SQL Window Functions

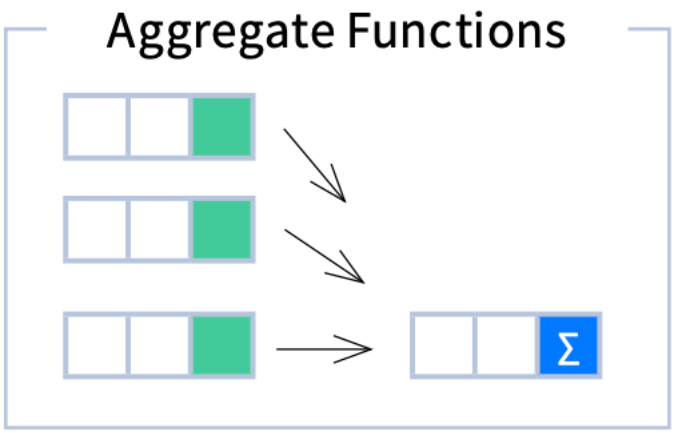
Kanru Wang

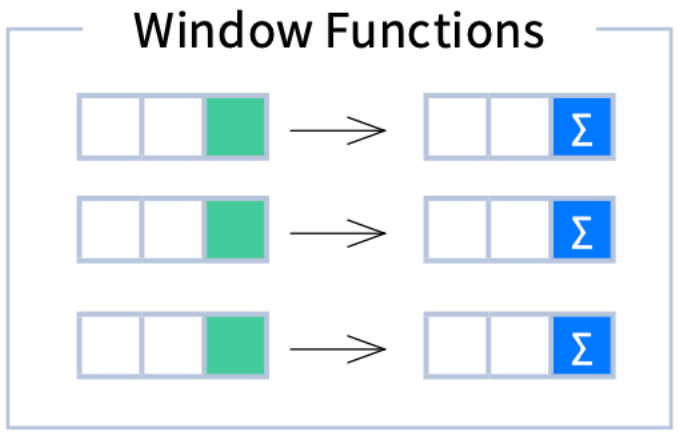
July 2022

AGGREGATE FUNCTIONS VS. WINDOW FUNCTIONS

Aggregate Functions

- avg()
- count()
- max()
- min()
- sum()





Ranking Functions

- row_number()
- rank()
- dense_rank()

Distribution Functions

- percent_rank()
- cume_dist()

Analytic Functions

- lead()
- lag()
- ntile()
- first_value()
- last_value()
- nth_value()

SYNTAX

Example

```
SELECT city, month,
    sum(sold) OVER (
        PARTITION BY city
        ORDER BY month
        RANGE UNBOUNDED PRECEDING) total
FROM sales;
```

General Pattern

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

PARTITION BY

divides rows into multiple groups, called **partitions**, to which the window function is applied.

month	city	sold
1	Rome	200
2	Paris	500
1	London	100
1	Paris	300
2	Rome	300
2	London	400
3	Rome	400

PARTITION BY city

month	city	sold	sum
1	Paris	300	800
2	Paris	500	800
1	Rome	200	900
2	Rome	300	900
3	Rome	400	900
1	London	100	500
2	London	400	500

ORDER BY

specifies the order of rows in each partition to which the window function is applied.

sold	city	month
200	Rome	1
500	Paris	2
100	London	1
300	Paris	1
300	Rome	2
400	London	2
400	Rome	3

PARTITION BY city ORDER BY month

sold	city	month
300	Paris	1
500	Paris	2
200	Rome	1
300	Rome	2
400	Rome	3
100	London	1
400	London	2

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	breed	running_total_weight
Charlie	British Shorthair	4.8
Smudge	British Shorthair	9.7
Tigger	British Shorthair	13.5
Millie	Maine Coon	5.4
Misty	Maine Coon	11.1
Ashes	Persian	4.5
Felix	Persian	9.5
Molly	Persian	13.7
Alfie	Siamese	5.5
Oscar	Siamese	11.6

Task

Order cats first by breed and second by name.
For each breed, want to know what the running total weight of the cats is.

