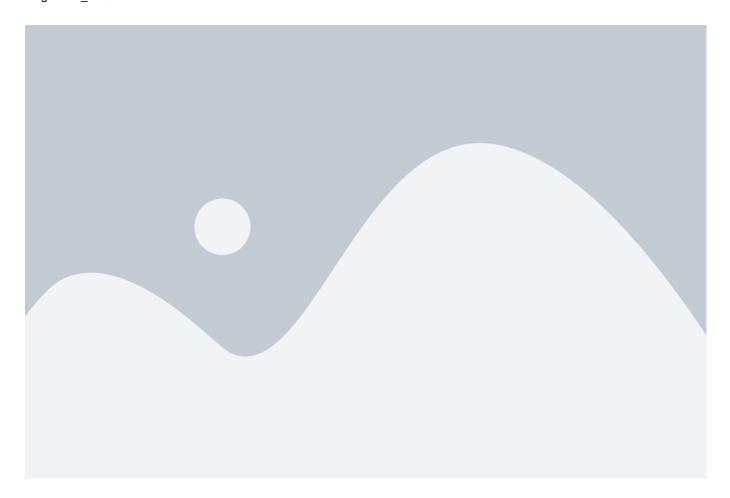
### b. Calculate the cumulative sum of total working years for each department

Department,
TotalWorkingYears,
SUM(TotalWorkingYears) OVER (PARTITION BY Department ORDER BY
TotalWorkingYears ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS
Cumulative\_sum\_year
FROM EmployeeData;



#### 3. Which Gender Has Higher Strength as Workforce in Each Department

```
WITH GenderCounts AS (
    SELECT
        Department,
        Gender,
        COUNT(*) as counts,
        SUM(CASE WHEN Gender = 'Male' THEN 1 ELSE 0 END) AS Males,
        SUM(CASE WHEN Gender = 'Female' THEN {f 1} ELSE {f 0} END) AS Females
    FROM
        EmployeeData
    GROUP BY
        Department,
        Gender
SELECT
    Department,
    Gender AS ProminentGender,
    counts,
     RANK() OVER (PARTITION BY Department ORDER BY counts DESC) AS Gender_Rank
FROM GenderCounts;
```



4. Create a New Column AGE\_BAND and Show Distribution of Employee's Age Band Group

ALTER TABLE EmployeeData
ADD Age\_Band NVARCHAR(50);



```
UPDATE EmployeeData

SET Age_Band = CASE

WHEN Age < 25 THEN 'Below 25'

WHEN Age BETWEEN 25 AND 34 THEN '25-34'

WHEN Age BETWEEN 35 AND 44 THEN '35-44'

WHEN Age BETWEEN 45 AND 55 THEN '45-55'

ELSE 'Above 55'

END;

SELECT

Age_Band,

COUNT(*) AS Count

FROM EmployeeData

GROUP BY Age_Band;
```



5. Compare All Marital Status of Employees and Find the Most Frequent Marital Status

Query to get marital status count and frequency rank

```
SELECT

MaritalStatus,

Count,

RANK() OVER (ORDER BY Count DESC) AS Freq_Rank

FROM (

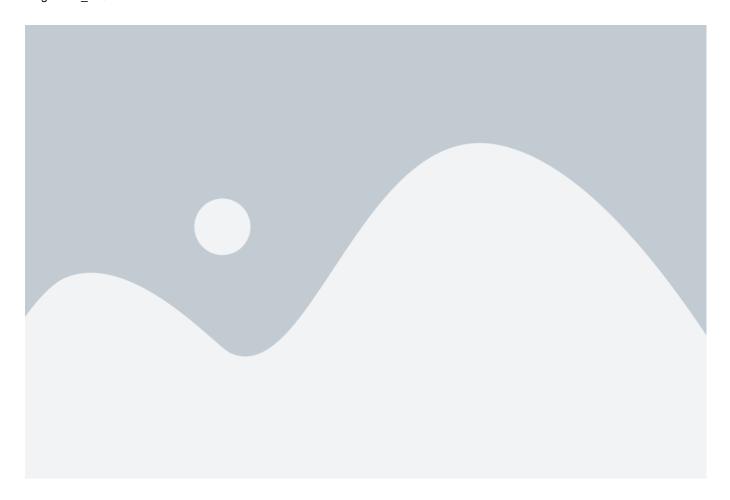
SELECT

MaritalStatus,

COUNT(*) AS Count

FROM EmployeeData

GROUP BY MaritalStatus
) AS _
```



