Window Functions in SQL

- Ranking Windows Function
- Aggregate Windows Function

Ranking Windows Function-

These functions provide rankings of rows within a partition based on specific criteria. Common ranking functions include:

RANK():

Assigns ranks to rows, skipping ranks for duplicates.

```
SELECT Name, Department, Salary,

RANK() OVER(PARTITION BY Department ORDER BY Salary DESC) AS Rank

FROM employee;
```

Employee table-

Name	Age	Department	Salary
Ramesh	20	Finance	50,000
Suresh	22	Finance	50,000
Ram	28	Finance	20,000
Deep	25	Sales	30,000
Pradeep	22	Sales	20,000

Output table-

Name	Age	Dept.	Salary	Rank
Ramesh	20	Finance	50,000	1
Suresh	22	Finance	50,000	1
Ram	28	Finance	20,000	3
Deep	25	Sales	30,000	1
Pradeep	22	Sales	20,000	2

DENSE_RANK():

Assigns ranks to rows without skipping rank numbers for duplicates.

```
SELECT Name, Department, Salary,

DENSE_RANK() OVER(PARTITION BY Department ORDER BY Salary DESC) AS
D_Rank

FROM employee;
```

Employee table-

Name	Age	Department	Salary
Ramesh	20	Finance	50,000
Suresh	22	Finance	50,000
Ram	28	Finance	20,000
Deep	25	Sales	30,000
Pradeep	22	Sales	20,000

Output table-

Name	Age	Dept.	Salary	D_Rank
Ramesh	20	Finance	50,000	1
Suresh	22	Finance	50,000	1
Ram	28	Finance	20,000	2
Deep	25	Sales	30,000	1
Pradeep	22	Sales	20,000	2

ROW_NUMBER():

Assigns a unique number to each row in the result set.

```
SELECT Name, Department, Salary,

ROW_NUMBER() OVER(PARTITION BY Department ORDER BY Salary DESC)
AS emp_row_no

FROM employee;
```

Employee table -

Name	Age	Department	Salary
Ramesh	20	Finance	50,000
Suresh	22	Finance	50,000
Ram	28	Finance	20,000
Deep	25	Sales	30,000
Pradeep	22	Sales	20,000

Output table -

Name	Age	Dept.	Salary	R_no.
Ramesh	20	Finance	50,000	1
Suresh	22	Finance	50,000	2
Ram	28	Finance	20,000	3
Deep	25	Sales	30,000	1
Pradeep	22	Sales	20,000	2

Aggregate Window Function-

Aggregate window functions calculate aggregates over a window of rows while retaining individual rows. Common aggregate functions include:

- **SUM()**: Sums values within a window.
- **AVG()**: Calculates the average value within a window.

- **COUNT()**: Counts the rows within a window.
- **MAX()**: Returns the maximum value in the window.
- **MIN()**: Returns the minimum value in the window.

SUM() over - Total salary per department

```
SELECT *,
  SUM(Salary) OVER (PARTITION BY Department) AS dept_total_salary
FROM Employees;
```

Employee table

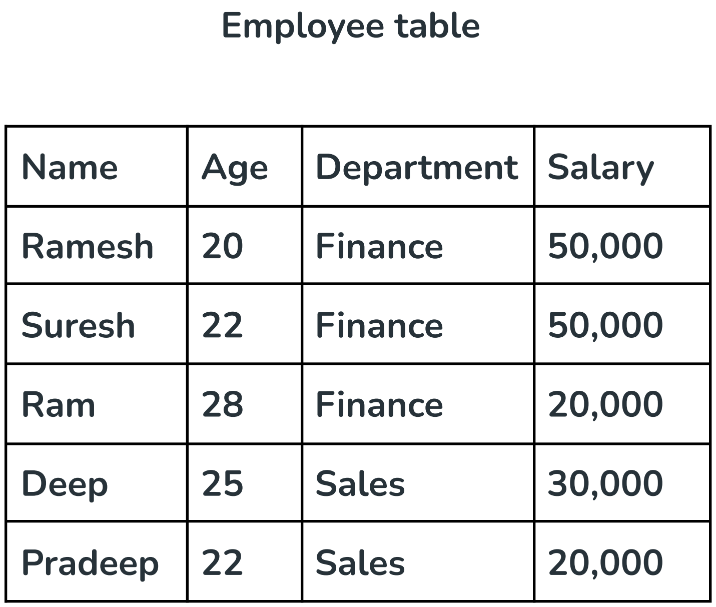
Name	Age	Department	Salary
Ramesh	20	Finance	50,000
Suresh	22	Finance	50,000
Ram	28	Finance	20,000
Deep	25	Sales	30,000
Pradeep	22	Sales	20,000

