William Su

Task 3.9 – Common Table Expressions

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

Step 1

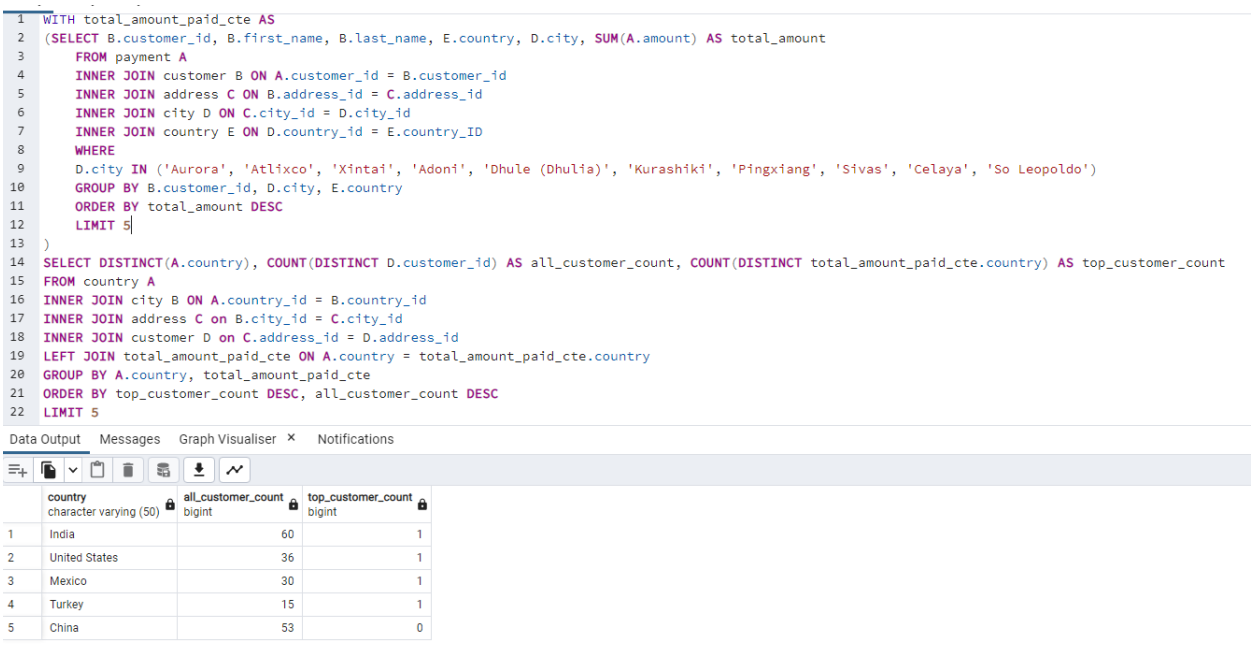
Query | Query History

```
WITH total_amount_paid_cte (customer_id, first_name, last_name, country, city, total_amount) AS
    (SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city, SUM(A.amount) AS total_amount
        FROM payment A
        INNER JOIN customer B ON A.customer_id = B.customer_id
        INNER JOIN address C ON B.address_id = C.address_id
        INNER JOIN city D ON C.city_id = D.city_id
        INNER JOIN country E ON D.country_id = E.country_ID
        WHERE
        D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
        GROUP BY B.customer_id, D.city, E.country
        ORDER BY total_amount DESC
        LIMIT 5)
SELECT AVG (total_amount) AS average
FROM total_amount_paid_cte
```

Data Output | Messages | Notifications

	average numeric
1	107.3540000000000000

Step 2



```
WITH total_amount_paid_cte AS
(SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city, SUM(A.amount) AS total_amount
    FROM payment A
    INNER JOIN customer B ON A.customer_id = B.customer_id
    INNER JOIN address C ON B.address_id = C.address_id
    INNER JOIN city D ON C.city_id = D.city_id
    INNER JOIN country E ON D.country_id = E.country_ID
    WHERE
    D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
    GROUP BY B.customer_id, D.city, E.country
    ORDER BY total_amount DESC
    LIMIT 5
)
SELECT DISTINCT(A.country), COUNT(DISTINCT D.customer_id) AS all_customer_count, COUNT(DISTINCT total_amount_paid_cte.country) AS top_customer_count
FROM country A
INNER JOIN city B ON A.country_id = B.country_id
INNER JOIN address C on B.city_id = C.city_id
INNER JOIN customer D on C.address_id = D.address_id
LEFT JOIN total_amount_paid_cte ON A.country = total_amount_paid_cte.country
GROUP BY A.country, total_amount_paid_cte
ORDER BY top_customer_count DESC, all_customer_count DESC
LIMIT 5
```

