

Task 3.9 Common Table Expressions

By Lee Heng Chuah

In this task, you will convert your subqueries from task 3.8 into CTEs to make your code easier to read.

Directions:

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

Step 1: Answer the business questions from step 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.

Rewrite Step 1:

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 WITH average(customer_id, first_name, last_name, city, country) AS
2 (SELECT A.customer_id,
3      A.first_name,
4      A.last_name,
5      C.city,
6      D.country,
7      SUM(E.amount) AS total_amount_paid
8 FROM customer A
9 INNER JOIN address B ON A.address_id = B.address_id
10 INNER JOIN city C ON B.city_id = C.city_id
11 INNER JOIN country D ON C.country_id = D.country_id
12 INNER JOIN payment E ON A.customer_id = E.customer_id
13 WHERE country IN ('India',
14                  'China',
15                  'United States',
16                  'Japan',
17                  'Mexico',
18                  'Brazil',
19                  'Russian Federation',
20                  'Philippines',
21                  'Turkey',
22                  'Indonesia')
23 AND city IN ('Aurora',
24             'Tokat',
25             'Tarsus',
26             'Atlixco',
27             'Emeishan',
28             'Pontianak',
29             'Shimoga',
30             'Aparecida de Goiania',
31             'Zalantun',
32             'Taguig')
33 GROUP BY A.customer_id,
34          A.first_name,
35          A.last_name,
36          C.city,
37          D.country)
38 ORDER BY total_amount_paid DESC
39 LIMIT 5)
40 SELECT AVG(total_amount_paid)
41 FROM average
```

The query is executed, and the output is displayed in a table:

avg
numeric
120.322

The status bar at the bottom indicates: Total rows: 1 of 1 Query complete 00:00:00.174 Ln 37, Col 19

Rewrite Step 2:

DashboardPropertiesSQLStatisticsDependenciesDependentsRockbuster/postgres@PostgreSQL 14*

Rockbuster/postgres@PostgreSQL 14

QueryQuery History

Scratch Pad

```
1 WITH top_5_customers(customer_id, first_name, last_name, city, country) AS
2 (SELECT A.customer_id,
3      A.first_name,
4      A.last_name,
5      C.city,
6      D.country,
7      SUM(E.amount) AS total_amount_paid
8 FROM customer A
9 INNER JOIN address B ON A.address_id = B.address_id
10 INNER JOIN city C ON B.city_id = C.city_id
11 INNER JOIN country D ON C.country_id = D.country_id
12 INNER JOIN payment E ON A.customer_id = E.customer_id
13 WHERE country IN ('India',
14                  'China',
15                  'United States',
16                  'Japan',
17                  'Mexico',
18                  'Brazil',
19                  'Russian Federation',
20                  'Philippines',
21                  'Turkey',
22                  'Indonesia')
23 AND city IN ('Aurora',
24             'Tokat',
25             'Tarsus',
26             'Atlixco',
27             'Emeishan',
28             'Pontianak',
29             'Shimoga',
30             'Aparecida de Goinia',
31             'Zalantun',
32             'Taguig')
33 GROUP BY A.customer_id,
34          A.first_name,
35          A.last_name,
36          C.city,
37          D.country
38 ORDER BY total_amount_paid DESC
39 LIMIT 5)
40
41 SELECT DISTINCT(D.country),
42        COUNT(DISTINCT A.customer_id) AS all_customer_count,
43        COUNT(DISTINCT D.country) AS top_customer_count
44 FROM customer A
45 INNER JOIN address B ON A.address_id = B.address_id
46 INNER JOIN city C ON B.city_id = C.city_id
47 INNER JOIN country D ON C.country_id = D.country_id
48 LEFT JOIN top_5_customers ON D.country = top_5_customers.Country
49 GROUP BY D.country, top_5_customers
50 ORDER BY all_customer_count DESC
51 LIMIT 5;
```

Data outputMessagesNotifications

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

Total rows: 5 of 5Query complete 00:00:00.105Ln 50, Col 29

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

I first define the database I required. And then, I run the query of total_amount_paid. After completing the query of total_amount_paid, I define the CTE using the WITH clause, give the CTE a name and provide the AS keyword. I finally add the total_amount_paid query into parentheses. I finish the main statement with SELECT, FROM CTE, GROUP BY, ORDER and LIMIT the results I need.

Step 2: Compare the performance of your CTEs and subqueries

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

I personally think the CTE would perform better as CTE is defined at the start of the main query, which makes them easier to spot in the main query. As CTE is determined at the beginning of the main query, it makes the CTE can replace any subquery easily.

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 each query has been run, a pop-up window will display its speed in milliseconds.

STEP 1: Exercise 3.8

Dashboard Properties SQL Statistics Dependencies Dependents **Rockbuster/postgres@PostgreSQL 14***

Rockbuster/postgres@PostgreSQL 14

No limit

Query Query History Scratch Pad x

