

Employee Dataset SQL Analysis on Azure by Rohit Raman

"CTE, Ranking Functions, and Aggregation in SQL: Employee Dataset Analysis"

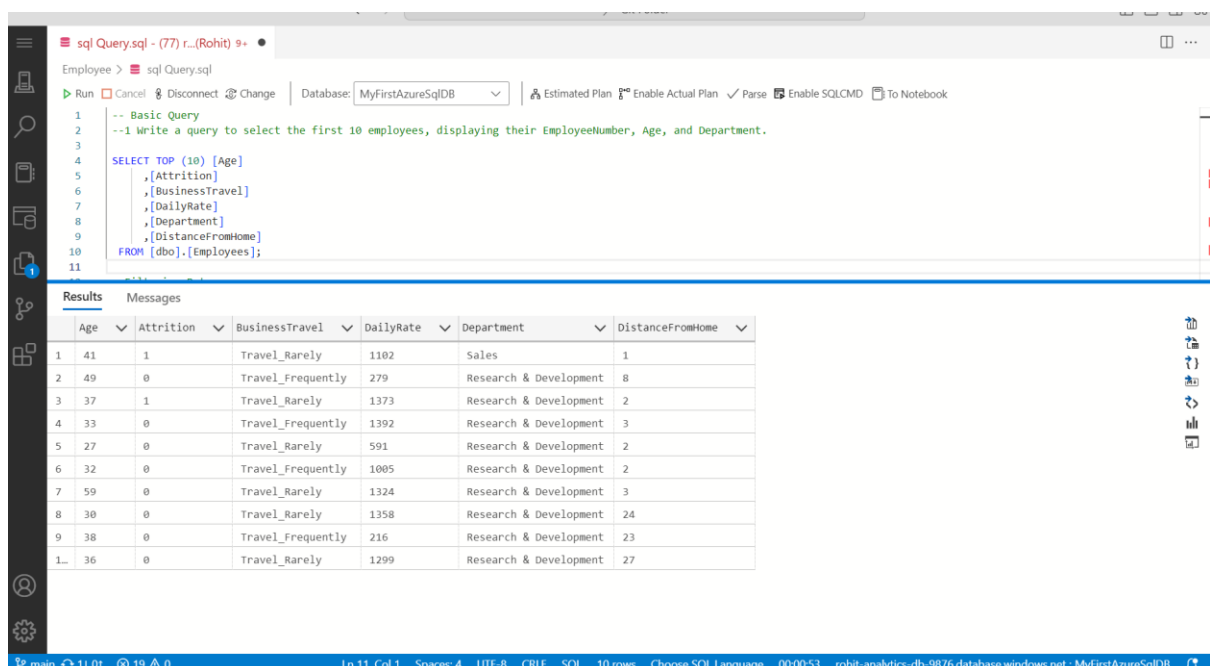
Introduction

This project focuses on analyzing employee data using SQL on Azure platform to extract insights on salary distribution, department composition, attrition rates, and employee rankings. The analysis is structured into fundamental SQL operations such as **basic queries, filtering, sorting, grouping, and aggregation**, along with more advanced **Common Table Expressions (CTEs) and ranking functions**.

Key analyses performed include retrieving specific employee details, filtering records based on conditions (e.g., employees older than 40 or those working in Sales), and sorting employees by salary. Aggregation techniques such as **COUNT (), AVG (), MIN (), and MAX ()** were used to measure attrition, calculate average department salaries, and determine the youngest and oldest employees per department.

CTEs were utilized for more complex queries, including identifying employees with over **10 years of experience and above-average salaries**, ranking employees by income within departments, and calculating department-wise employee counts. Window functions like **RANK (), DENSE_RANK (), and ROW_NUMBER ()** were employed to rank employees based on income and experience, as well as to determine the second-highest salary and top earners per department. **PARTITION BY** was used to compare salaries within departments efficiently.

1. Basic Query: Select the first 10 employees, displaying their EmployeeNumber, Age, and Department.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window with the following SQL code:

```
-- Basic Query
--1 Write a query to select the first 10 employees, displaying their EmployeeNumber, Age, and Department.

SELECT TOP (10) [Age]
, [Attrition]
, [BusinessTravel]
, [DailyRate]
, [Department]
, [DistanceFromHome]
FROM [dbo].[Employees];
```

The bottom pane shows the results of the query, displaying a table with 10 rows and 7 columns: Age, Attrition, BusinessTravel, DailyRate, Department, DistanceFromHome, and EmployeeNumber. The results are as follows:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome
1	41	1	Travel_Rarely	1102	Sales	1
2	49	0	Travel_Frequently	279	Research & Development	8
3	37	1	Travel_Rarely	1373	Research & Development	2
4	33	0	Travel_Frequently	1392	Research & Development	3
5	27	0	Travel_Rarely	591	Research & Development	2
6	32	0	Travel_Frequently	1005	Research & Development	2
7	59	0	Travel_Rarely	1324	Research & Development	3
8	30	0	Travel_Rarely	1358	Research & Development	24
9	38	0	Travel_Frequently	216	Research & Development	23
10	36	0	Travel_Rarely	1299	Research & Development	27

