**Answers 3.8**

**STEP 1:**

A screenshot of a computer

AI-generated content may be incorrect.

**CODE:**

SELECT AVG(total\_amount\_paid.total\_amount\_paid) AS average

FROM

(SELECT customer.customer\_id, customer.first\_name, customer.last\_name, city.city, country.country, SUM(payment.amount) AS total\_amount\_paid

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

(SELECT city.city

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

GROUP BY city, country

ORDER BY COUNT(customer.customer\_id) DESC

LIMIT 10)

GROUP BY customer.customer\_id, customer.first\_name, customer.last\_name, city.city, country.country

ORDER BY total\_amount\_paid DESC

LIMIT 5) AS total\_amount\_paid

**STEP 2:**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**CODE:**

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, customer.first\_name, customer.last\_name, city.city, country.country, SUM(payment.amount) AS total\_amount\_paid

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 (

SELECT city.city

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

GROUP BY city, country

ORDER BY COUNT(customer.customer\_id) desc

LIMIT 10)

GROUP BY customer.customer\_id, customer.first\_name, customer.last\_name, city.city, country.country

ORDER BY total\_amount\_paid desc

LIMIT 5) AS top\_5\_customers

ON customer.customer\_id = top\_5\_customers.customer\_id

GROUP BY country.country

ORDER BY top\_customer\_count desc

**STEP 3:**

* **Do you think steps 1 and 2 could be done without using subqueries?**

The queries from step 1 and step 2 rely heavily on subqueries to filter and aggregate data before performing calculations on the results. While it might be possible to rewrite them using JOINs functions instead of subqueries, doing so could make the queries more complex and harder to read.

* **When do you think subqueries are useful?**

Subqueries are useful here because they allow us to break down the problem into smaller, more manageable parts, first identifying the top paying customers, then calculating the average, and finally analyzing their distribution across countries.

Subqueries are particularly useful when we need to filter or aggregate data before using it in a main query. They help improve readability, especially when dealing with multistep calculations like ranking customers based on payments.