Step 1: Answer the business questions from step 1 and 2 of task 3.8 using CTEs
Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs. Copy-paste your CTEs and their outputs into your answers document. Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.

- Creating the CTE's from the previous subquery tasks were relatively simple. I identified the specific subquery statement and extracted it from its original position. Using that subquery, I rebuilt the statement into a CTE by placing it in front of the entire query statement as specified within the lesson. The second step required two CTEs as there were two subqueries. I listed those one after the other before finishing the rest of the main query command.

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

Data output | Messages | Notifications

	average_amount_paid numeric
1	107.3540000000000000

```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 D.city IN ('Aurora', 'Bhusawal', 'Shivapuri', 'Cianjur', 'Kuwana', 'Acua', 'Saint Louis', 'So Leopoldo', 'Iwaki', 'Eskisehir')
        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 ( A.customer_id) AS all_customer_count, COUNT ( 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 ( 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
```

Loading...

Data output | Messages | Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Mexico	30	1
2	Turkey	15	1
3	India	60	1
4	Japan	31	1
5	United States	36	1

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

Which approach do you think will perform better and why? Compare the costs of all the queries by creating query plans for each one. The EXPLAIN command gives you an *estimated* cost. To find out the actual speed of your queries, run them in pgAdmin 4. After each query has been run, a pop-up window will display its speed in milliseconds. Did the results surprise you? Write a few sentences to explain your answer.

- Step 1 – There results are to be expected as these two methods were nearly identical.
    - Subquery cost: 64.45 .. 64.46

Query | Query History

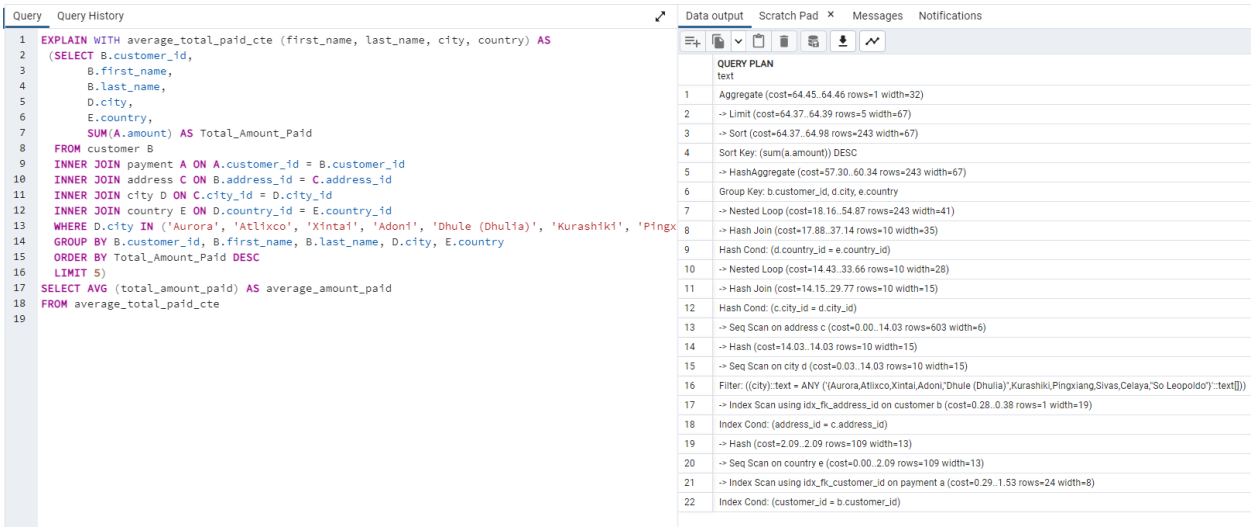
```sql
EXPLAIN SELECT AVG(total_amount_paid) AS average
FROM
(SELECT B.customer_id,
        B.first_name,
        B.last_name,
        D.city,
        E.country,
        SUM(A.amount) AS Total_Amount_Paid
FROM customer B
INNER JOIN payment A ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = 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', 'Pingxia
GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY Total_Amount_Paid DESC
LIMIT 5) AS total_amount_paid
```

Data output | Scratch Pad | Messages | 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=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=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=19)
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 00:00:00.035 | Ln 1, Col

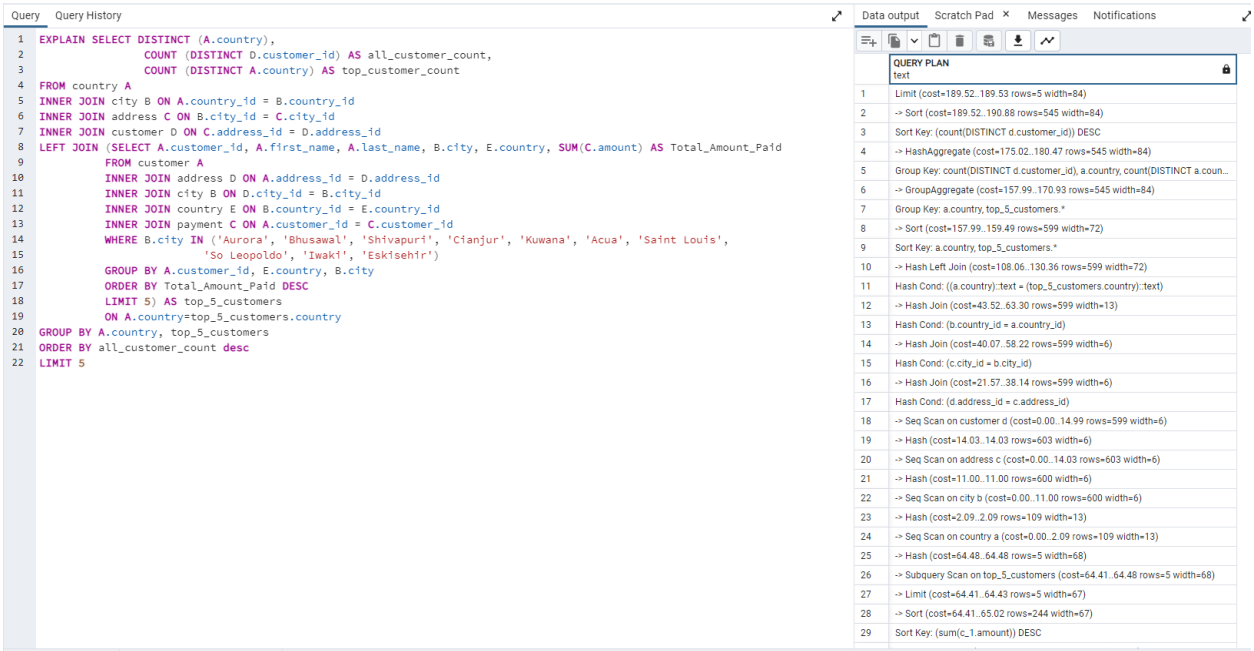
- CTE cost: 64.45 .. 64.46



```
EXPLAIN WITH average_total_paid_cte (first_name, last_name, city, country) AS
 (SELECT B.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM(A.amount) AS Total_Amount_Paid
  FROM customer B
  INNER JOIN payment A 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', 'Pingx
  GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
  ORDER BY Total_Amount_Paid DESC
  LIMIT 5)
SELECT AVG (total_amount_paid) AS average_amount_paid
FROM average_total_paid_cte
```

	QUERY PLAN text
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=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=19)
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)

