Aggregate window functions

WINDOW FUNCTIONS IN SNOWFLAKE

Jake Roach
Field Data Engineer

Aggregate window functions

m_id	wd	cb	num_workouts	total_cb	cb_vs_average
m_192	2024-01-01	105	4	432	-3
m_192	2024-01-03	156	4	432	48
m_192	2024-01-04	69	4	432	-39
m_192	2024-01-10	102	4	432	-6
m_74	2024-02-10	374	3	1274	-50.67
m_74	2024-02-13	396	3	1274	-28.67
m_74	2024-02-14	504	3	1274	-79.33
m_233	2024-03-05	51	2	132	-15
m_233	2024-03-12	81	2	132	15

Aggregate window functions, AVG

```
SELECT
    <fields>,

    -- No need to ORDER BY!
    AVG(<1>) OVER(
        PARTITION BY <2>
    ) AS <alias>

...
```

`<1>` : column to take the average of

`<2>` : field to partition data by

- Calculates a **summary metric** and shows the result for **each record in a window**
- Leverage the output of these functions using operators such as `+` , `-` , `*` , `/`
- `AVG` , `COUNT` , `SUM`
- No need to `ORDER BY` !

AVG

```
SELECT
    member_id AS m_id,
    calories_burned AS cb,

    -- First, the average calories burned
    AVG(calories_burned) OVER(
        PARTITION BY member_id
    ) AS avg_cb,

    -- Then, the difference vs. the average
    calories_burned - AVG(calories_burned) OVER(
        PARTITION BY member_id
    ) AS cb_vs_average

FROM fitness.workouts;
```

```
 m_id  |  cb  |  avg_cb  | cb_vs_average
------ | ---- | -------- | -------------
m_192  | 105  |   108    |      -3
m_192  | 156  |   108    |      48
m_192  |  69  |   108    |     -39
m_192  | 102  |   108    |      -6

m_74   | 374  |  424.67  |    -50.67
m_74   | 396  |  424.67  |    -28.67
m_74   | 504  |  424.67  |     79.33

m_233  |  51  |    66    |     -15
m_233  |  81  |    66    |      15
```

COUNT

```
SELECT
    member_id AS m_id,
    workout_date AS wd,

    -- Can pass '*' to COUNT
    COUNT(*) OVER(
        PARTITION BY member_id
    ) AS num_workouts

FROM fitness.workouts
ORDER BY member_id, workout_date;
```

Can pass a `*` to `COUNT`!

```
 m_id  |     wd     | num_workouts
------ | ---------- | -------------
m_192  |  2024-01-01  |         4
m_192  |  2024-01-03  |         4
m_192  |  2024-01-04  |         4
m_192  |  2024-01-10  |         4

m_74   |  2024-02-10  |         3
m_74   |  2024-02-13  |         3
m_74   |  2024-02-14  |         3

m_233  |  2024-03-05  |         2
m_233  |  2024-03-12  |         2
```

SUM

```
SELECT
    member_id AS m_id,
    calories_burned AS cb,

    -- Total calories burned for member id
    SUM(calories_burned) OVER(
        PARTITION BY member_id
    ) AS total_cb,

    -- Find proportion of total
    calories_burned / SUM(calories_burned) OVER(
        PARTITION BY member_id
    ) AS total_cb

FROM fitness.workouts;
```

```
 m_id  |  cb  | total_cb | prop_cb
------ | ---- | -------- | --------
m_192  | 105  |    432   |  0.2431
m_192  | 156  |    432   |  0.3611
m_192  |  69  |    432   |  0.1597
m_192  | 102  |    432   |  0.2361

m_74   | 374  |   1274   |  0.2935
m_74   | 396  |   1274   |  0.3108
m_74   | 504  |   1274   |  0.3956

m_233  | 51   |    132   |  0.3864
m_233  | 81   |    132   |  0.6136
```

Evaluating member workouts

```
SELECT
    member_id AS m_id, workout_date AS wd, calories_burned AS cb,

    COUNT(*) OVER(                                  -- Find the count of workouts
        PARTITION BY member_id
    ) AS num_workouts,

    SUM(calories_burned) OVER(                      -- Total # of calories burned for each member
        PARTITION BY member_id
    ) AS total_cb,

    calories_burned - AVG(calories_burned) OVER(    -- Compare calories burned in workout to average
        PARTITION BY member_id
    ) AS cb_vs_average

FROM fitness.workouts;
```

Evaluating member workouts

m_id	wd	cb	num_workouts	total_cb	cb_vs_average
m_192	2024-01-01	105	4	432	-3
m_192	2024-01-03	156	4	432	48
m_192	2024-01-04	69	4	432	-39
m_192	2024-01-10	102	4	432	-6
m_74	2024-02-10	374	3	1274	-50.67
m_74	2024-02-13	396	3	1274	-28.67
m_74	2024-02-14	504	3	1274	79.33
m_233	2024-03-05	51	2	132	-15
m_233	2024-03-12	81	2	132	15

Let's practice!