2. Filtering Data: Retrieve all employees who work in the "Sales" department.

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- Filtering Data
--2 Retrieve all employees who work in the "Sales" department.
Select * from [dbo].[Employees] where Department = 'Sales';
```

The results pane shows a table with 17 rows and 11 columns. The columns are: Age, Attrition, BusinessTravel, DailyRate, Department, DistanceFromHome, Education, EducationField, EmployeeCount, and EmployeeNumber. The data is filtered to show only employees in the 'Sales' department.

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
1	41	1	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1
2	53	0	Travel_Rarely	1219	Sales	2	4	Life Sciences	1	23
3	36	1	Travel_Rarely	1218	Sales	9	4	Life Sciences	1	27
4	42	0	Travel_Rarely	691	Sales	8	4	Marketing	1	35
5	46	0	Travel_Rarely	705	Sales	2	4	Marketing	1	38
6	39	1	Travel_Rarely	895	Sales	5	3	Technical Degree	1	42
7	50	1	Travel_Rarely	869	Sales	3	2	Marketing	1	47
8	35	0	Travel_Rarely	890	Sales	2	3	Marketing	1	49
9	33	0	Travel_Frequently	1141	Sales	1	3	Life Sciences	1	52
10	27	0	Travel_Frequently	994	Sales	8	3	Life Sciences	1	56
11	34	0	Non-Travel	1065	Sales	23	4	Marketing	1	60
12	46	0	Travel_Frequently	1211	Sales	5	4	Marketing	1	62
13	44	0	Travel_Rarely	1488	Sales	1	5	Marketing	1	68
14	26	0	Travel_Rarely	1443	Sales	23	3	Marketing	1	72
15	35	0	Travel_Frequently	853	Sales	18	5	Life Sciences	1	74
16	59	0	Travel_Rarely	1435	Sales	25	3	Life Sciences	1	81
17	59	0	Travel_Frequently	1225	Sales	1	1	Life Sciences	1	91

3. Sorting Data: Get the top 10 employees with the highest MonthlyIncome, sorted in descending order.

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- Filtering Data
--2 Retrieve all employees who work in the "Sales" department.
Select * from [dbo].[Employees] where Department = 'Sales';

-- Sorting Data
--3 Get the top 10 employees with the highest MonthlyIncome, sorted in descending order.
Select Top 10* from [dbo].[Employees] order by MonthlyIncome desc;
```

The results pane shows a table with 10 rows and 11 columns. The columns are: Age, Attrition, BusinessTravel, DailyRate, Department, DistanceFromHome, Education, EducationField, EmployeeCount, and EmployeeNumber. The data is sorted by MonthlyIncome in descending order.

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
1	52	0	Travel_Rarely	699	Research & Development	1	4	Life Sciences	1	259
2	41	0	Non-Travel	247	Research & Development	7	1	Life Sciences	1	1035
3	56	0	Travel_Rarely	718	Research & Development	4	4	Technical Degree	1	1191
4	50	0	Travel_Rarely	1452	Research & Development	11	3	Life Sciences	1	226
5	55	1	Travel_Rarely	725	Research & Development	2	3	Medical	1	787
6	51	0	Travel_Frequently	237	Sales	9	3	Life Sciences	1	1282
7	52	1	Travel_Rarely	266	Sales	2	1	Marketing	1	1038
8	40	0	Travel_Rarely	611	Sales	7	4	Medical	1	1740
9	43	0	Travel_Rarely	920	Research & Development	3	3	Life Sciences	1	1255
10	56	0	Travel_Rarely	206	Human Resources	8	4	Life Sciences	1	1338

4. Using WHERE Condition: Retrieve EmployeeNumber, Age, and Department of employees older than 40.

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- Using WHERE Condition:
--4 .Retrieve only the EmployeeNumber, Age, and Department of employees who are older than 40.
Select EmployeeNumber, Age, Department From [dbo].[Employees] where Age > 40;
```

The results pane shows a table with 17 rows of data:

	EmployeeNumber	Age	Department
1	1	41	Sales
2	2	49	Research & Development
3	10	59	Research & Development
4	23	53	Sales
5	32	53	Research & Development
6	35	42	Sales
7	36	44	Research & Development
8	38	46	Sales
9	40	44	Research & Development
10	46	43	Research & Development
11	47	50	Sales
12	58	41	Research & Development
13	62	46	Sales
14	64	48	Research & Development
15	68	44	Sales
16	80	50	Research & Development
17	81	59	Sales

5. Combining Conditions: Get employees in "Research & Development" earning more than 5000 per month.

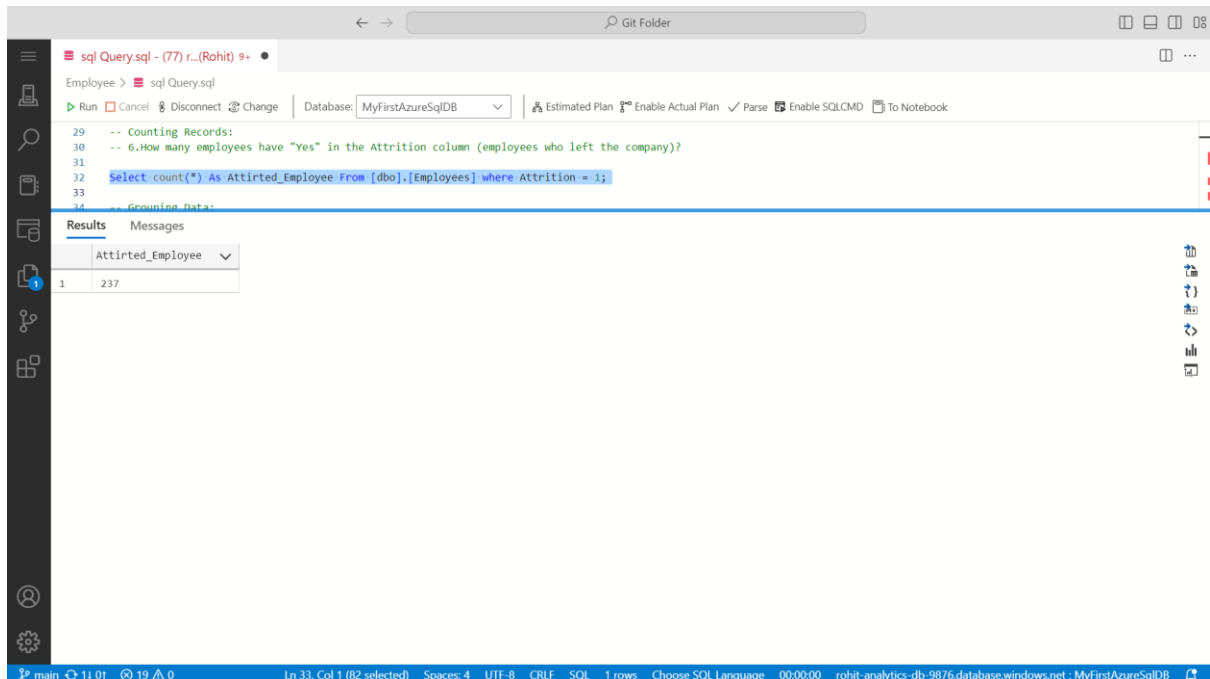
The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- Combining Conditions:
-- 5. Get a list of employees who are in the "Research & Development" department and earn more than 5000 per month.
Select EmployeeNumber, Age, Department, MonthlyIncome from [dbo].[Employees] where Department = 'Research & Development' and MonthlyIncome > 5000;
```

