Pivoting

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

Michel Semaan
Data Scientist

Transforming tables

Before

```
| Country | Year | Awards |
|---------|------|--------|
| CHN     | 2008 | 74     |
| CHN     | 2012 | 56     |
| RUS     | 2008 | 43     |
| RUS     | 2012 | 47     |
| USA     | 2008 | 125    |
| USA     | 2012 | 147    |
```

- Gold medals awarded to China, Russia, and the USA

After

```
| Country | 2008 | 2012 |
|---------|------|------|
| CHN     | 74   | 56   |
| RUS     | 43   | 47   |
| USA     | 125  | 147  |
```

- Pivoted by `Year`
- Easier to scan, especially if pivoted by a chronologically ordered column

Enter CROSSTAB

```
CREATE EXTENSION IF NOT EXISTS tablefunc;

SELECT * FROM CROSSTAB($$
  source_sql TEXT
$$) AS ct (column_1 DATA_TYPE_1,
           column_2 DATA_TYPE_2,
           ...,
           column_n DATA_TYPE_N);
```

Queries

Before

```
SELECT
  Country, Year, COUNT(*) AS Awards
FROM Summer_Medals
WHERE
  Country IN ('CHN', 'RUS', 'USA')
  AND Year IN (2008, 2012)
  AND Medal = 'Gold'
GROUP BY Country, Year
ORDER BY Country ASC, Year ASC;
```

After

```
CREATE EXTENSION IF NOT EXISTS tablefunc;

SELECT * FROM CROSSTAB($$
  SELECT
    Country, Year, COUNT(*) :: INTEGER AS Awards
  FROM Summer_Medals
  WHERE
    Country IN ('CHN', 'RUS', 'USA')
    AND Year IN (2008, 2012)
    AND Medal = 'Gold'
  GROUP BY Country, Year
  ORDER BY Country ASC, Year ASC;
$$) AS ct (Country VARCHAR, "2008" INTEGER, "2012" INTEGE

ORDER BY Country ASC;
```

Source query

```
WITH Country_Awards AS (
  SELECT
    Country, Year, COUNT(*) AS Awards
  FROM Summer_Medals
  WHERE
    Country IN ('CHN', 'RUS', 'USA')
    AND Year IN (2004, 2008, 2012)
    AND Medal = 'Gold' AND Sport = 'Gymnastics'
  GROUP BY Country, Year
  ORDER BY Country ASC, Year ASC)

SELECT
  Country, Year,
  RANK() OVER
    (PARTITION BY Year ORDER BY Awards DESC) :: INTEGER
    AS rank
FROM Country_Awards
ORDER BY Country ASC, Year ASC;
```

Source result

Country	Year	Rank
CHN	2004	3
CHN	2008	1
CHN	2012	1
RUS	2004	1
RUS	2008	2
RUS	2012	2
USA	2004	2
USA	2008	3
USA	2012	3

Pivot query

```
CREATE EXTENSION IF NOT EXISTS tablefunc;

SELECT * FROM CROSSTAB($$
  ...
$$) AS ct (Country VARCHAR,
           "2004" INTEGER,
           "2008" INTEGER,
           "2012" INTEGER)

ORDER BY Country ASC;
```

Pivot result

Country	2004	2008	2012
CHN	3	1	1
RUS	1	2	2
USA	2	3	3

Let's practice!

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

ROLLUP and CUBE

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

Michel Semaan
Data Scientist

Group-level totals

Chinese and Russian medals in the 2008 Summer Olympics per medal class

Country	Medal	Awards
CHN	Bronze	57
CHN	Gold	74
CHN	Silver	53
CHN	Total	184
RUS	Bronze	56
RUS	Gold	43
RUS	Silver	44
RUS	Total	143

The old way

```
SELECT
  Country, Medal, COUNT(*) AS Awards
FROM Summer_Medals
WHERE
  Year = 2008 AND Country IN ('CHN', 'RUS')
GROUP BY Country, Medal
ORDER BY Country ASC, Medal ASC
UNION ALL

SELECT
  Country, 'Total', COUNT(*) AS Awards
FROM Summer_Medals
WHERE
  Year = 2008 AND Country IN ('CHN', 'RUS')
GROUP BY Country, 2
ORDER BY Country ASC;
```

Enter ROLLUP

```
SELECT
  Country, Medal, COUNT(*) AS Awards
FROM Summer_Medals
WHERE
  Year = 2008 AND Country IN ('CHN', 'RUS')
GROUP BY Country, ROLLUP(Medal)
ORDER BY Country ASC, Medal ASC;
```

- `ROLLUP` is a `GROUP BY` subclause that includes extra rows for group-level aggregations
- `GROUP BY Country, ROLLUP(Medal)` will count all `Country` - and `Medal` -level totals, then count only `Country` -level totals and fill in `Medal` with `null` s for these rows

ROLLUP - Query

```
SELECT
  Country, Medal, COUNT(*) AS Awards
FROM summer_medals
WHERE
  Year = 2008 AND Country IN ('CHN', 'RUS')
GROUP BY ROLLUP(Country, Medal)
ORDER BY Country ASC, Medal ASC;
```

- `ROLLUP` is hierarchical, de-aggregating from the leftmost provided column to the right-most
  - `ROLLUP(Country, Medal)` includes `Country` -level totals
  - `ROLLUP(Medal, Country)` includes `Medal` -level totals
  - Both include grand totals

ROLLUP - Result

```
| Country | Medal  | Awards |
|---------|--------|--------|
| CHN     | Bronze | 57     |
| CHN     | Gold   | 74     |
| CHN     | Silver | 53     |
| CHN     | null   | 184    |
| RUS     | Bronze | 56     |
| RUS     | Gold   | 43     |
| RUS     | Silver | 44     |
| RUS     | null   | 143    |
| null    | null   | 327    |
```