Solution

```
SELECT
    name,
    breed,
    sum(weight) over (
        PARTITION BY breed
        ORDER BY name
    ) AS running_total_weight
FROM
    cats
ORDER BY
    breed,
    name
```

WINDOW FRAME

ROWS BETWEEN 1 PRECEDING
AND 1 FOLLOWING

city	sold	month
Paris	300	1
Rome	200	1
Paris	500	2
Rome	100	4
Paris	200	4
Paris	300	5
Rome	200	5
London	200	5
London	100	6
Rome	300	6

current row

1 row before the current row and
1 row after the current row

RANGE BETWEEN 1 PRECEDING
AND 1 FOLLOWING

city	sold	month
Paris	300	1
Rome	200	1
Paris	500	2
Rome	100	4
Paris	200	4
Paris	300	5
Rome	200	5
London	200	5
London	100	6
Rome	300	6

current row

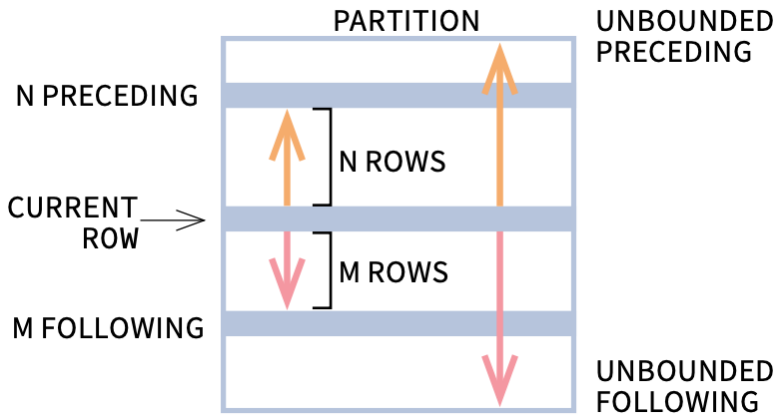
values in the range between 3 and 5
ORDER BY must contain a single expression

GROUPS BETWEEN 1 PRECEDING
AND 1 FOLLOWING

city	sold	month
Paris	300	1
Rome	200	1
Paris	500	2
Rome	100	4
Paris	200	4
Paris	300	5
Rome	200	5
London	200	5
London	100	6
Rome	300	6

current row

1 group before the current row and 1 group
after the current row regardless of the value




```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	weight	average_weight
Tigger	3.8	4.0
Molly	4.2	4.2
Ashes	4.5	4.5
Charlie	4.8	4.7
Smudge	4.9	4.9
Felix	5.0	5.0
Puss	5.1	5.2
Millie	5.4	5.3
Alfie	5.5	5.5
Misty	5.7	5.8
Oscar	6.1	6.0
Smokey	6.1	6.1

Task

Order by weight.
For each cat, calculate the average of the weight of: the cat, the cat just after it, and the cat just before it.
The first and last cats can just have an average weight of consisting of 2 cats not 3.

Solution

```
SELECT
    name,
    weight,
    avg(weight) over (
        ORDER BY weight
        ROWS BETWEEN 1 preceding AND 1 following
    ) AS average_weight
FROM
    cats
ORDER BY
    weight
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	running_total_weight
Smokey	6.1
Oscar	12.2
Misty	17.9
Alfie	23.4
Millie	28.8
Puss	33.9
Felix	38.9
Smudge	43.8
Charlie	48.6
Ashes	53.1
Molly	57.3

Task

Ordered by weight descending.
Calculate the running total weight.

Solution

```
SELECT
    name,
    sum(weight) over (
        ORDER BY weight DESC
        ROWS BETWEEN
            unbounded preceding AND current ROW
    ) AS running_total_weight
FROM
    cats
```

RANKING FUNCTIONS

city	price	row_number	rank	dense_rank
		over(order by price)		
Paris	7	1	1	1
Rome	7	2	1	1
London	8.5	3	3	2
Berlin	8.5	4	3	2
Moscow	9	5	5	3
Madrid	10	6	6	4
Oslo	10	7	6	4

`dense_rank()` – ranking within partition, with no gaps and same ranking for tied values

- `rank()` and `dense_rank()` require ORDER BY
- `row_number()` does not require ORDER BY
- None of the three ranking functions accept window frame definition (ROWS, RANGE, GROUPS).

