Analytical Functions

- Analytical functions are works on group of data like group functions.
- Group functions reduce number of rows in each group where as analytical function does not reduces number of rows in each group.

Syntax

analyticalfunction() over
(parition by column name order by coluname [asc/desc])

row_number()

- It assigns a unique sequential integer to each row within a partition of a result set.
- It starts from 1 for the first row in each partition.

Example#1

```
SELECT emp.*, row_number() over (partition by department_id order by salary desc) as rn
FROM employees emp;
```

Example#2

Write a query to display first 2 highest paid employees from each department.

```
SELECT * FROM (
SELECT emp.*, row_number() over (partition by department_id order by salary desc) as R FROM
employees emp) as e1
WHERE R <= 2;
```

rank()

- It assigns a unique rank to each distinct row value within a partition of a result set.
- In case of ties, assigned the same rank value, and subsequent ranks skipping over the number of tied values.

Example#1

```
SELECT emp.*, rank() over (partition by department_id order by salary desc) as rn
FROM employees emp;
```

dense_rank()

- It assigns a unique rank to each distinct row value within a partition of a result set.
- In case of ties, assigned the same rank value, and without skipping ranks for tied values.

Example#1

```
SELECT emp.*, dense_rank() over (partition by department_id order by salary desc) as rn
FROM employees emp;
```

LAG()

- It is used to compare current row value with previous row value.
- Syntax:
  LAG(columnname, offset, defaultvalue) over (partion by columnname order by column [asc/desc])

```
CREATE TABLE revenues (
   quarter_name VARCHAR(20),
   revenue DECIMAL(15, 2)
);
INSERT INTO revenues VALUES('Q1',20000000),
('Q2',50000000),
('Q3',10000000),
('Q4',60000000);

SELECT quarter_name, revenue, lag(revenue) OVER (order by quarter_name) prev_revenue
FROM  revenues;

SELECT quarter_name, revenue, lag(revenue,2) OVER (order by quarter_name) prev_revenue
FROM  revenues;

SELECT quarter_name, revenue, lag(revenue,2, 0) OVER (order by quarter_name) prev_revenue
FROM  revenues;
```

LEAD()

- It is used to compare current row value with next row value.

Example#1

```
SELECT quarter_name, revenue, lead(revenue) OVER (order by quarter_name) next_revenue
FROM  revenues;
```

Assignment

1. Write a query to display most recently joined employee in each department from employee table using analytical functions.
   Write a query to display most recently joined employee in each department from employee table using analytical functions.

   101 '2024-01-10' 1
   103 '2024-02-10' 1
   102 '2024-05-10' 2
   105 '2024-03-10' 2
   106 '2024-02-10' 2
   104 '2024-04-10' 3
   o/p
   -----
   103 '2024-02-10' 1
   102 '2024-05-10' 2
   104 '2024-04-10' 3

   sort based on joining date in desc
   103 '2024-02-10' 1
   101 '2024-01-10' 1
   102 '2024-05-10' 2
   105 '2024-03-10' 2
   106 '2024-02-10' 2
   104 '2024-04-10' 3

2. Write a query to display employee details along with max, min, avg and sum salary from employee table.

   ```
   SELECT emp.*,
   max(salary) over() as max_salary,
   ```

```
min(salary) over() as min_salary,
avg(salary) over() as avg_salary
FROM employees emp;
```

3. Write a query to display 2nd highest salary paid employee in each department using analytical functions.

```
SELECT * FROM (

SELECT *, dense_rank() over(partition by department_id order by salary desc) rn

FROM employees) as emp

WHERE rn = 2;
```

4. Write a query to display employees along with sum of salary department wise using analytical functions.
5. Write a query to display average salary department wise against each employee record without using analytical functions.

```
SELECT emp2.*, emp1.avg_salary FROM
(
SELECT department_id, avg(salary) as avg_salary FROM employees group by
department_id
) as emp1, employees emp2 WHERE emp1.department_id = emp2.department_id;
```