#56daysofSQL

Day 55
Window Functions
Part 5

Aditya Shenoy

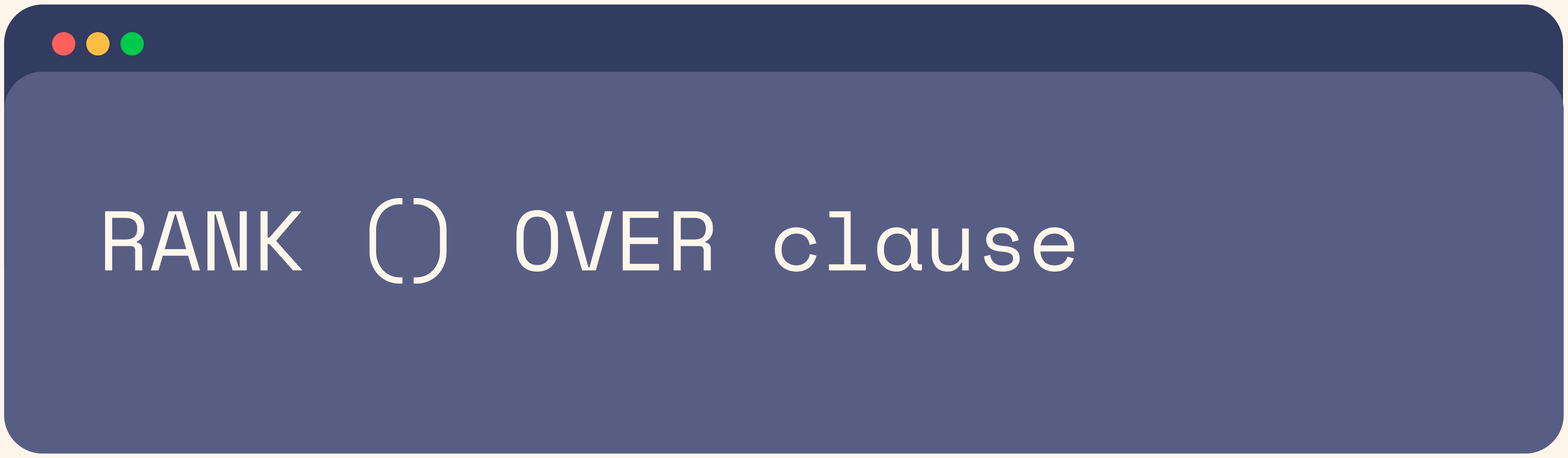
Ranking Window Functions

- RANK()
- DENSE_RANK()
- PERCENT_RANK()
- ROW_NUMBER()
- NTILE()

RANK()

- The **RANK** window function determines the rank of a value in a group of values.
- The ORDER BY expression in the OVER clause determines the value.
- Each value is ranked within its partition.
- Rows with equal values for the ranking criteria receive the same rank with the **next rank skipped**.

