Andrew Smith

Career Foundry Data Immersion

January 19, 2022

3.9: Common Table Expressions (Answers 3.9)

1a/b)Step 1 from Task 3.8 as CTE

WITH average\_amount\_cte AS (SELECT A.customer\_id, A.first\_name, A.last\_name, C.city, D.country, SUM(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 E ON A.customer\_id =E.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

ORDER BY COUNT(customer\_id) DESC

LIMIT 10)

GROUP BY D.country, C.city

ORDER BY COUNT(customer\_id) DESC

LIMIT 10)

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

ORDER BY sum(amount) DESC

LIMIT 5)

SELECT AVG (total\_amount\_paid) AS average

FROM average\_amount\_cte

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

1a/b)Step 2 from Task 3.8 as CTE

WITH top\_customer\_cte AS (SELECT A.customer\_id, A.first\_name, A.last\_name, C.city, D.country, SUM(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 E ON A.customer\_id = E.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

ORDER BY COUNT(customer\_id) DESC

LIMIT 10)

GROUP BY D.country, C.city

ORDER BY COUNT(customer\_id) DESC

LIMIT 10)

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

ORDER BY sum(amount) DESC

LIMIT 5),

all\_customer\_cte AS

(SELECT D.country,

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

COUNT(DISTINCT D.country) AS top\_customer\_count

FROM country D

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

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

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

GROUP BY D.country)

SELECT D.country, COUNT(DISTINCT A.customer\_id) AS

all\_customer\_count,

COUNT(DISTINCT top\_customer\_cte.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\_customer\_cte ON

D.country=top\_customer\_cte.country

GROUP BY D.country

ORDER BY all\_customer\_count DESC

LIMIT 5

Graphical user interface

Description automatically generated

Table

Description automatically generated

1c)

For each step I started by removing each outer query and replaced the beginning with a WITH….AS command. I named the WITH commands similar to the column names found in the results but with ‘\_cte’ at the end. After, I kept the inner queries the same from task 3.8. In step 1, I ended the query by adding a SELECT statement similar to the beginning of the step 1 outer query in the previous task(ex. “SELECT AVG(total\_amount\_paid) AS average FROM”…) but followed by the created cte name at the beginning of the new query (“FROM average\_amount\_cte”). Step 2 was similar to step 1 regarding the first few steps, but I had to name a whole new section since multiple variables were in play. I added the same beginning outer query from step 2 of the previous task followed by a portion of the inner query since this counted all customers rather than top customers. I ended with a LEFT JOIN to combine tables to show the top customer count per country.

2a)

I am unsure of which queries will perform better but I feel like the results will be close since the commands are similar.

2b/c)

**\*Subquery Step 1**

Graphical user interface, application, Word

Description automatically generated

-Runtime: 112 msec

-62 rows affected

-(cost=166.06..166.07 rows=1 width=32)

**\*CTE Step 1**

Graphical user interface, application, Word

Description automatically generated

-Runtime:156 msec

-62 rows affected

-(cost=166.06..166.07 rows=1 width=32)

**\*Subquery Step 2**

Graphical user interface, table

Description automatically generated

-Runtime:112 msec

-85 rows affected

-(cost=281.55..281.56 rows=5 width=84)

**\*CTE Step 2**

Graphical user interface, table

Description automatically generated

-Runtime:119 msec

-86 rows affected

-(cost=267.83..267.84 rows=5 width=25)

2d)

For step 1, the subquery had a faster runtime but both queries resulted in the same cost. In step 2, the subquery had a faster runtime but costed more than the CTE query. In short, yes the results surprised me since step 2 had a faster subquery that costed more and a slower CTE that costed less. I did notice that if you run the same query a few times that the runtime will change drastically which can result in random outcomes.

3)

Changing the step 1 subquery into a CTE was simple since there were only a few minor adjustments to the outer query. However, step 2 was a little bit challenging since more variables were in play and I had to experiment with different commands to make it work. Minor things like not knowing you can’t use a second WITH command on the second half of the query and only using a smaller section of the inner query followed by a LEFT JOIN made it confusing to end the CTE.