(https://databricks.com)

SQL Assignment

2

```
%sql
CREATE TABLE Emp (
    EmployeeID INT,
    Name VARCHAR(50),
    Department VARCHAR(50),
    Salary INT,
    JoiningDate DATE
);
```

OK

3

```
%sql
INSERT INTO Emp (EmployeeID, Name, Department, Salary, JoiningDate) VALUES
(1, 'Alice', 'Sales', 70000, '2021-03-15'),
(2, 'Bob', 'Sales', 68000, '2022-04-20'),
(3, 'Charlie', 'Marketing', 72000, '2020-07-30'),
(4, 'David', 'Marketing', 75000, '2021-11-25'),
(5, 'Eve', 'Sales', 69000, '2020-02-10'),
(6, 'Frank', 'HR', 66000, '2019-05-15'),
(7, 'Grace', 'HR', 64000, '2021-06-10'),
(8, 'Hannah', 'Finance', 73000, '2022-08-19'),
(9, 'Ian', 'Finance', 71000, '2020-03-05'),
(10, 'Jack', 'Sales', 78000, '2023-01-10'),
(11, 'Kara', 'Marketing', 80000, '2022-05-05'),
(12, 'Liam', 'Finance', 72000, '2021-01-30');
```

Table

	num_affected_rows	num_inserted_rows
1	12	12

1 row

Find the total number of employees and the average salary for each department.

5

```
%sql
select department as Department,count(*) as Total_Employee,avg(Salary) as Average_Salary from Emp group by Departm
```

Table

	Department	Total_Employee	Average_Salary
1	Sales	4	71250
2	HR	2	65000

3	Finance	3	72000
4	Marketing	3	75666.66666666667

4 rows

List each employee's name, department, salary, and their rank based on salary within their department.

7

```
%sql
SELECT
    Name,
    Department,
    Salary,
    RANK() OVER (PARTITION BY Department ORDER BY Salary DESC) AS Salary_Rank
FROM
    Emp;
```

Table

	Name	Department	Salary	Salary_Rank
1	Hannah	Finance	73000	1
2	Liam	Finance	72000	2
3	Ian	Finance	71000	3
4	Frank	HR	66000	1
5	Grace	HR	64000	2
6	Kara	Marketing	80000	1
7	David	Marketing	75000	2
8	Charlie	Marketing	72000	3
9	Jack	Sales	78000	1
10	Alice	Sales	70000	2
11	Eve	Sales	69000	3
12	Bob	Sales	68000	4

12 rows

For each department, find the employee with the highest salary.

9

```
%sql
select department as Department,max(Salary) as Highest_Salary from Emp group by department
```

Table

	Department	Highest_Salary
1	Sales	78000
2	HR	66000
3	Finance	73000
4	Marketing	80000

4 rows

Calculate the cumulative salary for each employee within their department, ordered by their salary in descending order.

11

```
%sql
SELECT
    Name,
    Department,
    Salary,
    SUM(Salary) OVER (
        PARTITION BY Department
        ORDER BY Salary DESC
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS Cumulative_Salary
FROM
    Emp;
```

Table

	Name	Department	Salary	Cumulative_Salary
1	Hannah	Finance	73000	73000
2	Liam	Finance	72000	145000
3	Ian	Finance	71000	216000
4	Frank	HR	66000	66000
5	Grace	HR	64000	130000
6	Kara	Marketing	80000	80000
7	David	Marketing	75000	155000
8	Charlie	Marketing	72000	227000
9	Jack	Sales	78000	78000
10	Alice	Sales	70000	148000
11	Eve	Sales	69000	217000
12	Bob	Sales	68000	285000

12 rows

Find the average salary for each department and list the employees who earn above their department's average salary.