General Syntax



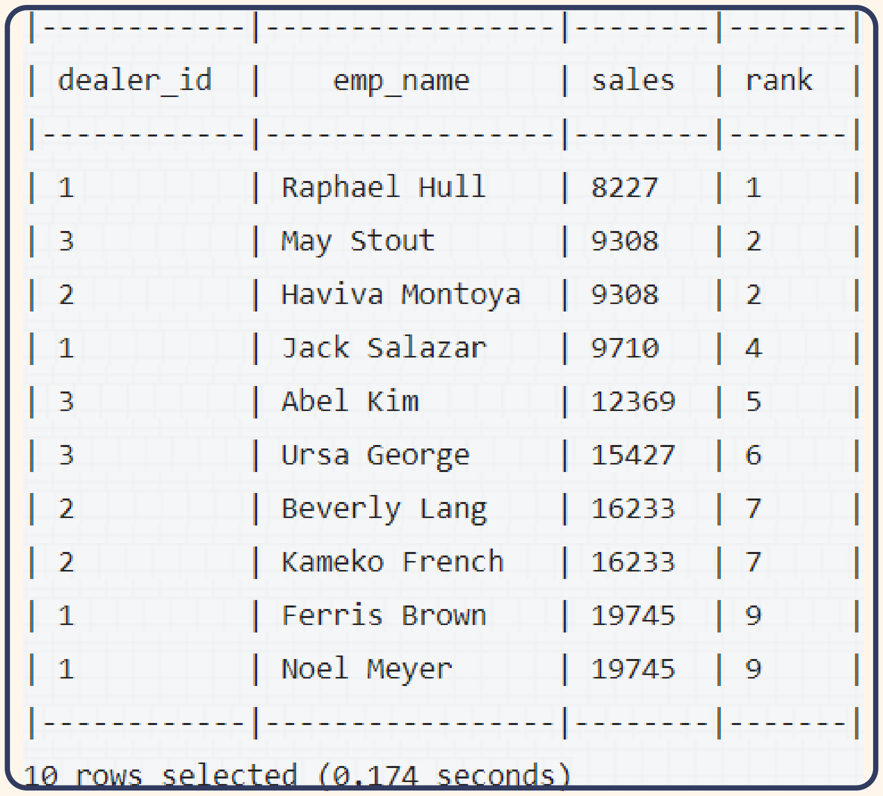
```
RANK () OVER clause
```

Use Case Example of RANK() :

The following query uses the **RANK()** window function to rank the employee sales for Q1.

```
SELECT dealer_id, emp_name, sales,
RANK() OVER(ORDER BY sales) as rank
FROM q1_sales;
```

Output :

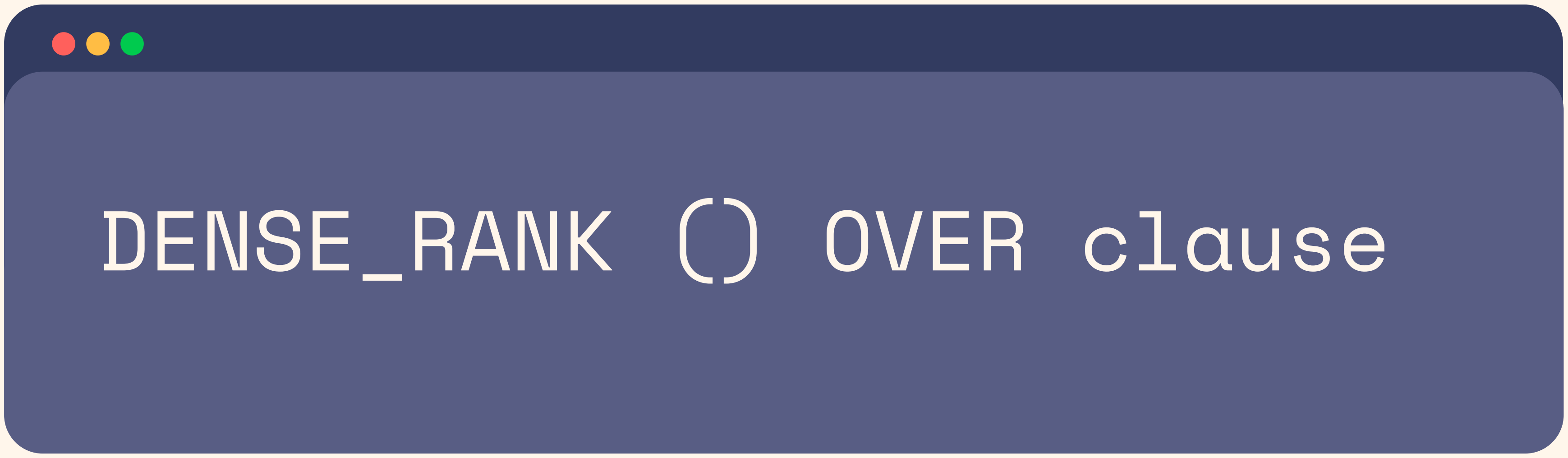


```
|------------|----------------|--------|-------|
| dealer_id  |    emp_name    | sales  | rank  |
|------------|----------------|--------|-------|
| 1          | Raphael Hull   | 8227   | 1     |
| 3          | May Stout      | 9308   | 2     |
| 2          | Haviva Montoya | 9308   | 2     |
| 1          | Jack Salazar   | 9710   | 4     |
| 3          | Abel Kim       | 12369  | 5     |
| 3          | Ursa George    | 15427  | 6     |
| 2          | Beverly Lang   | 16233  | 7     |
| 2          | Kameko French  | 16233  | 7     |
| 1          | Ferris Brown   | 19745  | 9     |
| 1          | Noel Meyer     | 19745  | 9     |
|------------|----------------|--------|-------|
10 rows selected (0.174 seconds)
```

DENSE_RANK()

- The **DENSE_RANK** window function determines the rank of a value in a group of values based on the ORDER BY expression and the OVER clause.
- Each value is ranked within its partition.
- Rows with equal values receive the same rank.
- There are **no gaps** in the sequence of ranked values if two or more rows have the same rank.

