

Assignment

HR Employee Dataset

Create db

```
DECLARE @DatabaseName VARCHAR(128) = 'employee_db';
-- Test condition to check is database exist
IF NOT EXISTS(SELECT 1 FROM sys.databases WHERE name = @DatabaseName)
--If the database doesn't exist
BEGIN
DECLARE @SQL NVARCHAR(MAX) = 'CREATE DATABASE '+ QUOTENAME(@DatabaseName);
EXEC sp_executesql @SQL;
END
USE employee_db;
CREATE TABLE EmployeeData (
Attrition VARCHAR(3),
Business_Travel VARCHAR(20),
CF_age_band VARCHAR(10),
CF_attrition_label VARCHAR(20),
Department VARCHAR(20),
Education_Field VARCHAR(20),
emp_no VARCHAR(10),
Employee_Number INT,
Gender VARCHAR(6),
Job_Role VARCHAR(30),
Marital_Status VARCHAR(10),
Over_Time VARCHAR(3),
Over18 VARCHAR(1),
Training_Times_Last_Year INT,
Age INT,
CF_current_Employee INT,
Daily_Rate INT,
Distance_From_Home INT,
Education VARCHAR(20),
Employee_Count INT,
Environment_Satisfaction INT,
Hourly_Rate INT,
Job_Involvement INT,
Job_Level INT,
Job_Satisfaction INT,
Monthly_Income INT,
Monthly_Rate INT,
Num_Companies_Worked INT,
```

```
Percent_Salary_Hike INT,
Performance_Rating INT,
Relationship_Satisfaction INT,
Standard_Hours INT,
Stock_Option_Level INT,
Total_Working_Years INT,
Work_Life_Balance INT,
Years_At_Company INT,
Years_In_Current_Role INT,
Years_Since_Last_Promotion INT,
Years_With_Curr_Manager INT
);
```

Insert Bulk Data

```
BULK INSERT employeeData FROM 'D:/CSV/HR_Employee.csv'
WITH
(
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n',
FIRSTROW = 2
);
```

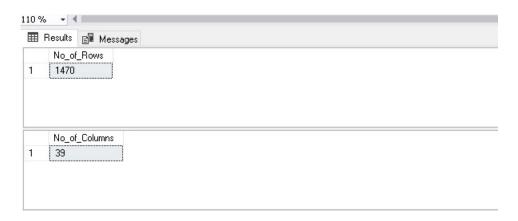


SELECT * FROM employeeData;

a) Return the shape of the table

```
SELECT
COUNT(*) AS No_of_Rows
FROM
employeeData;

SELECT
COUNT(*) AS No_of_Columns
FROM
INFORMATION_SCHEMA.columns;
```



```
Department,

Total_Working_Years,

SUM(Total_Working_Years) OVER(PARTITION BY Department

ORDER BY Total_Working_Years ROWS BETWEEN

UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_sum

FROM

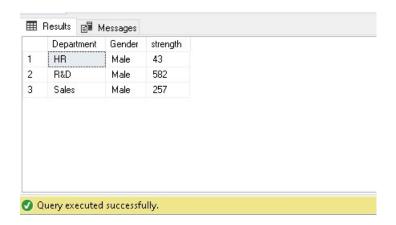
employeeData;
```

	Department	Total_Working_Years	cum_sum
58	HR	27	562
59	HR	30	592
60	HR	32	624
61	HR	33	657
52	HR	35	692
63	HR	36	728
64	R&D	0	0
65	R&D	0	0
66	R&D	0	0
67	R&D	0	0
68	R&D	0	0
69	R&D	0	0
70	R&D	0	0
71	R&D	1	1
70	DAD	4	-

c) Which gender have higher strength as workforce in each department

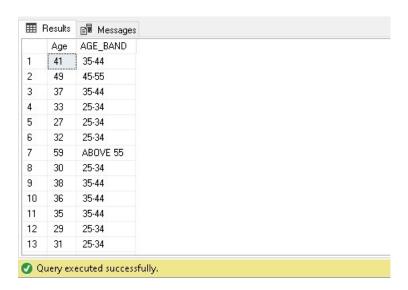
```
WITH GenderStrength AS (
    SELECT
        Department,
        Gender,
        COUNT(*) AS strength
    FROM
        EmployeeData
    GROUP BY
        Department,
        Gender
    ),
StrengthRankCTE AS(
SELECT
Department,
Gender,
strength,
DENSE_RANK() OVER(PARTITION BY Department ORDER BY strength) AS gender_rank
FROM GenderStrength
)
SELECT
Department,
Gender,
strength
FROM
StrengthRankCTE
WHERE
```

gender_rank = 2;