#### 6. Show the Job Role with Highest Attrition Rate (Percentage)

```
WITH AttritionRate AS (
    SELECT
        JobRole,
        (SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END) * 100.0) / COUNT(*) AS

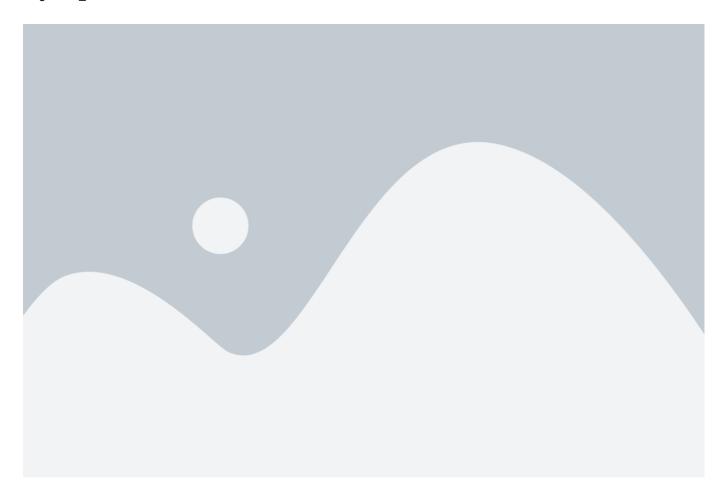
Attrition_Percentage
    FROM EmployeeData
    GROUP BY JobRole
)

SELECT TOP 1
    JobRole,
    Attrition_Percentage
FROM AttritionRate
ORDER BY Attrition_Percentage DESC;
```



# 7. Show Distribution of Employee's Promotion, Find the Maximum Chances of Employee Getting Promoted

SELECT YearsSinceLastPromotion, COUNT(\*) AS EmployeeCount FROM EmployeeData GROUP BY YearsSinceLastPromotion ORDER BY YearsSinceLastPromotion;



- 8. Show the Cumulative Sum of Total Working Years for Each Department
- 9. Find the Rank of Employees Within Each Department

Based on Their Monthly Income

```
SELECT
EmployeeNumber,
Department,
MonthlyIncome,
RANK() OVER (PARTITION BY Department ORDER BY MonthlyIncome DESC) AS
Income_Rank
FROM EmployeeData;
```



# 10. Calculate the Running Total of 'Total Working Years' for Each Employee Within Each Department and Age Band

```
SELECT

EmployeeNumber,

Department,

AGE_BAND,

TotalWorkingYears,

SUM(TotalWorkingYears) OVER (PARTITION BY Department, AGE_BAND ORDER BY EmployeeNumber) AS Running_Total_Working_Years

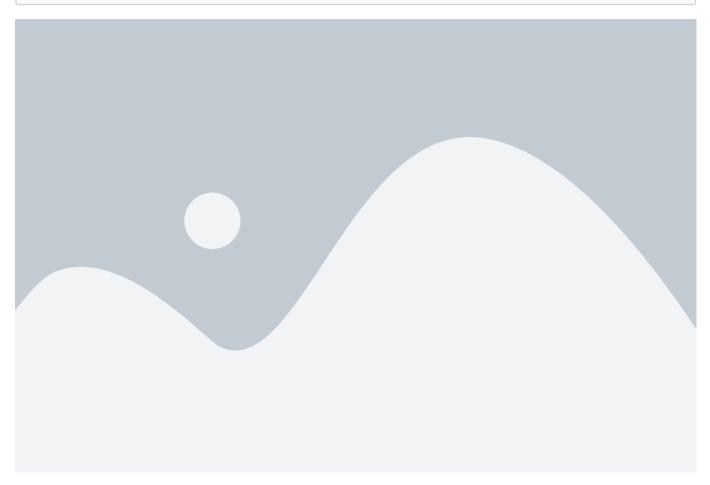
FROM EmployeeData;
```