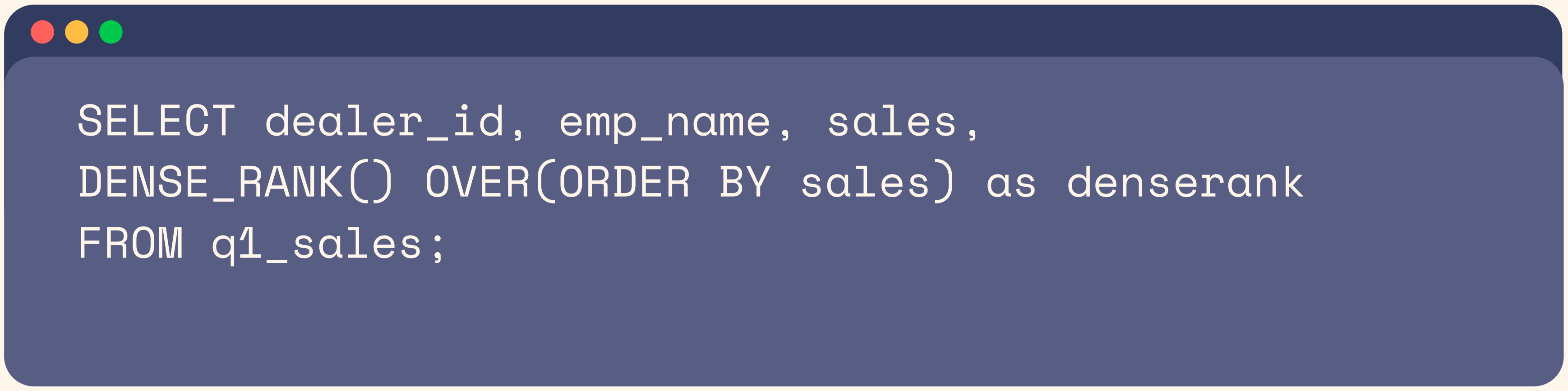
General Syntax



```
DENSE_RANK () OVER clause
```

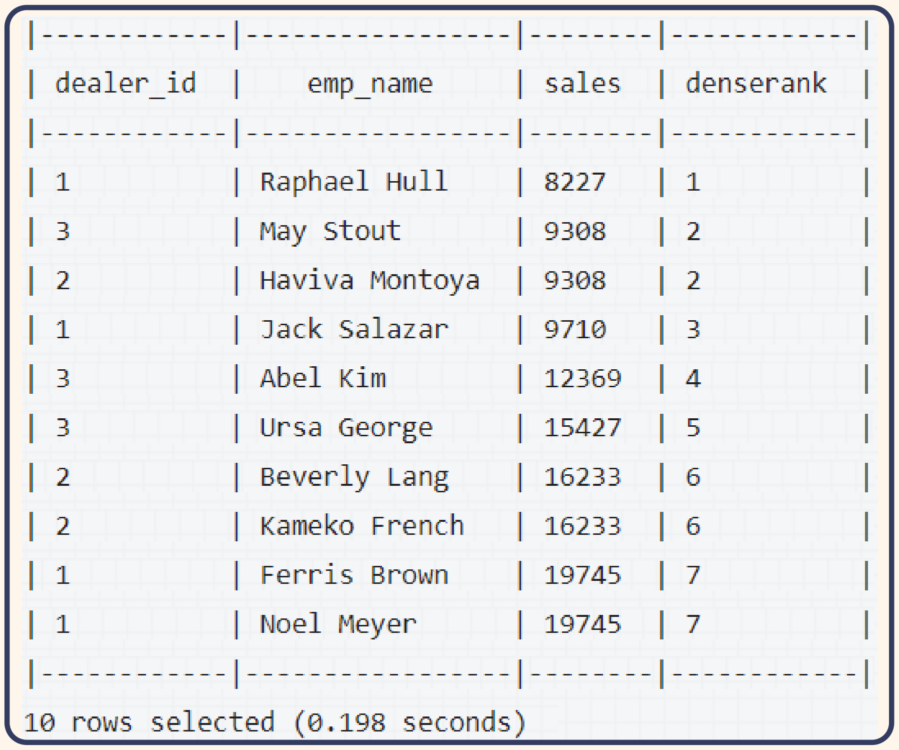
Use Case Example of DENSE_RANK():

The following query uses the **DENSE_RANK()** window function to rank the employee sales in Q1:



```
SELECT dealer_id, emp_name, sales,
DENSE_RANK() OVER(ORDER BY sales) as denserank
FROM q1_sales;
```

Output :

