Career Foundry Answers 3.9

Gabriel Pollicar

November 19, 2023

Step 1.

Question 1 from 3.8

```
-- CTE That Lists Total Payments from Top 5 Customers
WITH Top5_payments_cte ( country, city, amount) AS
(SELECT A.customer_id,
                        d.country,
                        c.city,
                        SUM(amount) AS Top5_Payments
        FROM customer A
        INNER JOIN payment E ON A.customer_id = E.customer_id
        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 c.city IN(SELECT c.city
                -- Finds Top 5 Cities in Top 5 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
                WHERE country IN (SELECT D.country
                        -- Finds Top 5 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(A.customer_id) DESC
                LIMIT 10)
        GROUP By A.customer_id, d.country, c.city
        ORDER BY SUM(amount) DESC
         LIMIT 5)

-- Finds Average of Total Payments from Top 5 Customers
SELECT AVG(Top5_Payments)
FROM Top5_payments_cte
```

```
;
```

```sql
-- CTE That Lists Total Payments from Top 5 Customers
WITH Top5_payments_cte ( country, city, amount) AS
(SELECT A.customer_id,
            d.country,
            c.city,
            SUM(amount) AS Top5_Payments
    FROM customer A
    INNER JOIN payment E ON A.customer_id = E.customer_id
    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 c.city IN(SELECT c.city
        -- Finds Top 5 Cities in Top 5 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
        WHERE country IN (SELECT D.country
            -- Finds Top 5 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(A.customer_id) DESC
        LIMIT 10)
    GROUP By A.customer_id, d.country, c.city
    ORDER BY SUM(amount) DESC
     LIMIT 5)

-- Finds Average of Total Payments from Top 5 Customers
SELECT AVG(Top5_Payments)
FROM Top5_payments_cte
;
```

Data Output | Messages | Notifications

	avg numeric
1	105.5540000000000000

Question 2 From 3.8

```
-- CTE That Lists the Top 5 Payements of the Top 5 Customers
WITH Top5_CustomerPayments_CTE (customer_id, amount) AS(
	SELECT A.customer_id AS customer_id,
			SUM(amount) AS Top5_Payments
	FROM customer A
	INNER JOIN payment E ON A.customer_id = E.customer_id
	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 c.city IN(SELECT c.city
		-- Finds Top 5 Cities in Top 5 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
		WHERE country IN (SELECT D.country
			-- Finds Top 5 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(A.customer_id) DESC
		LIMIT 10)
	GROUP By A.customer_id, A.first_name, A.last_name, d.country, c.city
	ORDER BY SUM(amount) DESC
	LIMIT 5)

SELECT  d.country AS Country,
		COUNT(DISTINCT A.customer_id) AS all_customer_count,
		COUNT(Top5_CustomerPayments_CTE.customer_id) AS top_5_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 Top5_CustomerPayments_CTE ON A.customer_id =
Top5_CustomerPayments_CTE.customer_id
GROUP BY d.country
ORDER BY top_5_customer_count DESC
```

```sql
-- CTE That Lists the Top 5 Payements of the Top 5 Customers
WITH Top5_CustomerPayments_CTE (customer_id, amount) AS(
    SELECT A.customer_id AS customer_id,
           SUM(amount) AS Top5_Payments
    FROM customer A
    INNER JOIN payment E ON A.customer_id = E.customer_id
    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 c.city IN(SELECT c.city
        -- Finds Top 5 Cities in Top 5 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
        WHERE country IN (SELECT D.country
            -- Finds Top 5 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(A.customer_id) DESC
        LIMIT 10)
    GROUP By A.customer_id, A.first_name, A.last_name, d.country, c.city
    ORDER BY SUM(amount) DESC
    LIMIT 5)

SELECT  d.country AS Country,
        COUNT(DISTINCT A.customer_id) AS all_customer_count,
        COUNT(Top5_CustomerPayments_CTE.customer_id) AS top_5_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 Top5_CustomerPayments_CTE ON A.customer_id = Top5_CustomerPayments_
GROUP BY d.country
ORDER BY top_5_customer_count DESC
```

Data Output | Messages | Notifications

	country character varying (50)	all_customer_count bigint	top_5_customer_count bigint
1	Mexico	30	1
2	India	60	1
3	China	53	1
4	United States	36	1
5	Japan	31	1
6	Argentina	13	0
7	Armenia	1	0
8	Austria	3	0
9	Azerbaijan	2	0

Step 1C.)

I first wrote the WITH statement, using the Select outputs from the subquery to fill the column names. I then took the main subquery which includes the inner subqueries as well, and placed them into the CTE statement. In the main Query, I replaced the subquery with the name of the CTE and ran the codes. For part 2 there was an issue with the syntax of the LEFT JOIN statement, which I figured required me to remove the parenthesis around the CTE name in the main QUERY. Overall, the process was simple. However, I think there are more I could do to make the entire code simpler and more readable such as removing repeating lines of code.

Step 2.)

Question 1. I think that the CTE approach will perform better than the multiple Subqueries. This is because the CTE approach creates a table that can be used in the main Query versus running a subquery in the main Query which can run multiple times.

Question 2.

Total Cost for Question 1:

- **Using Subquery**

  "Aggregate (cost=166.06..166.07 rows=1 width=32)"

  "Successfully run. Total query runtime: 105 msec. 1 rows affected."

- **Using CTE**

  "Aggregate (cost=166.06..166.07 rows=1 width=32)"

  "Successfully run. Total query runtime: 87 msec. 1 rows affected."

Total Cost for Question 2:

- **Using Subquery**

  "Sort (cost=270.11..270.38 rows=109 width=25)"

  "Successfully run. Total query runtime: 86 msec. 108 rows affected."

- **Using CTE**

  "Sort (cost=270.11..270.38 rows=109 width=25)"

  "Successfully run. Total query runtime: 62 msec. 108 rows affected."

Question 4. Based on the outputs from the Explain Function and the message popup detailing the runtime, using the total cost on the system is the same for both, but runtime is significantly lower when using CTE than with subqueries. I've also found that the runtime changes with each Code Execute. However, despite differing runtimes, the CTE for both queries performs faster than using subqueries. This could be because a CTE is run once and is held as a table in memory, Compared to a subquery which may need to run multiple times during a query.

Step 3

Converting the Subqueries into CTE was simple because I had previously labeled all subqueries with comments detailing their purpose and what they output. To create the CTE, I moved the lines of code from the original main query and moved them into the WHERE statement of the CTE. The subquery in the main Query was changed to the title of the CTE. The code ran without much syntax change.

Problems that I faced were mostly syntax issues when replacing the subquery with CTE statements. For example, the LEFT JOIN statement for Query 2 required some syntax change, the parentheses surrounding the subquery had to be removed when replacing with the CTE name. Another syntax issue was changing the table references for identifying the relationship between the customer Table and the CTE table (this line: “A.customer_id = Top5_CustomerPayments_CTE.customer_id”). Overall, getting the code to run the correct output was doable.