

Data Analytics Immersion

3.9: Common Table Expressions

Mehreen Becker

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.

```

SELECT AVG(total_amount_paid) AS average_amount_paid
FROM (
    SELECT
        SUM(p.amount) AS total_amount_paid
    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 payment P ON A.customer_id = P.customer_id
    WHERE C.city IN (
        SELECT C.city
        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
        WHERE D.country IN (
            SELECT
                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
            GROUP BY D.country
        )
        GROUP BY C.city
    )
    GROUP BY A.customer_id
    ORDER BY COUNT (A.customer_id) DESC
    LIMIT 10
)
GROUP BY D.country, C.city
ORDER BY total_amount_paid DESC
LIMIT 5) AS average_amount_paid
  
```

Total rows: 1 Query complete 00:00:00.098 LF Ln 13, Col 54

```

SELECT L1.c.city
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
WHERE D.country IN (
    SELECT
        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
    GROUP BY D.country
    ORDER BY COUNT (A.customer_id) DESC
    LIMIT 10
)
GROUP BY D.country, C.city
ORDER BY COUNT (A.customer_id) DESC
LIMIT 1
)
GROUP BY A.customer_id, A.first_name, A.last_name, D.country, C.city
ORDER BY total_amount_paid DESC
LIMIT 5) AS average_amount_paid
  
```

Total rows: 1 Query complete 00:00:00.098 LF Ln 13, Col 54

pgAdmin 4 Object Tools Edit View Window Help pgAdmin 4 Fri 24 Oct 18:57

Servers (1) My Server Databases (5) Actor Rockbuster Casts Catalogs (2) Event Triggers Extensions (1) Foreign Data W Languages (1) Publications Schemas (1) public Subscriptions Testing mehreenbecker postgres Login/Group Roles Tablespaces

```

WITH top_countries AS (
  SELECT
    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
  GROUP BY D.country
  ORDER BY COUNT(A.customer_id) DESC
  LIMIT 10
),
top_cities AS (
  SELECT
    D.country,
    C.city
  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
  WHERE D.country IN (SELECT country FROM top_countries)
  GROUP BY D.country, C.city
  ORDER BY COUNT(A.customer_id) DESC
)
TMFT 1a

```

Data Output Messages Explain Notifications

average_amount_paid	numeric
1	105.55400000000000

Total rows: 1 Query complete 00:00:00.073 LF Ln 11, Col 11

pgAdmin 4 Object Tools Edit View Window Help pgAdmin 4 Fri 24 Oct 18:58

Servers (1) My Server Databases (5) Actor Rockbuster Casts Catalogs (2) Event Triggers Extensions (1) Foreign Data W Languages (1) Publications Schemas (1) public Subscriptions Testing mehreenbecker postgres Login/Group Roles Tablespaces

```

SELECT BY CUUN(A.customer_id) DESC
LIMIT 10
),
customer_totals AS (
  SELECT
    A.customer_id,
    A.first_name,
    A.last_name,
    D.country,
    C.city,
    SUM(P.amount) AS total_amount_paid
  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 payment P ON A.customer_id = P.customer_id
  WHERE C.city IN (SELECT city FROM top_cities)
  GROUP BY A.customer_id, A.first_name, A.last_name, D.country, C.city
  ORDER BY total_amount_paid DESC
  LIMIT 5
)
SELECT AVG(total_amount_paid) AS average_amount_paid
FROM customer_totals;

```

Data Output Messages Explain Notifications

average_amount_paid	numeric
1	105.55400000000000

Total rows: 1 Query complete 00:00:00.073 LF Ln 17, Col 11

pgAdmin 4 Object Tools Edit View Window Help pgAdmin 4 Fri 24 Oct 22:26

Dependents X Processes X Testing/postgres... X Testing/postgres... X postgres/postgres... X Preferences X Rockbuster/postgres@My Server* X :

Servers (1) My Server Databases (5) Actor Casts Catalogs (2) Event Triggers Extensions (1) Foreign Data W Languages (1) Publications Schemas (1) public Subscriptions Testing mehreenbecker postgres Login/Group Roles Tablespaces

Query History

```

1   SELECT
2       d.country,
3       COUNT(DISTINCT a.customer_id) AS all_customer_count,
4       COUNT(DISTINCT CASE
5           WHEN (
6               SELECT SUM (p2.amount)
7                   FROM payment p2
8                   WHERE p2.customer_id = a.customer_id
9           ) > (
10               SELECT AVG(total_sum)
11                   FROM (
12                       SELECT SUM(p.amount) AS total_sum
13                           FROM payment p
14                           GROUP BY p.customer_id
15                   ) AS customer_totals
16           )
17           THEN a.customer_id
18           ELSE NULL
19       END) AS top_customer_count
20   FROM customer a
21   JOIN address b ON a.address_id = b.address_id
22   JOIN city c ON b.city_id = c.city_id
23   JOIN country d ON c.country_id = d.country_id
24   GROUP BY d.country
25   ORDER BY top_customer_count DESC
26   LIMIT 10;

```

Data Output Messages Explain Notifications

country	all_customer_count	top_customer_count
character varying (50)	bigint	bigint
1 India	60	26
2 China	53	25
3 United States	36	16
4 Japan	31	14
5 Russian Federation	28	13
6 Brazil	28	12
7 Mexico	30	11
8 Philippines	20	11
9 Taiwan	10	7
10 Turkey	15	7

Total rows: 10 Query complete 00:00:00.300 LF Ln 2, Col 12

pgAdmin 4 Object Tools Edit View Window Help pgAdmin 4 Fri 24 Oct 22:26

Dependents X Processes X Testing/postgres... X Testing/postgres... X postgres/postgres... X Preferences X Rockbuster/postgres@My Server* X :

Servers (1) My Server Databases (5) Actor Casts Catalogs (2) Event Triggers Extensions (1) Foreign Data W Languages (1) Publications Schemas (1) public Subscriptions Testing mehreenbecker postgres Login/Group Roles Tablespaces

Query History

```

1   SELECT
2       GROUP BY p.customer_id
3       ) AS customer_totals
4   )
5   THEN a.customer_id
6   ELSE NULL
7   END) AS top_customer_count
8   FROM customer a
9   JOIN address b ON a.address_id = b.address_id
10  JOIN city c ON b.city_id = c.city_id
11  JOIN country d ON c.country_id = d.country_id
12  GROUP BY d.country
13  ORDER BY top_customer_count DESC
14  LIMIT 10;

```

Data Output Messages Explain Notifications

country	all_customer_count	top_customer_count
character varying (50)	bigint	bigint
1 India	60	26
2 China	53	25
3 United States	36	16
4 Japan	31	14
5 Russian Federation	28	13
6 Brazil	28	12
7 Mexico	30	11
8 Philippines	20	11
9 Taiwan	10	7
10 Turkey	15	7

Total rows: 10 Query complete 00:00:00.300 LF Ln 2, Col 12

pgAdmin 4 Object Tools Edit View Window Help Fri 24 Oct 22:28

Dependents Processes Testing/postgres... Testing/postgres... postgres/postgres... Preferences Rockbuster/postgres@My Server*

Servers (1) My Server Databases (5) Actor Casts Catalogs (2) Event Triggers Extensions (1) Foreign Data W Languages (1) Publications Schemas (1) public Subscriptions Testing mehrennebecker postgres Login/Group Roles Tablespace

```

WITH customer_totals AS (
    SELECT
        p.customer_id,
        SUM(p.amount) AS total_sum
    FROM payment p
    GROUP BY p.customer_id
),
average_total AS (
    SELECT
        AVG(total_sum) AS avg_total
    FROM customer_totals
),
customer_with_country AS (
    SELECT
        a.customer_id,
        d.country
    FROM customer a
)
SELECT
    cw.country,
    COUNT(DISTINCT cw.customer_id) AS all_customer_count,
    COUNT(DISTINCT CASE
        WHEN ct.total_sum > at.avg_total THEN cw.customer_id
        ELSE NULL
    END) AS top_customer_count
FROM customer_with_country cw
JOIN customer_totals ct ON cw.customer_id = ct.customer_id
CROSS JOIN average_total at
GROUP BY cw.country
ORDER BY top_customer_count DESC
LIMIT 10;

```

Data Output Messages Explain Notifications Showing rows: 1 to 10 Page No: 1 of 1 LF Ln 34, Col 10

	country	all_customer_count	top_customer_count
1	India	60	26
2	China	53	25
3	United States	36	16
4	Japan	31	14
5	Russian Federation	28	13
6	Brazil	28	12

Total rows: 10 Query complete 00:00:00.098

pgAdmin 4 Object Tools Edit View Window Help Fri 24 Oct 22:28

Dependents Processes Testing/postgres... Testing/postgres... postgres/postgres... Preferences Rockbuster/postgres@My Server*

Servers (1) My Server Databases (5) Actor Casts Catalogs (2) Event Triggers Extensions (1) Foreign Data W Languages (1) Publications Schemas (1) public Subscriptions Testing mehrennebecker postgres Login/Group Roles Tablespace

```

JOIN address b ON a.address_id = b.address_id
JOIN city c ON b.city_id = c.city_id
JOIN country d ON c.country_id = d.country_id
)
SELECT
    cw.country,
    COUNT(DISTINCT cw.customer_id) AS all_customer_count,
    COUNT(DISTINCT CASE
        WHEN ct.total_sum > at.avg_total THEN cw.customer_id
        ELSE NULL
    END) AS top_customer_count
FROM customer_with_country cw
JOIN customer_totals ct ON cw.customer_id = ct.customer_id
CROSS JOIN average_total at
GROUP BY cw.country
ORDER BY top_customer_count DESC
LIMIT 10;

```

Data Output Messages Explain Notifications Showing rows: 1 to 10 Page No: 1 of 1 LF Ln 34, Col 10

	country	all_customer_count	top_customer_count
1	India	60	26
2	China	53	25
3	United States	36	16
4	Japan	31	14
5	Russian Federation	28	13
6	Brazil	28	12

Total rows: 10 Query complete 00:00:00.098

I first identified the top countries with the most customers by joining the customer, address, city and country tables and grouping them by country. Then, I narrowed the focus to the top 10 cities located within those countries, again by counting customers per city and ordering by that count. After that, I calculated information about the top 5 customers who paid the most using the payment table joined to the geographic data, and limited it to the customers from the top city. Finally I summarized the results in one query by finding the average of the top 5 total payments, and in the other by comparing counts of all customers versus top customers per country.

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

1. **Which approach do you think will perform better and why?** I'm sure that both CTEs and subqueries produce the same, if not a similar, result. With my limited experience with CTEs, I feel like subqueries allow for better optimization potentially and I also felt like I had to scan fewer rows.
2. **Compare the costs of all the queries by creating query plans for each one.**

Difference in cost:

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** Object Explorer - Servers (1) - My Server - Databases (5) - Rockbuster.
- Query:**

```

13   FROM customer A
14     INNER JOIN address B ON A.address_id = B.address_id
15     INNER JOIN city C ON B.city_id = C.city_id
16     INNER JOIN country D ON C.country_id = D.country_id
17   WHERE D.country IN (
18     SELECT
19       D.country
20         FROM customer A
21           INNER JOIN address B ON A.address_id = B.address_id
22             INNER JOIN city C ON B.city_id = C.city_id
23               INNER JOIN country D ON C.country_id = D.country_id
24                 GROUP BY D.country
25                   ORDER BY COUNT(a.customer_id) DESC
26                     LIMIT 10
27   )
28   GROUP BY D.country, C.city
  
