3.9: COMMON TABLE EXPRESSIONS

STEP 1

Rewrite the queries from steps 1 and 2 of task 3.8 as CTEs.

Average amount paid by TOP 5 customers.

```
24  WITH average_amount_cte (total_amount_paid,customer_id,first_name,last_name,city,country) AS
25  (SELECT SUM(A.amount) AS total_amount_paid,
26          B.customer_id,
27          B.first_name,
28          B.last_name,
29          D.city,
30          E.country
31  FROM payment A
32  INNER JOIN customer B on A.customer_id = B.customer_id
33  INNER JOIN address C on B.address_id = C.address_id
34  INNER JOIN city D on C.city_id = D.city_id
35  INNER JOIN country E on D.country_id = E.country_id
36  WHERE city IN ('Aurora','Acua','Citrus Heights','Iwaki','Ambattur','Shanwei',
37                 'So Leopoldo','Teboksary','Tianjin','Cianjur')
38  GROUP BY B.customer_id,
39           first_name,
40           last_name,
41           city,
42           country
43  ORDER BY total_amount_paid DESC
44  LIMIT 5)
45  SELECT AVG(total_amount_paid) AS average_amount_paid
46  FROM average_amount_cte
47
```

Data Output | Messages | Notifications

	average_amount_paid numeric
1	105.5540000000000000

Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.

1.1: I took the subquery from 3.8 and used it to define the CTE. I then queried the main statement by looking for the average_amount_paid.

1.2: I proceeded similarly as in previous step. However, here we needed two CTEs: top5_customer_count_cte and all_customer_count_cte

How many of the TOP 5 customers are based within each country.

Query | Query History

```
WITH
top5_customer_count_cte AS
(SELECT SUM(A.amount) AS total_amount_paid,
        B.customer_id,
        B.first_name,
        B.last_name,
        D.city,
        E.country
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 city IN ('Aurora','Acua','Citrus Heights','Iwaki','Ambattur','Shanwei',
               'So Leopoldo','Teboksary','Tianjin','Cianjur')
GROUP BY B.customer_id,
         first_name,
         last_name,
         city,
         country
ORDER BY total_amount_paid DESC
LIMIT 5),

all_customer_count_cte AS
(SELECT E.country,
        COUNT(DISTINCT B.customer_id) AS all_customer_count,
        COUNT(DISTINCT E.country) AS top5_customer_count
FROM country E
INNER JOIN city D ON E.country_id = D.country_id
INNER JOIN address C ON D.city_id = C.city_id
INNER JOIN customer B ON C.address_id = B.address_id
GROUP BY E.country)

SELECT E.country,
       COUNT(DISTINCT B.customer_id) AS all_customer_count,
       COUNT(DISTINCT top5_customer_count_cte.customer_id) AS top5_customer_count
FROM country E
INNER JOIN city D ON E.country_id = D.country_id
INNER JOIN address C ON D.city_id = C.city_id
INNER JOIN customer B ON C.address_id = B.address_id
LEFT JOIN
top5_customer_count_cte ON E.country = top5_customer_count_cte.country
GROUP BY E.country
ORDER BY all_customer_count DESC
LIMIT 5
```

Data Output | Messages | Notifications

	country character varying (50)	all_customer_count bigint	top5_customer_count bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1

STEP 2

Which approach do you think will perform better and why?

I think that in this task, both CTE and subquery methods will perform similarly since the complexity of the queries overall seems rather even. Although CTE queries are more readable.

Compare the costs of all the queries by creating query plans for each one.

Task 1 - Subquery

143

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=64.45..64.46 rows=1 width=32)
2	-> Limit (cost=64.37..64.39 rows=5 width=67)
3	-> Sort (cost=64.37..64.98 rows=243 width=67)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=67)

Task 1 - CTE

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=64.45..64.46 rows=1 width=32)
2	-> Limit (cost=64.37..64.39 rows=5 width=67)
3	-> Sort (cost=64.37..64.98 rows=243 width=67)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=67)

Task 2 - Subquery

Data Output | Messages | Notifications

	QUERY PLAN text
1	Limit (cost=166.23..166.24 rows=5 width=25)
2	-> Sort (cost=166.23..166.50 rows=109 width=25)
3	Sort Key: (count(DISTINCT b.customer_id)) DESC
4	-> GroupAggregate (cost=155.43..164.42 rows=109 width=25)
5	Group Key: e.country
6	-> Merge Left Join (cost=155.43..158.83 rows=599 width=17)

Total rows: 46 of 46 Query complete 00:00:00.658

Task - CTE

Data Output | Messages | Notifications

	QUERY PLAN text
1	Limit (cost=166.23..166.24 rows=5 width=25)
2	-> Sort (cost=166.23..166.50 rows=109 width=25)
3	Sort Key: (count(DISTINCT b.customer_id)) DESC
4	-> GroupAggregate (cost=155.43..164.42 rows=109 width=25)
5	Group Key: e.country
6	-> Merge Left Join (cost=155.43..158.83 rows=599 width=17)

The EXPLAIN command gives you an *estimated* cost. To find out the actual speed of your queries, run them in pgAdmin 4. After each query has been run, a pop-up window will display its speed in milliseconds.

The real time it took to process the queries varied every time I executed them. Sometimes it took a bit longer and sometimes a bit faster.

Overall, the results did not surprise me. The Query plans were similar in both cases.

STEP 3

This task was one of the most challenging exercises so far. The first task was relatively simple with replacing the subquery for CTE statement. However, the second task was hard mainly because a second CTE had to be added which was not discussed in the reading part of the exercise.

I believe with more practice I will be more comfortable with the syntax and it will be more intuitive.