

ANSWERS 3.9

In this task, you'll 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'll save your queries, outputs, and written answers in this document.

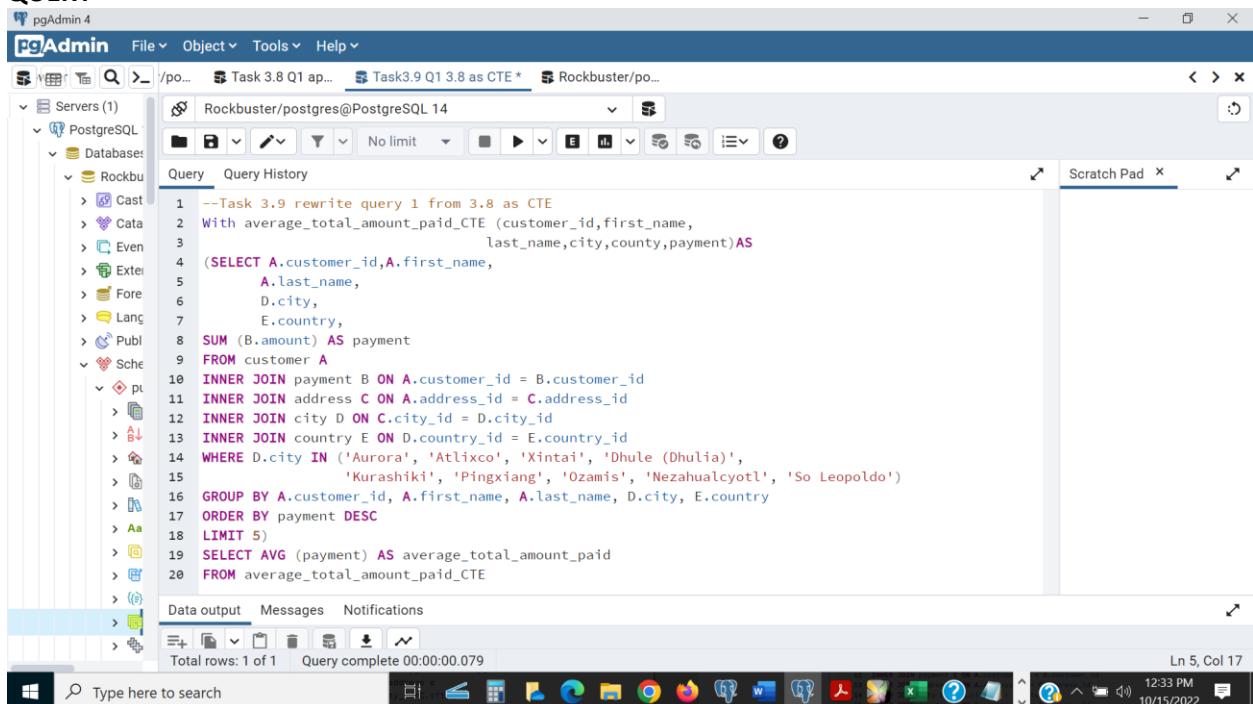
Step 1: Answer the business questions from step 1 and 2 of task 3.8 using CTEs

Business questions step 1 of task 3.8 using CTE's

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.

QUERY-1 REWRITTEN AS CTE - "Find the average amount paid by the top 5 customers".

QUERY



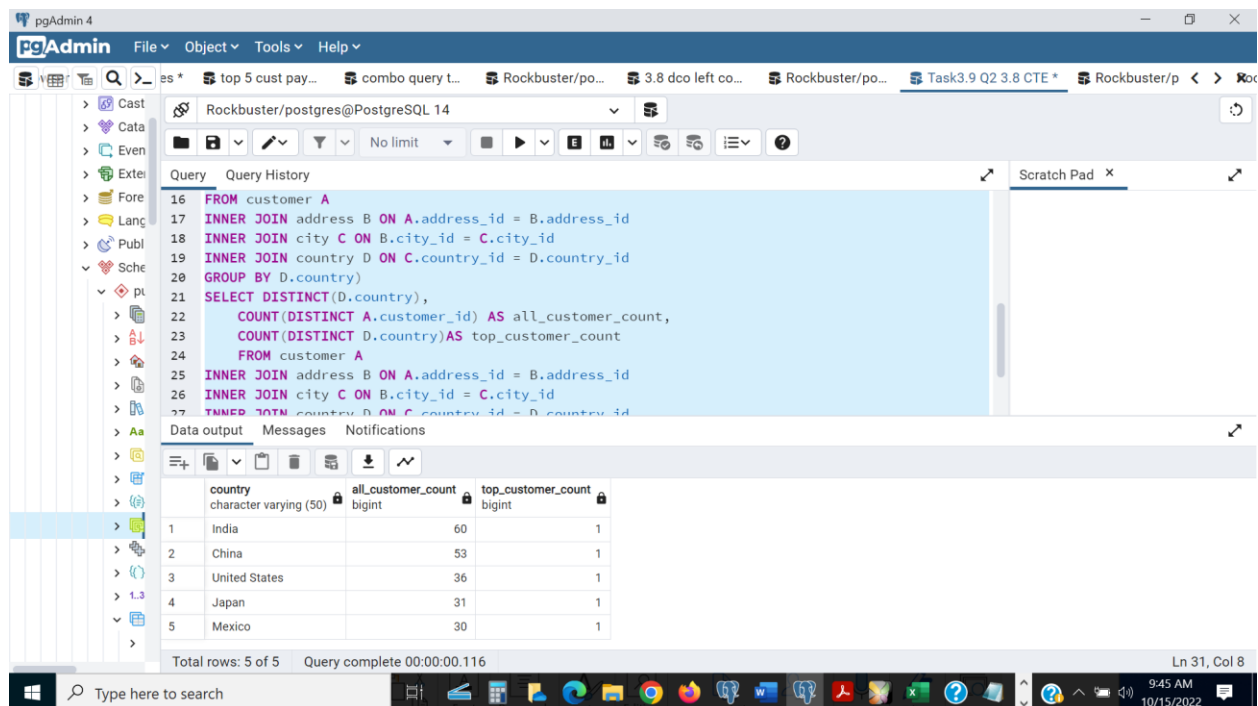
```
--Task 3.9 rewrite query 1 from 3.8 as CTE
1 With average_total_amount_paid_CTE (customer_id,first_name,
2 last_name,city,county,payment)AS
3
4 (SELECT A.customer_id,A.first_name,
5 A.last_name,
6 D.city,
7 E.country,
8 SUM (B.amount) AS payment
9 FROM customer A
10 INNER JOIN payment B ON A.customer_id = B.customer_id
11 INNER JOIN address C ON A.address_id = C.address_id
12 INNER JOIN city D ON C.city_id = D.city_id
13 INNER JOIN country E ON D.country_id = E.country_id
14 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Dhule (Dhulia)',
15 'Kurashiki', 'Pingxiang', 'Ozamis', 'Nezahualcyotl', 'So Leopoldo')
16 GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
17 ORDER BY payment DESC
18 LIMIT 5)
19 SELECT AVG (payment) AS average_total_amount_paid
20 FROM average_total_amount_paid_CTE
```

Total rows: 1 of 1 Query complete 00:00:00.079 Ln 5, Col 17


```

COUNT(DISTINCT A.customer_id) AS all_customer_count,
COUNT(DISTINCT D.country) 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_5_customers_CTE ON D.country = top_5_customers_CTE.country
GROUP BY D.country
ORDER BY all_customer_count DESC
LIMIT 5

```



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```

16 FROM customer A
17 INNER JOIN address B ON A.address_id = B.address_id
18 INNER JOIN city C ON B.city_id = C.city_id
19 INNER JOIN country D ON C.country_id = D.country_id
20 GROUP BY D.country
21 SELECT DISTINCT(D.country),
22        COUNT(DISTINCT A.customer_id) AS all_customer_count,
23        COUNT(DISTINCT D.country) AS top_customer_count
24 FROM customer A
25 INNER JOIN address B ON A.address_id = B.address_id
26 INNER JOIN city C ON B.city_id = C.city_id
27 INNER JOIN country D ON C.country_id = D.country_id

```

The results pane shows the following data:

country	all_customer_count	top_customer_count
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 5 Query complete 00:00:00.116 Ln 31, Col 8

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

1. Which approach do you think will perform better and why? [I would think that since the subquery has less lines of code it may run faster than the CTE.](#)
2. Compare the costs of all the queries by creating query plans for each one. [See below table](#)
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.
4. Did the results surprise you? Write a few sentences to explain your answer. [Results were surprisingly the same for costs but run times would change each time I would rerun the query, must have to do with accessing the data and possibly my system.](#)

QUERY-1 3.8 Subquery / REWRITTEN AS CTE

	Subquery	CTE
Cost	"Aggregate (cost=61.33..61.34 rows=1 width=32)"	"Aggregate (cost=61.33..61.34 rows=1 width=32)"
Time	Total query runtime: 146 msec. 1 rows affected	Total query runtime: 173 msec. 1 rows affected

pgAdmin 4

Rockbuster/postgres@PostgreSQL 14

Query

```
--Task 3.8 Q1 Subquery
EXPLAIN SELECT AVG (payment) AS average
FROM
(SELECT A.customer_id,
A.first_name,
A.last_name,
D.city,
E.country,
...
```

QUERY PLAN

```
1 Aggregate (cost=61.33..61.34 rows=1 width=32)
2 -> Limit (cost=61.26..61.27 rows=5 width=67)
3 -> Sort (cost=61.26..61.80 rows=219 width=67)
4 Sort Key: (sum(b.amount)) DESC
5 -> HashAggregate (cost=54.88..57.62 rows=219 width=67)
6 Group Key: a.customer_id, d.city, e.country
7 -> Nested Loop (cost=18.15..52.69 rows=219 width=41)
8 -> Hash Join (cost=17.86..36.73 rows=9 width=35)
```

Total rows: 22 of 22 Query complete 00:00:00.413 Ln 1, Col 23

pgAdmin 4

Rockbuster/postgres@PostgreSQL 14

Query

```
--Task 3.8 Q1 Subquery
SELECT AVG (payment) AS average
FROM
(SELECT A.customer_id,
A.first_name,
A.last_name,
D.city,
E.country,
...
```

Data output

average
106.7500000

Total rows: 1 of 1 Query complete 00:00:00.146 Ln 2, Col 1

Successfully run. Total query runtime: 146 msec. 1 rows affected.

pgAdmin 4

File Object Tools Help

Rockbuster/postgres@PostgreSQL 14

Query Query History

```
--Task 3.9 rewrite Q1 3.8 as CTE
EXPLAIN With average_total_amount_paid_CTE (customer_id,first_name,
last_name,city,county,payment)AS
(SELECT A.customer_id,A.first_name,
A.last_name,
D.city,
E.country,
SUM (B.amount) AS payment
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.address_id = C.address_id)
```

Data output Messages Notifications

QUERY PLAN

text

1 Aggregate (cost=61.33..61.34 rows=1 width=32)

2 -> Limit (cost=61.26..61.27 rows=5 width=67)

3 -> Sort (cost=61.26..61.80 rows=219 width=67)

4 Sort Key: (sum(b.amount)) DESC

5 -> HashAggregate (cost=54.88..57.62 rows=219 width=67)

6 Group Key: a.customer_id, d.city, e.country

Total rows: 22 of 22 Query complete 00:00:02.957

Ln 2, Col 9

pgAdmin 4

File Object Tools Help

Rockbuster/postgres@PostgreSQL 14

Query Query History

```
--Task 3.9 rewrite Q1 3.8 as CTE
With average_total_amount_paid_CTE (customer_id,first_name,
last_name,city,county,payment)AS
(SELECT A.customer_id,A.first_name,
A.last_name,
D.city,
E.country,
SUM (B.amount) AS payment
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.address_id = C.address_id)
```

Data output Messages Notifications

average_totalAmountPaid

numeric

1 106.750000000000000000000000000000

Execute/Refresh (F5)

Successfully run. Total query runtime: 173 msec. 1 rows affected.

Total rows: 1 of 1 Query complete 00:00:00.173

Ln 2, Col 1

QUERY-2 3.8 Subquery / REWRITTEN AS CTE

	Subquery	CTE
--	----------	-----

Cost	"Limit (cost=1177.64..1177.69 rows=5 width=84)"	"Limit (cost=1186.57..1186.62 rows=5 width=25)"
Time	Total query runtime 250 msec. 5 rows affected.	Total query runtime 1 secs. 38 msec. 5 rows affected.

The image displays two screenshots of the pgAdmin 4 web interface, showing a PostgreSQL query and its execution results.

Top Screenshot: The query editor shows a query labeled "Task 3.8 Q2 Subquery". The query is as follows:

```
--Task 3.8 Q2 Subquery
1 EXPLAIN SELECT DISTINCT (A.country),
2 COUNT(DISTINCT D.customer_id) AS all_customer_count,
3 COUNT(DISTINCT A.country) AS top_customer_count
4 FROM country A
5 INNER JOIN city B ON A.country_id = B.country_id
6 INNER JOIN address C ON B.city_id = C.city_id
7 INNER JOIN customer D ON C.address_id = D.address_id
8 LEFT JOIN (SELECT A.customer_id,A.first_name,A.last_name,
9 D.city,E.country,
10
```

The query plan is displayed below the query editor, showing the execution plan for the query. The plan includes a "Limit" step, followed by a "Sort" step, and a "GroupAggregate" step. The total rows affected are 47 of 47.

Bottom Screenshot: The query editor shows the same query, but the "Execute/Refresh" button is highlighted. The query results are displayed below the query editor, showing the execution plan and the data output.

The data output shows the following results:

country	all_customer_count	top_customer_count
India	60	1
China	53	1
United States	36	1
Japan	31	1
Mexico	30	1

The query results show 5 rows of data. The total query runtime is 250 msec. and 5 rows were affected.

The top screenshot shows the pgAdmin 4 interface with a SQL query being executed. The query is a complex join involving payment, customer, address, city, and country tables, grouped by customer and country to find the top 5 customers by total payment amount. The query plan is visible below the SQL editor.

```

--Task3.9 Q2 3.8 as CTE
2 EXPLAIN With top_5_customers_CTE (amount,customer_id,first_name,last_name,city,country,payment)AS
3 (SELECT A.amount,B.customer_id,B.first_name,B.last_name,
4      D.city,E.country,
5      SUM(A.amount) AS payment
6 FROM payment A
7 INNER JOIN customer B ON B.customer_id = A.customer_id
8 INNER JOIN address C ON C.address_id = B.address_id
9 INNER JOIN city D ON D.city_id = C.city_id
10 INNER JOIN country E ON E.country_id = D.country_id
11 GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country, A.amount

```

The bottom screenshot shows the same query executed successfully, displaying the results in a table. The table has 5 rows and 4 columns: country, all_customer_count, top_customer_count, and another top_customer_count.

country	all_customer_count	top_customer_count	top_customer_count
India	60	1	1
China	53	1	1
United States	36	1	1
Japan	31	1	1
Mexico	30	1	1

A green status bar at the bottom indicates: "Successfully run. Total query runtime: 1 secs 38 msec. 5 rows affected."

Step 3:

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

With creation of the subqueries the process was relatively simple with writing inner join queries given the examples, and then understanding the table relationships. In rewriting the CTE's it was more difficult with the second task with having to create two CTE's to join the second inner query, was not as straight.

Step 4:

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