The results pane shows a table with 18 rows of data:

	EmployeeNumber	Age	Department	MonthlyIncome
1	2	49	Research & Development	5130
2	12	38	Research & Development	9526
3	13	36	Research & Development	5237
4	20	29	Research & Development	9980
5	28	34	Research & Development	11994
6	32	53	Research & Development	19094
7	36	44	Research & Development	10248
8	40	44	Research & Development	6465
9	58	41	Research & Development	19545
10	64	48	Research & Development	5381
11	70	35	Research & Development	9884
12	73	33	Research & Development	13458
13	76	31	Research & Development	5915
14	77	37	Research & Development	5993
15	78	32	Research & Development	6162
16	80	50	Research & Development	18740
17	83	36	Research & Development	10096
18	84	55	Research & Development	14756

6. Counting Records: Count employees who left the company (Attrition = "Yes").



The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

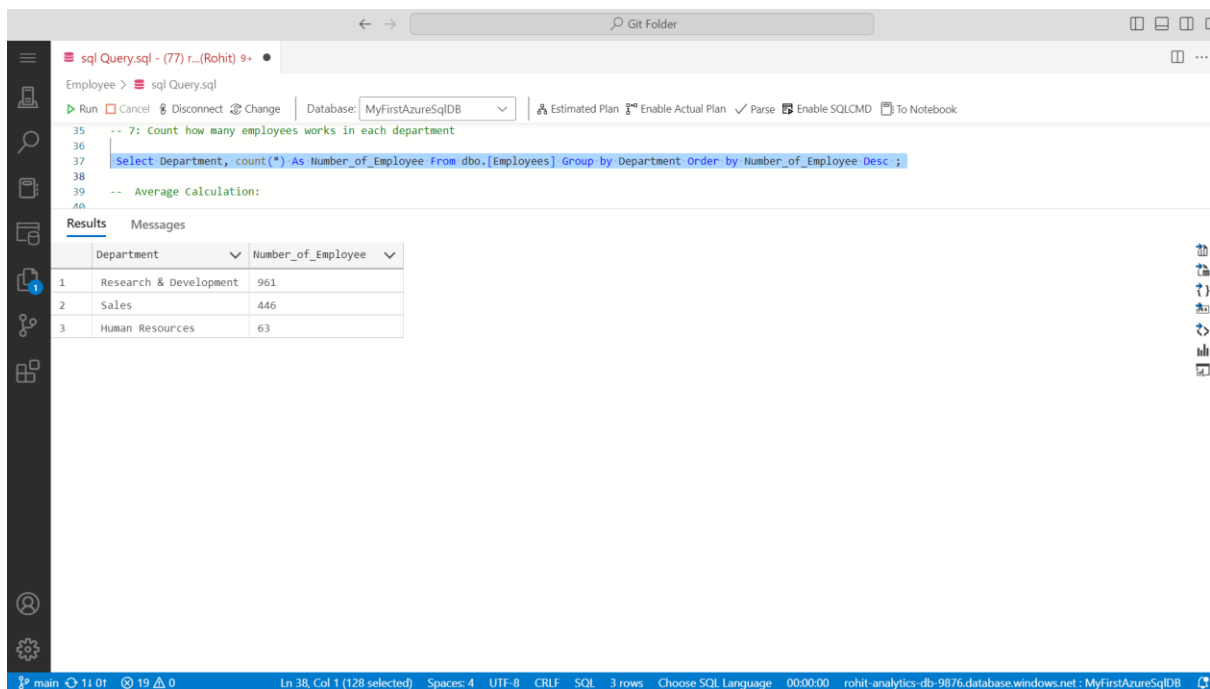
```
-- Counting Records:
-- 6.How many employees have "Yes" in the Attrition column (employees who left the company)?
Select count(*) As Attirted_Employee From [dbo].[Employees] where Attrition = 'Yes';
```

The query has been executed, and the results are displayed in the Results pane:

	Attirted_Employee
1	237

The status bar at the bottom indicates: Ln 33, Col 1 (82 selected) Spaces: 4 UTF-8 CRLF SQL 1 rows Choose SQL Language 00:00:00 rohit-analytics-db-9876.database.windows.net : MyFirstAzureSqlDB

