Introduction

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

Michel Semaan
Data Scientist

Motivation

USA total and running total of Summer Olympics gold medals since 2004

```
| Year | Medals | Medals_RT |
|------|--------|-----------|
| 2004 | 116    | 116       |
| 2008 | 125    | 241       |
| 2012 | 147    | 388       |
```

Discus throw reigning champion status

```
| Year | Champion | Last_Champion | Reigning_Champion |
|------|----------|---------------|-------------------|
| 1996 | GER      | null          | false             |
| 2000 | LTU      | GER           | false             |
| 2004 | LTU      | LTU           | true              |
| 2008 | EST      | LTU           | false             |
| 2012 | GER      | EST           | false             |
```

Course outline

1. Introduction to window functions
2. Fetching, ranking, and paging
3. Aggregate window functions and frames
4. Beyond window functions

Summer olympics dataset

- Each row represents a medal awarded in the Summer Olympics games

Columns

- `Year` , `City`
- `Sport` , `Discipline` , `Event`
- `Athlete` , `Country` , `Gender`
- `Medal`

Window functions

- Perform an operation across a set of rows that are somehow related to the current row
- Similar to `GROUP BY` aggregate functions, but all rows remain in the output

Uses

- Fetching values from preceding or following rows (e.g. fetching the previous row's value)
    - Determining reigning champion status
    - Calculating growth over time
- Assigning ordinal ranks (1rst, 2nd, *etc.*) to rows based on their values' positions in a sorted list
- Running totals, moving averages

Row numbers

Query

```
SELECT
  Year, Event, Country
FROM Summer_Medals
WHERE
  Medal = 'Gold';
```

Result

```
| Year | Event                       | Country |
|------|-----------------------------|---------|
| 1896 | 100M Freestyle              | HUN     |
| 1896 | 100M Freestyle For Sailors  | GRE     |
| 1896 | 1200M Freestyle             | HUN     |
| ...  | ...                         | ...     |
```

Enter ROW_NUMBER

Query

```
SELECT
  Year, Event, Country,
  ROW_NUMBER() OVER () AS Row_N
FROM Summer_Medals
WHERE
  Medal = 'Gold';
```

Result

```
| Year | Event                        | Country | Row_N |
|------|------------------------------|---------|-------|
| 1896 | 100M Freestyle               | HUN     | 1     |
| 1896 | 100M Freestyle For Sailors   | GRE     | 2     |
| 1896 | 1200M Freestyle              | HUN     | 3     |
| ...  | ...                          | ...     | ...   |
```

Anatomy of a window function

Query

```
SELECT
  Year, Event, Country,
  ROW_NUMBER() OVER () AS Row_N
FROM Summer_Medals
WHERE
  Medal = 'Gold';
```

- `FUNCTION_NAME() OVER (...)`
  - `ORDER BY`
  - `PARTITION BY`
  - `ROWS/RANGE PRECEDING/FOLLOWING/UNBOUNDED`

Let's practice!

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

ORDER BY

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

Michel Semaan
Data Scientist

Row numbers

Query

```
SELECT
  Year, Event, Country,
  ROW_NUMBER() OVER () AS Row_N
FROM Summer_Medals
WHERE
  Medal = 'Gold';
```

Result*

```
| Year | Event                      | Country | Row_N |
|------|----------------------------|---------|-------|
| 1896 | 100M Freestyle             | HUN     | 1     |
| 1896 | 100M Freestyle For Sailors | GRE     | 2     |
| 1896 | 1200M Freestyle            | HUN     | 3     |
| ...  | ...                        | ...     | ...   |
```

Enter ORDER BY

- `ORDER BY` in `OVER` orders the rows related to the current row
  - **Example**: Ordering by year in descending order in `ROW_NUMBER` 's `OVER` clause will assign 1 to the most recent year's rows

Ordering by Year in descending order

Query

```sql
SELECT
  Year, Event, Country,
  ROW_NUMBER() OVER (ORDER BY Year DESC) AS Row_N
FROM Summer_Medals
WHERE
  Medal = 'Gold';
```