11. For Each Employee Who Left, Calculate the Number of Years They Worked Before Leaving and Compare It with the Average Years Worked by Employees in the Same Department

```
WITH YearsWorked AS (
    SELECT
        EmployeeNumber,
        Department,
        TotalWorkingYears AS Years_Worked_Before_Leaving
    FROM EmployeeData
    WHERE Attrition = 'Yes'
),
AvgYears AS (
    SELECT
        Department,
        AVG(TotalWorkingYears) AS Avg_Years_Worked
    FROM EmployeeData
    GROUP BY Department
)
SELECT
    Y. Employee Number,
    Y.Department,
    Y.Years_Worked_Before_Leaving,
    A.Avg_Years_Worked
FROM YearsWorked Y, AvgYears A
WHERE Y.Department = A.Department AND Y.Years_Worked_Before_Leaving IS NOT NULL
ORDER BY
```

```
Department,
Years_Worked_Before_Leaving;
```



## 12. Rank the Departments by the Average Monthly Income of Employees Who Have Left

```
WITH DepartmentIncome AS (
SELECT
Department,
AVG(MonthlyIncome) AS Avg_Monthly_Income
FROM EmployeeData
WHERE Attrition = 'Yes'
GROUP BY Department
)
SELECT
Department,
Avg_Monthly_Income,
RANK() OVER (ORDER BY Avg_Monthly_Income DESC) AS Department_Rank
FROM DepartmentIncome;
```



### 13. Find If There Is Any Relation Between Attrition Rate and Marital Status of Employee

```
SELECT

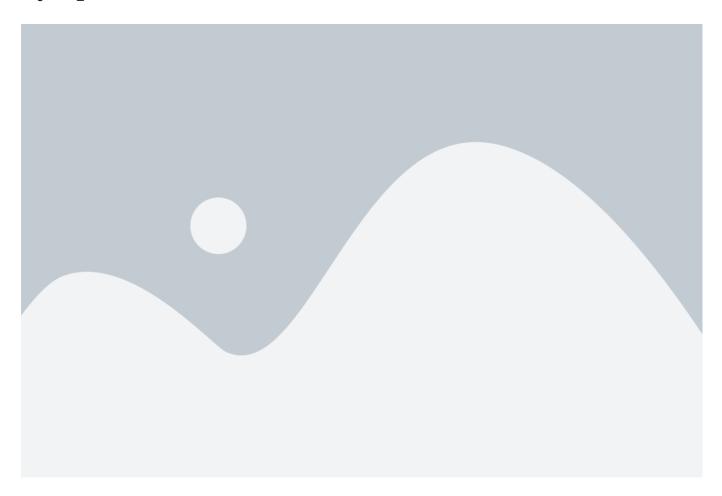
MaritalStatus,

(SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END) * 100.0) / COUNT(*) AS

Attrition_Percentage

FROM EmployeeData

GROUP BY MaritalStatus;
```



### Insight:

Attrition rate is higher in Singles. Attrition rate is lower in Married and Divorced.

14. Show the Department with Highest Attrition Rate (Percentage)

```
SELECT TOP 1

Department,

(SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END) * 100.0) / COUNT(*) AS

Attrition_Percentage

FROM EmployeeData

GROUP BY Department

ORDER BY Attrition_Percentage DESC;
```



