

## Directions

Create a new text document and call it “Answers 3.9.” You’ll save your queries, outputs, and written answers in this document.

### Step 1: Answer the business questions from steps 1 and 2 of task 3.8 using CTEs

1. Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.
2. Copy-paste your CTEs and their outputs into your answers document.
3. Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.

### WAS: SUBQUERY

```
SELECT ROUND(AVG(total_spend),3) AS average
FROM
(
SELECT A. customer_id,
A. first_name,
A.last_name,
C. city,
D. country,
SUM(F.amount) AS total_spend
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_id = D.country_id
INNER JOIN rental E ON A.customer_id = E.customer_id
INNER JOIN payment F ON E.rental_id = F.rental_id
WHERE city = 'Aurora' OR city = 'Acua' OR city = 'Citrus Heights' OR city = 'Iwaki' OR city =
'Ambattur' OR city = 'Shanwei' OR city = 'So Leopoldo' OR city = 'Teboksary' OR city = 'Tianjin' OR
city = 'Cianjur'
GROUP BY A. customer_id, A. first_name, A.last_name, C. city, D. country
ORDER BY total_spend DESC
LIMIT 5)
```

## NOW: CTE

WITH average (customer\_id, first\_name, last\_name, city, country, total\_spend) AS

(

SELECT A. customer\_id,

A. first\_name,

A.last\_name,

C. city,

D. country,

SUM(F.amount) AS total\_spend

FROM customer A

INNER JOIN address B ON A.address\_id = B.address\_id

INNER JOIN city C ON B.city\_id = C.city\_id

INNER JOIN country D ON C.country\_id = D.country\_id

INNER JOIN rental E ON A.customer\_id = E.customer\_id

INNER JOIN payment F ON E.rental\_id = F.rental\_id

WHERE city = 'Aurora' OR city = 'Acua' OR city = 'Citrus Heights' OR city = 'Iwaki' OR city =  
'Ambattur' OR city = 'Shanwei' OR city = 'So Leopoldo' OR city = 'Teboksary' OR city = 'Tianjin' OR  
city = 'Cianjur'

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

ORDER BY total\_spend DESC

LIMIT 5)

SELECT ROUND(AVG(total\_spend),3)

FROM average

Which gives the same result.

The screenshot shows a SQL IDE interface. The top pane displays a query with a CTE named 'average'. The query selects customer details and calculates the total spend for five specific cities. The bottom pane shows the result of the query, which is a single row with the average total spend rounded to three decimal places.

```
1 WITH average (customer_id, first_name, last_name, city, country, total_spend) AS
2 (
3 SELECT A. customer_id,
4 A. first_name,
5 A. last_name,
6 C. city,
7 D. country,
8 SUM(F.amount) AS total_spend
9 FROM customer A
10 INNER JOIN address B ON A.address_id = B.address_id
11 INNER JOIN city C ON B.city_id = C.city_id
12 INNER JOIN country D ON C.country_id = D.country_id
13 INNER JOIN rental E ON A.customer_id = E.customer_id
14 INNER JOIN payment F ON E.rental_id = F.rental_id
15 WHERE city = 'Aurora' OR city = 'Acua' OR city = 'Citrus Heights' OR city = 'Iwaki' OR city = 'Amba
16 GROUP BY A. customer_id, A. first_name, A. last_name, C. city, D. country
17 ORDER BY total_spend DESC
18 LIMIT 5)
19 SELECT ROUND(AVG(total_spend),3)
```

Data Output Messages Notifications

	round
	numeric
1	105.554

I stared off trying to get to the solution in one step, and I wasn't getting the syntax right.

Then I ran the CTE(/subquery) by itself and then I can visualise what I'm intergoating.

The screenshot shows the 'Data Output' pane of a SQL IDE. It displays the result of a CTE query, which is a table with 7 columns: customer\_id, first\_name, last\_name, city, country, and total\_spend. The data is sorted by total\_spend in descending order.

	customer_id	first_name	last_name	city	country	total_spend
	integer	character varying (45)	character varying (45)	character varying (50)	character varying (50)	numeric
1	225	Arlene	Harvey	Ambattur	India	111.76
2	424	Kyle	Spurlock	Shanwei	China	109.71
3	240	Marlene	Welch	Iwaki	Japan	106.77
4	486	Glen	Talbert	Acua	Mexico	100.77
5	537	Clinton	Buford	Aurora	United States	98.76

That made it a lot easier.

