

Achievement 3.9

PART1)

Query

Query History

```
1 WITH average_paid_cte (customer_id, first_name, last_name, Country, city, amount) AS
2
3 (SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city,
4  SUM(amount) AS total_amount_paid
5 FROM payment A
6  INNER JOIN customer B on A.customer_id = B.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 city IN ('Aurora', 'Acua', 'Citrus Heights',
11               'Iwaki', 'Ambattur', 'Shanwei',
12               'So Leopoldo', 'Teboksary',
13               'Tianjin', 'Cianjur')
14 GROUP BY country, city, B.customer_id
15 ORDER BY total_amount_paid DESC
16 LIMIT 5)
17
18 SELECT AVG(amount) AS Average_amount_paid
19 FROM average_paid_cte
```

Data output

Messages

Notifications

≡+

average_amount_paid

numeric

🔒

1	105.5540000000000000
---	----------------------

Query Query History

```
1 WITH country_count_cte (customer_id, first_name, last_name, Country, city, amount) AS
2
3 (SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city,
4  SUM(amount) AS total_customers
5 FROM payment A
6  INNER JOIN customer B on A.customer_id = B.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 Country IN ('India', 'China', 'United States',
11                   'Japan', 'Mexico', 'Brazil',
12                   'Russian Federation', 'Philippines',
13                   'Turkey', 'Indonesia')
14 GROUP BY country, city, B.customer_id
15 ORDER BY total_customers DESC)
16
17 SELECT e.country,
18 COUNT (DISTINCT b.customer_id) AS all_customer_count,
19 COUNT (DISTINCT e.country_id) AS top_customer_count
20 FROM country_count_cte
21  LEFT JOIN customer B on country_count_cte.customer_id = b.customer_id
22  LEFT JOIN address C ON b.address_id = c.address_id
23  LEFT JOIN city D ON c.city_id = d.city_id
24  LEFT JOIN country E ON d.country_ID = e.country_ID
25
26 GROUP BY e.country
27 LIMIT 5
```

Data output Messages Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Brazil	28	1
2	China	53	1
3	India	60	1
4	Indonesia	14	1
5	Japan	31	1

First of all, I need to think of what subquery I need to put in as a CTE, then construct the query and build an outer query with the CTE. It's like in Excel you have some data and you try to filter it out.

PART2)

Query Query History

```

1  EXPLAIN
2  SELECT AVG (total_amount_paid) AS average
3  FROM
4  (SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city,
5   SUM(amount) AS total_amount_paid

```

Data output Messages Notifications



QUERY PLAN	
	text
1	Aggregate (cost=65.98..65.99 rows=1 width=32)
2	-> Limit (cost=65.90..65.91 rows=5 width=270)
3	-> Sort (cost=65.90..66.54 rows=256 width=270)
4	Sort Key: (sum(a.amount)) DESC

Query Query History

```

1  EXPLAIN
2  WITH average_paid_cte (customer_id, first_name, last_name, Country, city, amount) AS
3
4  (SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city,
5   SUM(amount) AS total_amount_paid
6  FROM payment A
7   INNER JOIN customer B on A.customer_id = B.customer_id
8   INNER JOIN address C ON B.address_id = C.address_id

```

Data output Messages Notifications



QUERY PLAN	
	text
1	Aggregate (cost=65.98..65.99 rows=1 width=32)
2	-> Limit (cost=65.90..65.91 rows=5 width=270)
3	-> Sort (cost=65.90..66.54 rows=256 width=270)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=58.45..61.65 rows=256 width=270)
6	Group Key: e.country, d.city, b.customer_id
7	-> Nested Loop (cost=18.16..55.89 rows=256 width=28)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=22)

Query

Query History

1

EXPLAIN

2

SELECT e.country,

3

COUNT (DISTINCT b.customer_id) AS all_customer_count,

Data output

Messages

Notifications

≡+

▼

QUERY PLAN

text

1

Limit (cost=449.06..449.08 rows=5 width=29)

2

-> Sort (cost=449.06..449.34 rows=109 width=29)

3

Sort Key: (count(DISTINCT b.customer_id)) DESC

4

-> GroupAggregate (cost=432.08..447.25 rows=109 width=29)

5

Group Key: e.country_id

6

-> Sort (cost=432.08..435.60 rows=1408 width=17)

7

Sort Key: e.country_id

8

-> Hash Left Join (cost=325.85..358.45 rows=1408 width=17)

Data output

Messages

Notifications

≡+

▼

QUERY PLAN

text

1

Limit (cost=432.08..432.78 rows=5 width=25)

2

-> GroupAggregate (cost=432.08..447.25 rows=109 width=25)

3

Group Key: e.country

4

-> Sort (cost=432.08..435.60 rows=1408 width=17)

5

Sort Key: e.country

6

-> Hash Left Join (cost=325.85..358.45 rows=1408 width=17)

7

Hash Cond: (d.country_id = e.country_id)

8

-> Hash Left Join (cost=322.40..351.16 rows=1408 width=6)

9

Hash Cond: (c.city_id = d.city_id)

10

-> Hash Left Join (cost=303.90..328.94 rows=1408 width=6)

The cost of using subquery and CTE don't impact much. Maybe we need a larger data base to determine.

Since the query and CTE has similar usage. I wasn't surprised the cost were similar.

PART3)

Some of the challenges I face are that I am new to the software. I also believe there are better ways to organize the information and coding the tables more efficiently. I found it hard to locate the column names of a table and hence harder for me to match the primary keys. Even though this is an exercise, I realized if the wrong query is entered, it'll be harder to find what went wrong (meaning the query came out but names or functions are incorrect.) Overall this is a good experience and it's fun to explore for more ways to work with SQL.