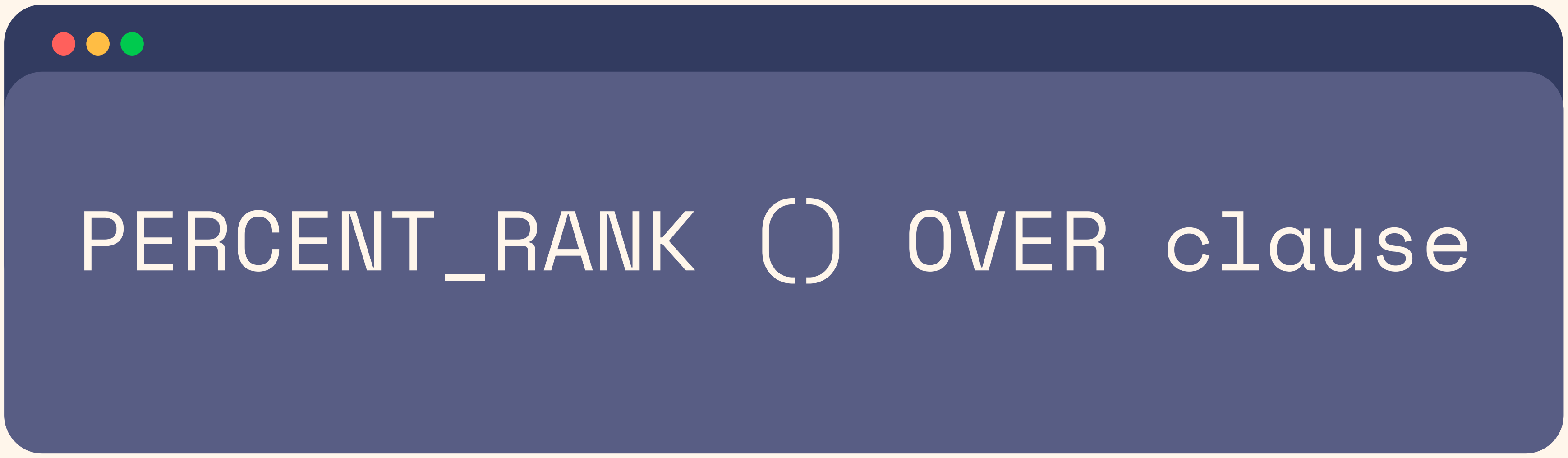


```
|-----------|----------------|--------|-----------|
| dealer_id |    emp_name    | sales  | denserank |
|-----------|----------------|--------|-----------|
| 1         | Raphael Hull   | 8227   | 1         |
| 3         | May Stout      | 9308   | 2         |
| 2         | Haviva Montoya | 9308   | 2         |
| 1         | Jack Salazar   | 9710   | 3         |
| 3         | Abel Kim       | 12369  | 4         |
| 3         | Ursa George    | 15427  | 5         |
| 2         | Beverly Lang   | 16233  | 6         |
| 2         | Kameko French  | 16233  | 6         |
| 1         | Ferris Brown   | 19745  | 7         |
| 1         | Noel Meyer     | 19745  | 7         |
|-----------|----------------|--------|-----------|
10 rows selected (0.198 seconds)
```

PERCENT_RANK()

- The **PERCENT_RANK** window function calculates the percent rank of the current row using the following formula:
    - **(x - 1) / (number of rows in window partition - 1)** where x is the rank of the current row.

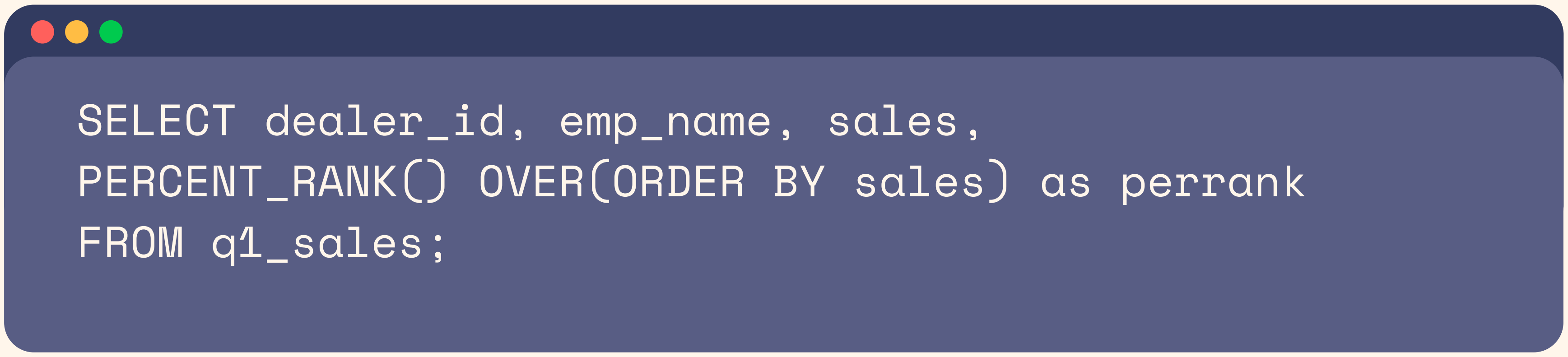
General Syntax



```
PERCENT_RANK () OVER clause
```