Output table

Name	Age	Dept.	Salary	dept_total_sal
Ramesh	20	Finance	50,000	120,000
Suresh	22	Finance	50,000	120,000
Ram	28	Finance	20,000	120,000
Deep	25	Sales	30,000	50,000
Pradeep	22	Sales	20,000	50,000

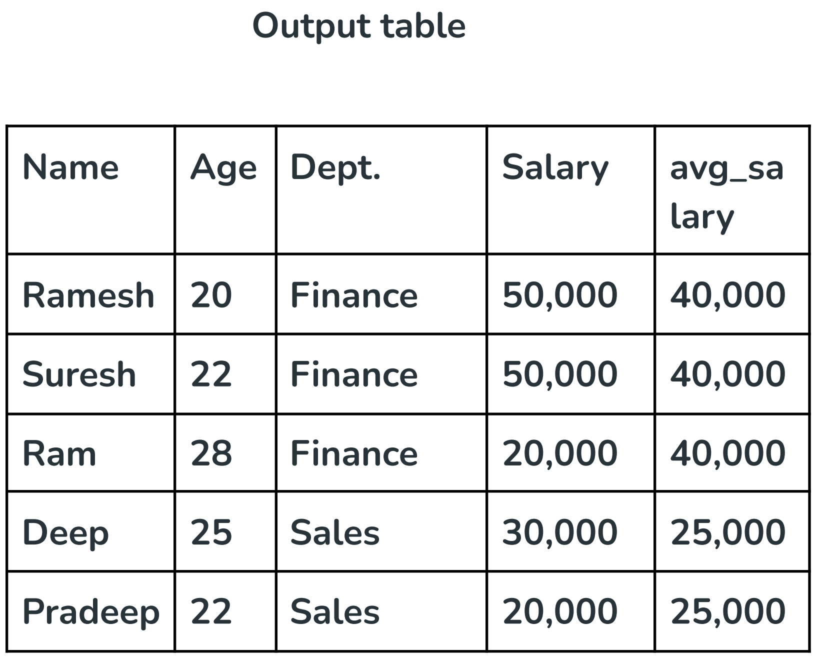
AVG() over - Calculate the Average Salary within each department

```
SELECT Name, Age, Department, Salary,
AVG(Salary) OVER( PARTITION BY Department) AS Avg_Salary
FROM employee
```



Employee table

Name	Age	Department	Salary
Ramesh	20	Finance	50,000
Suresh	22	Finance	50,000
Ram	28	Finance	20,000
Deep	25	Sales	30,000
Pradeep	22	Sales	20,000



Output table

Name	Age	Dept.	Salary	avg_salary
Ramesh	20	Finance	50,000	40,000
Suresh	22	Finance	50,000	40,000
Ram	28	Finance	20,000	40,000
Deep	25	Sales	30,000	25,000
Pradeep	22	Sales	20,000	25,000

`LAG()` and `LEAD()` – Previous and next salary in department

```
SELECT *,
LAG(Salary) OVER (PARTITION BY Department ORDER BY Salary) AS
prev_salary,
LEAD(Salary) OVER (PARTITION BY Department ORDER BY Salary) AS
next_salary
FROM Employees;
```