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

unique_number	name	colour
1	Ashes	Black
2	Charlie	Black
3	Molly	Black
4	Oscar	Black
5	Smudge	Black
6	Alfie	Brown
7	Misty	Brown
8	Smokey	Brown
9	Felix	Tortoiseshell
10	Millie	Tortoiseshell
11	Puss	Tortoiseshell
12	Tigger	Tortoiseshell

Task

Order by colour and name.
Assign each cat a unique number.

Solution

```
SELECT
    row_number() over (
        ORDER BY
            colour,
            name
    ) AS unique_number,
    name,
    colour
FROM
    cats
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

ranking	dense_ranking	weight	name
1	1	6.1	Oscar
1	1	6.1	Smokey
3	2	5.7	Misty
4	3	5.5	Alfie
5	4	5.4	Millie
6	5	5.1	Puss
7	6	5.0	Felix
8	7	4.9	Smudge
9	8	4.8	Charlie
10	9	4.5	Ashes
11	10	4.2	Molly
12	11	3.8	Tigger

Task

Rank cats by weight.
The two heaviest cats should both be 1st.
Then create two ranks:
- One would rank the next heaviest 3rd.
- The other would rank the next heaviest 2nd.

Order by ranking and name.

Solution

```
SELECT
    rank() over (
        ORDER BY weight DESC
    ) AS ranking,
    dense_rank() over (
        ORDER BY weight DESC
    ) AS dense_ranking,
    weight,
    name
FROM
    cats
ORDER BY
    ranking,
    name
```

DISTRIBUTION FUNCTIONS

`percent_rank() OVER(ORDER BY sold)`

city	sold	percent_rank
Paris	100	0
Berlin	150	0.25
Rome	200	0.5
Moscow	200	0.5
London	300	1

without this row 50% of values are less than this row's value

`cume_dist() OVER(ORDER BY sold)`

city	sold	cume_dist
Paris	100	0.2
Berlin	150	0.4
Rome	200	0.8
Moscow	200	0.8
London	300	1

80% of values are less than or equal to this one

`percent_rank()` is
`(rank - 1) / (total number of rows - 1)`

`cume_dist()` is
`(number of rows <= the value) / total number of rows`

`percent_rank()` and `cume_dist()`
- require ORDER BY
- do not accept window frame definition (ROWS, RANGE, GROUPS)

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	weight	running_percent	running_percentile
Tigger	3.8	0.0	8
Molly	4.2	9.1	17
Ashes	4.5	18.2	25
Charlie	4.8	27.3	33
Smudge	4.9	36.4	42
Felix	5.0	45.5	50
Puss	5.1	54.5	58
Millie	5.4	63.6	67
Alfie	5.5	72.7	75
Misty	5.7	81.8	83
Oscar	6.1	90.9	100
Smokey	6.1	90.9	100

Task

Each cat would like to know

- What percentage of other cats weigh less than it.
- What weight percentile it is in. Cast to an integer.

Order by weight.

Solution

```
SELECT
    name,
    weight,
    percent_rank() over (
        ORDER BY weight
    ) * 100 AS running_percent,
    cast(
        cume_dist() over (
            ORDER BY weight
        ) * 100 AS integer
    ) AS running_percentile
FROM
    cats
ORDER BY
    weight
```

ANALYTIC FUNCTIONS

`lead(sold) OVER(ORDER BY month)`

order by month

month	sold	
1	500	300
2	300	400
3	400	100
4	100	500
5	500	NULL

`lag(sold) OVER(ORDER BY month)`

order by month

month	sold	
1	500	NULL
2	300	500
3	400	300
4	100	400
5	500	100

`lead(sold, 2, 0) OVER(ORDER BY month)`

order by month

month	sold	
1	500	400
2	300	100
3	400	500
4	100	0
5	500	0

offset=2

`lag(sold, 2, 0) OVER(ORDER BY month)`

order by month

month	sold	
1	500	0
2	300	0
3	400	500
4	100	300
5	500	400

offset=2

`ntile(3)`

city	sold		
Rome	100	1	1
Paris	100		1
London	200		1
Moscow	200	2	2
Berlin	200		2
Madrid	300		2
Oslo	300	3	3
Dublin	300		3

`ntile(n)` – divides rows within a partition as equally as possible into n groups, and assign each row its group number.

`ntile()`, `lead()`, and `lag()`

- require an ORDER BY
- do not accept window frame definition (ROWS, RANGE, GROUPS).

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	weight	weight_quartile
Tigger	3.8	1
Molly	4.2	1
Ashes	4.5	1
Charlie	4.8	2
Smudge	4.9	2
Felix	5.0	2
Puss	5.1	3
Millie	5.4	3
Alfie	5.5	3
Misty	5.7	4
Oscar	6.1	4
Smokey	6.1	4

Task

Group the cats into quartiles by their weight.
Order by weight.

Solution