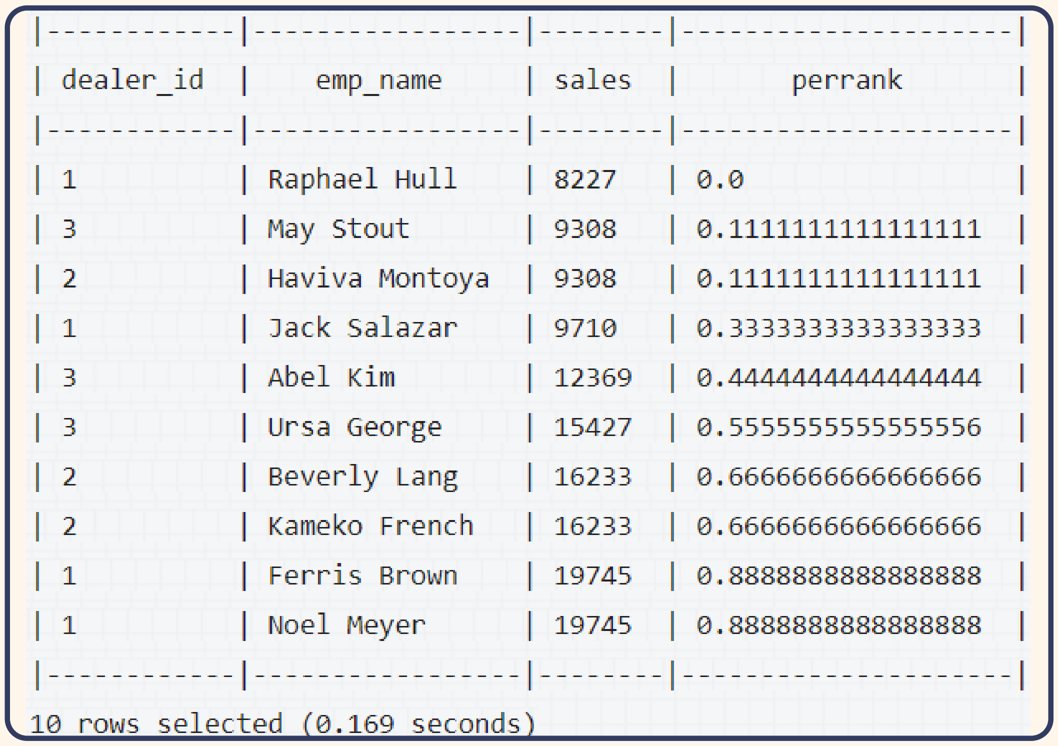
Use Case Example of PERCENT_RANK()

The following query uses the **PERCENT_RANK()** window function to calculate the percent rank for employee sales in Q1:



```
SELECT dealer_id, emp_name, sales,
PERCENT_RANK() OVER(ORDER BY sales) as perrank
FROM q1_sales;
```

Output :