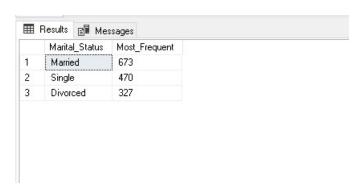
d) Create a new column AGE_BAND and Show Distribution of Employee's Age band group (Below 25, 25-34, 35-44, 45-55. ABOVE 55).

```
SELECT
Age,
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 AS AGE_BAND
FROM
EmployeeData;
```



e) Compare all marital status of employee and find the most frequent marital status

```
SELECT
Marital_Status,
COUNT(Marital_Status) AS Most_Frequent
FROM
EmployeeData
GROUP BY Marital_Status
ORDER BY COUNT(Marital_Status) DESC;
```

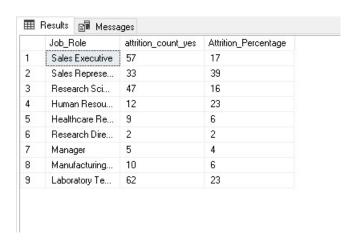




- Insight: The most frequent is Married

f) Show the Job Role with Highest Attrition Rate (Percentage)

```
SELECT
Job_Role,
SUM(
CASE
WHEN Attrition = 'Yes' THEN 1 ELSE 0
END) AS attrition_count_yes,
SUM(
CASE
WHEN Attrition = 'Yes' THEN 1 ELSE 0
END)*100/COUNT(Attrition) AS Attrition_Percentage
FROM
EmployeeData
GROUP BY
Job_Role;
```



promoted.

```
Years_Since_Last_Promotion,
Attrition,
AVG(Job_Involvement) AS Job_Involvement,
AVG(Performance_rating) AS avg_performance_rating,
AVG(Relationship_satisfaction) AS avg_relationship_satisfaction,
COUNT(*) AS no_of_employees
FROM
EmployeeData
GROUP BY Years_Since_Last_Promotion,Attrition
ORDER BY Years_Since_Last_Promotion ASC;
```

	Years_Since_Last_Promotion	Attrition	Job_Involvement	avg_performance_rating	avg_relationship_satisfaction	no_of_employees
1	0	No	2	3	2	471
2	0	Yes	2	3	2	110
3	1	No	2	3	2	308
4	1	Yes	2	3	2	49
5	2	No	2	3	2	132
6	2	Yes	2	3	2	27
7	3	No	2	3	2	43
8	3	Yes	2	3	2	9
9	4	No	2	3	2	56
10	4	Yes	3	3	1	5
11	5	No	2	3	2	43
12	5	Yes	2	3	3	2
13	6	No	2	3	2	26
14	6	Yes	2	3	3	6
15	7	No	2	3	2	60
16	7	Yes	2	3	2	16
17	8	No	2	3	2	18
18	9	No	2	3	2	13
19	9	Yes	2	3	3	4

• - insight : if attrition is no there is higher chance of getting promoted

h) Show the cumulative sum of total working years for each department.

```
SELECT
Department,
Total_Working_Years,
SUM(Total_Working_Years)
OVER(PARTITION BY Department ORDER BY Total_Working_Years ROWS
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
AS cum_sum
FROM
employeeData;
```

	Department	Total_Working_Years	cum_sum
1	HR	1	1
2	HR	1	2
3	HR	1	3
4	HR	1	4
5	HR	2	6
6	HR	2	8
7	HR	3	11
8	HR	3	14
9	HR	4	18
10	HR	4	22
11	HR	4	26
12	HR	5	31
13	HR	6	37
14	HR	6	43

i) Find the rank of employees within each department based on their monthly income

```
SELECT
Department,
Employee_Number,
Monthly_Income,
DENSE_RANK() OVER(PARTITION BY Department ORDER BY Monthly_Income DESC)
as Income_Rank
FROM
EmployeeData;
```

	Department	Employee_Number	Monthly_Income	Income_Rank
1	HR	1338	19717	1
2	HR	1625	19658	2
3	HR	1973	19636	3
4	HR	734	19189	4
5	HR	731	19141	5
6	HR	140	18844	6
7	HR	644	18200	7
8	HR	148	17328	8
9	HR	1408	16799	9
10	HR	1550	16437	10
11	HR	1352	14026	11
12	HR	698	10725	12
13	HR	1098	10482	13
14	HB	590	9950	14

