Window Functions

Naved Akhtar

Syntax

- window_function(arg1, arg2,..)
  OVER ([PARTITION BY
  partition_expression] [ORDER BY
  sort_expression [ASC | DESC]
  [NULLS {FIRST | LAST }])

PARTITION BY CLAUSE

- The PARTITION BY clause divides rows into multiple groups or partitions to which the window function is applied.
- The PARTITION BY clause is optional. If you skip the PARTITION BY clause, the window function will treat the whole result set as a single partition.

ORDER BY CLAUSE

- The ORDER BY clause specifies the order of rows in each partition to which the window function is applied.
- The ORDER BY clause uses the NULLS FIRST or NULLS LAST option to specify whether nullable values should be first or last in the result set. The default is NULLS LAST option.

Most Used Window Functions

Name	Description
ROW_NUMBER	Number the current row within its partition starting from 1.
RANK	Rank the current row within its partition with gaps.
DENSE_RANK	Rank the current row within its partition without gaps.
FIRST_VALUE	Return a value evaluated against the first row within its partition.
LAST_VALUE	Return a value evaluated against the last row within its partition.
LAG	Return a value evaluated at the row that is at a specified physical offset row before the current row within the partition.
LEAD	Return a value evaluated at the row that is offset rows after the current row within the partition.

Sample products and product_groups tables

Results Messages

	product_id	product_name	price	group_id
1	1	Microsoft Lumia	200.00	1
2	2	HTC One	400.00	1
3	3	Nexus	500.00	1
4	4	iPhone	900.00	1
5	5	HP Elite	1200.00	2
6	6	Lenovo Thinkpad	700.00	2
7	7	Sony VAIO	700.00	2
8	8	Dell Vostro	800.00	2
9	9	iPad	700.00	3
10	10	Kindle Fire	150.00	3
11	11	Samsung Galaxy Tab	200.00	3

	group_id	group_name
1	1	Smartphone
2	2	Laptop
3	3	Tablet

Row Number

The ROW_NUMBER() function assigns a sequential number to each row in each partition. See the following query

```
SELECT
product_name,
group_name,
price,
ROW_NUMBER () OVER (
PARTITION BY group_name
ORDER BY
price
) rn
FROM
products p
INNER JOIN product_groups pg
ON p.group_id=pg.group_id;
```

Row Number Output

	product_name	group_name	price	rn
1	Lenovo Thinkpad	Laptop	700.00	1
2	Sony VAIO	Laptop	700.00	2
3	Dell Vostro	Laptop	800.00	3
4	HP Elite	Laptop	1200.00	4
5	Microsoft Lumia	Smartphone	200.00	1
6	HTC One	Smartphone	400.00	2
7	Nexus	Smartphone	500.00	3
8	iPhone	Smartphone	900.00	4
9	Kindle Fire	Tablet	150.00	1
10	Samsung Galaxy Tab	Tablet	200.00	2
11	iPad	Tablet	700.00	3

RANK

The RANK() function assigns ranking within an ordered partition. If rows have the same values, the RANK() function assigns the same rank, with the next ranking(s) skipped.

DENSE_RANK

Similar to the RANK() function, the DENSE_RANK() function assigns a rank to each row within an ordered partition, but the ranks have no gap. In other words, the same ranks are assigned to multiple rows and no ranks are skipped.

FIRST_VALUE, LAST_VALUE

- The FIRST_VALUE() function returns a value evaluated against the first row within its partition, whereas the LAST_VALUE() function returns a value evaluated against the last row in its partition.

```
SELECT
 product_name,
 group_name,
 price,
 FIRST_VALUE (price) OVER (
 PARTITION BY group_name
 ORDER BY
 price
 ) AS lowest_price_per_group
 FROM
 products p
 INNER JOIN product_groups pg
 ON p.group_id=pg.group_id;
```