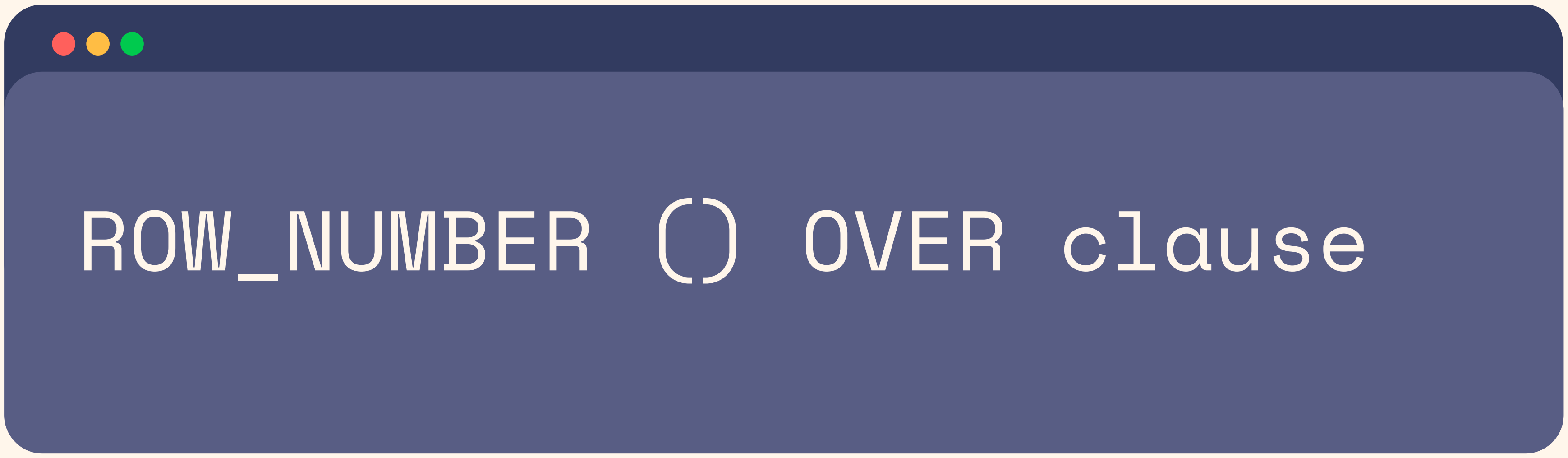


```
|-----------|----------------|--------|---------------------|
| dealer_id |    emp_name    | sales  |       perrank       |
|-----------|----------------|--------|---------------------|
| 1         | Raphael Hull   | 8227   | 0.0                 |
| 3         | May Stout      | 9308   | 0.1111111111111111  |
| 2         | Haviva Montoya | 9308   | 0.1111111111111111  |
| 1         | Jack Salazar   | 9710   | 0.3333333333333333  |
| 3         | Abel Kim       | 12369  | 0.4444444444444444  |
| 3         | Ursa George    | 15427  | 0.5555555555555556  |
| 2         | Beverly Lang   | 16233  | 0.6666666666666666  |
| 2         | Kameko French  | 16233  | 0.6666666666666666  |
| 1         | Ferris Brown   | 19745  | 0.8888888888888888  |
| 1         | Noel Meyer     | 19745  | 0.8888888888888888  |
|-----------|----------------|--------|---------------------|
10 rows selected (0.169 seconds)
```

ROW_NUMBER()

- The **ROW_NUMBER** window function determines the ordinal number of the current row within its partition.
- The ORDER BY expression in the OVER clause determines the number.
- Each value is ordered within its partition.
- Rows with equal values for the ORDER BY expressions receive different row numbers nondeterministically.

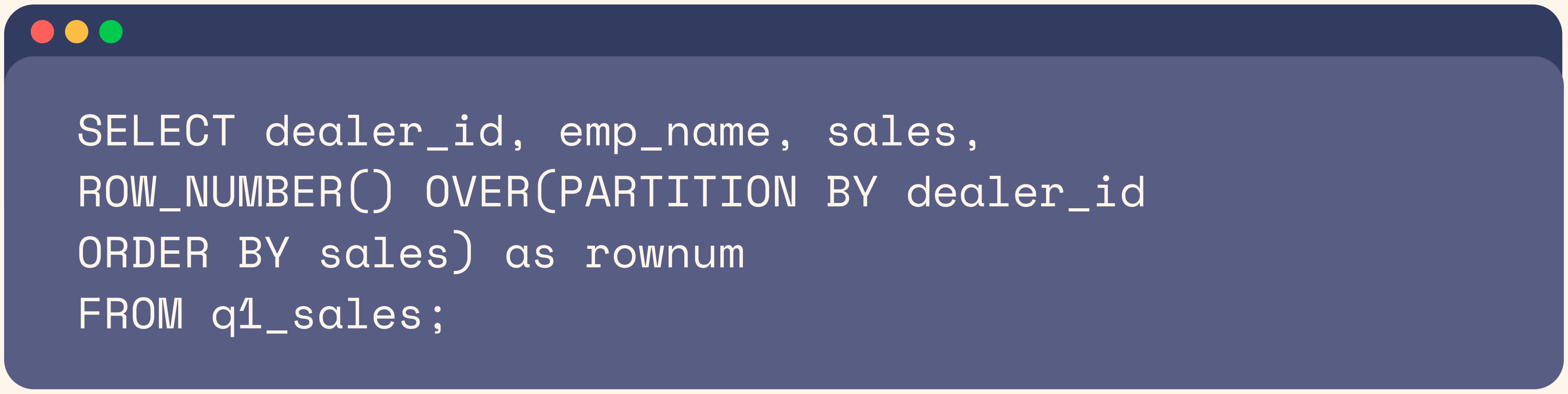
General Syntax



```
ROW_NUMBER () OVER clause
```