Data Output | Messages | Graph Visualiser | Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	1
2	United States	36	1
3	Mexico	30	1
4	Turkey	15	1
5	China	53	0

From my understand, CTEs and Subqueries are fairly interchangeable. For the first step, I simply moved the subquery to the top and gave it the CTE syntax. For the second step, I did the same, choosing to move the subquery for finding the total amount of top customers to the top of the query and giving it the CTE syntax. I debated trying to write another CTE for the inner joins left in the query, but decided against it.

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

I'm not too sure which one would perform better, but I think I may use CTEs over subqueries in the long run due to CTEs being easier to read and only having to be defined once. I would imagine that in the first query, the subquery is faster, but in the second query, the CTE would be faster. This is because the first query is much simpler, and only asks for one thing, while the second query requires some more work and comparisons to be made.

Step 1

	Subqueries	CTE
Cost	cost=64.45..64.46 rows=1 width=32	64.45..64.46, rows=1, width=32
Speed	Total runtime = 49ms	Total runtime = 59ms

Subqueries

	QUERY PLAN text
1	Aggregate (cost=64.45..64.46 rows=1 width=32)
2	-> Limit (cost=64.37..64.39 rows=5 width=270)
3	-> Sort (cost=64.37..64.98 rows=243 width=270)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=270)
6	Group Key: b.customer_id, d.city, e.country
7	-> Nested Loop (cost=18.16..54.87 rows=243 width=28)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=22)
9	Hash Cond: (d.country_id = e.country_id)
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=15)
11	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
12	Hash Cond: (c.city_id = d.city_id)
13	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
14	-> Hash (cost=14.03..14.03 rows=10 width=15)
15	-> Seq Scan on city d (cost=0.03..14.03 rows=10 width=15)
16	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulia)",Kurashiki,Pingxiang,Sivas,Celaya,"So Leopoldo"}'::text[]))
17	-> Index Scan using idx_fk_address_id on customer b (cost=0.28..0.38 rows=1 width=6)
18	Index Cond: (address_id = c.address_id)
19	-> Hash (cost=2.09..2.09 rows=109 width=13)
20	-> Seq Scan on country e (cost=0.00..2.09 rows=109 width=13)
21	-> Index Scan using idx_fk_customer_id on payment a (cost=0.29..1.53 rows=24 width=8)
22	Index Cond: (customer_id = b.customer_id)

Total rows: 22 of 22 | Query complete 00:00:00.049

CTE

Data Output | Messages | Graph Visualiser | Notifications

	QUERY PLAN text
1	Aggregate (cost=64.45..64.46 rows=1 width=32)
2	-> Limit (cost=64.37..64.39 rows=5 width=270)
3	-> Sort (cost=64.37..64.98 rows=243 width=270)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=270)
6	Group Key: b.customer_id, d.city, e.country
7	-> Nested Loop (cost=18.16..54.87 rows=243 width=28)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=22)
9	Hash Cond: (d.country_id = e.country_id)
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=15)
11	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
12	Hash Cond: (c.city_id = d.city_id)
13	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
14	-> Hash (cost=14.03..14.03 rows=10 width=15)
15	-> Seq Scan on city d (cost=0.03..14.03 rows=10 width=15)
16	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulia)",Kurashiki,Pingxiang,Sivas,Celaya,"So Leopoldo"}'::text[]))
17	-> Index Scan using idx_fk_address_id on customer b (cost=0.28..0.38 rows=1 width=6)
18	Index Cond: (address_id = c.address_id)
19	-> Hash (cost=2.09..2.09 rows=109 width=13)
20	-> Seq Scan on country e (cost=0.00..2.09 rows=109 width=13)
21	-> Index Scan using idx_fk_customer_id on payment a (cost=0.29..1.53 rows=24 width=8)
22	Index Cond: (customer_id = b.customer_id)