## 15. Calculate the Moving Average of Monthly Income Over the Past 3 Employees for Each Job Role

```
WITH RankedEmployees AS (

SELECT

JobRole,

MonthlyIncome,

ROW_NUMBER() OVER (PARTITION BY JobRole ORDER BY EmployeeNumber DESC) AS

rn

FROM EmployeeData
)

SELECT

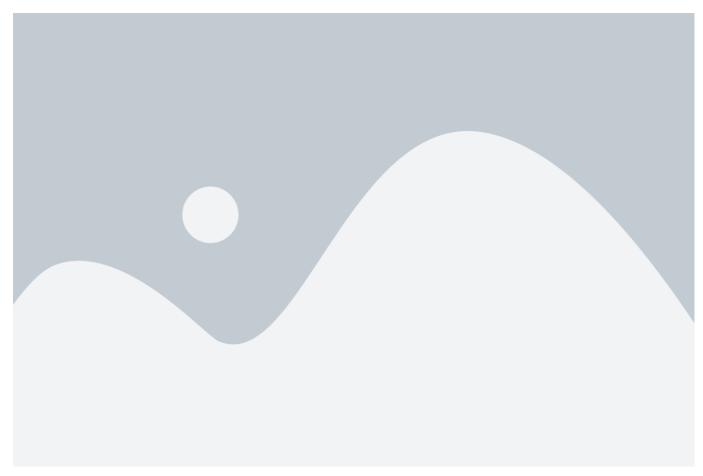
JobRole,

MonthlyIncome,

AVG(MonthlyIncome) OVER (PARTITION BY JobRole ORDER BY rn ROWS BETWEEN 2

PRECEDING AND CURRENT ROW) AS MovingAverage

FROM RankedEmployees;
```



#### ###-

```
WITH MovingAvg AS (

SELECT

JobRole,

EmployeeNumber,

MonthlyIncome,

AVG(MonthlyIncome) OVER (PARTITION BY JobRole ORDER BY EmployeeNumber ROWS

BETWEEN 2 PRECEDING AND CURRENT ROW) AS Moving_Avg_Income

FROM EmployeeData
)

SELECT

JobRole,

EmployeeNumber,

MonthlyIncome,

Moving_Avg_Income

FROM MovingAvg;
```



#### 16. Identify Employees with Outliers in Monthly Income Within Each Job Role

```
WITH IncomeStats AS (
    SELECT
        JobRole,
        EmployeeNumber,
        MonthlyIncome,
        PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY MonthlyIncome) OVER
(PARTITION BY JobRole) AS Q1,
        PERCENTILE CONT(0.75) WITHIN GROUP (ORDER BY MonthlyIncome) OVER
(PARTITION BY JobRole) AS Q3
    FROM EmployeeData
)
SELECT
    EmployeeNumber,
    JobRole,
    MonthlyIncome,
    CASE
        WHEN MonthlyIncome < Q1 - (Q3 - Q1) * 1.5 THEN 'Low'
        WHEN MonthlyIncome > Q3 + (Q3 - Q1) * 1.5 THEN 'High'
        ELSE 'Not an Outlier'
    END AS OutlierType
FROM IncomeStats
WHERE MonthlyIncome < Q1 - (Q3 - Q1) * 1.5
   OR MonthlyIncome > Q3 + (Q3 - Q1) * 1.5;
```



17. Gender Distribution Within Each Job Role, Show Each Job Role with Its Gender Domination

```
WITH GenderCount AS
(
SELECT
    JobRole,
    SUM(CASE WHEN Gender = 'Male' THEN 1 ELSE 0 END) AS Male_Count,
    SUM(CASE WHEN Gender = 'Female' THEN 1 ELSE 0 END) AS Female_Count
FROM EmployeeData
GROUP BY JobRole
)
SELECT
    CASE
        WHEN Male_count > Female_Count THEN 'Male'
        WHEN Female_Count > Male_count THEN 'Female'
        ELSE 'Equal'
    END AS Dominant_Gender
FROM
    GenderCount;
```



### 18. Percent Rank of Employees Based on Training Times Last Year

```
SELECT
EmployeeNumber,
TrainingTimesLastYear,
PERCENT_RANK() OVER (ORDER BY TrainingTimesLastYear) * 100 AS PercentRank
FROM EmployeeData;
```



## 19. Divide Employees into 5 Groups Based on Training Times Last Year

```
SELECT
EmployeeNumber,
TrainingTimesLastYear,
NTILE(5) OVER (ORDER BY TrainingTimesLastYear) AS Training_Group
FROM EmployeeData;
```