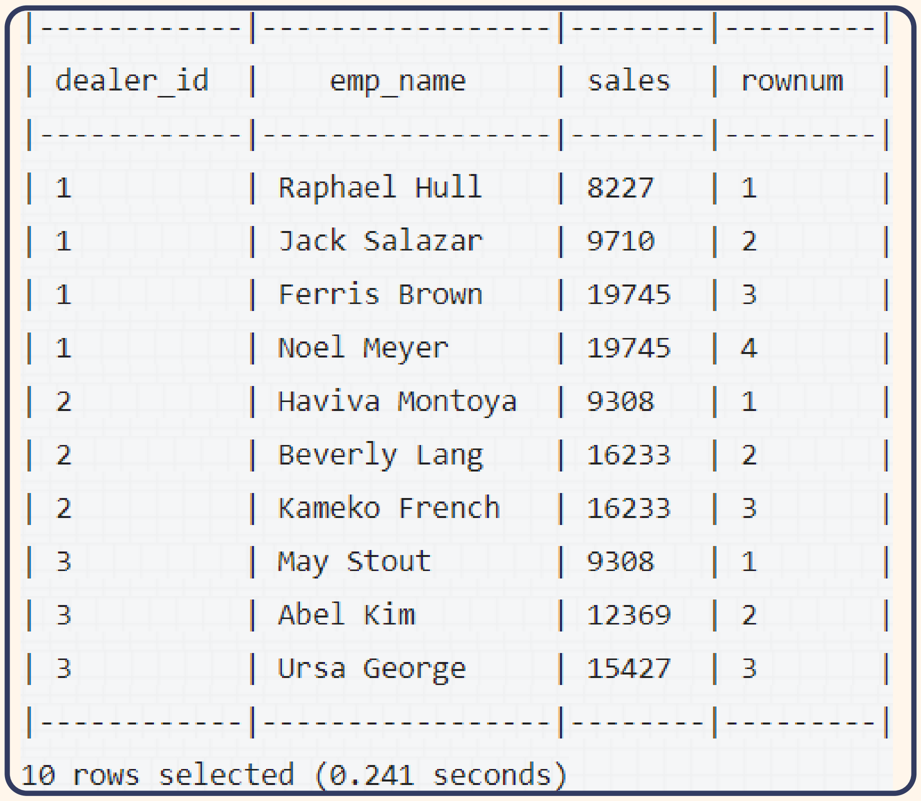
Use Case Example of ROW_NUMBER()

The following query uses the **ROW_NUMBER()** window function to number the sales for each dealer_id:



```
SELECT dealer_id, emp_name, sales,
ROW_NUMBER() OVER(PARTITION BY dealer_id
ORDER BY sales) as rownum
FROM q1_sales;
```

Output :

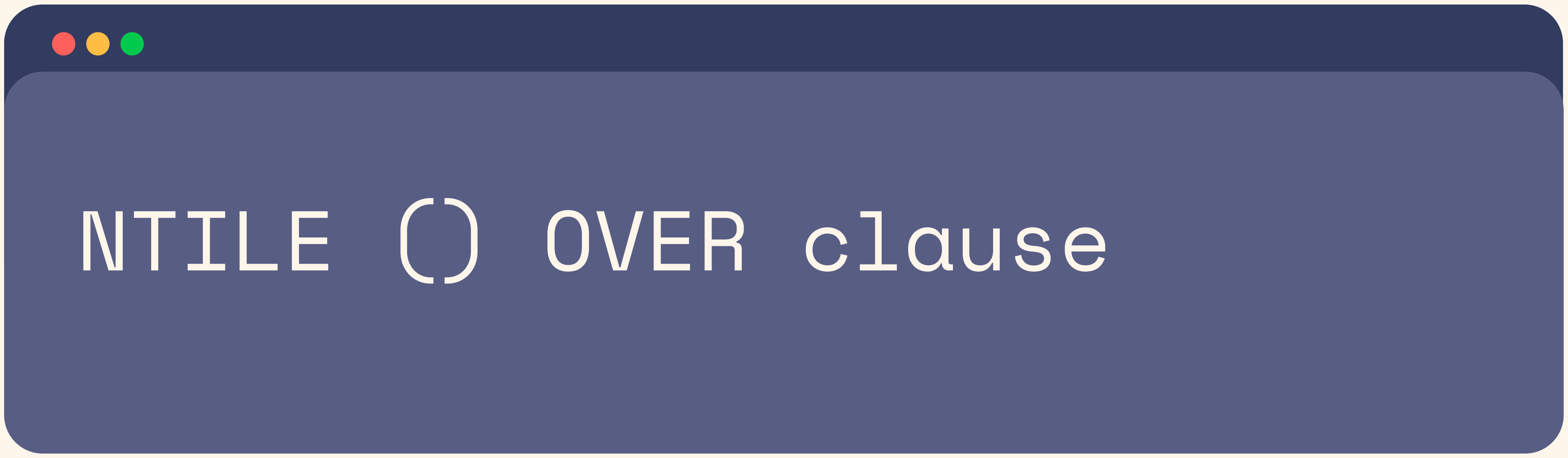


```
|-----------|-----------------|--------|---------|
| dealer_id |    emp_name     | sales  | rownum  |
|-----------|-----------------|--------|---------|
| 1         | Raphael Hull    | 8227   | 1       |
| 1         | Jack Salazar    | 9710   | 2       |
| 1         | Ferris Brown    | 19745  | 3       |
| 1         | Noel Meyer      | 19745  | 4       |
| 2         | Haviva Montoya  | 9308   | 1       |
| 2         | Beverly Lang    | 16233  | 2       |
| 2         | Kameko French   | 16233  | 3       |
| 3         | May Stout       | 9308   | 1       |
| 3         | Abel Kim        | 12369  | 2       |
| 3         | Ursa George     | 15427  | 3       |
|-----------|-----------------|--------|---------|
10 rows selected (0.241 seconds)
```