FIRST_VALUE OUTPUT

product_name	group_name	price	lowest_price_per_group
Lenovo Thinkpad	Laptop	700.00	700.00
Sony VAIO	Laptop	700.00	700.00
Dell Vostro	Laptop	800.00	700.00
HP Elite	Laptop	1200.00	700.00
Microsoft Lumia	Smartphone	200.00	200.00
HTC One	Smartphone	400.00	200.00
Nexus	Smartphone	500.00	200.00
iPhone	Smartphone	900.00	200.00
Kindle Fire	Tablet	150.00	150.00
Samsung Galaxy Tab	Tablet	200.00	150.00
iPad	Tablet	700.00	150.00

LEAD, LAG

- The LAG() function has the ability to access data from the previous row, while the LEAD() function can access data from the next row.

- Both LAG() and LEAD() functions have the same syntax as follows:
- LAG (expression [,offset] [,default]) over_clause;
- LEAD (expression [,offset] [,default]) over_clause;

LEAD, LAG CONTD.

```
SELECT

product_name,

group_name,

price,

LAG (price, 1) OVER (PARTITION BY
group_name ORDER BY price) AS prev_price,

price - LAG (price, 1) OVER (PARTITION BY
group_name ORDER BY price) AS
cur_prev_diff

FROM

products p

INNER JOIN product_groups pg

ON p.group_id=pg.group_id;
```

LEAD,LAG Output

product_name	group_name	price	prev_price	cur_prev_diff
Lenovo Thinkpad	Laptop	700.00	NULL	NULL
Sony VAIO	Laptop	700.00	700.00	0.00
Dell Vostro	Laptop	800.00	700.00	100.00
HP Elite	Laptop	1200.00	800.00	400.00
Microsoft Lumia	Smartphone	200.00	NULL	NULL
HTC One	Smartphone	400.00	200.00	200.00
Nexus	Smartphone	500.00	400.00	100.00
iPhone	Smartphone	900.00	500.00	400.00
Kindle Fire	Tablet	150.00	NULL	NULL
Samsung Galaxy Tab	Tablet	200.00	150.00	50.00
iPad	Tablet	700.00	200.00	500.00

Window Frame Clauses

ROWS BETWEEN

- This clause defines the window frame based on the number of rows before and after the current row.
- Example : `ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING`

RANGE BETWEEN

- This clause defines the window frame based on the values of the rows before and after the current row, rather than the number of rows.
- Example :`RANGE BETWEEN INTERVAL '1' DAY PRECEDING AND INTERVAL '1' DAY`

Window Frame Clauses (Absolute Boundaries)

UNBOUNDED PRECEEDING

- Starting point of a window frame
- Indicates that the window frame includes all rows from the partition's first row up to the current row.

UNBOUNDED FOLLOWING

- Ending point of a window frame
- Indicates that the window frame includes all rows from the current row upto the partition's last row

CURRENT ROW

- This keyword specifies the current row within the window frame

Window Frame Clauses (Absolute Boundaries) Contd.

```
SELECT
product_name,
group_name,
price,
LAST_VALUE (price) OVER (
PARTITION BY group_name
ORDER BY
price RANGE BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING
) AS highest_price_per_group
FROM
products p
INNER JOIN product_groups pg ON p.group_id=pg.group_id;
```

Window Frame Clauses (Absolute Boundaries) Output

product_name	group_name	price	highest_price_per_group
Lenovo Thinkpad	Laptop	700.00	1200.00
Sony VAIO	Laptop	700.00	1200.00
Dell Vostro	Laptop	800.00	1200.00
HP Elite	Laptop	1200.00	1200.00
Microsoft Lumia	Smartphone	200.00	900.00
HTC One	Smartphone	400.00	900.00
Nexus	Smartphone	500.00	900.00
iPhone	Smartphone	900.00	900.00
Kindle Fire	Tablet	150.00	700.00
Samsung Galaxy Tab	Tablet	200.00	700.00
iPad	Tablet	700.00	700.00

Thanks !

Any Questions?