7. Grouping Data: Count employees in each department.



The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

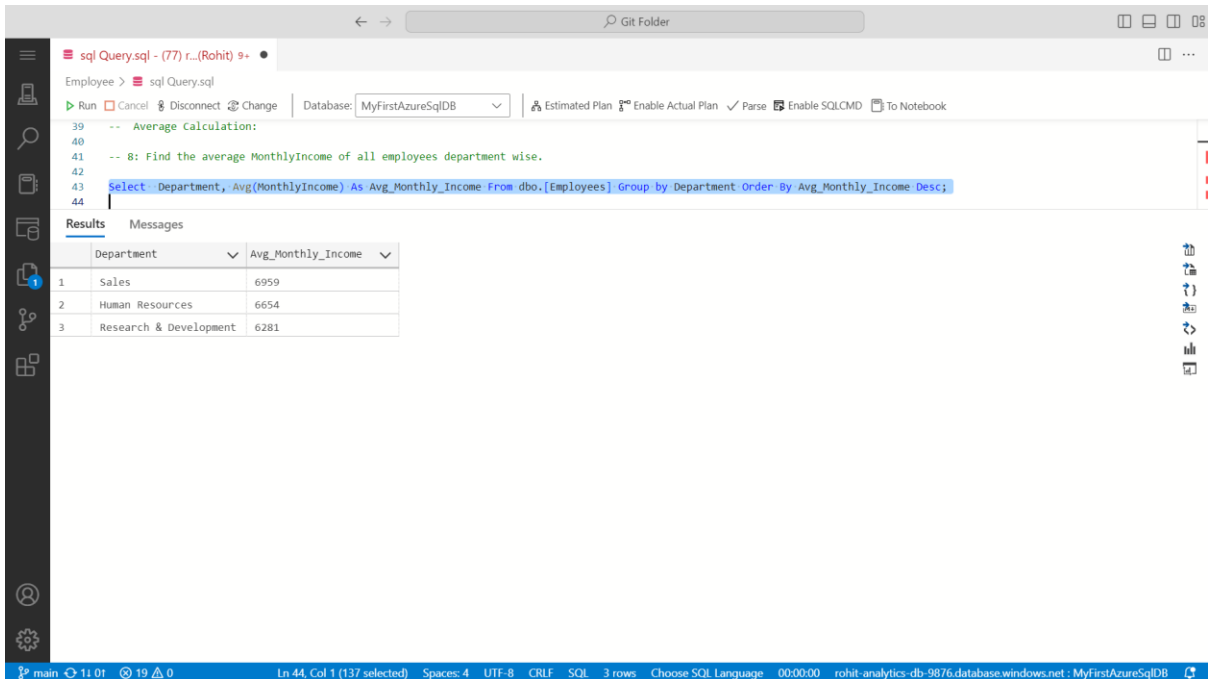
```
-- 7: Count how many employees works in each department
Select Department, count(*) As Number_of_Employee From [dbo].[Employees] Group by Department Order by Number_of_Employee Desc ;
```

The query has been executed, and the results are displayed in the Results pane:

	Department	Number_of_Employee
1	Research & Development	961
2	Sales	446
3	Human Resources	63

The status bar at the bottom indicates: Ln 38, Col 1 (128 selected) Spaces: 4 UTF-8 CRLF SQL 3 rows Choose SQL Language 00:00:00 rohit-analytics-db-9876.database.windows.net : MyFirstAzureSqlDB

8. Average Calculation: Find the average MonthlyIncome by department.



The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

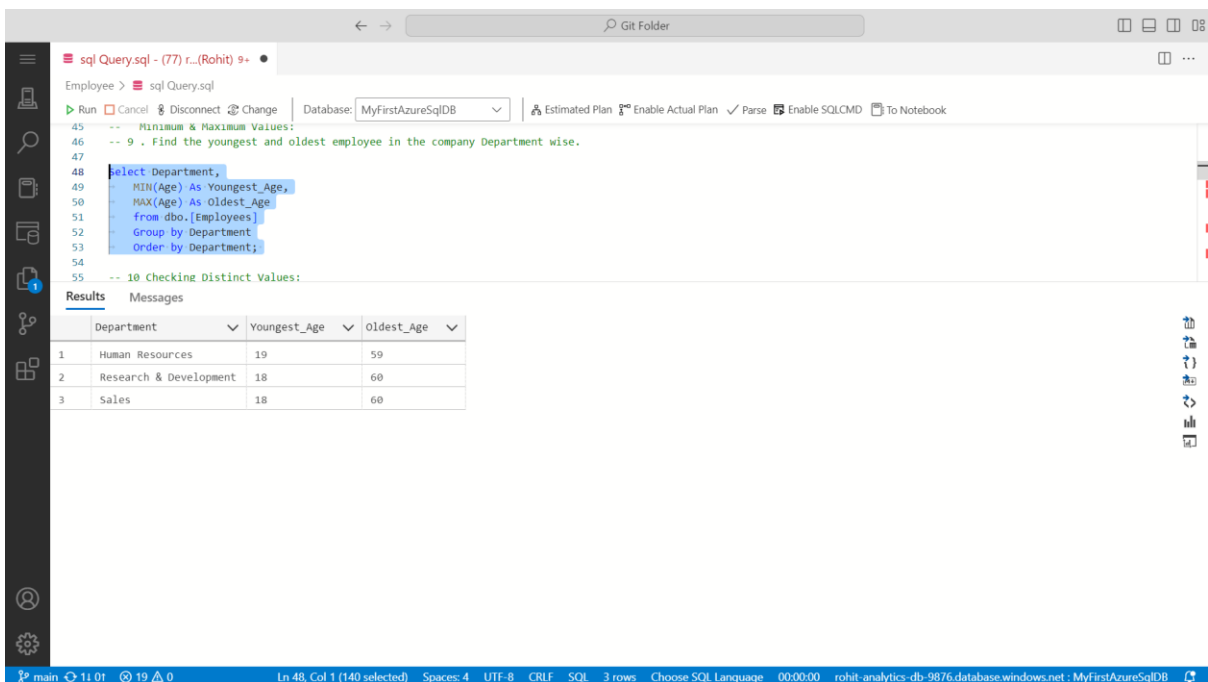
```
-- Average Calculation:
-- 8: Find the average MonthlyIncome of all employees department wise.
Select Department, Avg(MonthlyIncome) As Avg_Monthly_Income From dbo.[Employees] Group by Department Order By Avg_Monthly_Income Desc;
```

The Results tab shows the following data:

	Department	Avg_Monthly_Income
1	Sales	6959
2	Human Resources	6654
3	Research & Development	6281