NTILE()

- The **NTILE** window function divides the rows for each window partition, as equally as possible, into a specified number of ranked groups.
- The **NTILE** window function requires the **ORDER BY** clause in the OVER clause.

General Syntax



```
NTILE () OVER clause
```

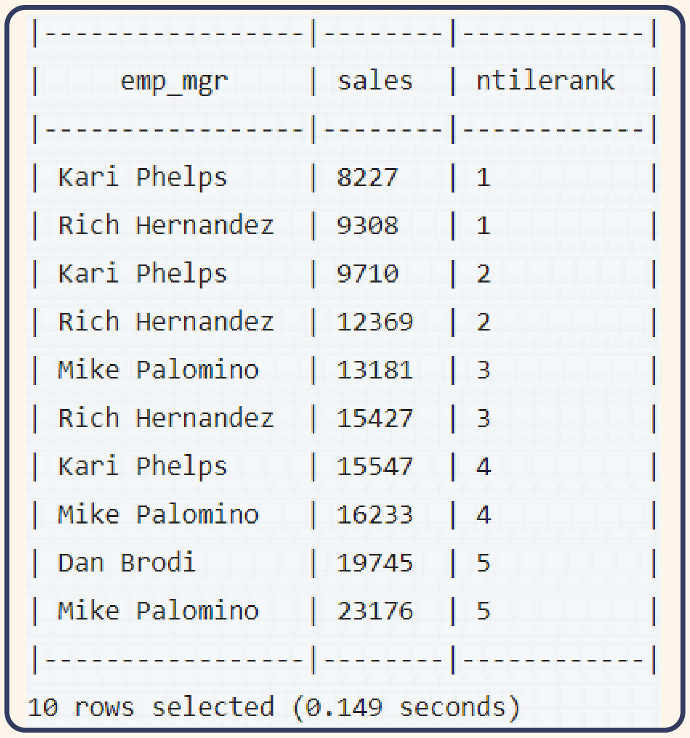
#56daysofSQL

Use Case Example of NTILE()

The following example uses the **NTILE** window function to divide the Q1 sales into five groups and list the sales in ascending order:

```
SELECT emp_mgr, sales,
NTILE(5) OVER(ORDER BY sales) as ntilerank
FROM q1_sales;
```

Output :



```
|-----------------|--------|------------|
|     emp_mgr     | sales  | ntilerank  |
|-----------------|--------|------------|
| Kari Phelps     | 8227   | 1          |
| Rich Hernandez  | 9308   | 1          |
| Kari Phelps     | 9710   | 2          |
| Rich Hernandez  | 12369  | 2          |
| Mike Palomino   | 13181  | 3          |
| Rich Hernandez  | 15427  | 3          |
| Kari Phelps     | 15547  | 4          |
| Mike Palomino   | 16233  | 4          |
| Dan Brodi       | 19745  | 5          |
| Mike Palomino   | 23176  | 5          |
|-----------------|--------|------------|
10 rows selected (0.149 seconds)
```

Aditya Shenoy