WINDOW FUNCTIONS IN SNOWFLAKE

Window frames

WINDOW FUNCTIONS IN SNOWFLAKE

Jake Roach
Field Data Engineer

Window frames

member_id	workout_date	calories_burned	average_calories_burned
m_192	2025-01-01	105	204.2
m_192	2025-01-03	156	204.2
m_192	2025-01-04	69	204.2
m_192	2025-01-10	102	204.2
m_74	2025-02-10	374	204.2
m_74	2025-02-13	396	204.2
m_74	2025-02-14	504	204.2
m_233	2025-03-05	51	204.2
m_233	2025-03-12	81	204.2

Window frames

member_id	workout_date	calories_burned	average_calories_burned
m_192	2025-01-01	105	108
m_192	2025-01-03	156	108
m_192	2025-01-04	69	108
m_192	2025-01-10	102	108
m_74	2025-02-10	374	424.67
m_74	2025-02-13	396	424.67
m_74	2025-02-14	504	424.67
m_233	2025-03-05	51	66
m_233	2025-03-12	81	66

Dynamic window frames

member_id	workout_date	calories_burned	average_calories_burned
m_192	2025-01-01	105	105.0
m_192	2025-01-03	156	130.5
m_192	2025-01-04	69	110.0
m_192	2025-01-10	102	108.0
m_74	2025-02-10	374	374.0
m_74	2025-02-13	396	385.0
m_74	2025-02-14	504	426.7
m_233	2025-03-05	51	51.0
m_233	2025-03-12	81	66.0

Finding a running calculation

```
SELECT
    ...

    AVG(<1>) OVER(
        PARTITION BY ...
        ORDER BY <2>

        -- Window between the first
        -- and current row

        ROWS BETWEEN UNBOUNDED PRECEDING
            AND CURRENT ROW
    )
...
```

`ROWS BETWEEN` allows us to create a dynamic window frame

- `UNBOUNDED PRECEEDING AND CURRENT ROW`
- First row in sequence until current row
- "Running" calculation, not rolling
- `CURRENT ROW AND UNBOUNDED FOLLOWING`

`<1>` : field to take calculation of

`<2>` : sequences results, builds window frame

Running total of calories burned

```
SELECT
    member_id AS m_id,
    calories_burned AS cb,

    -- Running total!
    SUM(calories_burned) OVER(
        PARTITION BY member_id
        ORDER BY workout_date

        ROWS BETWEEN UNBOUNDED PRECEDING
            AND CURRENT ROW

    ) AS running_total

FROM fitness.workouts;
```

```
 m_id  |  cb  | running_total
-------|------|--------------
 m_192 |  105 |       105
 m_192 |  156 |       261
 m_192 |   69 |       330
 m_192 |  102 |       432

 m_74  |  374 |       374
 m_74  |  396 |       770
 m_74  |  504 |      1274

 m_233 |   51 |        51
 m_233 |   81 |       132
```

CURRENT ROW AND UNBOUNDED FOLLOWING

```
SELECT
    member_id AS m_id,
    calories_burned AS cb,

    SUM(calories_burned) OVER(
        PARTITION BY member_id
        ORDER BY workout_date

        -- Window between current row and last
        ROWS BETWEEN CURRENT ROW
            AND UNBOUNDED FOLLOWING

    ) AS left_to_burn

FROM fitness.workouts;
```

```
 m_id  |  cb  | left_to_burn
-------|------|-------------
 m_192 | 105  |      432
 m_192 | 156  |      327
 m_192 |  69  |      171
 m_192 | 102  |      102

 m_74  | 374  |     1274
 m_74  | 396  |      900
 m_74  | 504  |      504

 m_233 |  51  |      132
 m_233 |  81  |       81
```

Calorie-burning trends

```
SELECT
    member_id,
    workout_date,
    calories_burned,

    AVG(calories_burned) OVER(   -- Running average of calories burned
        PARTITION BY member_id

        -- Crate a window by workout date, from the first workout to the current workout
        ORDER BY workout_date
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

    ) AS avg_calories_burned

FROM FITNESS.workouts;
```

Calorie-burning trends

member_id	workout_date	calories_burned	average_calories_burned
m_192	2025-01-01	105	105.0
m_192	2025-01-03	156	130.5
m_192	2025-01-04	69	110.0
m_192	2025-01-10	102	108.0
m_74	2025-02-10	374	374.0
m_74	2025-02-13	396	385.0
m_74	2025-02-14	504	426.7
m_233	2025-03-05	51	51.0
m_233	2025-03-12	81	66.0

Let's practice!

