Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generatedNumber each row in the dataset.

SELECT

  \*,

  -- Assign numbers to each row

  ROW\_NUMBER() OVER() AS Row\_N

FROM Summer\_Medals

ORDER BY Row\_N ASC;

Assign a number to each year in which Summer Olympic games were held.

SELECT

  Year,

  -- Assign numbers to each year

  ROW\_NUMBER() OVER() AS Row\_N

FROM (

  SELECT DISTINCT YEAR

  FROM Summer\_Medals

  ORDER BY Year ASC

) AS Years

ORDER BY Year ASC;

A picture containing graphical user interface

Description automatically generated

Table

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text, table

Description automatically generated

Assign a number to each year in which Summer Olympic games were held so that rows with the most recent years have lower row numbers.

SELECT

  Year,

  -- Assign the lowest numbers to the most recent years

  ROW\_NUMBER() OVER (ORDER BY YEAR DESC) AS Row\_N

FROM (

  SELECT DISTINCT Year

  FROM Summer\_Medals

) AS Years

ORDER BY Year;

For each athlete, count the number of medals he or she has earned. Having wrapped the previous query in the Athlete\_Medals CTE, rank each athlete by the number of medals they've earned.

WITH Athlete\_Medals AS (

  SELECT

    -- Count the number of medals each athlete has earned

    Athlete,

    COUNT(\*) AS Medals

  FROM Summer\_Medals

  GROUP BY Athlete)

SELECT

  -- Number each athlete by how many medals they've earned

  Athlete,

  ROW\_NUMBER() OVER (ORDER BY Medals DESC) AS Row\_N

FROM Athlete\_Medals

ORDER BY Medals DESC;

**Reigning weightlifting champions**

A reigning champion is a champion who's won both the previous and current years' competitions. To determine if a champion is reigning, the previous and current years' results need to be in the same row, in two different columns.

Return each year's gold medalists in the Men's 69KG weightlifting competition. Having wrapped the previous query in the Weightlifting\_Gold CTE, get the previous year's champion for each year.

WITH Weightlifting\_Gold AS (

  SELECT

    -- Return each year's champions' countries

    Year,

    Country AS champion

  FROM Summer\_Medals

  WHERE

    Discipline = 'Weightlifting' AND

    Event = '69KG' AND

    Gender = 'Men' AND

    Medal = 'Gold')

SELECT

  Year, Champion,

  -- Fetch the previous year's champion

  LAG(Champion) OVER

    (ORDER BY year ASC) AS Last\_Champion

FROM Weightlifting\_Gold

ORDER BY Year ASC;

Graphical user interface, text, application

Description automatically generatedGraphical user interface

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with medium confidence

# Reigning champions by gender

You've already fetched the previous year's champion for one event. However, if you have multiple events, genders, or other metrics as columns, you'll need to split your table into partitions to avoid having a champion from one event or gender appear as the previous champion of another event or gender. Return the previous champions of each year's event by gender.

WITH Tennis\_Gold AS (

  SELECT DISTINCT

    Gender, Year, Country

  FROM Summer\_Medals

  WHERE

    Year >= 2000 AND

    Event = 'Javelin Throw' AND

    Medal = 'Gold')

SELECT

  Gender, Year,

  Country AS Champion,

  -- Fetch the previous year's champion by gender

  Lag(Country) OVER (partition by gender ORDER BY year ASC) AS Last\_Champion

FROM Tennis\_Gold

ORDER BY Gender ASC, Year ASC;

Return the previous champions of each year's events by gender and event.

WITH Athletics\_Gold AS (

  SELECT DISTINCT

    Gender, Year, Event, Country

  FROM Summer\_Medals

  WHERE

    Year >= 2000 AND

    Discipline = 'Athletics' AND

    Event IN ('100M', '10000M') AND

    Medal = 'Gold')

SELECT

  Gender, Year, Event,

  Country AS Champion,

  -- Fetch the previous year's champion by gender and event

  Lag(Country) OVER (Partition by gender, event

            ORDER BY Year ASC) AS Last\_Champion

FROM Athletics\_Gold

ORDER BY Event ASC, Gender ASC, Year ASC;

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

# Future gold medalists

Fetching functions allow you to get values from different parts of the table into one row. If you have time-ordered data, you can "peek into the future" with the LEAD fetching function. This is especially useful if you want to compare a current value to a future value.

For each year, fetch the current gold medalist and the gold medalist 3 competitions ahead of the current row.