```
SELECT
    name,
    weight,
    ntile(4) over (
        ORDER BY weight
    ) AS weight_quartile
FROM
    cats
ORDER BY
    weight
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	breed	weight	weight_lose_from_prev	weight_lose_by_breed
Tigger	British Shorthair	3.8	0.0	0.0
Molly	Persian	4.2	0.4	0.0
Ashes	Persian	4.5	0.3	0.3
Charlie	British Shorthair	4.8	0.3	1.0
Smudge	British Shorthair	4.9	0.1	0.1
Felix	Persian	5.0	0.1	0.5
Puss	Maine Coon	5.1	0.1	0.0
Millie	Maine Coon	5.4	0.3	0.3
Alfie	Siamese	5.5	0.1	0.0
Misty	Maine Coon	5.7	0.2	0.3
Oscar	Siamese	6.1	0.4	0.6
Smokey	Maine Coon	6.1	0.0	0.4

Task

- Each cat wants to lose weight to be the equivalent weight of the cat weighting just less than it.
- Each cat wants to lose weight to be the equivalent weight of the cat in the same breed weighing just less than it.

Find out how much weight should they lose. Order by weight.

Solution

```
SELECT
    name,
    breed,
    weight,
    coalesce(
        weight - lag(weight, 1) over (
            ORDER BY weight
        ),
        0
    ) AS weight_lose_from_prev,
    coalesce(
        weight - lag(weight, 1) over (
            PARTITION by breed
            ORDER BY weight
        ),
        0
    ) AS weight_lose_by_breed
FROM
    cats
ORDER BY
    weight
```

`COALESCE()` returns the first non-null value in a list

ANALYTIC FUNCTIONS

`first_value(sold) OVER (PARTITION BY city ORDER BY month)`

city	month	sold	first_value
Paris	1	500	500
Paris	2	300	500
Paris	3	400	500
Rome	2	200	200
Rome	3	300	200
Rome	4	500	200

`last_value(sold) OVER (PARTITION BY city ORDER BY month` **`RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING`**`)`

city	month	sold	last_value
Paris	1	500	400
Paris	2	300	400
Paris	3	400	400
Rome	2	200	500
Rome	3	300	500
Rome	4	500	500

`nth_value(sold, 2) OVER (PARTITION BY city ORDER BY month RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)`

city	month	sold	nth_value
Paris	1	500	300
Paris	2	300	300
Paris	3	400	300
Rome	2	200	300
Rome	3	300	300
Rome	4	500	300
Rome	5	300	300
London	1	100	NULL

If ORDER BY is specified, the default window frame is RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
So if we want `last_value()`, we must specify RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

`first_value()`, `last_value()`, and `nth_value()`
- do not require an ORDER BY
- accept window frame definition (ROWS, RANGE, GROUPS)

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>;
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	colour	weight_by_colour
Ashes	Black	4.2
Charlie	Black	4.2
Molly	Black	4.2
Oscar	Black	4.2
Smudge	Black	4.2
Alfie	Brown	5.5
Misty	Brown	5.5
Smokey	Brown	5.5
Felix	Tortoiseshell	3.8
Millie	Tortoiseshell	3.8
Puss	Tortoiseshell	3.8
Tigger	Tortoiseshell	3.8

