SQL . EXAMPLES . 1 . SELECT

Step 1

Open your **browser** (Chrome, Firefox, Edge, Safari)
Open **GBQ Public Datasets**
Select **bigquery-public-data** project
Select **america_health_rankings** dataset and **ahr** table

Step 2

Find out all the **state_name** values for **edition = 2021** and **order** them from Z to A

```
SELECT DISTINCT(state_name)
FROM `bigquery-public-data.america_health_rankings.ahr`
WHERE edition = 2021
ORDER BY state_name DESC
```

Step 3

Find out the **measure_name values** that start with **A**. They must appear only once in the list.

```
SELECT DISTINCT(measure_name)
FROM `bigquery-public-data.america_health_rankings.ahr`
WHERE measure_name LIKE "A%"
```

Step 4

Find out the **maximum value** for **measure** = **Excessive Drinking**

```
SELECT MAX(value)
FROM `bigquery-public-data.america_health_rankings.ahr`
WHERE measure_name = "Excessive Drinking"
```

Step 5

Find out the **average value** for each **state**. Columns must be named **STATE** and **AVG VALUE** and data must be sorted starting from the lowest **AVG VALUE**

```
SELECT state_name as STATE, AVG(value) as AVG_VALUE
FROM `bigquery-public-data.america_health_rankings.ahr`
GROUP BY state_name
ORDER BY AVG_VALUE ASC
```

SQL . EXAMPLES . 2 . SUBSELECT

Step 1

Open your **browser** (Chrome, Firefox, Edge, Safari)
Open **GBQ Public Datasets**
Select **bigquery-public-data** project
Select **census_bureau_international** dataset and select **birth_death_growth_rates** table

Step 2

Find out all the **years** when **more than 100 countries but less than 200** were analysed

```
SELECT COUNT(DISTINCT(country_name)) as countries, year
FROM
`bigquery-public-data.census_bureau_international.birth_death_growth_rates`
GROUP BY year
HAVING countries BETWEEN 100 AND 200
```

Step 3

Find out, for **every year in ascending order**, the **ranking of all countries** sorted by the **highest net migration**

```
SELECT year, country_name, net_migration,
  RANK() OVER (PARTITION BY year ORDER BY net_migration DESC) AS rank
FROM
`bigquery-public-data.census_bureau_international.birth_death_growth_rates`
ORDER BY year ASC
```

Step 4

Use **Step 3 query** to obtain the **top country** for **every year** in **growth rate** using a **SUBSELECT**

```
SELECT * FROM
  (SELECT year, country_name, net_migration,
    RANK() OVER (PARTITION BY year ORDER BY net_migration DESC) AS rank
  FROM
`bigquery-public-data.census_bureau_international.birth_death_growth_rates`)
WHERE rank = 1
ORDER BY year ASC
```

Step 5

Rearrange your **Step 4 query** using **WITH**

```
WITH ranking as
  (SELECT year, country_name, net_migration,
    RANK() OVER (PARTITION BY year ORDER BY net_migration DESC) AS rank
  FROM
`bigquery-public-data.census_bureau_international.birth_death_growth_rates`)
SELECT * FROM ranking WHERE rank = 1 ORDER BY year ASC
```

SQL . EXAMPLES . 3 . TWO TABLES

Step 1

Open your **browser** (Chrome, Firefox, Edge, Safari)
Open **GBQ Public Datasets**
Select **bigquery-public-data** project
Select **census_utility** dataset and check **flips_codes_all** and **flips_codes_states** tables

Step 2

Find out all the **area_names** (in **flips_codes_all**) for each **state_name** (in **flips_codes_states**)
Tip: use the field **state_fips_code** to combine both tables

```
SELECT s.state_name, a.area_name
FROM
  `bigquery-public-data.census_utility.fips_codes_all` a,
  `bigquery-public-data.census_utility.fips_codes_states` s
WHERE a.state_fips_code = s.state_fips_code
ORDER BY state_name ASC, area_name ASC
```

SQL . EXAMPLES . 4 . JOINS

Step 1

Open your **browser** (Chrome, Firefox, Edge, Safari)
Open **GBQ Public Datasets**
Select **bigquery-public-data** project
Click **Compose New Query** link

Step 2

Play with the following **query** to understand joins

```
with left_table as (select * from unnest([
  struct('a' as strings, 1 as numbers),
  struct('b' as strings, 2 as numbers),
  struct('c' as strings, 3 as numbers)
])),
right_table as (select * from unnest([
  struct('a' as strings, 4 as numbers),
  struct('b' as strings, 5 as numbers),
  struct('d' as strings, 6 as numbers)
]))

select
  *
from
  left_table
--cross left right inner full outer
join
  right_table
on left_table.strings = right_table.strings
```

SQL . EXAMPLES . 5 . JOIN

Step 1

Open your **browser** (Chrome, Firefox, Edge, Safari)
Open **GBQ Public Datasets**
Select **bigquery-public-data** project
Select **austin_bikeshare** dataset and check **bikeshare_stations** and **bikeshare_trips** tables

Step 2

Having in mind that the fields **start_station_id** and **end_station_id** in **bikeshare_trips** correspond to field **station_id** in **bikeshare_stations** table, find out how many **bikeshare_trips** started in a **bikeshare_station** in **council_district 9** and ended in a **bikeshare_station** in **council_district 8**.

```
SELECT COUNT(*)
FROM
  `bigquery-public-data.austin_bikeshare.bikeshare_trips` t
JOIN `bigquery-public-data.austin_bikeshare.bikeshare_stations` s_s
  ON t.start_station_id = s_s.station_id
JOIN `bigquery-public-data.austin_bikeshare.bikeshare_stations` s_e
  ON t.end_station_id = CAST(s_e.station_id AS STRING)
WHERE
  s_s.council_district = 9
  AND s_e.council_district = 8
  AND t.bike_type = 'electric'
```

SQL . EXAMPLES . 6 . JOIN

Step 1

Open your **browser** (Chrome, Firefox, Edge, Safari)
Open **GBQ Public Datasets**
Select **bigquery-public-data** project
Select **london_bicycles** dataset and check **cycle_stations** and **cycle_hire** tables

Step 2

Having in mind that the fields **start_station_id** and **end_station_id** in **cycle_hire** correspond to field **id** in **cycle_stations** table, find out all the **cycle_hire** records where the **distance** between the origin's and end's latitudes is **greater than 0.2**.
Show the **names** of both stations, their **latitudes**, the **distance** between them and the **duration** of the trip.

```
WITH routes AS
  (SELECT
    s1.name as origin, s1.longitude as origin_lat,
    s2.name as destination, s2.longitude as destination_lat,
    case
      when (s1.longitude > s2.longitude) then (s1.longitude - s2.longitude)
      else (s2.longitude - s1.longitude)
    end as distance,
--  abs(s1.longitude - s2.longitude) as distance,
    h.duration
  FROM
    `bigquery-public-data.london_bicycles.cycle_stations` s1,
    `bigquery-public-data.london_bicycles.cycle_stations` s2,
    `bigquery-public-data.london_bicycles.cycle_hire` h
  WHERE h.start_station_id = s1.id
  AND h.end_station_id = s2.id)

SELECT *
FROM routes
WHERE distance > 0.2
ORDER BY distance ASC
```