SELECT

  -- For each year, fetch the current and future medalists

  year,

  Athlete,

  LEAD(Athlete, 3) OVER (ORDER BY year ASC) AS Future\_Champion

FROM Discus\_Medalists

ORDER BY Year ASC;

# First athlete by name

It's often useful to get the first or last value in a dataset to compare all other values to it. With absolute fetching functions like FIRST\_VALUE, you can fetch a value at an absolute position in the table, like its beginning or end. Return all athletes and the first athlete ordered by alphabetical order.

SELECT

  -- Fetch all athletes and the first athlete alphabetically

  athlete,

  FIRST\_VALUE(athlete) OVER (

    ORDER BY athlete ASC

  ) AS First\_Athlete

FROM All\_Male\_Medalists;

# Last country by name

Just like you can get the first row's value in a dataset, you can get the last row's value. This is often useful when you want to compare the most recent value to previous values. Return the year and the city in which each Olympic games were held. Fetch the last city in which the Olympic games were held.

SELECT

  Year,

  City,

  -- Get the last city in which the Olympic games were held

  last\_value(city) OVER (

   ORDER BY year ASC

   RANGE BETWEEN

     UNBOUNDED PRECEDING AND

     UNBOUNDED FOLLOWING

  ) AS Last\_City

FROM Hosts

ORDER BY Year ASC;

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

Table

Description automatically generatedTable

Description automatically generated with medium confidenceTable

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generated

# Ranking athletes by medals earned

In chapter 1, you used ROW\_NUMBER to rank athletes by awarded medals. However, ROW\_NUMBER assigns different numbers to athletes with the same count of awarded medals, so it's not a useful ranking function; if two athletes earned the same number of medals, they should have the same rank.

* Rank each athlete by the number of medals they've earned -- the higher the count, the higher the rank -- with identical numbers in case of identical values.

SELECT

  Athlete,

  Medals,

  -- Rank athletes by the medals they've won

  rank() OVER (ORDER BY medals DESC) AS Rank\_N

FROM Athlete\_Medals

ORDER BY Medals DESC;

# Ranking athletes from multiple countries

In the previous exercise, you used RANK to assign rankings to one group of athletes. In real-world data, however, you'll often find numerous groups within your data. Without partitioning your data, one group's values will influence the rankings of the others.

Also, while RANK skips numbers in case of identical values, the most natural way to assign rankings is not to skip numbers. If two countries are tied for second place, the country after them is considered to be third by most people.

* Rank each country's athletes by the count of medals they've earned -- the higher the count, the higher the rank -- without skipping numbers in case of identical values.

SELECT

  Country,

  -- Rank athletes in each country by the medals they've won

  Athlete,

  dense\_rank() OVER (PARTITION BY country

                ORDER BY Medals DESC) AS Rank\_N

FROM Athlete\_Medals

ORDER BY Country ASC, RANK\_N ASC;

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

# Paging events

There are exactly 666 unique events in the Summer Medals Olympics dataset. If you want to chunk them up to analyze them piece by piece, you'll need to split the events into groups of approximately equal size.

* Split the distinct events into exactly 111 groups, ordered by event in alphabetical order.

WITH Events AS (

  SELECT DISTINCT Event

  FROM Summer\_Medals)

SELECT

  --- Split up the distinct events into 111 unique groups

  event,

  NTILE(111) OVER (ORDER BY event ASC) AS Page

FROM Events

ORDER BY Event ASC;

# Top, middle, and bottom thirds

Splitting your data into thirds or quartiles is often useful to understand how the values in your dataset are spread. Getting summary statistics (averages, sums, standard deviations, etc.) of the top, middle, and bottom thirds can help you determine what distribution your values follow.

* Split the athletes into top, middle, and bottom thirds based on their count of medals.

SELECT

  Athlete,

  Medals,

  -- Split athletes into thirds by their earned medals

  NTILE(3) OVER (ORDER BY Medals DESC) AS Third

FROM Athlete\_Medals

ORDER BY Medals DESC, Athlete ASC;

* Return the average of each third.