The status bar at the bottom indicates: main 11:01 19 0 Ln 44, Col 1 (137 selected) Spaces: 4 UTF-8 CRLF SQL 3 rows Choose SQL Language 00:00:00 rohit-analytics-db-9876.database.windows.net : MyFirstAzureSqlDB

9. Minimum & Maximum Values: Find the youngest and oldest employee in each department.



The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- Minimum & Maximum Values:
-- 9 . Find the youngest and oldest employee in the company Department wise.
Select Department,
MIN(Age) As Youngest_Age,
MAX(Age) As Oldest_Age,
from dbo.[Employees]
Group by Department
Order by Department;
```

The Results tab shows the following data:

	Department	Youngest_Age	Oldest_Age
1	Human Resources	19	59
2	Research & Development	18	60
3	Sales	18	60

The status bar at the bottom indicates: main 11:01 19 0 Ln 48, Col 1 (140 selected) Spaces: 4 UTF-8 CRLF SQL 3 rows Choose SQL Language 00:00:00 rohit-analytics-db-9876.database.windows.net : MyFirstAzureSqlDB

10. Checking Distinct Values: Count and list unique job roles in the dataset.

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- 10 Checking Distinct Values:
-- How many unique job roles are there in the dataset and what are those
SELECT Count(Distinct(Department)) as Distinct_Department FROM dbo.[Employees]
SELECT Distinct(Department) as Distinct_Department FROM dbo.[Employees]
```

The Results pane shows the output of the second query, listing the distinct departments:

Distinct_Department
1 Sales
2 Research & Development
3 Human Resources

11. CTE - Salary & Experience: Find employees with over 10 years of experience earning above the average salary.

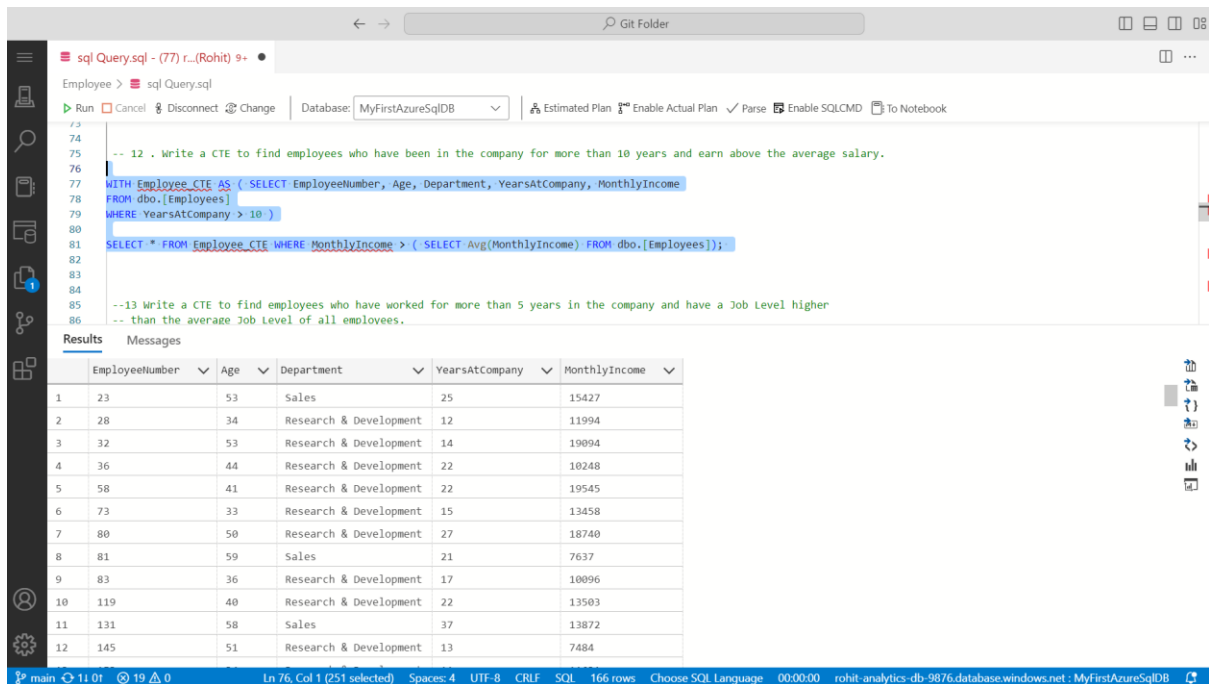
The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
--- Common table Expressions:
-- 11. Write a CTE to find employees who have been in the company for more than 10 years and earn above the average salary.
With Employee_CTE AS ( Select EmployeeNumber, Age, Department, YearsAtCompany, MonthlyIncome
from dbo.[Employees]
where YearsAtCompany > 10 )
select * from Employee_CTE where MonthlyIncome > ( select Avg(MonthlyIncome) from dbo.[Employees]);
select * from Employees
```

The Results pane shows the output of the CTE query, listing employees with over 10 years of experience and above-average salary:

EmployeeNumber	Age	Department	YearsAtCompany	MonthlyIncome
1	23	Sales	25	15427
2	28	Research & Development	12	11994
3	32	Research & Development	14	19094
4	36	Research & Development	22	10248
5	58	Research & Development	22	19545
6	73	Research & Development	15	13458
7	80	Research & Development	27	18740
8	81	Sales	21	7637
9	83	Research & Development	17	10096
10	119	Research & Development	22	13503

12. CTE - Experience & Salary: Repeat the above with a refined query.

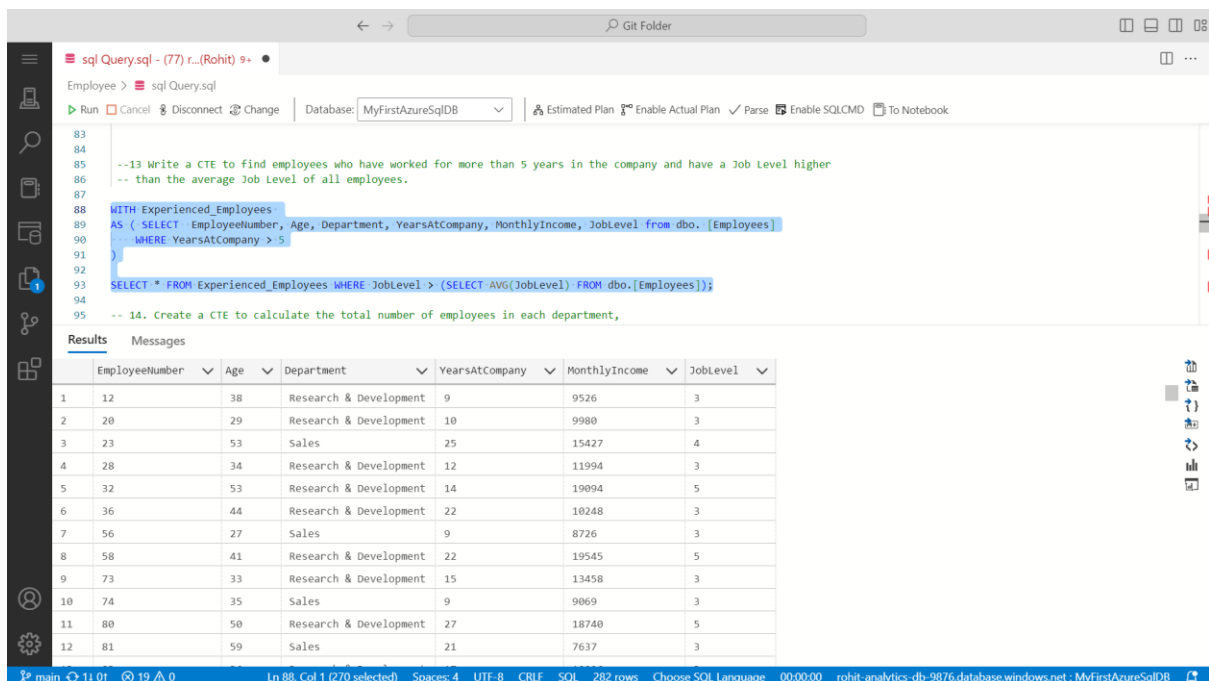


The screenshot shows a SQL query window in SQL Server Enterprise Manager. The query is a CTE that filters employees based on their years at the company and their monthly income relative to the average. The results table shows 12 rows of employee data.