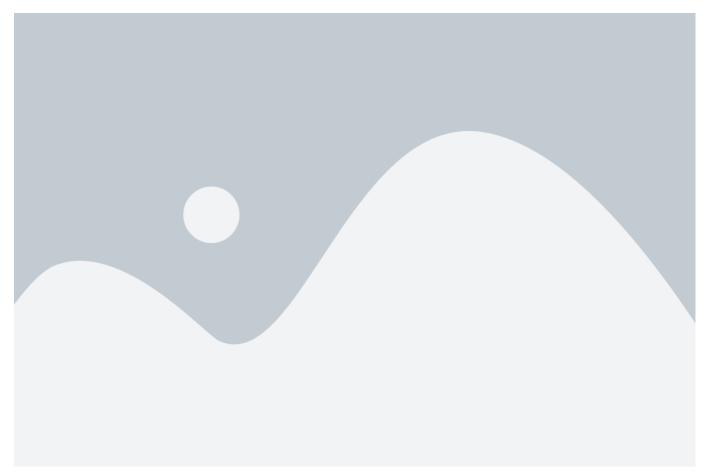
## 20. Categorize Employees Based on Training Times Last Year as - Frequent Trainee, Moderate Trainee, Infrequent Trainee

```
WITH TrainingTimeTiling AS
(
SELECT
    EmployeeNumber,
    TrainingTimesLastYear,
    NTILE(3) OVER (ORDER BY TrainingTimesLastYear) AS TrainingTile
FROM EmployeeData
)
SELECT
    EmployeeNumber,
    TrainingTimesLastYear,
        WHEN TrainingTile = 3 THEN 'Frequent Trainee'
        WHEN TrainingTile = 2 THEN 'Moderate Trainee'
        ELSE 'Infrequent Trainee'
    END AS TraineeCategory
FROM TrainingTimeTiling;
```



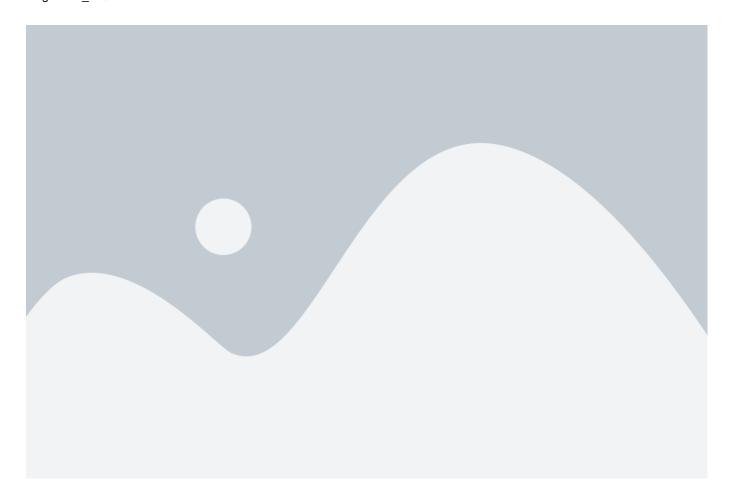
# 21. Categorize Employees as 'High', 'Medium', or 'Low' Performers Based on Their Performance Rating

```
SELECT
   EmployeeNumber,
   PerformanceRating,
   CASE
     WHEN PerformanceRating >= 4 THEN 'High'
     WHEN PerformanceRating = 3 THEN 'Medium'
     ELSE 'Low'
   END AS PerformanceCategory
FROM EmployeeData;
```



###-

```
WITH PerformanceCategories AS
(
SELECT
    EmployeeNumber,
    PerformanceRating,
    NTILE(3) OVER(ORDER BY PerformanceRating DESC) AS Tile
FROM EmployeeData
)
SELECT
    EmployeeNumber,
    PerformanceRating,
    CASE
        WHEN Tile = 1 THEN 'High'
        WHEN TIle = 2 THEN 'Medium'
        ELSE 'Low'
    END AS PerformanceCategory
FROM PerformanceCategories;
```



22. Use a CASE WHEN Statement to Categorize Employees into 'Poor', 'Fair', 'Good', or 'Excellent' Work-Life Balance Based on Their Work-Life Balance Score

```
SELECT

EmployeeNumber,
WorkLifeBalance,
CASE

WHEN WorkLifeBalance = 1 THEN 'Poor'
WHEN WorkLifeBalance = 2 THEN 'Fair'
WHEN WorkLifeBalance = 3 THEN 'Good'
ELSE 'Excellent'
END AS Work_Life_Balance_Category
FROM EmployeeData;
```