WITH Athlete\_Medals AS (

  SELECT Athlete, COUNT(\*) AS Medals

  FROM Summer\_Medals

  GROUP BY Athlete

  HAVING COUNT(\*) > 1),

  Thirds AS (

  SELECT

    Athlete,

    Medals,

    NTILE(3) OVER (ORDER BY Medals DESC) AS Third

  FROM Athlete\_Medals)

SELECT

  -- Get the average medals earned in each third

  THIRD,

  AVG(Medals) AS Avg\_Medals

FROM Thirds

GROUP BY Third

ORDER BY Third ASC;

Graphical user interface

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with medium confidence Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

# Running totals of athlete medals

The running total (or cumulative sum) of a column helps you determine what each row's contribution is to the total sum.

* Return the athletes, the number of medals they earned, and the medals running total, ordered by the athletes' names in alphabetical order.

WITH Athlete\_Medals AS (

  SELECT

    Athlete, COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE

    Country = 'USA' AND Medal = 'Gold'

    AND Year >= 2000

  GROUP BY Athlete)

SELECT

  -- Calculate the running total of athlete medals

  Athlete,

  Medals,

  SUM(Medals) OVER (ORDER BY Athlete ASC) AS Max\_Medals

FROM Athlete\_Medals

ORDER BY Athlete ASC;

# Maximum country medals by year

Getting the maximum of a country's earned medals so far helps you determine whether a country has broken its medals record by comparing the current year's earned medals and the maximum so far.

* Return the year, country, medals, and the maximum medals earned so far for each country, ordered by year in ascending order.

WITH Country\_Medals AS (

  SELECT

    Year, Country, COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE

    Country IN ('CHN', 'KOR', 'JPN')

    AND Medal = 'Gold' AND Year >= 2000

  GROUP BY Year, Country)

SELECT

  -- Return the max medals earned so far per country

  YEAR,

  COUNTRY,

  MEDALS,

  MAX(MEDALS) OVER (PARTITION BY COUNTRY

                ORDER BY YEAR ASC) AS Max\_Medals

FROM Country\_Medals

ORDER BY Country ASC, Year ASC;

# Minimum country medals by year

So far, you've seen MAX and SUM, aggregate functions normally used with GROUP BY, being used as window functions. You can also use the other aggregate functions, like MIN, as window functions.

* Return the year, medals earned, and minimum medals earned so far.

WITH France\_Medals AS (

  SELECT

    Year, COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE

    Country = 'FRA'

    AND Medal = 'Gold' AND Year >= 2000

  GROUP BY Year)

SELECT

  YEAR,

  Medals,

  MIN(Medals) OVER (ORDER BY YEAR ASC) AS Min\_Medals

FROM France\_Medals

ORDER BY Year ASC;

Graphical user interface, text, application

Description automatically generated with medium confidence

Table

Description automatically generated

A picture containing table

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedA picture containing graphical user interface

Description automatically generatedTable

Description automatically generated

Table

Description automatically generated with medium confidence

Table

Description automatically generated with low confidence

# Moving maximum of Scandinavian athletes' medals

Frames allow you to restrict the rows passed as input to your window function to a sliding window for you to define the start and finish.

Adding a frame to your window function allows you to calculate "moving" metrics, inputs of which slide from row to row.

* Return the year, medals earned, and the maximum medals earned, comparing only the current year and the next year.

WITH Scandinavian\_Medals AS (

  SELECT

    Year, COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE

    Country IN ('DEN', 'NOR', 'FIN', 'SWE', 'ISL')

    AND Medal = 'Gold'

  GROUP BY Year)

SELECT

  -- Select each year's medals

  year,

  medals,

  -- Get the max of the current and next years'  medals

  max(medals) OVER (ORDER BY year ASC

             ROWS BETWEEN current row

             AND 1 following) AS Max\_Medals

FROM Scandinavian\_Medals

ORDER BY Year ASC;

# Moving maximum of Chinese athletes' medals

Frames allow you to "peek" forwards or backward without first using the relative fetching functions, LAG and LEAD, to fetch previous rows' values into the current row.

* Return the athletes, medals earned, and the maximum medals earned, comparing only the last two and current athletes, ordering by athletes' names in alphabetical order.

WITH Chinese\_Medals AS (

  SELECT

    Athlete, COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE

    Country = 'CHN' AND Medal = 'Gold'

    AND Year >= 2000

  GROUP BY Athlete)

SELECT

  -- Select the athletes and the medals they've earned

  athlete,

  medals,

  -- Get the max of the last two and current rows' medals

  max(medals) OVER (ORDER BY athlete ASC

            ROWS BETWEEN 2 preceding

            AND current row) AS Max\_Medals

FROM Chinese\_Medals

ORDER BY Athlete ASC;

Graphical user interface, text, application, email

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, text

Description automatically generated

# Moving average of Russian medals

Using frames with aggregate window functions allow you to calculate many common metrics, including moving averages and totals. These metrics track the change in performance over time.