Employee table

Name	Age	Dept.	Salary
Ramesh	20	Finance	50,000
Suresh	22	Finance	50,000
Ram	28	Finance	20,000
Deep	25	Sales	30,000
Pradeep	22	Sales	20,000

Output table

Name	Age	Dept.	Salary	prev_salary	next_salary
Ramesh	20	Finance	50,000	NULL	50,000
Suresh	22	Finance	50,000	50,000	20,000
Ram	28	Finance	20,000	50,000	NULL
Deep	25	Sales	30,000	NULL	20,000
Pradeep	22	Sales	20,000	30,000	NULL

LIMIT OFFSET –

ID	Name	Salary
1	Alice	50000
2	Bob	60000
3	Carol	55000
4	David	70000
5	Eva	65000

```
SELECT * FROM table_name
ORDER BY some_column
LIMIT count OFFSET skip;
```

`LIMIT`: Number of rows to return.
`OFFSET`: Number of rows to skip before starting to return results.

Get Top 2 Salaries:

```
SELECT * FROM Employees
ORDER BY Salary DESC
LIMIT 2;
```

Output table -

Name	Salary
David	70000
Eva	65000

Skip Top 3, Get Next 2

```
SELECT * FROM Employees
ORDER BY Salary DESC
```

LIMIT 2 OFFSET 3; //   or LIMIT 3,2 ;

Output table -

```
3  Carol  55000
1  Alice   50000
```

Exercise 1

Sample `employees` Table

employee_id	name	department	salary	experience	city
1	Alice	IT	90000	7	New York
2	Bob	HR	85000	6	Chicago
3	Charlie	IT	85000	5	San Francisco
4	David	Finance	80000	4	Boston
5	Eve	HR	75000	3	Los Angeles
6	Frank	Finance	75000	2	Miami
7	Grace	IT	70000	5	Chicago
8	Henry	HR	72000	4	New York
9	Ian	IT	68000	3	San Diego
10	Julia	Finance	77000	6	Dallas

1-Whis is the highest salary?
2-Second highest salary?
3-Fourth highest salary? Fetch all the employees who have the fourth highest salaries?
4-Group departmentwise and salarywise?
5-Select the top 2 highest salaries from each dept.
6-Return three columns of Rank,dense rank,row number and compare.

CASE WHEN THEN:

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    ...
    ELSE resultN
END
```

Sample output table-

Name	Dept	Salary
Alice	HR	50,000
Bob	HR	60,000
Carol	HR	70,000
David	IT	90,000
Eva	IT	70,000

Name	Dept	Salary	Salary level
Alice	HR	50,000	Low
Bob	HR	60,000	Medium
Carol	HR	70,000	Medium
David	IT	90,000	High
Eva	IT	70,000	Medium

Exercise 2

Ques 1 -

Write a SQL query to display each employee’s name, salary, and a new column called `Salary_Level` which categorizes the salary as follows:

- `'High'` if the salary is greater than or equal to 80000
- `'Medium'` if the salary is between 60000 and 79999
- `'Low'` if the salary is less than 60000

```
SELECT Name, Salary,
  CASE
    WHEN Salary >= 80000 THEN 'High'
    WHEN Salary >= 60000 THEN 'Medium'
    ELSE 'Low'
  END AS Salary_Level
FROM Employees;
```

Ques 2 -

Show patient_id, weight, height, isObese from the patients table.
Display isObese as a boolean 0 or 1.
Obese is defined as weight(kg)/(height(m)^{2}) >= 30.
weight is in units kg.
height is in units cm.
(refer table mentioned at the end)

```
SELECT patient_id, weight, height,
  (CASE
      WHEN weight/(POWER(height/100.0,2)) >= 30 THEN
         1
      ELSE
         0
      END) AS isObese
FROM patients;
```

JOINS -