```
-- 12. Write a CTE to find employees who have been in the company for more than 10 years and earn above the average salary.
WITH Employee_CTE AS ( SELECT EmployeeNumber, Age, Department, YearsAtCompany, MonthlyIncome
FROM dbo.[Employees]
WHERE YearsAtCompany > 10 )
SELECT * FROM Employee_CTE WHERE MonthlyIncome > ( SELECT Avg(MonthlyIncome) FROM dbo.[Employees]);
```

	EmployeeNumber	Age	Department	YearsAtCompany	MonthlyIncome
1	23	53	Sales	25	15427
2	28	34	Research & Development	12	11994
3	32	53	Research & Development	14	19094
4	36	44	Research & Development	22	10248
5	58	41	Research & Development	22	19545
6	73	33	Research & Development	15	13458
7	80	50	Research & Development	27	18740
8	81	59	Sales	21	7637
9	83	36	Research & Development	17	10096
10	119	40	Research & Development	22	13503
11	131	58	Sales	37	13872
12	145	51	Research & Development	13	7484

13. CTE - Job Level: Find employees with more than 5 years of experience and a Job Level higher than the average.



The screenshot shows a SQL query window in SQL Server Enterprise Manager. The query is a CTE that filters employees based on their years at the company and their job level relative to the average. The results table shows 12 rows of employee data.

```
-- 13 Write a CTE to find employees who have worked for more than 5 years in the company and have a Job Level higher
-- than the average Job Level of all employees.
WITH Experienced_Employees
AS ( SELECT EmployeeNumber, Age, Department, YearsAtCompany, MonthlyIncome, JobLevel from dbo. [Employees]
WHERE YearsAtCompany > 5 )
SELECT * FROM Experienced_Employees WHERE JobLevel > (SELECT AVG(JobLevel) FROM dbo.[Employees]);
```

	EmployeeNumber	Age	Department	YearsAtCompany	MonthlyIncome	JobLevel
1	12	38	Research & Development	9	9526	3
2	20	29	Research & Development	10	9980	3
3	23	53	Sales	25	15427	4
4	28	34	Research & Development	12	11994	3
5	32	53	Research & Development	14	19094	5
6	36	44	Research & Development	22	10248	3
7	56	27	Sales	9	8726	3
8	58	41	Research & Development	22	19545	5
9	73	33	Research & Development	15	13458	3
10	74	35	Sales	9	9069	3
11	80	50	Research & Development	27	18740	5
12	81	59	Sales	21	7637	3

14. CTE - Employee Count: Find departments with more than 500 employees.

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- 14. Create a CTE to calculate the total number of employees in each department,
-- then select only those departments where employee count is more than 500.

WITH Total_Employees AS (SELECT Count(*) AS TotEmployee, Department FROM dbo.[Employees] GROUP BY Department)
SELECT * FROM Total_Employees WHERE TotEmployee > 500;
```

The Results pane shows the following data:

	TotEmployee	Department
1	961	Research & Development

The status bar at the bottom indicates: Ln 99, Col 1 (167 selected) Spaces: 4 UTF-8 CRLF SQL 1 rows Choose SQL Language 00:00:00 rohit-analytics-db-9876.database.windows.net : MyFirstAzureSqlDB

15. CTE - Ranking Departments: Rank departments by total employees in descending order.

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```
-- 15 Create a CTE that calculates the total employees in each department,
-- then assign a RANK() based on the number of employees in descending order.