* Calculate the 3-year moving average of medals earned.

WITH Russian\_Medals AS (

  SELECT

    Year, COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE

    Country = 'RUS'

    AND Medal = 'Gold'

    AND Year >= 1980

  GROUP BY Year)

SELECT

  Year, Medals,

  --- Calculate the 3-year moving average of medals earned

  avg(medals) OVER

    (ORDER BY Year ASC

     ROWS BETWEEN

     2 preceding AND current row) AS Medals\_MA

FROM Russian\_Medals

ORDER BY Year ASC;

# Moving total of countries' medals

What if your data is split into multiple groups spread over one or more columns in the table? Even with a defined frame, if you can't somehow separate the groups' data, one group's values will affect the average of another group's values.

* Calculate the 3-year moving sum of medals earned per country.

WITH Country\_Medals AS (

  SELECT

    Year, Country, COUNT(\*) AS Medals

  FROM Summer\_Medals

  GROUP BY Year, Country)

SELECT

  Year, Country, Medals,

  -- Calculate each country's 3-game moving total

  sum(Medals) OVER

    (PARTITION BY country

     ORDER BY Year ASC

     ROWS BETWEEN

     2 preceding AND current row) AS Medals\_MA

FROM Country\_Medals

ORDER BY Country ASC, Year ASC;

Graphical user interface

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedTable

Description automatically generated

Table

Description automatically generatedText

Description automatically generated

# A basic pivot

You have the following table of Pole Vault gold medalist countries by gender in 2008 and 2012.

| Gender | Year | Country |

|--------|------|---------|

| Men | 2008 | AUS |

| Men | 2012 | FRA |

| Women | 2008 | RUS |

| Women | 2012 | USA |

Pivot it by Year to get the following reshaped, cleaner table.

| Gender | 2008 | 2012 |

|--------|------|------|

| Men | AUS | FRA |

| Women | RUS | USA |

* Create the correct extension.
* Fill in the column names of the pivoted table.

-- Create the correct extention to enable CROSSTAB

CREATE EXTENSION IF NOT EXISTS tablefunc;

SELECT \* FROM CROSSTAB($$

  SELECT

    Gender, Year, Country

  FROM Summer\_Medals

  WHERE

    Year IN (2008, 2012)

    AND Medal = 'Gold'

    AND Event = 'Pole Vault'

  ORDER By Gender ASC, Year ASC;

-- Fill in the correct column names for the pivoted table

$$) AS ct (Gender VARCHAR,

           year VARCHAR,

           country VARCHAR)

ORDER BY Gender ASC;

# Pivoting with ranking

You want to produce an easy scannable table of the rankings of the three most populous EU countries by how many gold medals they've earned in the 2004 through 2012 Olympic games. The table needs to be in this format:

| Country | 2004 | 2008 | 2012 |

|---------|------|------|------|

| FRA | ... | ... | ... |

| GBR | ... | ... | ... |

| GER | ... | ... | ... |

You'll need to count the gold medals each country has earned, produce the ranks of each country by medals earned, then pivot the table to this shape.

* Count the gold medals that France (FRA), the UK (GBR), and Germany (GER) have earned per country and year.

-- Count the gold medals per country and year

SELECT

  country,

  year,

  count(medal) AS Awards

FROM Summer\_Medals

WHERE

  Country IN ('FRA', 'GBR', 'GER')

  AND Year IN (2004, 2008, 2012)

  AND Medal = 'Gold'

GROUP BY country, year

ORDER BY Country ASC, Year ASC

Select the country and year columns, then rank the three countries by how many gold medals they earned per year.

SELECT

  -- Select Country and Year

  Country,

  year,

  -- Rank by gold medals earned per year

  rank() over(partition by country order by awards asc) :: INTEGER AS rank

FROM Country\_Awards

ORDER BY Country ASC, Year ASC;

Pivot the query's results by Year by filling in the new table's correct column names.