Task

Assign each cat the lowest weight for its colour.
Order by colour and name

Solution

```
SELECT
    name,
    colour,
    first_value(weight) over (
        PARTITION by colour
        ORDER BY weight
    ) AS weight_by_colour
FROM
    cats
ORDER BY
    colour,
    name
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	weight	breed	next_heaviest
Tigger	3.8	British Shorthair	4.8
Molly	4.2	Persian	4.5
Ashes	4.5	Persian	5
Charlie	4.8	British Shorthair	4.9
Smudge	4.9	British Shorthair	fattest cat
Felix	5.0	Persian	fattest cat
Puss	5.1	Maine Coon	5.4
Millie	5.4	Maine Coon	5.7
Alfie	5.5	Siamese	6.1
Misty	5.7	Maine Coon	6.1
Oscar	6.1	Siamese	fattest cat
Smokey	6.1	Maine Coon	fattest cat

Task

Assign each cat the next heaviest cat's weight when grouped by breed.
If there is no heavier cat, print 'fattest cat'.
Order by weight.

Solution

```
SELECT
    name,
    weight,
    breed,
    coalesce(
        cast(
            lead(weight, 1) over (
                PARTITION by breed
                ORDER BY weight
            ) AS varchar
        ),
        'fattest cat'
    ) AS next_heaviest
FROM
    cats
ORDER BY
    weight
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	weight	imagined_weight
Tigger	3.8	3.9
Molly	4.2	3.9
Ashes	4.5	3.9
Charlie	4.8	4.8
Smudge	4.9	4.8
Felix	5.0	4.8
Puss	5.1	4.8
Millie	5.4	4.8
Alfie	5.5	4.8
Misty	5.7	4.8
Oscar	6.1	4.8
Smokey	6.1	4.8

Task

Assign each cat the weight of the 4th lightest cat. All cats should have this weight, except for the lightest three, which should each be assigned a weight of 3.9.
Order by weight.

Solution

```
SELECT
    name,
    weight,
    coalesce(
        nth_value(weight, 4) over (
            ORDER BY weight
        ),
        3.9
    ) AS imagined_weight
FROM
    cats
ORDER BY
    weight
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

breed	imagined_weight
British Shorthair	4.8
Maine Coon	5.4
Persian	4.5
Siamese	6.1

Task

Use the 2nd lightest cat's weight of each breed to represent the weight of each breed.

Solution

```
SELECT
    DISTINCT
    breed,
    nth_value(weight, 2) over (
        PARTITION by breed
        ORDER BY weight
        RANGE BETWEEN
            UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
    ) AS imagined_weight
FROM
    cats
ORDER BY
    breed
```

Named Window Definition

Example

```
SELECT country, city,
    rank() OVER country_sold_avg
FROM sales
WHERE month BETWEEN 1 AND 6
GROUP BY country, city
HAVING sum(sold) > 10000
WINDOW country_sold_avg AS (
    PARTITION BY country
    ORDER BY avg(sold) DESC)
ORDER BY country, city;
```

General Pattern

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER <window_name>
FROM <table_name>
WHERE <...>
GROUP BY <...>
HAVING <...>
WINDOW <window_name> AS (
    PARTITION BY <...>
    ORDER BY <...>
    <window_frame>)
ORDER BY <...>;
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

name	weight	by_half	thirds	quart
Tigger	3.8	1	1	1
Molly	4.2	1	1	1
Ashes	4.5	1	1	1
Charlie	4.8	1	1	2
Smudge	4.9	1	2	2
Felix	5.0	1	2	2
Puss	5.1	2	2	3
Millie	5.4	2	2	3
Alfie	5.5	2	3	3
Misty	5.7	2	3	4
Oscar	6.1	2	3	4
Smokey	6.1	2	3	4

Task

Calculate for each cat, the half, third and quartile they are in for their weight.

Solution

```
SELECT
    name,
    weight,
    ntile(2) over ntile_window AS by_half,
    ntile(3) over ntile_window AS thirds,
    ntile(4) over ntile_window AS quart
FROM
    cats
WINDOW
    ntile_window AS (
        ORDER BY weight
    )
ORDER BY
    weight
```

LOGICAL ORDER OF OPERATIONS IN SQL

1. FROM, JOIN
2. WHERE
3. GROUP BY
4. aggregate functions
5. HAVING
6. **window functions**
7. SELECT
8. DISTINCT
9. UNION/INTERSECT/EXCEPT
10. ORDER BY
11. OFFSET
12. LIMIT/FETCH/TOP

Can use window functions in SELECT and ORDER BY.
Cannot put window functions in the FROM, WHERE, GROUP BY, or HAVING clauses.