j) Calculate the running total of 'Total Working Years' for each employee within each department and age band.

```
SELECT
Department,

CF_age_band,

Total_Working_Years,

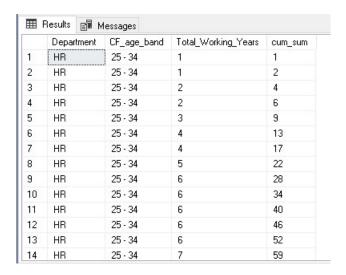
SUM(Total_Working_Years)

OVER(PARTITION BY Department, CF_age_band ORDER BY Total_Working_Years ROWS

BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

AS cum_sum

FROM
```



k) Foreach 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.

```
SELECT
e.Employee_Number,
e.Department,
e.Years_At_Company,
-- Average years of employees worked at each department mapping to the
-- main query department
(SELECT AVG(Years_At_Company)
    FROM EmployeeData WHERE Department = e.Department)
 AS Avg_of_department,
-- Find the difference between current working years and average
--working years in each department
ABS(e.Years_At_Company - (SELECT AVG(Years_At_Company)
        EmployeeData WHERE
Department = e.Department)) AS Comparison
FROM
    EmployeeData as e
WHERE
    CF_current_Employee = 0
ORDER BY
    Department;
```

	Employee_Number	Department	Years_At_Company	Avg_of_department	Comparison
1	133	HR	3	7	4
2	566	HR	1	7	6
3	590	HR	3	7	4
4	608	HR	7	7	0
5	1098	HR	20	7	13
6	1747	HR	4	7	3
7	1944	HR	1	7	6
8	1467	HR	2	7	5
9	1842	HR	1	7	6
10	1714	HR	1	7	6
11	1818	HR	5	7	2
12	1844	HR	2	7	5
13	2027	R&D	2	6	4
14	1604	R&D	1	6	5

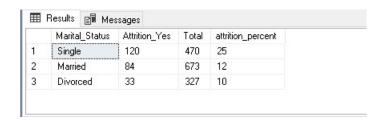
I) Rank the departments by the average monthly income of employees who have left

```
SELECT
Department,
AVG(Monthly_Income) AS Average_Monthly_Income,
DENSE_RANK() OVER(ORDER BY AVG(Monthly_Income)) AS EM_RANK
FROM
EmployeeData
WHERE
CF_current_Employee = 0
GROUP BY Department;
```

	Department	Average_Monthly_Income	EM_RANK
1	HR	3715	1
2	R&D	4108	2
3	Sales	5908	3

m) Find the if there is any relation between Attrition Rate and Marital Status of Employee.

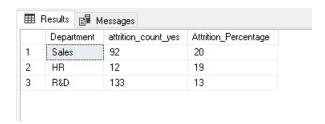
```
SELECT
   Marital_Status,
   SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END) AS Attrition_Yes,
   COUNT(*) AS Total,
   SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END)*100/COUNT(*)
   as attrition_percent
FROM
   EmployeeData
GROUP BY
   Marital_Status
ORDER BY attrition_percent DESC;
```





n) Show the Department with Highest Attrition Rate (Percentage)

```
SELECT
Department,
SUM(
CASE
WHEN Attrition = 'Yes' THEN 1 ELSE 0
END) AS attrition_count_yes,
SUM(
CASE
WHEN Attrition = 'Yes' THEN 1 ELSE 0
END)*100/COUNT(Attrition) AS Attrition_Percentage
FROM
EmployeeData
GROUP BY
Department;
ORDER BY
Attrition_Percentage DESC
```



o) Calculate the moving average of monthly income over the past 3 employees for each job role.

```
SELECT
Job_Role,
Employee_Number,
Monthly_Income,
AVG(Monthly_Income) OVER (PARTITION BY Job_Role ORDER BY Employee_Number
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS Moving_Avg_Income
FROM
EmployeeData
ORDER BY
Job_Role, Employee_Number;
```

	Job_Role	Employee_Number	Monthly_Income	Moving_Avg_Income
1	Healthcare Representative	13	5237	5237
2	Healthcare Representative	36	10248	7742
3	Healthcare Representative	40	6465	7316
4	Healthcare Representative	70	9884	8865
5	Healthcare Representative	83	10096	8815
6	Healthcare Representative	117	4152	8044
7	Healthcare Representative	119	13503	9250
8	Healthcare Representative	124	10673	9442
9	Healthcare Representative	139	5163	9779
10	Healthcare Representative	145	7484	7773
11	Healthcare Representative	165	10312	7653
12	Healthcare Representative	223	9439	9078
13	Healthcare Representative	258	13734	11161
14	Healthcare Representative	282	6673	9948