Result

```
| Year | Event          | Country | Row_N |
|------|----------------|---------|-------|
| 2012 | Wg 96 KG       | IRI     | 1     |
| 2012 | 4X100M Medley  | USA     | 2     |
| 2012 | Wg 84 KG       | RUS     | 3     |
| ...  | ...            | ...     | ...   |
| 2008 | 50M Freestyle  | BRA     | 637   |
| 2008 | 96 - 120KG     | CUB     | 638   |
| ...  | ...            | ...     | ...   |
```

Ordering by multiple columns

Query

```
SELECT
  Year, Event, Country,
  ROW_NUMBER() OVER
    (ORDER BY Year DESC, Event ASC) AS Row_N
FROM Summer_Medals
WHERE
  Medal = 'Gold';
```

Result

```
| Year | Event    | Country | Row_N |
|------|---------|---------|-------|
| 2012 | + 100KG | FRA     | 1     |
| 2012 | + 67 KG | SRB     | 2     |
| 2012 | + 78KG  | CUB     | 3     |
| ...  | ...     | ...     | ...   |
```

Ordering in- and outside OVER

Query

```
SELECT
  Year, Event, Country,
  ROW_NUMBER() OVER
    (ORDER BY Year DESC, Event ASC) AS Row_N
FROM Summer_Medals
WHERE
  Medal = 'Gold'
ORDER BY Country ASC, Row_N ASC;
```

Result

```
| Year | Event     | Country | Row_N |
|------|-----------|---------|-------|
| 2012 | 1500M     | ALG     | 36    |
| 2000 | 1500M     | ALG     | 1998  |
| 1996 | 1500M     | ALG     | 2662  |
| ...  | ...       | ...     | ...   |
```

- `ORDER BY` inside `OVER` takes effect before `ORDER BY` outside `OVER`

Reigning champion

- A reigning champion is a champion who's won both the previous and current years' competitions
- The previous and current year's champions need to be in the same row (in two different columns)

Enter LAG

- `LAG(column, n) OVER (...)` returns `column` 's value at the row `n` rows before the current row
  - `LAG(column, 1) OVER (...)` returns the previous row's value

Current champions

Query

```
SELECT
  Year, Country AS Champion
FROM Summer_Medals
WHERE
  Year IN (1996, 2000, 2004, 2008, 2012)
  AND Gender = 'Men' AND Medal = 'Gold'
  AND Event = 'Discus Throw';
```

Result

```
| Year | Champion |
|------|----------|
| 1996 | GER      |
| 2000 | LTU      |
| 2004 | LTU      |
| 2008 | EST      |
| 2012 | GER      |
```

Current and last champions

Query

```
WITH Discus_Gold AS (
  SELECT
    Year, Country AS Champion
  FROM Summer_Medals
  WHERE
    Year IN (1996, 2000, 2004, 2008, 2012)
    AND Gender = 'Men' AND Medal = 'Gold'
    AND Event = 'Discus Throw')

SELECT
  Year, Champion,
  LAG(Champion, 1) OVER
    (ORDER BY Year ASC) AS Last_Champion
FROM Discus_Gold
ORDER BY Year ASC;
```

Result

```
| Year | Champion | Last_Champion |
|------|----------|---------------|
| 1996 | GER      | null          |
| 2000 | LTU      | GER           |
| 2004 | LTU      | LTU           |
| 2008 | EST      | LTU           |
| 2012 | GER      | EST           |
```

Let's practice!

