

bite-sized.sql

SQL: NAMED WINDOWS

bite-sized.sql

**Window
functions are
incredibly
powerful**

E.g. Monthly aggregates

```
SELECT
  trip_month,
  trip_duration,
  AVG(trip_duration)
    OVER (PARTITION BY trip_month) AS av_duration,
  MIN(trip_duration)
    OVER (PARTITION BY trip_month) AS mn_duration,
  MAX(trip_duration)
    OVER (PARTITION BY trip_month) AS mx_duration,
  ROW_NUMBER()
    OVER (PARTITION BY trip_month
          ORDER BY trip_duration) AS row_num
FROM trips;
```

The function is applied over a “window” defined by the partition. In this case, calculate the aggregates over each unique value of trip_month

Some window functions include an ORDER BY clause



If we include neither PARTITION BY nor ORDER BY, the window becomes the entire result set, allowing aggregates without GROUP BY. For example, this returns the average over the whole table:

```
AVG(trip_duration) OVER () AS av_duration
```

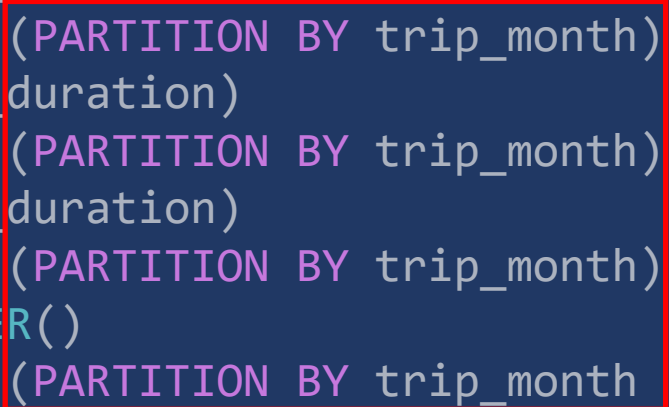
bite-sized.sql

**But this can
lead to messy
code**

E.g. The same PARTITION BY is repeated four times

```
SELECT
  trip_month,
  trip_duration,
  AVG(trip_duration)
    OVER (PARTITION BY trip_month) AS av_duration,
  MIN(trip_duration)
    OVER (PARTITION BY trip_month) AS mn_duration,
  MAX(trip_duration)
    OVER (PARTITION BY trip_month) AS mx_duration,
  ROW_NUMBER()
    OVER (PARTITION BY trip_month
          ORDER BY trip_duration) AS row_num
FROM trips;
```

Repetition



bite-sized.sql

The WINDOW clause lets us name windows

E.g. Use the WINDOW clause to name the window

- 2 The window name **w** is re-used in each window function

```
SELECT
  trip_month,
  trip_duration,
  AVG(trip_duration) OVER w AS av_duration,
  MIN(trip_duration) OVER w AS mn_duration,
  MAX(trip_duration) OVER w AS mx_duration,
  ROW_NUMBER() OVER (w ORDER BY trip_duration) AS row
FROM trips
WINDOW w AS (PARTITION BY trip_month);
```

The diagram illustrates the reuse of the window name 'w'. A red arrow points from the 'w' in the first three window functions (AVG, MIN, MAX) to the 'w' in the ROW_NUMBER() function. Another red arrow points from the 'w' in the ROW_NUMBER() function to the 'w' in the WINDOW clause definition. A third red arrow points from the 'w' in the WINDOW clause definition back to the 'w' in the first three window functions, forming a cycle that highlights how the same name is used for all windows in the query.

- 1 The common feature of each of the windows (i.e. the **PARTITION BY**) is given the name **w**

- 3 If needed, we can extend the window definition for specific functions. Here we're using **w**, then adding the **ORDER BY** clause needed for **ROW_NUMBER()**

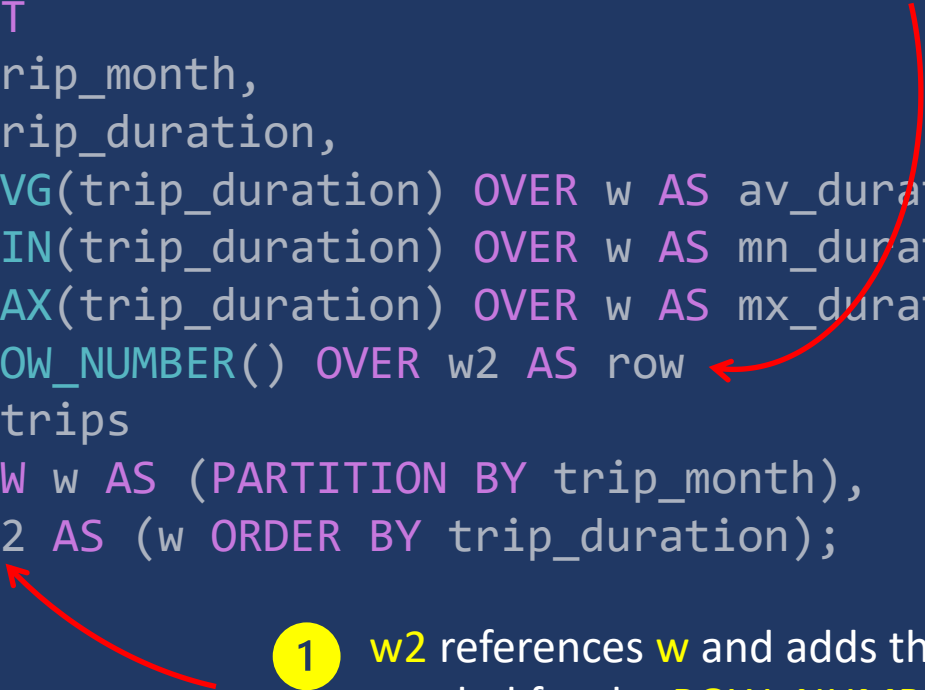
bite-sized.sql

Names can reference other names

Names can reference other names

- 2 **ROW_NUMBER()** now simply uses the **w2** name

```
SELECT
    trip_month,
    trip_duration,
    AVG(trip_duration) OVER w AS av_duration,
    MIN(trip_duration) OVER w AS mn_duration,
    MAX(trip_duration) OVER w AS mx_duration,
    ROW_NUMBER() OVER w2 AS row
FROM trips
WINDOW w AS (PARTITION BY trip_month),
       w2 AS (w ORDER BY trip_duration);
```



- 1 **w2** references **w** and adds the **ORDER BY** clause needed for the **ROW_NUMBER()** function

WINDOW clause:

1. Tidier code
2. Easy to change existing windows consistently
3. Easy to create new windows by extending existing windows