SQL Exercise 8: subqueries

Step 1: Find the average amount paid by the top 5 customers.

Query | Query History

```
SELECT AVG (total_payment_amount) AS avg_paid_top5
FROM
(SELECT customer.customer_id,
       customer.first_name,
       customer.last_name,
       city.city,
       country.country,
       SUM(payment.amount) AS total_payment_amount
FROM payment
INNER JOIN customer ON payment.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
WHERE city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni',
               'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang',
               'Sivas', 'Celaya', 'So Leopoldo')
  AND country IN ('India', 'China', 'United States', 'Japan',
                  'Mexico', 'Brazil', 'Russian Federation',
                  'Philippines', 'Turkey', 'Indonesia')
GROUP BY customer.customer_id, customer.first_name, customer.last_name,
         city.city, country.country
ORDER BY total_payment_amount DESC
LIMIT 5)  AS avg_amount_paid
```

Data Output | Messages | Notifications

SQL

	avg_paid_top5 numeric
1	107.3540000000000000

Step 2: Find out how many of the top 5 customers you identified in step 1 are based within each country.

Query Query History

```
SELECT country.country,
       COUNT(DISTINCT customer.customer_id) AS all_customer_count,
       COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count
FROM customer
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
LEFT JOIN (
    SELECT customer.customer_id, country.country
    FROM payment
    INNER JOIN customer ON payment.customer_id = customer.customer_id
    INNER JOIN address ON customer.address_id = address.address_id
    INNER JOIN city ON address.city_id = city.city_id
    INNER JOIN country ON city.country_id = country.country_id
    WHERE city.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni','Dhule (Dhulia)', 'Kurashiki', 'Pingxiang',
                        'Sivas', 'Celaya', 'So Leopoldo')
      AND country.country IN ('India', 'China', 'United States', 'Japan','Mexico', 'Brazil', 'Russian Federation',
                              'Philippines', 'Turkey', 'Indonesia')
    GROUP BY customer.customer_id, country.country
    ORDER BY SUM(payment.amount) DESC
    LIMIT 5
) AS top_5_customers
  ON top_5_customers.country = country.country
GROUP BY country.country
HAVING COUNT(DISTINCT top_5_customers.customer_id) >= 0
ORDER BY top_customer_count DESC, all_customer_count DESC LIMIT 5;
```

Data Output Messages Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Mexico	30	2
2	India	60	1
3	United States	36	1
4	Turkey	15	1
5	China	53	0

Total rows: 5 Query complete 00:00:00.134

Step 3:

1. Write 1 to 2 short paragraphs on the following:
    a. Do you think steps 1 and 2 could be done without using subqueries?

Well, by searching and reviewing, I assume that steps 1 and 2 could be done without subqueries by using multiple JOINs, filtering, or grouping directly in a single query.

For example, we could join all the relevant tables, group the data by customer or country, and then use ORDER BY and LIMIT to identify the top customers before counting them by country. However, this approach would make the query harder to read and maintain because the logic for identifying the top 5 customers and the logic for counting them would be mixed together in one long query.

b. When do you think subqueries are useful?

I think they are useful when we want to break a problem into smaller, more understandable parts. They let us create an intermediate result set—like the list of top 5 customers; that can be reused in a larger query without repeating the same calculations. They also make it easier to apply limits or filters to a specific part of data before joining it to other tables. This way our main query will be simpler and more readable.