COMMON TABLE EXPRESSION

```
Ann Mariya Francis
Exercise 3.9
5 Feb 2025
1.1)
WITH top_countries AS (
  SELECT D.country
  FROM customer A
  JOIN address B ON A.address_id = B.address_id
  JOIN city C ON B.city_id = C.city_id
  JOIN country D ON C.country_id = D.country_id
  GROUP BY D.country
  ORDER BY COUNT(A.customer_id) DESC
  LIMIT 10
),
top_cities AS (
  SELECT D.country, C.city
  FROM customer A
  JOIN address B ON A.address_id = B.address_id
  JOIN city C ON B.city_id = C.city_id
  JOIN country D ON C.country_id = D.country_id
  WHERE D.country IN (SELECT country FROM top_countries)
  GROUP BY D.country, C.city
  ORDER BY COUNT(A.customer_id) DESC
  LIMIT 10
),
total_amount_per_customer AS (
  SELECT
```

B.customer_id,

B.first_name,

```
B.last_name,
    E.country,
    D.city,
    SUM(A.amount) AS total_amount_paid
  FROM payment A
  INNER JOIN customer B ON A.customer_id = B.customer_id
  INNER JOIN address C ON B.address_id = C.address_id
  INNER JOIN city D ON C.city_id = D.city_id
  INNER JOIN country E ON D.country_id = E.country_id
  WHERE (E.country, D.city) IN (SELECT country, city FROM top_cities)
  GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
),
top_5_customers AS (
  SELECT customer_id, total_amount_paid
  FROM total_amount_per_customer
  ORDER BY total_amount_paid DESC
  LIMIT 5
SELECT AVG(total_amount_paid) AS average
FROM top_5_customers;
```



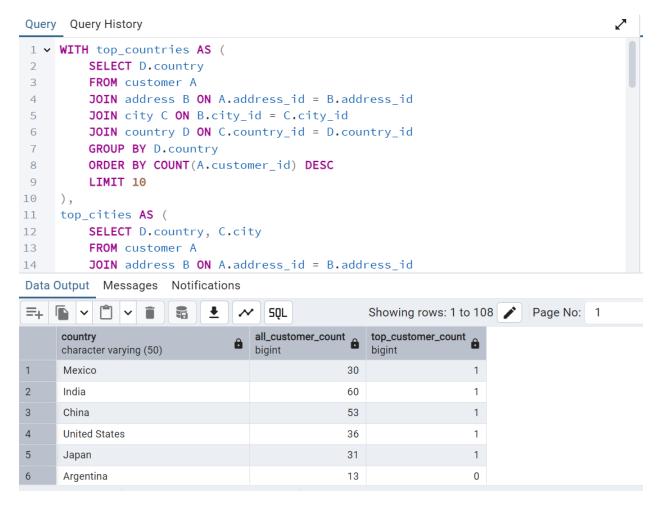
we first identify the **top 10 countries** with the most customers, then filter the **top 10 cities** in those countries. Next, we calculate the **total amount paid** by each customer in those locations. After that, we select the **top 5 customers** based on their total payments and finally compute the **average amount paid** by them.

```
1.2)
WITH top_countries AS (
    SELECT D.country
FROM customer A
    JOIN address B ON A.address_id = B.address_id
    JOIN city C ON B.city_id = C.city_id
    JOIN country D ON C.country_id = D.country_id
    GROUP BY D.country
    ORDER BY COUNT(A.customer_id) DESC
    LIMIT 10
),
top_cities AS (
    SELECT D.country, C.city
```

FROM customer A

```
JOIN address B ON A.address id = B.address id
  JOIN city C ON B.city_id = C.city_id
  JOIN country D ON C.country_id = D.country_id
  WHERE D.country IN (SELECT country FROM top_countries)
  GROUP BY D.country, C.city
  ORDER BY COUNT(A.customer_id) DESC
  LIMIT 10
),
total_amount_per_customer AS (
  SELECT
    B.customer_id,
    B.first_name,
    B.last_name,
    E.country,
    D.city,
    SUM(A.amount) AS total_amount_paid
  FROM payment A
  INNER JOIN customer B ON A.customer_id = B.customer_id
  INNER JOIN address C ON B.address_id = C.address_id
  INNER JOIN city D ON C.city_id = D.city_id
  INNER JOIN country E ON D.country_id = E.country_id
  WHERE (E.country, D.city) IN (SELECT country, city FROM top_cities)
  GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
),
top_5_customers AS (
  SELECT customer_id, country
  FROM total_amount_per_customer
  ORDER BY total_amount_paid DESC
  LIMIT 5
),
country_customer_count AS (
```

```
SELECT
    E.country,
    COUNT(DISTINCT B.customer_id) AS all_customer_count
  FROM customer B
  INNER JOIN address C ON B.address_id = C.address_id
  INNER JOIN city D ON C.city_id = D.city_id
  INNER JOIN country E ON D.country_id = E.country_id
  GROUP BY E.country
)
SELECT
  ccc.country,
  ccc.all_customer_count,
  COUNT(DISTINCT tc.customer_id) AS top_customer_count
FROM country_customer_count ccc
LEFT JOIN top_5_customers tc ON ccc.country = tc.country
GROUP BY ccc.country, ccc.all_customer_count
ORDER BY top_customer_count DESC;
```



we first count the **total number of customers per country**. Then, using the previous logic, we determine the **top 5 customers** and identify which country they belong to. Finally, we join this data to see how many of those top 5 customers exist in each country while also displaying the total customer count per country.

2)

CTEs generally perform better than subqueries because they help avoid repeated calculations and make queries easier to read. CTEs improve query structure and allow PostgreSQL to optimize execution more efficiently.

Query #1	Subquery	CTE
Cost	127.72	127.74
Execution Time (Milliseconds)	00:00:00.105	00:00:00.102

Query #2	Subquery	CTE
Cost	231.8	235.07
Execution Time	00:00:00.146	00:00:00.113

³⁾ One challenge I faced when replacing subqueries with CTEs was making sure the query logic stayed clear and correct. Subqueries are often more compact, while CTEs require breaking the query into multiple parts, which can be tricky to organize properly.