- Group-level totals contain `nulls`; the row with all `null`s is the grand total
- Notice that it didn't include `Medal`-level totals, since it's `ROLLUP(Country, Medal)` and not `ROLLUP(Medal, Country)`

Enter CUBE

```
SELECT
  Country, Medal, COUNT(*) AS Awards
FROM summer_medals
WHERE
  Year = 2008 AND Country IN ('CHN', 'RUS')
GROUP BY CUBE(Country, Medal)
ORDER BY Country ASC, Medal ASC;
```

- `CUBE` is a non-hierarchical `ROLLUP`
- It generates all possible group-level aggregations
  - `CUBE(Country, Medal)` counts `Country` -level, `Medal` -level, and grand totals

CUBE - Result

```
| Country | Medal  | Awards |
|---------|--------|--------|
| CHN     | Bronze | 57     |
| CHN     | Gold   | 74     |
| CHN     | Silver | 53     |
| CHN     | null   | 184    |
| RUS     | Bronze | 56     |
| RUS     | Gold   | 43     |
| RUS     | Silver | 44     |
| RUS     | null   | 143    |
| null    | Bronze | 113    |
| null    | Gold   | 117    |
| null    | Silver | 97     |
| null    | null   | 327    |
```

- Notice that `Medal` -level totals are included

ROLLUP vs CUBE

Source

```
| Year | Quarter | Sales |
|------|---------|-------|
| 2008 | Q1      | 12    |
| 2008 | Q2      | 15    |
| 2009 | Q1      | 21    |
| 2009 | Q2      | 27    |
```

- Use `ROLLUP` when you have hierarchical data (e.g., date parts) and don't want all possible group-level aggregations
- Use `CUBE` when you want all possible group-level aggregations

`ROLLUP(Year, Quarter)`

```
| Year | Quarter | Sales |
|------|---------|-------|
| 2008 | null    | 27    |
| 2009 | null    | 48    |
| null | null    | 75    |
```

`CUBE(Year, Quarter)`

Above rows + the following

```
| Year | Quarter | Sales |
|------|---------|-------|
| null | Q1      | 33    |
| null | Q2      | 42    |
```

Let's practice!

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

A survey of useful functions

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS

Michel Semaan
Data Scientist

Nulls ahoy

Query

```
SELECT
  Country, Medal, COUNT(*) AS Awards
FROM summer_medals
WHERE
  Year = 2008 AND Country IN ('CHN', 'RUS')
GROUP BY ROLLUP(Country, Medal)
ORDER BY Country ASC, Medal ASC;
```

- `null` s signify group totals

Result

```
| Country | Medal  | Awards |
|---------|--------|--------|
| CHN     | Bronze | 57     |
| CHN     | Gold   | 74     |
| CHN     | Silver | 53     |
| CHN     | null   | 184    |
| RUS     | Bronze | 56     |
| RUS     | Gold   | 43     |
| RUS     | Silver | 44     |
| RUS     | null   | 143    |
| null    | null   | 327    |
```

Enter COALESCE

- `COALESCE()` takes a list of values and returns the first non-`null` value, going from left to right
- `COALESCE(null, null, 1, null, 2) ? 1`
- Useful when using SQL operations that return `null` s
  - `ROLLUP` and `CUBE`
  - Pivoting
  - `LAG` and `LEAD`

Annihilating nulls

Query

```
SELECT
  COALESCE(Country, 'Both countries') AS Country,
  COALESCE(Medal, 'All medals') AS Medal,
  COUNT(*) AS Awards
FROM summer_medals
WHERE
  Year = 2008 AND Country IN ('CHN', 'RUS')
GROUP BY ROLLUP(Country, Medal)
ORDER BY Country ASC, Medal ASC;
```

Result

```
| Country         | Medal      | Awards |
|-----------------|------------|--------|
| Both countries | All medals | 327    |
| CHN             | All medals | 184    |
| CHN             | Bronze     | 57     |
| CHN             | Gold       | 74     |
| CHN             | Silver     | 53     |
| RUS             | All medals | 143    |
| RUS             | Bronze     | 56     |
| RUS             | Gold       | 43     |
| RUS             | Silver     | 44     |
```

Compressing data

Before

```
| Country | Rank |
|---------|------|
| CHN     | 1    |
| RUS     | 2    |
| USA     | 3    |
```

- `Rank` is redundant because the ranking is implied

After

```
CHN, RUS, USA
```

- Succinct and provides all information needed because the ranking is implied

Enter STRING_AGG

- `STRING_AGG(column, separator)` takes all the values of a column and concatenates them, with `separator` in between each value

`STRING_AGG(Letter, ', ')` transforms this...

```
| Letter |
|--------|
| A      |
| B      |
| C      |
```

...into this

```
A, B, C
```

Query and result

Before

```
WITH Country_Medals AS (
  SELECT
    Country, COUNT(*) AS Medals
  FROM Summer_Medals
  WHERE Year = 2012
    AND Country IN ('CHN', 'RUS', 'USA')
    AND Medal = 'Gold'
    AND Sport = 'Gymnastics'
  GROUP BY Country),

  SELECT
    Country,
    RANK() OVER (ORDER BY Medals DESC) AS Rank
  FROM Country_Medals
  ORDER BY Rank ASC;
```

After

```
WITH Country_Medals AS (...),

  Country_Ranks AS (...)

  SELECT STRING_AGG(Country, ', ')
  FROM Country_Medals;
```

Result

```
CHN, RUS, USA
```

Let's practice!

POSTGRESQL SUMMARY STATS AND WINDOW FUNCTIONS