WINDOW FUNCTIONS IN SNOWFLAKE

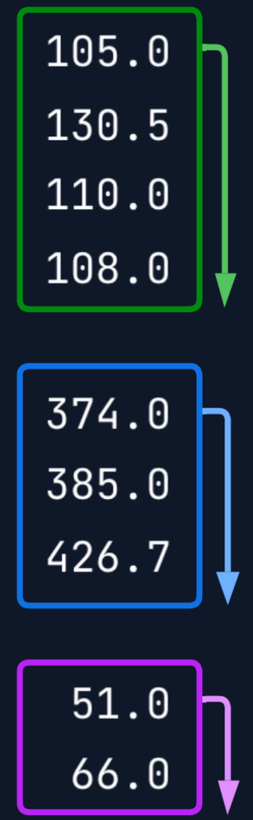
Moving Averages and Totals

WINDOW FUNCTIONS IN SNOWFLAKE

Jake Roach
Field Data Engineer

Running averages and totals

member_id	workout_date	calories_burned	average_calories_burned
m_192	2025-01-01	105	105.0
m_192	2025-01-03	156	130.5
m_192	2025-01-04	69	110.0
m_192	2025-01-10	102	108.0
m_74	2025-02-10	374	374.0
m_74	2025-02-13	396	385.0
m_74	2025-02-14	504	426.7
m_233	2025-03-05	51	51.0
m_233	2025-03-12	81	66.0



"Moving" averages and totals

member_id	workout_date	calories_burned	moving_avg_cb	last_3_cb
m_192	2024-01-01	105	130.5	105
m_192	2024-01-03	156	110.0	261
m_192	2024-01-04	69	109.0	330
m_192	2024-01-10	102	120.0	327
m_192	2024-01-11	189	127.3	360
m_192	2024-01-12	91	145.0	382
m_192	2024-01-16	155	127.7	435
m_192	2024-01-19	137	133.7	383
m_192	2024-01-20	109	123.0	401
m_74	2024-02-10	374	385.0	374
		...		

"Moving" averages and totals

member_id	workout_date	calories_burned	moving_avg_cb	last_3_cb
m_192	2024-01-01	105	130.5	105
m_192	2024-01-03	156	110.0	261
m_192	2024-01-04	69	109.0	330
m_192	2024-01-10	102	120.0	327
m_192	2024-01-11	189	127.3	360
m_192	2024-01-12	91	145.0	382
m_192	2024-01-16	155	127.7	435
m_192	2024-01-19	137	133.7	383
m_192	2024-01-20	109	123.0	401
m_74	2024-02-10	374	385.0	374
		...		

Moving calculations

```
SELECT
    ...

    AVG(<1>) OVER(
        PARTITION BY ...
        ORDER BY <2>

        -- Window between the X preceding
        -- and Y following records

        ROWS BETWEEN <X> PRECEDING
            AND <Y> FOLLOWING
    )
...
```

Specify the number of records before and after the current row

```
ROWS BETWEEN X PRECEDING AND Y FOLLOWING
```

- **"Moving" calculation**, not running
- Can still use `CURRENT ROW`

`<X>` : # of rows to look back

`<Y>` : # of rows to look ahead

Creating a moving average

```
SELECT
    member_id, workout_date, calories_burned,

    -- Create a moving average using a window using the previous workout,
    -- current workout, and next workout

    AVG(calories_burned) OVER(
        PARTITION BY member_id
        ORDER BY workout_date
        ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
    ) AS moving_avg_cb

FROM FITNESS.workouts;
```

Creating a moving average

member_id	workout_date	calories_burned	moving_avg_cb
m_192	2024-01-01	105	130.5
m_192	2024-01-03	156	110.0
m_192	2024-01-04	69	109.0
m_192	2024-01-10	102	120.0
m_192	2024-01-11	189	127.3
m_192	2024-01-12	91	145.0
m_192	2024-01-16	155	127.7
m_192	2024-01-19	137	133.7
m_192	2024-01-20	109	123.0
m_74	2024-02-10	374	385.0
...			

Informing members with moving totals

```
SELECT
    member_id, workout_date, calories_burned,

    ...

    -- Use CURRENT ROW to avoid "look ahead"
    SUM(calories_burned) OVER(
        PARTITION BY member_id
        ORDER BY workout_date
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ) AS last_3_cb

FROM FITNESS.workouts;
```

Informing members with moving totals

member_id	workout_date	calories_burned	moving_avg_cb	last_3_cb
m_192	2024-01-01	105	130.5	105
m_192	2024-01-03	156	110.0	261
m_192	2024-01-04	69	109.0	330
m_192	2024-01-10	102	120.0	327
m_192	2024-01-11	189	127.3	360
m_192	2024-01-12	91	145.0	382
m_192	2024-01-16	155	127.7	435
m_192	2024-01-19	137	133.7	383
m_192	2024-01-20	109	123.0	401
m_74	2024-02-10	374	385.0	374
		...		

Let's practice!

WINDOW FUNCTIONS IN SNOWFLAKE

Congratulations!

WINDOW FUNCTIONS IN SNOWFLAKE

Jake Roach
Field Data Engineer

datacamp

Thank you!

WINDOW FUNCTIONS IN SNOWFLAKE