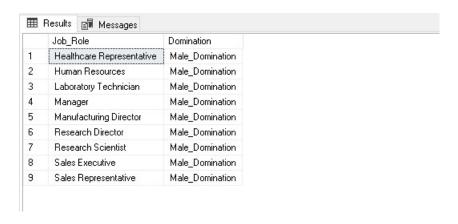
p) Identify employees with outliers in monthly income within each job role. [Condition: Monthly_Income < Q1 - (Q3 - Q1) * 1.5 OR Monthly_Income > Q3 + (Q3 - Q1)]

```
WITH Quartiles AS (
    SELECT
        Job_Role,
        Monthly_Income,
        PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY MONTHLY_INCOME)
        OVER (PARTITION BY job_role) AS Q1,
        PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY MONTHLY_INCOME)
        OVER (PARTITION BY job_role) AS Q3
    FROM
        EmployeeData
),
Outliers AS (
    SELECT
        job_role,
        Monthly_Income,
        Q1,
        Q3,
        (Q3 - Q1) AS IQR,
        CASE
            WHEN Monthly_Income < Q1 - (1.5 * (Q3 - Q1)) THEN Monthly_Income
            WHEN Monthly_Income > Q3 + (1.5 * (Q3 - Q1)) THEN Monthly_Income
            ELSE NULL
        END AS Outlier_Value
    FROM
        Quartiles
)
SELECT
    job_role,
    Outlier_Value
FROM
    Outliers
WHERE
    Outlier_Value >=0;
```

	Job_Role	Monthly_Income
1	Laboratory Technician	
2	Laboratory Technician	6472
3	Laboratory Technician	6674
4	Laboratory Technician	6782
5	Laboratory Technician	7403
6	Manager	11244
7	Manager	11557
8	Manager	11631
9	Manager	11849
10	Manager	11878
11	Manager	11904
12	Manager	11916
13	Manager	11996
14	Research Scientist	6220
15	Research Scientist	6322
16	Research Scientist	6545
17	Research Scientist	6646

q) Gender distribution within each job role, show each job role with its gender domination. [Male_Domination or Female_Domination]

```
WITH GenderCTE AS(
SELECT
Job_Role,
Gender,
COUNT(Gender) AS Gender_count,
DENSE_RANK() OVER(PARTITION BY Job_Role ORDER BY Gender) AS gender_rank
FROM
EmployeeData
GROUP BY
Job_Role, Gender
)
SELECT
Job_Role,
CASE
    WHEN gender_rank = 2 THEN 'Male_Domination'
    ELSE 'Female_Domination'
END AS Domination
FROM
GenderCTE
WHERE
gender_rank = 2
ORDER BY
Job_Role, Gender ASC;
```

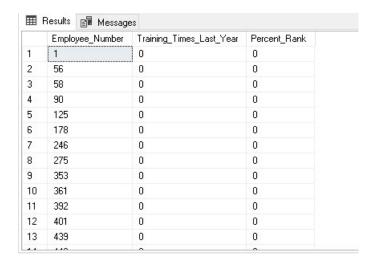




Insights: Males are the most dominant in Each department

r) Percent rank of employees based on training times last year

```
SELECT
Employee_Number,
Training_Times_Last_Year,
PERCENT_RANK() OVER (ORDER BY Training_Times_Last_Year) AS Percent_Rank
FROM
EmployeeData
ORDER BY
Percent_Rank;
```





Insight: The person with most training in last year has the higher percent rank

s) Divide employees into 5 groups based on training times last year [Use NTILE ()]

```
WITH groupCTE AS(
SELECT
Employee_Number,
Training_Times_Last_Year,
NTILE(5) OVER(ORDER BY Training_Times_Last_Year) AS group_tile
FROM
EmployeeData
```

```
SELECT
Employee_Number,
Training_Times_Last_Year,
CASE
WHEN group_tile = 1 THEN 'group 1'
WHEN group_tile = 2 THEN 'group 2'
WHEN group_tile = 3 THEN 'group 3'
WHEN group_tile = 4 THEN 'group 4'
ELSE 'group 5'
END AS Employee_Training_Group
FROM
groupCTE;
```

	Employee_Number	Training_Times_Last_Year	Employee_Training_Group
1	1	0	group 1
2	56	0	group 1
3	58	0	group 1
4	90	0	group 1
5	125	0	group 1
6	178	0	group 1
7	246	0	group 1
8	275	0	group 1
9	353	0	group 1
10	361	0	group 1
11	392	0	group 1
12	401	0	group 1
13	439	0	group 1
	440		

t) Categorize employees based on training times last year as - Frequent Trainee, Moderate Trainee, Infrequent Trainee