WITH Total_Emp AS (SELECT Department, Count(*) AS Num_of_Emp FROM dbo.[Employees] GROUP BY Department)
SELECT Department, Num_of_Emp, RANK() OVER(ORDER BY Num_of_Emp DESC) AS Rank_of_Department FROM Total_Emp;
```

The Results pane shows the following data:

	Department	Num_of_Emp	Rank_of_Department
1	Research & Development	961	1
2	Sales	446	2
3	Human Resources	63	3

The status bar at the bottom indicates: Ln 107, Col 1 (217 selected) Spaces: 4 UTF-8 CRLF SQL 3 rows Choose SQL Language 00:00:00 rohit-analytics-db-9876.database.windows.net : MyFirstAzureSqlDB

16. PARTITION BY - Income: Count employees earning above their department's average salary.

The screenshot shows a SQL query in the 'Employee' database, 'MyFirstAzureSqlDB'. The query is designed to count employees whose monthly income is greater than the average monthly income of their department, using the PARTITION BY clause.

```
--16 Retrieve Total employee whose MonthlyIncome is more than the average MonthlyIncome of their department using PARTITION BY.

WITH Department_Income AS (
    SELECT EmployeeNumber, Department, MonthlyIncome, AVG(MonthlyIncome)
    OVER (PARTITION BY Department )AS AvgDepartmentIncome FROM dbo.[Employees])
SELECT Department , count(*) AS Total_Employee_with_Handsome_salary
FROM Department_Income WHERE MonthlyIncome > AvgDepartmentIncome GROUP BY Department;
```

The results table shows the following data:

Department	Total_Employee_with_Handsome_salary
1 Human Resources	17
2 Research & Development	306
3 Sales	160

17. RANK() - Top Earners: Rank the top 5 highest-paid employees in each department.

The screenshot shows a SQL query in the 'Employee' database, 'MyFirstAzureSqlDB'. The query uses the RANK() function to assign a rank to employees based on their monthly income in descending order for each department, and then filters for the top 5 employees in each department.

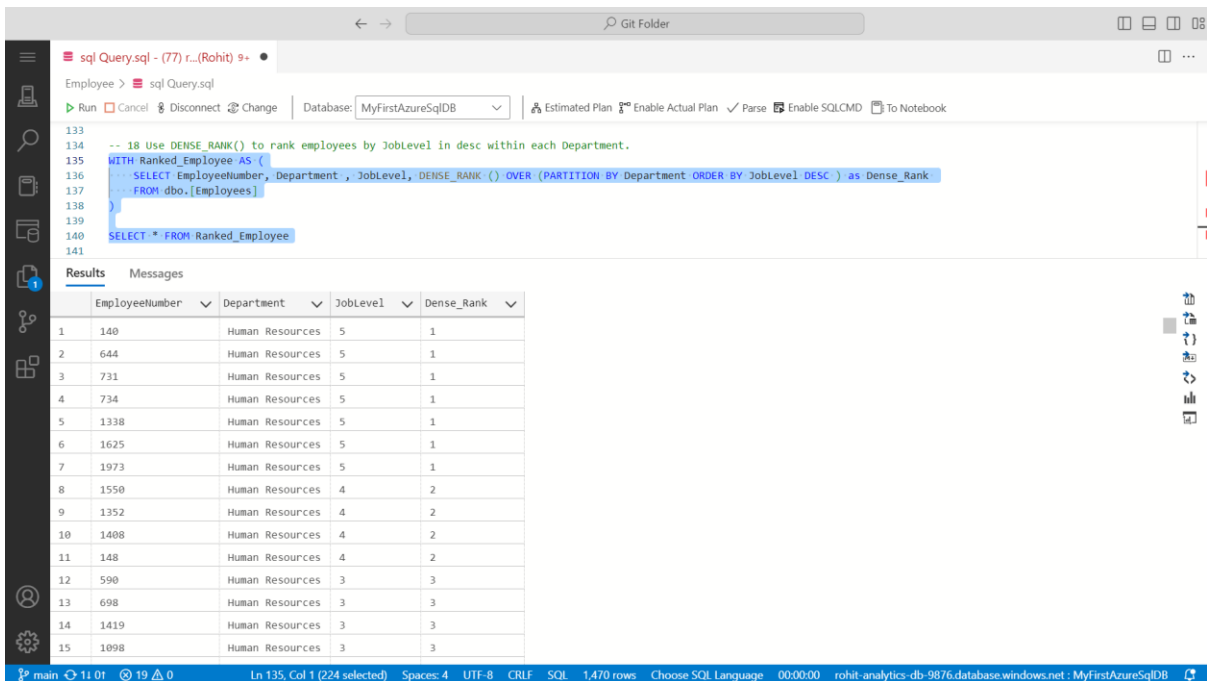
```
-- Advanced SQL Questions:
--17 Assign a rank to employees based on their MonthlyIncome in descending order for each department
--only for top 5 employees for each of the department.