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

PARTITION BY

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

Michel Semaan
Data Scientist

Motivation

Query

```
WITH Discus_Gold AS (
  SELECT
    Year, Event, Country AS Champion
  FROM Summer_Medals
  WHERE
    Year IN (2004, 2008, 2012)
    AND Gender = 'Men' AND Medal = 'Gold'
    AND Event IN ('Discus Throw', 'Triple Jump')
    AND Gender = 'Men')

SELECT
  Year, Event, Champion,
  LAG(Champion) OVER
    (ORDER BY Event ASC, Year ASC) AS Last_Champion
FROM Discus_Gold
ORDER BY Event ASC, Year ASC;
```

Result

```
| Year | Event        | Champion | Last_Champion |
|------|--------------|----------|---------------|
| 2004 | Discus Throw | LTU      | null          |
| 2008 | Discus Throw | EST      | LTU           |
| 2012 | Discus Throw | GER      | EST           |
| 2004 | Triple Jump  | SWE      | GER           |
| 2008 | Triple Jump  | POR      | SWE           |
| 2012 | Triple Jump  | USA      | POR           |
```

- When `Event` changes from `Discus Throw` to `Triple Jump` , `LAG` fetched `Discus Throw` 's last champion as opposed to a `null`

Enter PARTITION BY

- `PARTITION BY` splits the table into partitions based on a column's unique values
  - The results aren't rolled into one column
- Operated on separately by the window function
  - `ROW_NUMBER` will reset for each partition
  - `LAG` will only fetch a row's previous value if its previous row is in the same partition

Partitioning by one column

Query

```
WITH Discus_Gold AS (...)

SELECT
  Year, Event, Champion,
  LAG(Champion) OVER
    (PARTITION BY Event
     ORDER BY Event ASC, Year ASC) AS Last_Champion
FROM Discus_Gold
ORDER BY Event ASC, Year ASC;
```

Result

```
| Year | Event        | Champion | Last_Champion |
|------|--------------|----------|---------------|
| 2004 | Discus Throw | LTU      | null          |
| 2008 | Discus Throw | EST      | LTU           |
| 2012 | Discus Throw | GER      | EST           |
| 2004 | Triple Jump  | SWE      | null          |
| 2008 | Triple Jump  | POR      | SWE           |
| 2012 | Triple Jump  | USA      | POR           |
```

More complex partitioning

```
| Year | Country | Event                 | Row_N |
|------|---------|----------------------|-------|
| 2008 | CHN     | + 78KG (Heavyweight) | 1     |
| 2008 | CHN     | - 49 KG              | 2     |
| ...  | ...     | ...                  | ...   |
| 2008 | JPN     | 48 - 55KG            | 27    |
| 2008 | JPN     | 48 - 55KG            | 28    |
| ...  | ...     | ...                  | ...   |
| 2012 | CHN     | +75KG                | 32    |
| 2012 | CHN     | - 49 KG              | 33    |
| ...  | ...     | ...                  | ...   |
| 2012 | JPN     | +75KG                | 51    |
| 2012 | JPN     | - 49 KG              | 52    |
| ...  | ...     | ...                  | ...   |
```

- Row number should reset per `Year` and `Country`

Partitioning by multiple columns

Query

```
WITH Country_Gold AS (
  SELECT
    DISTINCT Year, Country, Event
  FROM Summer_Medals
  WHERE
    Year IN (2008, 2012)
    AND Country IN ('CHN', 'JPN')
    AND Gender = 'Women' AND Medal = 'Gold')

SELECT
  Year, Country, Event,
  ROW_NUMBER() OVER (PARTITION BY Year, Country)
FROM Country_Gold;
```

Result

```
| Year | Country | Event                | Row_N |
|------|---------|----------------------|-------|
| 2008 | CHN     | + 78KG (Heavyweight) | 1     |
| 2008 | CHN     | - 49 KG              | 2     |
| ...  | ...     | ...                  | ...   |
| 2008 | JPN     | 48 - 55KG            | 1     |
| 2008 | JPN     | 48 - 55KG            | 2     |
| ...  | ...     | ...                  | ...   |
| 2012 | CHN     | +75KG                | 1     |
| 2012 | CHN     | - 49 KG              | 2     |
| ...  | ...     | ...                  | ...   |
| 2012 | JPN     | +75KG                | 1     |
| 2012 | JPN     | - 49 KG              | 2     |
| ...  | ...     | ...                  | ...   |
```

Let's practice!

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS