Answers 3.9

Step 1)

Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.

Step 1 (3.8)

Query Query History

```sql
WITH total_paid_cte(customer_id, first_name, last_name,
                    address, city, country, total_amount_paid)
AS (SELECT A.customer_id, A.first_name, A.last_name, B.address,
      C.city, D.country, SUM(E.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 ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulia)',
                 'Kurashiki','Pingxang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id, A.first_name, A.last_name, B.address, C.city, D.country
ORDER BY total_amount_paid DESC
LIMIT 5)
SELECT AVG (total_amount_paid) AS average_total_amount_paid
FROM total_paid_cte
```

Data output Messages Notifications

	average_total_amount_paid numeric
1	107.3540000000000000

Approach: First I retrieved the subqueries from 3.8. I then removed the outer query and replaced it with the “WITH” statement, creating a retrievable CTE. With the second one, this was more complicated, as I had to reformat multiple sections in the multiple sub queried statement in order to conform with the CTE syntax.

Step 2 (3.8)

Query Query History

```sql
WITH top_customer_count_cte(amount, customer_id,first_name, last_name,
                            city, country, total_amount_paid)
AS (SELECT A.amount, B.customer_id, B.first_name, B.last_name,
           D.city, E.country,
SUM(amount) AS total_amount_paid
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 city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulia)',
                'Kurashiki','Pingxang','Sivas','Celaya','So Leopoldo')
GROUP BY A.amount, B.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY SUM(amount) DESC LIMIT 5),
customer_count_cte AS
(SELECT D.country, COUNT(DISTINCT A.customer_id) AS all_customer_count,
COUNT(DISTINCT D.country) 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
GROUP BY D.country)
SELECT D.country, COUNT(DISTINCT A.customer_id) AS all_customer_count,
COUNT(DISTINCT top_customer_count_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_count_cte on D.country = top_customer_count_cte.country
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 5
```

Data output Messages Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	2
2	Mexico	30	2
3	United States	36	1

Step 2)

Which approach do you think will perform better and why?

For single use purposes, I would imagine creating a CTE takes more time than it saves. However, for frequent repeated use, the time the CTE's creation would save, rather than recreating the entire subqueries each time, would surely be more optimal. I would imagine that the subqueries run slightly quicker than CTE's on the system, just based on being shorter, though I would imagine compared to real productivity, this small difference in processing speed would be negligible.

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

3.8 Step 1

	SUBQUERY	CTE
COST	68.64..68.65	68.64..68.65
TIME	0.066secs	0.073secs

3.8 Step 2

	SUBQUERY	CTE
COST	199.85..205.30	172.25..172.53
TIME	0.051secs	0.107secs

3.8 Step 1 Subquery

Query Query History

```
EXPLAIN(SELECT AVG(total_payment) AS average
FROM (SELECT A.customer_id, A.first_name, A.last_name, B.address,
      C.city, D.country, SUM(E.amount) AS total_payment
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 ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulia)',
                 'Kurashiki','Pingxang','Sivas','Celaya','So Leopoldo')
```

Data output Messages Notifications

	QUERY PLAN text
1	Aggregate (cost=68.64..68.65 rows=1 width=32)
2	-> Limit (cost=68.56..68.58 rows=5 width=87)
3	-> Sort (cost=68.56..69.26 rows=278 width=87)
4	Sort Key: (sum(e.amount)) DESC
5	-> HashAggregate (cost=60.47..63.95 rows=278 width=87)
6	Group Key: a.customer_id, b.address, c.city, d.country
7	-> Nested Loop (cost=18.17..57.00 rows=278 width=61)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=55)
9	Hash Cond: (c.country_id = d.country_id)
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=48)
11	-> Hash Join (cost=14.15..29.77 rows=10 width=35)
12	Hash Cond: (b.city_id = c.city_id)
13	-> Seq Scan on address b (cost=0.00..14.03 rows=603 width=26)
14	-> Hash (cost=14.03..14.03 rows=10 width=15)
15	-> Seq Scan on city c (cost=0.03..14.03 rows=10 width=15)
16	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulia)",Kurashiki,Pingxang,Sivas,Celaya,"So Leopoldo"}'::text[]))
17	-> Index Scan using idx_fk_address_id on customer a (cost=0.28..0.38 rows=1 width=19)
18	Index Cond: (address_id = b.address_id)
19	-> Hash (cost=2.09..2.09 rows=109 width=13)
20	-> Seq Scan on country d (cost=0.00..2.09 rows=109 width=13)
21	-> Index Scan using idx_fk_customer_id on payment e (cost=0.29..1.71 rows=28 width=8)
22	Index Cond: (customer_id = a.customer_id)

Total rows: 22 of 22 Query complete 00:00:00.066

3.8 Step 1 CTE

Query Query History

```
EXPLAIN(WITH total_paid_cte(customer_id, first_name, last_name,
                            address, city, country, total_amount_paid)
AS (SELECT A.customer_id, A.first_name, A.last_name, B.address,
      C.city, D.country, SUM(E.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 ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulia)',
```

Data output Messages Notifications

	QUERY PLAN text
1	Aggregate (cost=68.64..68.65 rows=1 width=32)
2	-> Limit (cost=68.56..68.58 rows=5 width=87)
3	-> Sort (cost=68.56..69.26 rows=278 width=87)
4	Sort Key: (sum(e.amount)) DESC
5	-> HashAggregate (cost=60.47..63.95 rows=278 width=87)
6	Group Key: a.customer_id, b.address, c.city, d.country
7	-> Nested Loop (cost=18.17..57.00 rows=278 width=61)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=55)
9	Hash Cond: (c.country_id = d.country_id)
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=48)
11	-> Hash Join (cost=14.15..29.77 rows=10 width=35)
12	Hash Cond: (b.city_id = c.city_id)
13	-> Seq Scan on address b (cost=0.00..14.03 rows=603 width=26)
14	-> Hash (cost=14.03..14.03 rows=10 width=15)
15	-> Seq Scan on city c (cost=0.03..14.03 rows=10 width=15)
16	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulia)",Kurashiki,Pingxang,Sivas,Celaya,"So Leopoldo"}'::text[]))
17	-> Index Scan using idx_fk_address_id on customer a (cost=0.28..0.38 rows=1 width=19)
18	Index Cond: (address_id = b.address_id)
19	-> Hash (cost=2.09..2.09 rows=109 width=13)
20	-> Seq Scan on country d (cost=0.00..2.09 rows=109 width=13)
21	-> Index Scan using idx_fk_customer_id on payment e (cost=0.29..1.71 rows=28 width=8)
22	Index Cond: (customer_id = a.customer_id)

Total rows: 22 of 22 Query complete 00:00:00.073

3.8 Step 2 Subquery

Query Query History

```
EXPLAIN(SELECT DISTINCT (A.country),
COUNT(DISTINCT D.customer_id) AS all_customer_count,
COUNT (distinct top_5_customers.customer_id) AS top_customer_count
FROM country A
```

Data output Messages Notifications

	QUERY PLAN text
1	Unique (cost=199.85..205.30 rows=545 width=34)
2	-> Sort (cost=199.85..201.21 rows=545 width=34)
3	Sort Key: (count(DISTINCT d.customer_id)) DESC, a.country, (count(DISTINCT top_5_customers.customer_id))
4	-> GroupAggregate (cost=162.14..175.08 rows=545 width=34)
5	Group Key: a.country, top_5_customers.country
6	-> Sort (cost=162.14..163.64 rows=599 width=26)
7	Sort Key: a.country, top_5_customers.country
8	-> Hash Left Join (cost=112.21..134.51 rows=599 width=26)
9	Hash Cond: ((a.country)::text = (top_5_customers.country)::text)
10	-> Hash Join (cost=43.52..63.30 rows=599 width=13)
11	Hash Cond: (b.country_id = a.country_id)
12	-> Hash Join (cost=40.07..58.22 rows=599 width=6)
13	Hash Cond: (c.city_id = b.city_id)
14	-> Hash Join (cost=21.57..38.14 rows=599 width=6)
15	Hash Cond: (d.address_id = c.address_id)
16	-> Seq Scan on customer d (cost=0.00..14.99 rows=599 width=6)
17	-> Hash (cost=14.03..14.03 rows=603 width=6)
18	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
19	-> Hash (cost=11.00..11.00 rows=600 width=6)
20	-> Seq Scan on city b (cost=0.00..11.00 rows=600 width=6)
21	-> Hash (cost=2.09..2.09 rows=109 width=13)
22	-> Seq Scan on country a (cost=0.00..2.09 rows=109 width=13)
23	-> Hash (cost=68.63..68.63 rows=5 width=13)
24	-> Subquery Scan on top_5_customers (cost=68.56..68.63 rows=5 width=13)
25	-> Limit (cost=68.56..68.58 rows=5 width=87)
26	-> Sort (cost=68.56..69.26 rows=278 width=87)
27	Sort Key: (sum(e.amount)) DESC
28	-> HashAggregate (cost=60.47..63.95 rows=278 width=87)
29	Group Key: a_1.customer_id, b_1.address, c_1.city, d_1.country
30	-> Nested Loop (cost=18.17..57.00 rows=278 width=61)
31	-> Hash Join (cost=17.88..37.14 rows=10 width=55)
32	Hash Cond: (c_1.country_id = d_1.country_id)
33	-> Nested Loop (cost=14.43..33.66 rows=10 width=48)
34	-> Hash Join (cost=14.15..29.77 rows=10 width=35)
35	Hash Cond: (b_1.city_id = c_1.city_id)
36	-> Seq Scan on address b_1 (cost=0.00..14.03 rows=603 width=26)
37	-> Hash (cost=14.03..14.03 rows=10 width=15)
38	-> Seq Scan on city c_1 (cost=0.03..14.03 rows=10 width=15)
39	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulia)",Kurashiki,Pingxang,Sivas,Celaya,"So Leopoldo"}'::text[]))
40	-> Index Scan using idx_fk_address_id on customer a_1 (cost=0.28..0.38 rows=1 width=19)
41	Index Cond: (address_id = b_1.address_id)
42	-> Hash (cost=2.09..2.09 rows=109 width=13)
43	-> Seq Scan on country d_1 (cost=0.00..2.09 rows=109 width=13)
44	-> Index Scan using idx_fk_customer_id on payment e (cost=0.29..1.71 rows=28 width=8)
45	Index Cond: (customer_id = a_1.customer_id)

Total rows: 45 of 45 Query complete 00:00:00.051

3.8 Step 2 CTE

Query Query History

```
EXPLAIN(WITH top_customer_count_cte(amount, customer_id,first_name, last_name,
                                    city, country, total_amount_paid)
AS (SELECT A.amount, B.customer_id, B.first_name, B.last_name,
           D.city, E.country,
SUM(amount) AS total_amount_paid
```

Data output Messages Notifications

	QUERY PLAN text
1	Sort (cost=172.25..172.53 rows=109 width=25)
2	Sort Key: (count(DISTINCT top_customer_count_cte.customer_id)) DESC
3	-> GroupAggregate (cost=159.58..168.56 rows=109 width=25)
4	Group Key: d.country
5	-> Merge Left Join (cost=159.58..162.98 rows=599 width=17)
6	Merge Cond: ((d.country)::text = (top_customer_count_cte.country)::text)
7	-> Sort (cost=90.94..92.44 rows=599 width=13)
8	Sort Key: d.country
9	-> Hash Join (cost=43.52..63.30 rows=599 width=13)
10	Hash Cond: (c.country_id = d.country_id)
11	-> Hash Join (cost=40.07..58.22 rows=599 width=6)
12	Hash Cond: (b.city_id = c.city_id)
13	-> Hash Join (cost=21.57..38.14 rows=599 width=6)
14	Hash Cond: (a.address_id = b.address_id)
15	-> Seq Scan on customer a (cost=0.00..14.99 rows=599 width=6)
16	-> Hash (cost=14.03..14.03 rows=603 width=6)
17	-> Seq Scan on address b (cost=0.00..14.03 rows=603 width=6)
18	-> Hash (cost=11.00..11.00 rows=600 width=6)
19	-> Seq Scan on city c (cost=0.00..11.00 rows=600 width=6)
20	-> Hash (cost=2.09..2.09 rows=109 width=13)
21	-> Seq Scan on country d (cost=0.00..2.09 rows=109 width=13)
22	-> Sort (cost=68.64..68.66 rows=5 width=13)
23	Sort Key: top_customer_count_cte.country
24	-> Subquery Scan on top_customer_count_cte (cost=68.52..68.59 rows=5 width=13)
25	-> Limit (cost=68.52..68.54 rows=5 width=73)
26	-> Sort (cost=68.52..69.22 rows=277 width=73)
27	Sort Key: (sum(a_1.amount)) DESC
28	-> HashAggregate (cost=60.46..63.92 rows=277 width=73)
29	Group Key: a_1.amount, b_1.customer_id, d_1.city, e.country
30	-> Nested Loop (cost=18.17..57.00 rows=277 width=41)
31	-> Hash Join (cost=17.88..37.14 rows=10 width=35)
32	Hash Cond: (d_1.country_id = e.country_id)
33	-> Nested Loop (cost=14.43..33.66 rows=10 width=28)
34	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
35	Hash Cond: (c_1.city_id = d_1.city_id)
36	-> Seq Scan on address c_1 (cost=0.00..14.03 rows=603 width=6)
37	-> Hash (cost=14.03..14.03 rows=10 width=15)
38	-> Seq Scan on city d_1 (cost=0.03..14.03 rows=10 width=15)
39	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulia)",Kurashiki,Pingxang,...
40	-> Index Scan using idx_fk_address_id on customer b_1 (cost=0.28..0.38 rows=1 width=19)
41	Index Cond: (address_id = c_1.address_id)
42	-> Hash (cost=2.09..2.09 rows=109 width=13)
43	-> Seq Scan on country e (cost=0.00..2.09 rows=109 width=13)
44	-> Index Scan using idx_fk_customer_id on payment a_1 (cost=0.29..1.71 rows=28 width=8)
45	Index Cond: (customer_id = b_1.customer_id)

Step 3)

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

For Step 1 it was relatively simple to understand, and without multiple subqueries replacing the subquery with a CTE was straightforward. For Step 2 it was more challenging, and initially perplexing as I wasn't sure how I would create a CTE for a statement with multiple subqueries. There were a lot of attempts that resulted in invalid statements, it took a lot of trial and error to eventually have the CTE provide an output.