

3.9: Common Table Expressions

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

- Find the average amount paid by the top 5 customers.

```
1 WITH average_total_amount_paid_cte(customer_id,first_name,last_name,city,country,total_amount_paid)AS
2 (SELECT B.customer_id, B.first_name, B.last_name,
3 D.city, E.country,
4 SUM(A.amount) AS Total_Amount_Paid FROM customer B
5 INNER JOIN payment A ON B.customer_id = A.customer_id
6 INNER JOIN address C ON B.address_id = C.address_id
7 INNER JOIN city D ON C.city_id = D.city_id
8 INNER JOIN country E ON D.country_id = E.country_id
9 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang',
10 GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
11 ORDER BY Total_Amount_Paid DESC
12 LIMIT 5)
13 SELECT AVG(total_amount_paid) AS average_total_amount_paid
14 FROM average_total_amount_paid_cte
```

Data output

average_total_amount_paid
107.3540000000000000

- Find the out how many of the top 5 customers are based within each country.

```
1 WITH top_customer_count_cte (amount,country_id,first_name,last_name,city,country,total_amount_paid) AS
2 (SELECT A.amount,B.customer_id, B.first_name, B.last_name, D.city, E.country,
3 SUM(amount)AS Total_Amount_Paid FROM payment A
4 INNER JOIN customer B ON A.customer_id = B.customer_id
5 INNER JOIN address C ON B.address_id = C.address_id
6 INNER JOIN city D ON C.city_id = D.city_id
7 INNER JOIN country E ON D.country_id = E.country_id
8 WHERE city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
9 GROUP BY A.amount,B.customer_id,B.first_name,B.last_name,D.city,E.country
10 ORDER BY SUM(amount)DESC LIMIT 5),
11 customer_count_cte AS(SELECT D.country, COUNT(DISTINCT A.customer_id) AS all_customer_count,
12 COUNT(DISTINCT D.country) AS top_customer_count
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 GROUP BY D.country)
18 SELECT D.country, COUNT(DISTINCT A.customer_id) AS all_customer_count,
19 COUNT(DISTINCT top_customer_count_cte)AS top_customer_count
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
```

Data Output

country	all_customer_count	top_customer_count
Mexico	30	2
United States	36	1
China	53	1
India	60	1
American Samoa	1	0

- Explain how you approached this step.

The first step is to copy the query for (finding the average amount paid by the top 5 customers) from the previous task and remove the OUTER query then adding the cte query. The cte query is added by using the WITH function as well as adding the AS function at the end of the syntax and adding parenthesis around inner query. The second step is the same as the first however 2 ctes are added to which include one for the count of customers and the other for count of the top customers.

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

1. Which approach do you think will perform better and why?

I think the subquery approach is better as it is easier to write however performance wise they seem to be the same.

2. Compare the costs of all the queries by creating query plans for each one.

Subquery task1.

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```

1 EXPLAIN
2 SELECT AVG(Total_amount_paid) AS Average
3 FROM
4 (SELECT B.customer_id, B.first_name, B.last_name, D.city, E.
5 SUM(A.amount) AS Total_Amount_Paid
6 FROM customer B
7 INNER JOIN payment A ON B.customer_id = A.customer_id
8 INNER JOIN address C ON B.address_id = C.address_id
9 INNER JOIN city D ON C.city_id = D.city_id
10 INNER JOIN country E ON D.country_id = E.country_id
11 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni',
12 'Sivas', 'Celaya', 'So Leopoldo'))
13 GROUP BY B.customer_id, B.first_name, B.last_name, D.city
14 ORDER BY Total_Amount_Paid DESC
15 LIMIT 5) AS Total_amount_paid

```

The execution plan (QUERY PLAN) is shown on the right:

Step	Operation	Cost	Rows	Width
1	Aggregate	(cost=64.45..64.46)	rows=1	width=32
2	-> Limit	(cost=64.37..64.39)	rows=5	width=67
3	-> Sort	(cost=64.37..64.98)	rows=243	width=67
4	Sort Key: (sum(a.amount)) DESC			
5	-> HashAggregate	(cost=57.30..60.34)	rows=243	width=67
6	Group Key: b.customer_id, d.city, e.country			
7	-> Nested Loop	(cost=18.16..54.87)	rows=243	width=41
8	-> Hash Join	(cost=17.88..37.14)	rows=10	width=35
9	Hash Cond: (d.country_id = e.country_id)			
10	-> Nested Loop	(cost=14.43..33.66)	rows=10	width=28
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=15
14	-> Hash	(cost=14.03..14.03)	rows=10	width=15

At the bottom, the status bar shows: Total rows: 22 of 22, Query complete 00:00:00.065, Rows selected: 22, Ln 1, Col 8.

Cte Task1.

The screenshot shows the DBeaver SQL editor interface. The top toolbar includes icons for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The main toolbar has icons for file operations, query execution, and settings. The SQL editor contains the following query:

```
1 EXPLAIN
2 WITH Average_total_amount_paid_cte(customer_id,first_name,la
3 (SELECT B.customer_id,B.first_name,B.last_name,D.city,E.coun
4 SUM(A.amount) AS Total_Amount_Paid
5 FROM customer B
6 INNER JOIN payment A ON B.customer_id = A.customer_id
7 INNER JOIN address C ON B.address_id = C.address_id
8 INNER JOIN city D ON C.city_id = D.city_id
9 INNER JOIN country E ON D.country_id = E.country_id
10 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dh
11 'Sivas', 'Celaya', 'So Leopoldo')
12 GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E
13 ORDER BY Total_Amount_Paid DESC
14 LIMIT 5)
15 SELECT AVG(Total_amount_paid)AS Average_total_amount_paid
16 FROM Average_total_amount_paid_cte
```

The query plan is displayed on the right side of the editor. It shows the following steps:

- 1. Aggregate (cost=64.45..64.46 rows=1 width=32)
- 2. -> Limit (cost=64.37..64.39 rows=5 width=67)
- 3. -> Sort (cost=64.37..64.98 rows=243 width=67)
- 4. Sort Key: (sum(a.amount)) DESC
- 5. -> HashAggregate (cost=57.30..60.34 rows=243 wid
- 6. Group Key: b.customer_id, d.city, e.country
- 7. -> Nested Loop (cost=18.16..54.87 rows=243 width=
- 8. -> Hash Join (cost=17.88..37.14 rows=10 width=35)
- 9. Hash Cond: (d.country_id = e.country_id)
- 10. -> Nested Loop (cost=14.43..33.66 rows=10 width=2
- 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=60
- 14. -> Hash (cost=14.03..14.03 rows=10 width=15)
- 15. -> Hash (cost=14.03..14.03 rows=10 width=15)

The status bar at the bottom indicates: Total rows: 22 of 22, Query complete 00:00:00.085, Rows selected: 22, Ln 6, Col 54.

The costs for both subquery and cte approach are the same for task1 which is 64.45.

Subquery Task 2.

The screenshot shows the DBeaver SQL editor interface. The top toolbar includes icons for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The main toolbar has icons for file operations, query execution, and settings. The SQL editor contains the following query:

```
1 EXPLAIN
2 WITH top_customer_count_cte(customer_id,first_name,la
3 (SELECT B.customer_id,B.first_name,B.last_name,D.city,E.coun
4 COUNT(*) AS top_customer_count
5 FROM customer B
6 INNER JOIN payment A ON B.customer_id = A.customer_id
7 INNER JOIN address C ON B.address_id = C.address_id
8 INNER JOIN city D ON C.city_id = D.city_id
9 INNER JOIN country E ON D.country_id = E.country_id
10 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dh
11 'Sivas', 'Celaya', 'So Leopoldo')
12 GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E
13 ORDER BY top_customer_count DESC
14 LIMIT 5)
15 SELECT AVG(Total_amount_paid)AS Average_total_amount_paid
16 FROM Average_total_amount_paid_cte
```

The query plan is displayed on the right side of the editor. It shows the following steps:

- 1. Limit (cost=166.83..166.85 rows=5 width=25)
- 2. -> Sort (cost=166.83..167.11 rows=109 width=25)
- 3. Sort Key: (count(DISTINCT top_customer_count_cte.*)) DESC
- 4. -> GroupAggregate (cost=156.04..165.02 rows=109 width=25)
- 5. Group Key: d.country
- 6. -> Merge Left Join (cost=156.04..159.44 rows=599 width=78)
- 7. Merge Cond: ((d.country)::text = (top_customer_count_cte.country)::text)
- 8. -> Sort (cost=90.94..92.44 rows=599 width=13)
- 9. Sort Key: d.country
- 10. -> Hash Join (cost=43.52..63.30 rows=599 width=13)
- 11. Hash Cond: (c.country_id = d.country_id)
- 12. -> Hash Join (cost=40.07..58.22 rows=599 width=6)
- 13. Hash Cond: (b.city_id = c.city_id)
- 14. -> Hash Join (cost=21.57..38.14 rows=599 width=6)
- 15. Hash Cond: (a.address_id = b.address_id)

The status bar at the bottom indicates: Total rows: 46 of 46, Query complete 00:00:00.115, Rows selected: 46.

Cte Task 2.

The screenshot shows a PostgreSQL query editor with a complex SQL query and its corresponding query plan. The query is as follows:

```
1 EXPLAIN
2 SELECT DISTINCT(A.country),
3 COUNT(DISTINCT D.customer_id) AS all_customer_count,
4 COUNT(DISTINCT A.country) AS top_customer_count
5 FROM country A
6 INNER JOIN city B ON A.country_id = B.country_id
7 INNER JOIN address C ON B.city_id = C.city_id
8 INNER JOIN customer D ON C.address_id = D.address_id
9 LEFT JOIN (SELECT B.customer_id, B.first_name,
10 B.last_name, D.city, E.country,
11 SUM(A.amount) AS Total_Amount_Paid FROM customer B
12 INNER JOIN payment A ON B.customer_id = A.customer_id
13 INNER JOIN address C ON B.address_id = C.address_id
14 INNER JOIN city D ON C.city_id = D.city_id
15 INNER JOIN country E ON D.country_id = E.country_id
16 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni
17 GROUP BY B.customer_id, B.first_name, B.last_name, D.c
18 LIMIT 5) AS top_5_customers
19 ON A.country=top_5_customers.COUNTRY
20 GROUP BY A.country, top_5_customers
21 ORDER BY all_customer_count DESC
22 LIMIT 5;
```

The query plan shows the following steps:

- Limit (cost=189.48..189.49 rows=5 width=84)
- > Sort (cost=189.48..190.84 rows=545 width=84)
- Sort Key: (count(DISTINCT d.customer_id)) DESC
- > HashAggregate (cost=174.98..180.43 rows=545 width=84)
- Group Key: count(DISTINCT d.customer_id), a.country, count(DISTINCT a...
- > GroupAggregate (cost=157.95..170.89 rows=545 width=84)
- Group Key: a.country, top_5_customers.*
- > Sort (cost=157.95..159.45 rows=599 width=72)
- Sort Key: a.country, top_5_customers.*
- > Hash Left Join (cost=108.02..130.32 rows=599 width=72)
- Hash Cond: ((a.country)::text = (top_5_customers.country)::text)
- > Hash Join (cost=43.52..63.30 rows=599 width=13)
- Hash Cond: (b.country_id = a.country_id)
- > Hash Join (cost=40.07..58.22 rows=599 width=6)
- Hash Cond: (c.city_id = b.city_id)

Total rows: 47 of 47 Query complete 00:00:00.177 Rows selected: 47 Ln 1, Col 8

3. Did the results surprise you? Write a few sentences to explain your answer.

The cost for the subquery in task 2 is 166.83, which is lower than the cte cost which is 189.48 for the same task. I presume that the added syntax for the cte also adds to the cost of the query.

Step 3:

Challenges faced when replacing subqueries with CTEs.

Replacing subquery with cte was a bit challenging because the cte stamen had to added multiple times within the query thus posing a challenge of where the statement would need to be added especially on the second task. It therefore means the more complex the query is the more challenging it would be to add the cte part.