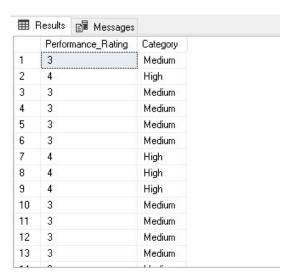
```
WITH TrainingStats AS (
SELECT
Employee_Number,
Training_Times_Last_Year,
PERCENT_RANK() OVER (ORDER BY Training_Times_Last_Year ) AS Percent_Rank
FROM
EmployeeData
)
SELECT
Employee_Number,
Training_Times_Last_Year,
CASE
WHEN Percent_Rank >= 0.75 THEN 'Frequent Trainee'
WHEN Percent_Rank >= 0.50 THEN 'Moderate Trainee'
ELSE 'Infrequent Trainee'
END AS Trainee_Category
FROM
TrainingStats
ORDER BY
Training_Times_Last_Year;
```

3339	sults 📑 Messagi -		T . O.
	Employee_Number	Training_Times_Last_Year	Trainee_Category
1	1	0	Infrequent Trainee
2	56	0	Infrequent Trainee
3	58	0	Infrequent Trainee
4	90	0	Infrequent Trainee
5	125	0	Infrequent Trainee
6	178	0	Infrequent Trainee
7	246	0	Infrequent Trainee
8	275	0	Infrequent Trainee
9	353	0	Infrequent Trainee
10	361	0	Infrequent Trainee
11	392	0	Infrequent Trainee
12	401	0	Infrequent Trainee
13	439	0	Infrequent Trainee
	440	_	

u) Categorize employees as 'High', 'Medium', or 'Low' performers based on their performance rating, using a CASE WHEN statement.

```
SELECT
PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Performance_Rating) OVER () AS _
FROM
EmployeeData

SELECT
Performance_Rating,
CASE
WHEN Performance_Rating >3 THEN 'High'
WHEN Performance_Rating <3 THEN 'low'
ELSE 'Medium'
END AS Category
FROM
EmployeeData
```

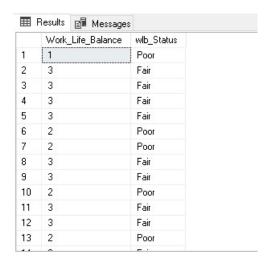


v) Use a CASE WHEN statement to categorize employees into 'Poor', 'Fair', 'Good', or 'Excellent' work-life balance based on their work-life balance score.

SELECT
DISTINCT
Work_Life_Balance

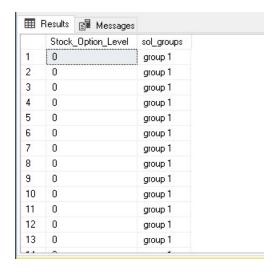
```
FROM
EmployeeData

SELECT
Work_Life_Balance,
CASE
WHEN Work_Life_Balance > 4 THEN 'Excellent'
WHEN Work_Life_Balance > 3 THEN 'Good'
WHEN Work_Life_Balance > 2 THEN 'Fair'
ELSE 'Poor'
END AS wlb_Status
FROM
EmployeeData;
```



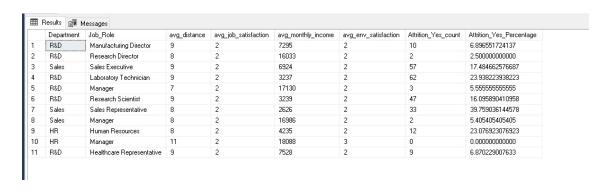
w) Group employees into 3 groups based on their stock option level using the [NTILE] function.

```
SELECT
Stock_Option_Level,
CASE
WHEN SOL_dist = 1 THEN 'group 1'
WHEN SOL_dist = 2 THEN 'group 2'
ELSE 'group 3'
END As sol_groups
FROM(
SELECT
Stock_Option_Level,
NTILE(3) OVER(ORDER BY Stock_Option_Level) AS SOL_dist
FROM
EmployeeData) AS _
```



x) Find key reasons for Attrition in Company

```
SELECT
Department,
Job_Role,
AVG(Distance_From_Home) AS avg_distance,
AVG(Job_Satisfaction) AS avg_job_satisfaction,
AVG(Monthly_Income) AS avg_monthly_income,
AVG(Environment_Satisfaction) AS avg_env_satisfaction,
SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END) AS Attrition_Yes_count,
(SUM(CASE WHEN Attrition = 'Yes' THEN 1 ELSE 0 END)*100.0/COUNT(Attrition))
AS Attrition_Yes_Percentage
FROM
EmployeeData
GROUP BY Department, Job_Role;
```





Insight: From the above query, It has came to know that the "avg_monthly_income" is affecting the attrition. More the average salary in each department lesser the attrition rate.