```
- Analysis Tab:** Shows the execution plan with 5 nodes:

#	Node	Rows	Plan
1.	Aggregate (cost=166.06..166.07 rows=1 width=32)	1	
2.	Limit (cost=165.98..165.99 rows=5 width=67)	5	
3.	Sort (cost=165.98..166.59 rows=244 width=67)	244	
4.	Aggregate (cost=158.87..161.92 rows=244 width=67)	244	
5.	Nested Loop Inner Join (cost=117.48..156.43 rows=244 width=41)	244	
- Statistics Tab:** Total rows: 1 Query complete 00:00:00.341

Figure 1: Cost (CTE)

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** Object Explorer - Servers (1) - My Server - Databases (5) - Rockbuster.
- Query:**

```

7   E.country,
8     SUM(A.amount) AS total_amount_paid
9   FROM payment A
10  INNER JOIN customer B ON A.customer_id = B.customer_id
11  INNER JOIN address C ON B.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
15    ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki',
16    'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
17    AND E.country IN
18    ('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil',
19    'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
20  GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
21  ORDER BY total_amount_paid DESC
22  LIMIT 5) AS average;
  
```
- Analysis Tab:** Shows the execution plan with 5 nodes:

#	Node	Rows	Plan
1.	Aggregate (cost=24.64..24.65 rows=1 width=32)	1	
2.	Limit (cost=24.56..24.57 rows=5 width=67)	5	
3.	Sort (cost=24.56..24.62 rows=22 width=67)	22	
4.	Aggregate (cost=23.92..24.2 rows=22 width=67)	22	
5.	Nested Loop Inner Join (cost=3.65..23.7 rows=22 width=41)	22	
- Statistics Tab:** Total rows: 1 Query complete 00:00:00.063

Figure 2: Cost (Subquery)

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.

Difference in query run time:

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is as follows:

```

13    FROM customer A
14      INNER JOIN address B ON A.address_id = B.address_id
15      INNER JOIN city C ON B.city_id = C.city_id
16      INNER JOIN country D ON C.country_id = D.country_id
17      WHERE D.country IN (
18          SELECT
19              D.country
20          FROM customer A
21          INNER JOIN address B ON A.address_id = B.address_id
22          INNER JOIN city C ON B.city_id = C.city_id
23          INNER JOIN country D ON C.country_id = D.country_id
24          GROUP BY D.country
25          ORDER BY COUNT(a.customer_id) DESC
26          LIMIT 10
27      )
28      GROUP BY D.country, C.city

```

Below the query, the status bar indicates "Total rows: 1" and "Query complete 00:00:00.341".

Figure 1: Total Query Runtime: 341 msec (CTE)

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is as follows:

```

7      E.country,
8      SUM(A.amount) AS total_amount_paid
9      FROM payment A
10     INNER JOIN customer B ON A.customer_id = B.customer_id
11     INNER JOIN address C ON B.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
15         ('Aurora', 'Atlixco', 'Xintai', 'Addoni', 'Dhule (Dhulia)', 'Kurashiki',
16         'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
17     AND E.country IN
18         ('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil',
19         'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
20     GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
21     ORDER BY total_amount_paid DESC
22     LIMIT 5) AS average;

```

Below the query, the status bar indicates "Total rows: 1" and "Query complete 00:00:00.032".

Figure 2: Total Query Runtime: 32 msec (Subquery)

4. **Did the results surprise you?** Write a few sentences to explain your answer. To be honest, no. I did think that subqueries would run faster (not that I noticed the few milisecond difference) but more because they feel more straightforward and less complex.

Step 3: Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs. Since we were asked to use CTEs and not subqueries, one of the main challenges for me was to make sure that everything made sense. I did find myself feeling a little lost and confused since it's easier with CTEs to misplace filters and to also join things together. I

found myself scanning rows multiple times to ensure that the string made sense and that everything was connected with each other and connected in a way that it should be.