WITH Ranked_Employees AS(
    SELECT EmployeeNumber, Department, MonthlyIncome, RANK() OVER(PARTITION BY Department ORDER BY MonthlyIncome) AS Rank
    FROM dbo.[Employees]
)
SELECT * FROM Ranked_Employees
where rank<=5;
```

The results table shows the following data:

EmployeeNumber	Department	MonthlyIncome	Rank
1 1714	Human Resources	1555	1
2 1499	Human Resources	2064	2
3 133	Human Resources	2073	3
4 1461	Human Resources	2109	4
5 829	Human Resources	2143	5
6 701	Research & Development	1009	1
7 1012	Research & Development	1051	2
8 243	Research & Development	1102	3
9 1974	Research & Development	1129	4
10 1270	Research & Development	1223	5
11 1056	Sales	1052	1
12 1876	Sales	1081	2

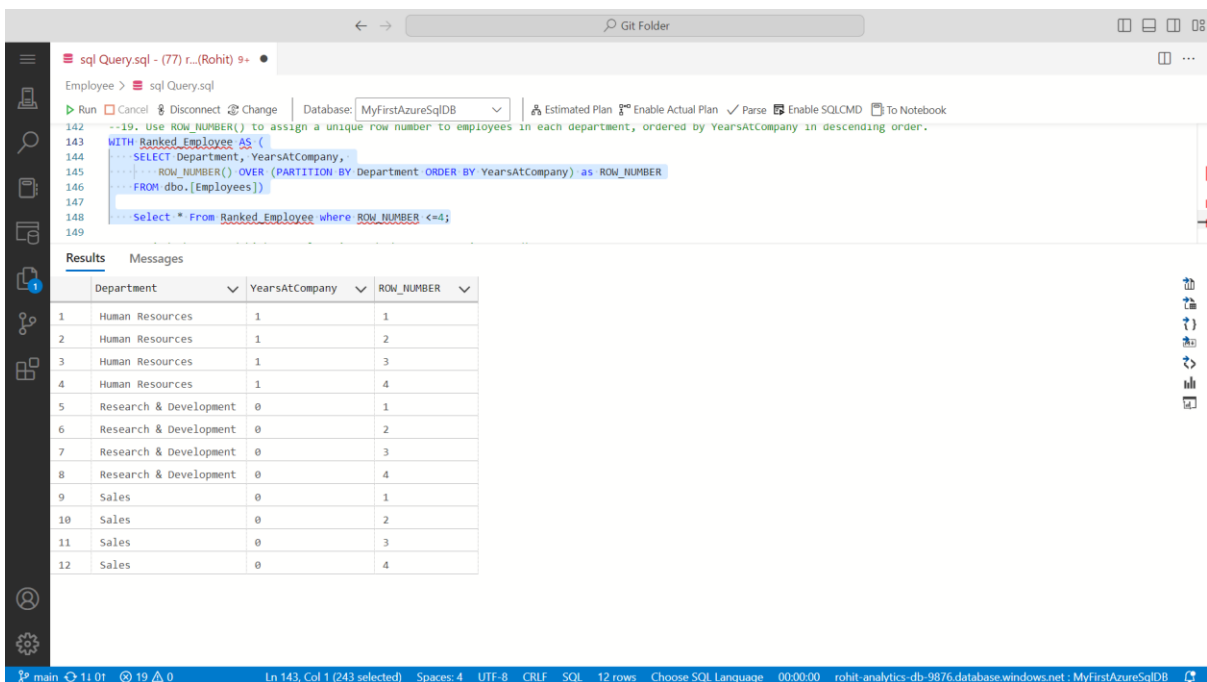
18. DENSE_RANK() - Job Level: Rank employees by Job Level within each department.



The screenshot shows a SQL query in SQL Server Enterprise Manager. The query uses `DENSE_RANK()` to rank employees by `JobLevel` within each `Department`. The results table displays the following data:

	EmployeeNumber	Department	JobLevel	Dense_Rank
1	140	Human Resources	5	1
2	644	Human Resources	5	1
3	731	Human Resources	5	1
4	734	Human Resources	5	1
5	1338	Human Resources	5	1
6	1625	Human Resources	5	1
7	1973	Human Resources	5	1
8	1550	Human Resources	4	2
9	1352	Human Resources	4	2
10	1408	Human Resources	4	2
11	148	Human Resources	4	2
12	590	Human Resources	3	3
13	698	Human Resources	3	3
14	1419	Human Resources	3	3
15	1098	Human Resources	3	3

19. ROW_NUMBER() - Tenure: Assign row numbers to employees in each department based on YearsAtCompany.



The screenshot shows a SQL query in SQL Server Enterprise Manager. The query uses `ROW_NUMBER()` to assign a unique row number to employees in each department, ordered by `YearsAtCompany` in descending order. The results table displays the following data:

	Department	YearsAtCompany	ROW_NUMBER
1	Human Resources	1	1
2	Human Resources	1	2
3	Human Resources	1	3
4	Human Resources	1	4
5	Research & Development	0	1
6	Research & Development	0	2
7	Research & Development	0	3
8	Research & Development	0	4
9	Sales	0	1
10	Sales	0	2
11	Sales	0	3
12	Sales	0	4

20. RANK() - Second Highest Salary: Find the second-highest salary in each department.

The screenshot shows a SQL query in SQL Server Enterprise Manager. The query is as follows:

```
--20: Find the second highest salary in each department using RANK().
WITH Second_Highest_Salary AS (
    SELECT EmployeeNumber, Department, MonthlyIncome,
           RANK() OVER (PARTITION BY Department ORDER BY MonthlyIncome DESC) AS rnk
    FROM dbo.[Employees]
)
SELECT * FROM Second_Highest_Salary WHERE rnk = 2;
```

The results table shows the following data:

EmployeeNumber	Department	MonthlyIncome	rnk
1625	Human Resources	19658	2
1035	Research & Development	19973	2
1038	Sales	19845	2

21. RANK()/DENSE_RANK() - Top 3 Earners: Find the top 3 highest-paid employees in each department.

The screenshot shows a SQL query in SQL Server Enterprise Manager. The query is as follows:

```
--21: Find the top 3 highest-paid employees in each department using RANK() or DENSE_RANK().
WITH Highest_Paid_Emp AS (
    SELECT EmployeeNumber, Department, MonthlyIncome,
           RANK() OVER (PARTITION BY Department ORDER BY MonthlyIncome DESC) AS rnk
    FROM dbo.[Employees]
)
SELECT * FROM Highest_Paid_Emp WHERE rnk <= 3;
```

The results table shows the following data:

EmployeeNumber	Department	MonthlyIncome	rnk
1338	Human Resources	19717	1
1625	Human Resources	19658	2
1973	Human Resources	19636	3
259	Research & Development	19999	1
1035	Research & Development	19973	2
1191	Research & Development	19943	3
1282	Sales	19847	1
1038	Sales	19845	2
1740	Sales	19833	3