13

```
%sql
SELECT
    e.Name,
    e.Department,
    e.Salary
FROM
    Emp e
JOIN
    (SELECT
         Department,
         AVG(salary) AS Average_Salary
     FROM
         Emp
     GROUP BY
         Department) d
ON
    e.Department = d.Department
WHERE
    e.Salary > d.Average_Salary;
```

Table

	Name	Department	Salary
1	Frank	HR	66000
2	Hannah	Finance	73000
3	Jack	Sales	78000
4	Kara	Marketing	80000

4 rows

For each department, determine the difference between each employee's salary and the highest salary in that department.

15

```
%sql
SELECT
    Name,
    Department,
    Salary,
    MAX(Salary) OVER (PARTITION BY Department) - Salary AS Salary_Difference
FROM
    Emp;
```

Table

	Name	Department	Salary	Salary_Difference
1	Hannah	Finance	73000	0
2	Ian	Finance	71000	2000
3	Liam	Finance	72000	1000
4	Frank	HR	66000	0
5	Grace	HR	64000	2000
6	Charlie	Marketing	72000	8000
7	David	Marketing	75000	5000
8	Kara	Marketing	80000	0
9	Alice	Sales	70000	8000
10	Bob	Sales	68000	10000
11	Eve	Sales	69000	9000
12	Jack	Sales	78000	0

12 rows

List the number of employees hired each year, ordered by year.

17

```
%sql
SELECT
    YEAR(JoiningDate) AS Hire_Year,
    COUNT(*) AS Employee_Count
FROM
    Emp
GROUP BY
    YEAR(JoiningDate)
ORDER BY
    Hire_Year;
```

Table

	Hire_Year	Employee_Count
1	2019	1
2	2020	3
3	2021	4
4	2022	3
5	2023	1

5 rows

Find the top two highest-paid employees from each department.

19

```
%sql
SELECT
    Name,
    Department,
    Salary
FROM (
    SELECT
        Name,
        Department,
        Salary,
        ROW_NUMBER() OVER (PARTITION BY Department ORDER BY Salary DESC) AS rank
    FROM
        Emp
) ranked
WHERE
    rank <= 2;
```

Table

	Name	Department	Salary
1	Hannah	Finance	73000
2	Liam	Finance	72000
3	Frank	HR	66000
4	Grace	HR	64000
5	Kara	Marketing	80000
6	David	Marketing	75000
7	Jack	Sales	78000
8	Alice	Sales	70000

8 rows

Calculate the running average salary for each department, ordered by salary in descending order.

21

```
%sql
SELECT
    Name,
    Department,
    Salary,
    AVG(Salary) OVER (
        PARTITION BY Department
        ORDER BY Salary DESC
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS Running_Average_Salary
FROM
    Emp;
```

Table

	Name	Department	Salary	Running_Average_Salary
1	Hannah	Finance	73000	73000
2	Liam	Finance	72000	72500
3	Ian	Finance	71000	72000
4	Frank	HR	66000	66000
5	Grace	HR	64000	65000
6	Kara	Marketing	80000	80000
7	David	Marketing	75000	77500
8	Charlie	Marketing	72000	75666.66666666667
9	Jack	Sales	78000	78000
10	Alice	Sales	70000	74000
11	Eve	Sales	69000	72333.33333333333
12	Bob	Sales	68000	71250

12 rows

Find each employee's tenure in years (as of today) and rank employees by tenure within each department.

23

Table

	Name	Department	JoiningDate	Tenure_Years	Tenure_Rank
1	Ian	Finance	2020-03-05	4.70	1
2	Liam	Finance	2021-01-30	3.80	2
3	Hannah	Finance	2022-08-19	2.25	3
4	Frank	HR	2019-05-15	5.51	1
5	Grace	HR	2021-06-10	3.44	2
6	Charlie	Marketing	2020-07-30	4.30	1
7	David	Marketing	2021-11-25	2.98	2
8	Kara	Marketing	2022-05-05	2.54	3
9	Eve	Sales	2020-02-10	4.77	1
10	Alice	Sales	2021-03-15	3.68	2
11	Bob	Sales	2022-04-20	2.58	3
12	Jack	Sales	2023-01-10	1.85	4

12 rows

This result is stored as `_sqldf` and can be used in other Python cells.