I got the WITH ... AS part of the query sorted – it was easy to see what columns, column names I needed to include

Then it was really easy to understand that I just needed to add:

```
SELECT ROUND(AVG(total_spend),3)
```

```
FROM average
```

## WAS: SUBQUERY

SELECT

D.country,

COUNT(DISTINCT A.customer\_id) AS all\_customer\_count,

COUNT(DISTINCT T.customer\_id) AS top\_customer\_count

FROM customer A

INNER JOIN address B ON A.address\_id = B.address\_id

INNER JOIN city C ON B.city\_id = C.city\_id

INNER JOIN country D ON C.country\_id = D.country\_id

LEFT JOIN (

SELECT A.customer\_id, D.country

FROM customer A

INNER JOIN address B ON A.address\_id = B.address\_id

INNER JOIN city C ON B.city\_id = C.city\_id

INNER JOIN country D ON C.country\_id = D.country\_id

INNER JOIN rental E ON A.customer\_id = E.customer\_id

INNER JOIN payment F ON E.rental\_id = F.rental\_id

WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur',

'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')

GROUP BY A.customer\_id, D.country

ORDER BY SUM(F.amount) DESC

LIMIT 5

) AS T ON A.customer\_id = T.customer\_id

GROUP BY D.country

ORDER BY all\_customer\_count DESC;

## NOW: CTE

```
WITH top_customers AS (  
    SELECT A.customer_id, D.country  
    FROM customer A  
    INNER JOIN address B ON A.address_id = B.address_id  
    INNER JOIN city C ON B.city_id = C.city_id  
    INNER JOIN country D ON C.country_id = D.country_id  
    INNER JOIN rental E ON A.customer_id = E.customer_id  
    INNER JOIN payment F ON E.rental_id = F.rental_id  
    WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur',  
        'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')  
    GROUP BY A.customer_id, D.country  
    ORDER BY SUM(F.amount) DESC  
    LIMIT 5  
)  
SELECT  
    D.country,  
    COUNT(DISTINCT A.customer_id) AS all_customer_count,  
    COUNT(DISTINCT T.customer_id) AS top_customer_count  
FROM customer A  
INNER JOIN address B ON A.address_id = B.address_id  
INNER JOIN city C ON B.city_id = C.city_id  
INNER JOIN country D ON C.country_id = D.country_id  
LEFT JOIN top_customers T ON A.customer_id = T.customer_id  
GROUP BY D.country  
ORDER BY top_customer_count DESC;
```



## Step 2: Compare the performance of your CTEs and subqueries.

1. Which approach do you think will perform better and why?

I honestly don't know. AT this scale, they are pretty much of the same level of complexity. I can see that if you need many, having a complete block of CTEs at the start is going to be much clearer than looking for the subqueries.

2. Compare the costs of all the queries by creating query plans for each one.
3. The EXPLAIN command gives you an *estimated* cost. To find out the actual speed of your queries, run them in pgAdmin 4. After you've run each query, a popup window will display its speed in milliseconds.

Step 1 Subquery	"Aggregate (cost=525.92..525.93 rows=1 width=32)"
Step 1 CTE	"Aggregate (cost=525.92..525.93 rows=1 width=32)"
Step 2 Subquery	"Sort (cost=617.38..617.66 rows=109 width=25)"
Step 2 CTE	"Sort (cost=617.38..617.66 rows=109 width=25)"

4. Did the results surprise you? Write a few sentences to explain your answer.

There's no difference in cost for the subqueries and CTEs

I'm not surprised they are doing the same thing, just using a very slightly different method.

## Step 3:

Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.

The main challenge is continuing to understand the syntax and how to visualise what I'm trying to do and write instructions, then when I write the wrong instructions, working out why that doesn't do what I think.

I'm starting to lean a little less heavily on Co-Pilot and more on trial and error, and breaking the query down into chunks that I definitely DO understand.