CREATE EXTENSION IF NOT EXISTS tablefunc;

SELECT \* FROM CROSSTAB($$

  WITH Country\_Awards AS (

    SELECT

      Country,

      Year,

      COUNT(\*) AS Awards

    FROM Summer\_Medals

    WHERE

      Country IN ('FRA', 'GBR', 'GER')

      AND Year IN (2004, 2008, 2012)

      AND Medal = 'Gold'

    GROUP BY Country, Year)

  SELECT

    Country,

    Year,

    RANK() OVER

      (PARTITION BY Year

       ORDER BY Awards DESC) :: INTEGER AS rank

  FROM Country\_Awards

  ORDER BY Country ASC, Year ASC;

-- Fill in the correct column names for the pivoted table

$$) AS ct (country VARCHAR,

           "2004" INTEGER,

           "2008" INTEGER,

           "2012" INTEGER)

Order by Country ASC;

A picture containing logo

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated Graphical user interface, text, application, email

Description automatically generated

Shape

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

Graphical user interface

Description automatically generated

# Country-level subtotals

You want to look at three Scandinavian countries' earned gold medals per country and gender in the year 2000. You're also interested in Country-level subtotals to get the total medals earned for each country, but Gender-level subtotals don't make much sense in this case, so disregard them.

* Count the gold medals awarded per country and gender.
* Generate Country-level gold award counts.

-- Count the gold medals per country and gender

SELECT

  country,

  gender,

  COUNT(\*) AS Gold\_Awards

FROM Summer\_Medals

WHERE

  Year = 2004

  AND Medal = 'Gold'

  AND Country IN ('DEN', 'NOR', 'SWE')

-- Generate Country-level subtotals

GROUP BY country, rollup(gender)

ORDER BY Country ASC, Gender ASC;

# All group-level subtotals

You want to break down all medals awarded to Russia in the 2012 Olympic games per gender and medal type. Since the medals all belong to one country, Russia, it makes sense to generate all possible subtotals (Gender- and Medal-level subtotals), as well as a grand total.

Generate a breakdown of the medals awarded to Russia per country and medal type, including all group-level subtotals and a grand total.

* Count the medals awarded per gender and medal type.
* Generate all possible group-level counts (per gender and medal type subtotals and the grand total).

-- Count the medals per country and medal type

SELECT

  Gender,

  medal,

  count(\*) AS Awards

FROM Summer\_Medals

WHERE

  Year = 2012

  AND Country = 'RUS'

-- Get all possible group-level subtotals

GROUP BY cube (Gender, medal)

ORDER BY Gender ASC, Medal ASC;

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

# Cleaning up results

Returning to the breakdown of Scandinavian awards you previously made, you want to clean up the results by replacing the nulls with meaningful text.

* Turn the nulls in the Country column to All countries, and the nulls in the Gender column to All genders.

SELECT

  -- Replace the nulls in the columns with meaningful text

  COALESCE(Country, 'All countries') AS Country,

  COALESCE(Gender, 'All genders') AS Gender,

  COUNT(\*) AS Awards

FROM Summer\_Medals

WHERE

  Year = 2004

  AND Medal = 'Gold'

  AND Country IN ('DEN', 'NOR', 'SWE')

GROUP BY ROLLUP(Country, Gender)

ORDER BY Country ASC, Gender ASC;

# Summarizing results

After ranking each country in the 2000 Olympics by gold medals awarded, you want to return the top 3 countries in one row, as a comma-separated string. In other words, turn this:

| Country | Rank |

|---------|------|

| USA | 1 |

| RUS | 2 |

| AUS | 3 |

| ... | ... |

into this:

USA, RUS, AUS

* Rank countries by the medals they've been awarded.

WITH Country\_Medals AS (

  SELECT

    Country,

    COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE Year = 2000

    AND Medal = 'Gold'

  GROUP BY Country)

  SELECT

    Country,

    -- Rank countries by the medals awarded

    RANK() OVER(Order by Medals Desc) AS Rank

  FROM Country\_Medals

  ORDER BY Rank ASC;

* Return the top 3 countries by medals awarded as one comma-separated string.

WITH Country\_Medals AS (

  SELECT

    Country,

    COUNT(\*) AS Medals

  FROM Summer\_Medals

  WHERE Year = 2000

    AND Medal = 'Gold'

  GROUP BY Country),

  Country\_Ranks AS (

  SELECT

    Country,

    RANK() OVER (ORDER BY Medals DESC) AS Rank

  FROM Country\_Medals

  ORDER BY Rank ASC)

-- Compress the countries column

SELECT STRING\_AGG(Country,', ')

FROM Country\_Ranks

-- Select only the top three ranks

WHERE rank < 4;

**Subqueries inside WHERE and SELECT Clauses**