```
SELECT <column_1>, <column_2>,
    <window_function>() OVER (
        PARTITION BY <...>
        ORDER BY <...>
        <window_frame>) <window_column_alias>
FROM <table_name>
WHERE <...>
GROUP BY <...>
HAVING <...>
ORDER BY <...>
```

Bonus

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

colour	names
Tortoiseshell	Felix,Tigger,Millie,Puss
Brown	Alfie,Misty,Smokey
Black	Ashes,Molly,Smudge,Oscar,Charlie

Task

Group cats by colour.
Each row containing a colour and a list of cat names.

Solution

```
SELECT
    colour,
    array_agg(name) AS NAMES
FROM
    cats
GROUP BY
    colour
```

Cat Table

Name	Breed	Weight	Colour
Andy	Persian	50	Black
Bob	Siamese	100	Black
Cedric	British Shorthair	50	Black
Dave	Siamese	70	Brown
Eric	Maine Coon	70	Brown

Desired Output

breed	average_weight	average_old_weight
Persian	4.6	4.8
British Shorthair	4.5	4.5
Maine Coon	5.6	5.6
Siamese	5.8	5.5

Task

Group

Solution

```
SELECT
    breed,
    avg(weight) AS average_weight,
    avg(weight) filter (
        WHERE
            age > 1
    ) average_old_weight
FROM
    cats
GROUP BY
    breed
```

SQL Window Functions	Python Pandas

`ROW_NUMBER() over (`
`  PARTITION BY ticker, exchange`
`  ORDER BY date DESC, period`
`) as example`
	`df['example'] = (`
	`  df.sort_values(['date', 'period'], ascending=[False, True])`
	`  .groupby(['ticker', 'exchange'])`
	`  .cumcount() + 1`
	`)`

`LEAD(price) over (`
`  PARTITION BY ticker`
`  ORDER BY date`
`) as next_day_px`
	`df['next_day_px'] = (`
	`  df.sort_values(by=['date'], ascending=True)`
	`  .groupby(['ticker'])['price']`
	`  .shift(-1)  # shift(1) for LAG()`
	`)`

`PERCENT_RANK() OVER (`
`  PARTITION BY ticker, year`
`  ORDER BY price`
`) as perc_price`
	`df['perc_price'] = (`
	`  df.groupby(['ticker', 'year'])['price']`
	`  .rank(pct=True)`
	`)`

`SUM(trade_vol) OVER (`
`  PARTITION BY ticker`
`  ORDER BY date`
`  ROWS BETWEEN 3 PRECEEDING AND CURRENT ROW`
`) as volume_3day`
	`df['volume_3day'] = (`
	`  df.sort_values(by=['date'], ascending=True)`
	`  .groupby(['ticker'])['trade_vol']`
	`  .rolling(3, min_periods = 1).sum()  # .mean() to get average`
	`  .reset_index(drop=True, level=0)`
	`)`

`SUM(trade_vol) OVER (`
`  PARTITION BY ticker`
`  ORDER BY date`
`  ROWS BETWEEN UNBOUNDED PRECEEDING AND CURRENT ROW`
`) as cum_total_vol`
	`df['cum_total_vol'] = (`
	`  df.sort_values(by=['date'], ascending=True)`
	`  .groupby(['ticker'])['trade_vol']`
	`  .cumsum()`
	`)`

`SUM(trade_vol) OVER (`
`  PARTITION BY ticker`
`) as total_vol`
	`df['total_vol'] = (`
	`  df.groupby(['ticker','year'])['trade_vol']`
	`  .transform('sum')  # .transform('mean') to get average`
	`)  # can provide a function to .transform()`

10 Tips For Writing Efficient And Faster SQL Queries

https://andrewrgoss.com/pdf/sql_query_optimization_techniques.pdf

REFERENCE

- https://learnsql.com/blog/sql-window-functions-cheat-sheet/
- https://www.windowfunctions.com/
- https://stackoverflow.com/questions/19175734/tsql-partition-by-with-order-by
- https://towardsdatascience.com/sql-window-functions-in-python-pandas-data-science-dc7c7a63cbb4
- https://engineeringfordatascience.com/posts/sql_like_window_functions_in_pandas/