```

16      'United States',
17      'Japan',
18      'Mexico',
19      'Brazil',
20      'Russian Federation',
21      'Philippines',
22      'Turkey',
23      'Indonesia')
24 AND city IN ('Aurora',
25             'Tokat',
26             'Tarsus',
27             'Atlixco',
28             'Emeishan',
29             'Pontianak',
30             'Shimoga',
31             'Aparecida de Goiania',
32             'Zalantun',
33             'Taguig')
34 GROUP BY A.customer_id,
35          A.first_name,
36          A.last_name,
37          C.city,
38          D.country
39 ORDER BY total_amount_paid DESC
40 LIMIT 5) AS top_5_customers

```

Data output Messages Notifications

QUERY PLAN	
	text
1	Aggregate (cost=29.56..29.57 rows=1 width=32)
2	-> Limit (cost=29.48..29.50 rows=5 width=67)
3	-> Sort (cost=29.48..29.55 rows=28 width=67)
4	Sort Key: (sum(e.amount)) DESC
5	-> HashAggregate (cost=28.67..29.02 rows=28 width=67)
6	Group Key: a.customer_id, c.city, d.country
7	-> Nested Loop (cost=3.62..28.39 rows=28 width=41)
8	-> Nested Loop (cost=3.33..26.40 rows=1 width=35)
9	-> Nested Loop (cost=3.06..26.01 rows=1 width=22)
10	-> Hash Join (cost=2.79..21.31 rows=1 width=22)
11	Hash Cond: (c.country_id = d.country_id)
12	-> Seq Scan on city c (cost=0.00..18.50 rows=10 width=15)
13	Filter: ((city)::text = ANY ('{Aurora,Tokat,Tarsus,Atlixco,Emeishan,Pontianak,Shimoga,Aparecida de Goiania,Zalantun,Taguig}::text[]))
14	-> Hash (cost=2.66..2.66 rows=10 width=13)
15	-> Seq Scan on country d (cost=0.03..2.66 rows=10 width=13)
16	Filter: ((country)::text = ANY ('{India,China,'United States','Japan,Mexico,Brazil','Russian Federation','Philippines,Turkey,Indonesia'}::text[]))
17	-> Index Scan using idx_customer_id on address e (cost=0.28..4.60 rows=1 width=6)
Total rows:	22 of 22 Query complete 00:00:00.052

Ln 40, Col 27

STEP 1: Exercise 3.9

DashboardPropertiesSQLStatisticsDependenciesDependentsRockbuster/postgres@PostgreSQL 14*

Rockbuster/postgres@PostgreSQL 14

No limit

QueryQuery HistoryScratch Pad

```
1 EXPLAIN
2 WITH average(customer_id, first_name, last_name, city, country) AS
3 (SELECT A.customer_id,
4      A.first_name,
5      A.last_name,
6      C.city,
7      D.country,
8      SUM(E.amount) AS total_amount_paid
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 payment E ON A.customer_id = E.customer_id
14 WHERE country IN ('India',
15                  'China',
16                  'United States',
17                  'Japan',
18                  'Mexico',
19                  'Brazil',
20                  'Russian Federation'.

```

Data outputMessagesNotifications

QUERY PLAN
text

1	Aggregate (cost=29.56..29.57 rows=1 width=32)
2	-> Limit (cost=29.48..29.50 rows=5 width=67)
3	-> Sort (cost=29.48..29.55 rows=28 width=67)
4	Sort Key: (sum(e.amount)) DESC
5	-> HashAggregate (cost=28.67..29.02 rows=28 width=67)
6	Group Key: a.customer_id, c.city, d.country
7	-> Nested Loop (cost=3.62..28.39 rows=28 width=41)
8	-> Nested Loop (cost=3.33..26.40 rows=1 width=35)
9	-> Nested Loop (cost=3.06..26.01 rows=1 width=22)
10	-> Hash Join (cost=2.79..21.31 rows=1 width=22)
11	Hash Cond: (c.country_id = d.country_id)
12	-> Seq Scan on city c (cost=0.00..18.50 rows=10 width=15)
13	Filter: ((city)::text = ANY ('(Aurora,Tokat,Tarsus,Atlixco,Emeishan,Pontianak,Shimoga,Aparecida de Goinia',Zalantun,Taguig)::text[]))
14	-> Hash (cost=2.66..2.66 rows=10 width=13)
15	-> Seq Scan on country d (cost=0.03..2.66 rows=10 width=13)
16	Filter: ((country)::text = ANY ('(India,China,United States',Japan,Mexico,Brazil,Russian Federation',Philippines,Turkey,Indonesia)::text[]))
17	-> Index Scan using idx_fk_city_id on address b (cost=0.28..4.69 rows=1 width=6)
18	Index Cond: (city_id = c.city_id)
19	-> Index Scan using idx_fk_address_id on customer a (cost=0.28..0.38 rows=1 width=19)
20	Index Cond: (address_id = b.address_id)
21	-> Index Scan using idx_fk_customer_id on payment e (cost=0.29..1.71 rows=28 width=8)

Total rows: 22 of 22Query complete 00:00:00.053Ln 1, Col 8

STEP 2: Exercise 3.8

Dashboard Properties SQL Statistics Dependencies Dependents **Rockbuster/postgres@PostgreSQL 14***

Rockbuster/postgres@PostgreSQL 14

No limit

Query Query History

Scratch Pad

```
1 EXPLAIN
2 SELECT DISTINCT(D.country),
3       COUNT(DISTINCT A.customer_id) AS all_customer_count,
4       COUNT(DISTINCT D.country) AS top_customer_count
5 FROM customer A
6 INNER JOIN address B ON A.address_id = B.address_id
7 INNER JOIN city C ON B.city_id = C.city_id
8 INNER JOIN country D ON C.country_id = D.country_id
9 LEFT JOIN(SELECT A.customer_id,
10              A.first_name,
11              A.last_name,
12              C.city,
13              D.country,
14              SUM(E.amount) AS total_amount_paid
15 FROM customer A
16 INNER JOIN address B ON A.address_id = B.address_id
17 INNER JOIN city C ON B.city_id = C.city_id
18 INNER JOIN country D ON C.country_id = D.country_id
19 INNER JOIN payment E ON A.customer_id = E.customer_id
20 WHERE country IN ('India',
21                  'China',
22                  'United States',
23                  'Japan',
24                  'Mexico',
25                  'Brazil',
```

Data output Messages Notifications

QUERY PLAN

text

1	Limit (cost=154.59..154.60 rows=5 width=84)
2	-> Sort (cost=154.59..155.95 rows=545 width=84)
3	Sort Key: (count(DISTINCT a.customer_id)) DESC
4	-> HashAggregate (cost=140.09..145.54 rows=545 width=84)
5	Group Key: count(DISTINCT a.customer_id), d.country, count(DISTINCT d.country)
6	-> GroupAggregate (cost=123.06..136.00 rows=545 width=84)
7	Group Key: d.country, top_5_customers.*
8	-> Sort (cost=123.06..124.56 rows=599 width=72)
9	Sort Key: d.country, top_5_customers.*
10	-> Hash Left Join (cost=73.13..95.43 rows=599 width=72)
11	Hash Cond: ((d.country)::text = (top_5_customers.country)::text)
12	-> Hash Join (cost=43.52..63.30 rows=599 width=13)
13	Hash Cond: (c.country_id = d.country_id)
14	-> Hash Join (cost=40.07..58.22 rows=599 width=6)
15	Hash Cond: (b.city_id = c.city_id)
16	-> Hash Join (cost=21.57..38.14 rows=599 width=6)
17	Hash Cond: (a.address_id = b.address_id)

Total rows: 47 of 47 Query complete 00:00:00.102

Ln 1, Col 8

STEP 2: Exercise 3.9

The screenshot shows a PostgreSQL IDE interface. The top bar includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The main editor displays a SQL query using a CTE to find the top 5 customers by total amount paid in India, China, United States, Japan, Mexico, Brazil, Russian Federation, and Philippines. The query is as follows:

```
1 EXPLAIN
2 WITH top_5_customers(customer_id, first_name, last_name, city, country) AS
3 (SELECT A.customer_id,
4      A.first_name,
5      A.last_name,
6      C.city,
7      D.country,
8      SUM(E.amount) AS total_amount_paid
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 payment E ON A.customer_id = E.customer_id
14 WHERE country IN ('India',
15                  'China',
16                  'United States',
17                  'Japan',
18                  'Mexico',
19                  'Brazil',
20                  'Russian Federation',
21                  'Philippines',
```

Below the query editor, the 'Data output' tab shows the 'QUERY PLAN' for the executed query. The plan details the execution steps, including limits, sorts, hash aggregates, and various joins, along with their estimated costs and row counts.

Step	Operation	Cost	Rows	Width
1	Limit	154.59..154.60	5	84
2	Sort	154.59..155.95	545	84
3	Sort Key: (count(DISTINCT a.customer_id)) DESC			
4	HashAggregate	140.09..145.54	545	84
5	Group Key: count(DISTINCT a.customer_id), d.country, count(DISTINCT d.country)			
6	GroupAggregate	123.06..136.00	545	84
7	Group Key: d.country, top_5_customers.*			
8	Sort	123.06..124.56	599	72
9	Sort Key: d.country, top_5_customers.*			
10	Hash Left Join	73.13..95.43	599	72
11	Hash Cond: ((d.country)::text = (top_5_customers.country)::text)			
12	Hash Join	43.52..63.30	599	13
13	Hash Cond: (c.country_id = d.country_id)			
14	Hash Join	40.07..58.22	599	6
15	Hash Cond: (b.city_id = c.city_id)			
16	Hash Join	21.57..38.14	599	6
17	Hash Cond: (a.address_id = b.address_id)			
18	Seq Scan on customer a	0.00..14.99	599	6
19	Hash	14.03..14.03	603	6

Total rows: 47 of 47 Query complete 00:00:00.114 Ln 1, Col 8

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

Yes, the results totally surprised me. Both STEP 1 & STEP 2 run the same costs, either subqueries or CTE. Both CTE and subquery achieved the same results. From an accessibility, readability, and performance perspective, CTE makes life much easier.

Step 3:

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

Compared with subqueries, I found CTE is handy as it lets me create the query with the information I need at the start of the main query, making it easier to spot in the main query. I then just write the main query after the CTE. This makes my code much cleaner and easier to read than subqueries. While subquery could be challenged as these queries have been nesting or referenced in different statements under the main statement. I personally found subqueries are far more challenging than CTEs.

Step 4:

Save your “Answers 3.9” document as a PDF and upload it here for your tutor to review.

- a. Copy-paste your query and its output into your answers document

Dashboard Properties SQL Statistics Dependencies Dependents [Rockbuster/postgres@PostgreSQL 14*](#)

Rockbuster/postgres@PostgreSQL 14

Query Query History Scratch Pad

```

1 SELECT D.country,
2       COUNT(A.customer_id) AS customer_number
3 FROM customer A
4 INNER JOIN address B ON A.address_id = B.address_id
5 INNER JOIN city C ON B.city_id = C.city_id
6 INNER JOIN country D ON C.country_id = D.country_id
7 GROUP BY country
8 ORDER BY customer_number DESC
9 LIMIT 10

```

Data output Messages Notifications

	country character varying (50)	customer_number bigint
1	India	60
2	China	53
3	United States	36
4	Japan	31
5	Mexico	30
6	Brazil	28
7	Russian Federation	28
8	Philippines	20
9	Turkey	15
10	Indonesia	14

Total rows: 10 of 10 Query complete 00:00:00.112 Ln 4, Col 23

- b. Write a few sentences on how you approached this query and why. It is important that you can explain your thought process when writing queries, especially for future interviews.

I revisited and understood the customer, address, city, and country data. I then joined the customer data with the address, city, and country data with the inner join query. I put the GROUP BY query and counted the customer based on country. Later, I put the ORDER BY command based on customer numbers from highest to lowest and LIMIT to 10.

2. Write a query to find the top 10 cities within the top 10 countries identified in step 1.

- a. Copy-paste your query and its output into your answers document

The screenshot shows a PostgreSQL query editor interface. The top bar includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The main editor area displays a SQL query that selects the top 10 cities based on the number of customers, filtered by a list of 10 countries. The query is as follows:

```
1 SELECT C.city,
2       COUNT(A.customer_id) AS customer_number
3 FROM customer A
4 INNER JOIN address B ON A.address_id = B.address_id
5 INNER JOIN city C ON B.city_id = C.city_id
6 INNER JOIN country D ON C.country_id = D.country_id
7 WHERE country IN ('India',
8                  'China',
9                  'United States',
10                 'Japan',
11                 'Mexico',
12                 'Brazil',
13                 'Russian Federation',
14                 'Philippines',
15                 'Turkey',
16                 'Indonesia')
17 GROUP BY city
18 ORDER BY customer_number DESC
19 LIMIT 10
```

Below the query editor, the 'Data output' tab shows the results of the query in a table format:

	city character varying (50)	customer_number bigint
1	Aurora	2
2	Tokat	1
3	Tarsus	1
4	Atlixco	1
5	Emeishan	1
6	Pontianak	1
7	Shimoga	1
8	Aparecida de Goinia	1
9	Zalantun	1
10	Taguig	1

The bottom status bar indicates 'Total rows: 10 of 10', 'Query complete 00:00:00.060', and 'Ln 16, Col 31'.

- b. Write a short explanation of how you approached this query and why.

I revisited my existing query. I changed the SELECT from country to city. To find the top 10 cities within the top 10 countries, I listed out the top 10 countries by the WHERE query. Lastly, I changed the GROUP BY from country to city.

3. Write a query to find the top 5 customers in the top 10 cities who have paid the highest total amounts to Rockbuster. The customer team would like to reward them for their loyalty.
- Tip: After the join syntax, you will need to use the WHERE clause with an operator, followed by GROUP BY and ORDER BY. Your output should include the following columns: Customer ID, Customer First Name and Last Name, Country, City, Total Amount Paid
 - Copy-past your query and its output into your answers document

Dashboard Properties SQL Statistics Dependencies Dependents [Rockbuster/postgres@PostgreSQL 14*](#)

Rockbuster/postgres@PostgreSQL 14

No limit

Query Query History Scratch Pad

```

1 SELECT A.customer_id,
2       A.first_name,
3       A.last_name,
4       C.city,
5       D.country,
6       SUM(E.amount) AS total_payment
7 FROM customer A
8 INNER JOIN address B ON A.address_id = B.address_id
9 INNER JOIN city C ON B.city_id = C.city_id
10 INNER JOIN country D ON C.country_id = D.country_id
11 INNER JOIN payment E ON A.customer_id = E.customer_id
12 WHERE city IN ('Aurora',
13              'Tokat',
14              'Tarsus',
15              'Atlixco',
16              'Emeishan',
17              'Pontianak',
18              'Shimoga',
19              'Aparecida de Goiania',
20              'Zalantun',
21              'Taguig')
22 GROUP BY A.customer_id, C.city, D.country
23 ORDER BY total_payment DESC
24 LIMIT 5

```

Data output Messages Notifications

	customer_id integer	first_name character varying (45)	last_name character varying (45)	city character varying (50)	country character varying (50)	total_payment numeric
1	566	Casey	Mena	Tokat	Turkey	130.68
2	84	Sara	Perry	Atlixco	Mexico	128.7
3	506	Leslie	Seward	Pontianak	Indonesia	123.72
4	389	Alan	Kahn	Emeishan	China	119.75
5	537	Clinton	Buford	Aurora	United States	98.76

Total rows: 5 of 5 Query complete 00:00:00.059 Ln 21, Col 25

- Save your "Answer 3.7" document as a PDF and upload it here for your tutor to review.