Window functions in SQL are used to perform calculations across a specified range of rows related to the current row. They are often used in conjunction with the `OVER` clause to define the window of rows to which the function is applied. Here are detailed explanations and two code examples for each of the specified window functions:

1. ROW_NUMBER():

The `ROW_NUMBER()` function assigns a unique number to each row within a partition of a result set, starting at 1 for the first row.

Example 1:

```sql
SELECT
  ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_num,
  employee_id,
  salary
FROM employees;
```

This example assigns a unique row number to each employee based on their salary in descending order.

Example 2:

```sql
SELECT
  ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY hire_date) AS row_num,
  employee_id,
  hire_date
FROM employees;
```

This example assigns a row number within each department based on the hire date.

2. RANK():

The `RANK()` function assigns a rank to each row within a result set, with ties receiving the same rank. The next rank is then skipped.

Example 1:

```sql
SELECT
```

```
  RANK() OVER (ORDER BY total_sales DESC) AS sales_rank,
  employee_id,
  total_sales
FROM sales_summary;
```

This example assigns a rank to employees based on their total sales in descending order.

Example 2:

```sql
SELECT
  RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS salary_rank,
  employee_id,
  salary
FROM employees;
```

This example assigns a rank within each department based on the employee's salary in descending order.

3. DENSE_RANK():

The `DENSE_RANK()` function is similar to `RANK()`, but it does not skip rank values for ties.

Example 1:

```sql
SELECT
  DENSE_RANK() OVER (ORDER BY order_date) AS order_rank,
  order_id,
  order_date
FROM orders;
```

This example assigns a dense rank to orders based on their order dates.

Example 2:

```sql
SELECT
  DENSE_RANK() OVER (PARTITION BY product_category ORDER BY revenue DESC) AS
revenue_rank,
  product_id,
  revenue
FROM product_sales;
```

This example assigns a dense rank within each product category based on revenue in descending order.

4. LAG() and LEAD():

The `LAG()` function retrieves data from a previous row, and the `LEAD()` function retrieves data from a subsequent row within the result set.

Example 1:

```sql
SELECT
  employee_id,
  salary,
  LAG(salary) OVER (ORDER BY hire_date) AS prev_salary
FROM employees;
```

This example retrieves the previous salary for each employee based on their hire date.

Example 2:

```sql
SELECT
  product_id,
  revenue,
  LEAD(revenue, 2) OVER (PARTITION BY product_category ORDER BY order_date) AS
next_two_revenue
FROM product_sales;
```

Certainly! Let's provide more examples for each window function:

1. ROW_NUMBER():

Example 3:

```sql
SELECT
```

```
  ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY hire_date DESC) AS
row_num,
  employee_id,
  hire_date
FROM employees;
```

This example assigns a row number within each department based on the hire date in descending order.

Example 4:

```sql
SELECT
  ROW_NUMBER() OVER (ORDER BY product_price) AS row_num,
  product_id,
  product_price
FROM products;
```

This example assigns a row number to each product based on its price.

2. RANK():

Example 3:

```sql
SELECT
  RANK() OVER (PARTITION BY project_id ORDER BY task_priority) AS task_rank,
  task_id,
  task_priority
FROM project_tasks;
```

This example assigns a rank to tasks within each project based on their priority.

Example 4:

```sql
SELECT
  RANK() OVER (ORDER BY total_profit DESC) AS profit_rank,
  region,
  total_profit
FROM regional_performance;
```

This example assigns a rank to each region based on total profit in descending order.

3. DENSE_RANK():

Example 3:

```sql
SELECT
  DENSE_RANK() OVER (ORDER BY start_date) AS event_rank,
  event_id,
  start_date
FROM events;
```

This example assigns a dense rank to events based on their start date.

Example 4:

```sql
SELECT
  DENSE_RANK() OVER (PARTITION BY category ORDER BY rating DESC) AS
category_rank,
  movie_id,
  rating
FROM movies;
```

This example assigns a dense rank within each movie category based on the rating in descending order.

4. LAG() and LEAD():

Example 3:

```sql
SELECT
  product_id,
  sales_date,
  LAG(sales, 1, 0) OVER (PARTITION BY product_id ORDER BY sales_date) AS prev_sales,
  LEAD(sales, 1, 0) OVER (PARTITION BY product_id ORDER BY sales_date) AS next_sales
FROM product_sales;
```

This example retrieves the previous and next sales for each product based on the sales date, with default values set to 0.

Example 4:

```sql
SELECT
  employee_id,
  salary,
  LAG(salary) OVER (ORDER BY hire_date DESC) AS prev_salary,
  LEAD(salary) OVER (ORDER BY hire_date DESC) AS next_salary
FROM employees;
```

This example retrieves the previous and next salary for each employee based on the hire date in descending order.