Total rows: 22 of 22 | Query complete 00:00:00.059

Step 2

	Subqueries	CTE
Cost	cost=189.48..189.49 rows=5 width=84	189.48...189.49, rows=5 width=84
Speed	Total runtime = 61ms	Total runtime = 49ms

Subqueries

	QUERY PLAN text
1	Limit (cost=189.48..189.49 rows=5 width=84)
2	-> Sort (cost=189.48..190.84 rows=545 width=84)
3	Sort Key: (count(DISTINCT d.customer_id)) DESC
4	-> HashAggregate (cost=174.98..180.43 rows=545 width=84)
5	Group Key: count(DISTINCT d.customer_id), a.country, count(DISTINCT a.country)
6	-> GroupAggregate (cost=157.95..170.89 rows=545 width=84)
7	Group Key: a.country, total_amount_paid.*
8	-> Sort (cost=157.95..159.45 rows=599 width=72)
9	Sort Key: a.country, total_amount_paid.*
10	-> Hash Left Join (cost=108.02..130.32 rows=599 width=72)
11	Hash Cond: ((a.country)::text = (total_amount_paid.country)::text)
12	-> Hash Join (cost=43.52..63.30 rows=599 width=13)
13	Hash Cond: (b.country_id = a.country_id)
14	-> Hash Join (cost=40.07..58.22 rows=599 width=6)
15	Hash Cond: (c.city_id = b.city_id)
16	-> Hash Join (cost=21.57..38.14 rows=599 width=6)
17	Hash Cond: (d.address_id = c.address_id)
18	-> Seq Scan on customer d (cost=0.00..14.99 rows=599 width=6)
19	-> Hash (cost=14.03..14.03 rows=603 width=6)
20	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
21	-> Hash (cost=11.00..11.00 rows=600 width=6)
22	-> Seq Scan on city b (cost=0.00..11.00 rows=600 width=6)
Total rows: 47 of 47	Query complete 00:00:00.061	

CTE

	QUERY PLAN text
1	Limit (cost=189.48..189.49 rows=5 width=84)
2	-> Sort (cost=189.48..190.84 rows=545 width=84)
3	Sort Key: (count(DISTINCT total_amount_paid_cte.country)) DESC, (count(DISTINCT d.customer_id)) DESC
4	-> HashAggregate (cost=174.98..180.43 rows=545 width=84)
5	Group Key: count(DISTINCT total_amount_paid_cte.country), count(DISTINCT d.customer_id), a.country
6	-> GroupAggregate (cost=157.95..170.89 rows=545 width=84)
7	Group Key: a.country, total_amount_paid_cte.*
8	-> Sort (cost=157.95..159.45 rows=599 width=81)
9	Sort Key: a.country, total_amount_paid_cte.*
10	-> Hash Left Join (cost=108.02..130.32 rows=599 width=81)
11	Hash Cond: ((a.country)::text = (total_amount_paid_cte.country)::text)
12	-> Hash Join (cost=43.52..63.30 rows=599 width=13)
13	Hash Cond: (b.country_id = a.country_id)
14	-> Hash Join (cost=40.07..58.22 rows=599 width=6)
15	Hash Cond: (c.city_id = b.city_id)
16	-> Hash Join (cost=21.57..38.14 rows=599 width=6)
17	Hash Cond: (d.address_id = c.address_id)
18	-> Seq Scan on customer d (cost=0.00..14.99 rows=599 width=6)
19	-> Hash (cost=14.03..14.03 rows=603 width=6)
20	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
21	-> Hash (cost=11.00..11.00 rows=600 width=6)
22	-> Seq Scan on city b (cost=0.00..11.00 rows=600 width=6)

Total rows: 47 of 47 Query complete 00:00:00.049

The results didn't exactly surprise me as my guess was fairly correct. In the first query, the subquery method was faster, and in the second query, the CTE method was faster. I was surprised, however, when I found out that both methods had the same cost amount.

Step 3: Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.

The first step I found to be fairly simple because there was just the one subquery that I had to replace. The second query I had a little bit more difficulty with as it was a more complicated query. I also noticed that when I completed the CTE version of the second query, the output was something that I was not expecting. In a previous task, the query outputted five 1s in the top_customer_count column, but this time, it did not. Looking deeper, this made me realize that this seems to be the correct answer, although Mexico should have a 2 in the column instead of a 1. I would definitely like to get more practice in with CTEs in the future.