## 23. Group Employees into 3 Groups Based on Their Stock Option Level Using the [NTILE] Function

```
SELECT
EmployeeNumber,
StockOptionLevel,
NTILE(3) OVER (ORDER BY StockOptionLevel) AS StockOption_Group
FROM EmployeeData;
```



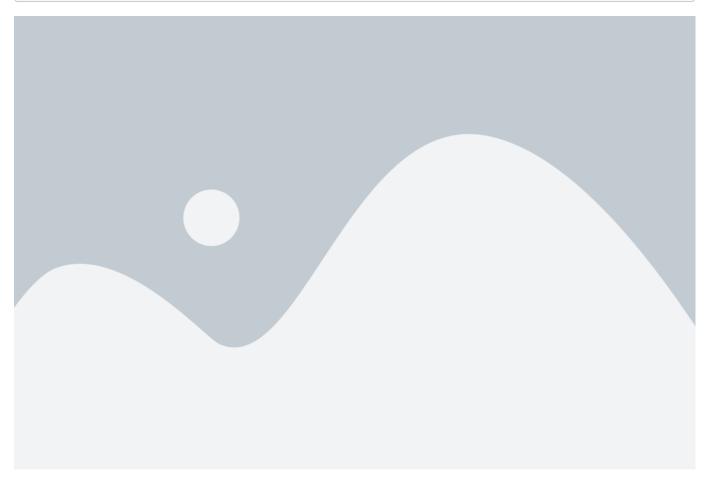
### 24. Find Key Reasons for Attrition in Company

```
SELECT
Attrition,
BusinessTravel,
Department,
MaritalStatus,
JobRole,
AVG(MonthlyIncome) AS AvgIncome,
COUNT(*) AS Count
FROM EmployeeData
WHERE Attrition = 'Yes'
GROUP BY Attrition, BusinessTravel, Department, MaritalStatus, JobRole
ORDER BY Count DESC;
```



```
WITH WorkingYears AS
(
SELECT
    *,
    CASE
        WHEN YearsAtCompany <= 5 THEN '0-5'
        WHEN YearsAtCompany <= 10 THEN '5-10'
        WHEN YearsAtCompany <= 15 THEN '10-15'
        WHEN YearsAtCompany <= 20 THEN '10-20'
        ELSE '0-5'
    END AS WorkingYearBand
FROM
    EmployeeData
)
SELECT
    Attrition,
    WorkingYearBand,
    COUNT(*) AS Total_Employees,
    AVG(Age) AS Avg_Age,
    AVG(MonthlyIncome) AS Avg_MonthlyIncome,
    AVG(WorkLifeBalance) AS Avg_WorkLifeBalance,
    AVG(JobSatisfaction) AS Avg_JobSatisfaction,
    AVG(EnvironmentSatisfaction) AS Avg_EnvironmentSatisfaction,
    AVG(JobInvolvement) AS Avg_JobInvolvement,
    AVG(PerformanceRating) AS Avg_PerformanceRating
FROM WorkingYears
WHERE Attrition = 'Yes'
```

```
GROUP BY Attrition, WorkingYearBand
ORDER BY Attrition, WorkingYearBand, Total_Employees DESC;
```



#### ###-

```
Attrition,
YearsAtCompany,
COUNT(*) AS Total_Employees,
AVG(Age) AS Avg_Age,
AVG(MonthlyIncome) AS Avg_MonthlyIncome,
AVG(WorkLifeBalance) AS Avg_WorkLifeBalance,
AVG(JobSatisfaction) AS Avg_JobSatisfaction,
AVG(EnvironmentSatisfaction) AS Avg_EnvironmentSatisfaction,
AVG(JobInvolvement) AS Avg_JobInvolvement,
AVG(PerformanceRating) AS Avg_PerformanceRating
FROM EmployeeData
WHERE Attrition = 'Yes'
GROUP BY Attrition, YearsAtCompany
ORDER BY Attrition, YearsAtCompany, Total_Employees DESC;
```