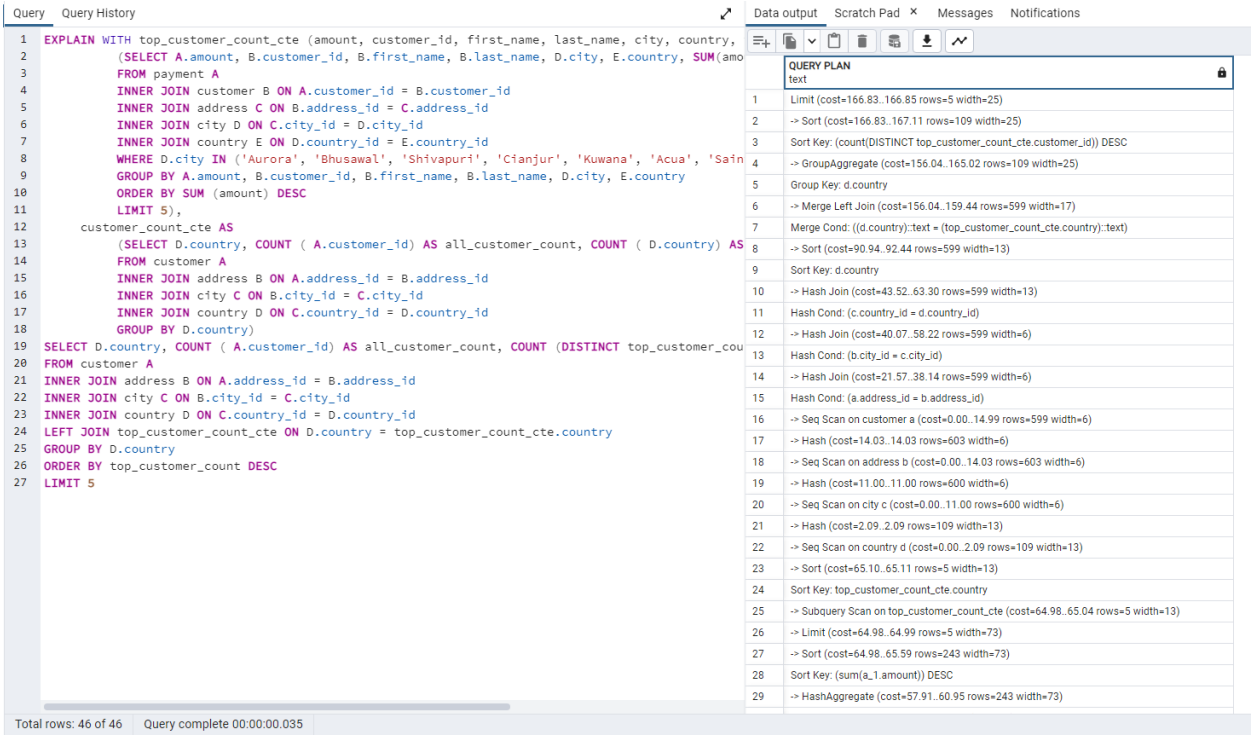
- Step 2 – These results were surprising as I didn't expect a significant difference between the two methods. On the previous step there was no difference in cost, however it seems that including more than one subquery increases the cost of running queries. Also, the CTE method seemed a bit more complicated than the subquery method so I assumed it would cost more.
    - Subquery cost: 189.52 .. 189.53



```
EXPLAIN SELECT DISTINCT (A.country),
               COUNT (DISTINCT D.customer_id) AS all_customer_count,
               COUNT (DISTINCT A.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 (SELECT A.customer_id, A.first_name, A.last_name, B.city, E.country, SUM(C.amount) AS Total_Amount_Paid
          FROM customer A
          INNER JOIN address D ON A.address_id = D.address_id
          INNER JOIN city B ON D.city_id = B.city_id
          INNER JOIN country E ON B.country_id = E.country_id
          INNER JOIN payment C ON A.customer_id = C.customer_id
          WHERE B.city IN ('Aurora', 'Bhusawal', 'Shivapuri', 'Cianjur', 'Kuwana', 'Acua', 'Saint Louis',
                           'So Leopoldo', 'Iwaki', 'Eskisehir')
          GROUP BY A.customer_id, E.country, B.city
          ORDER BY Total_Amount_Paid DESC
          LIMIT 5) AS top_5_customers
          ON A.country=top_5_customers.country
GROUP BY A.country, top_5_customers
ORDER BY all_customer_count desc
LIMIT 5
```

	QUERY PLAN text
1	Limit (cost=189.52..189.53 rows=5 width=84)
2	-> Sort (cost=189.52..190.88 rows=545 width=84)
3	Sort Key: (count(DISTINCT d.customer_id)) DESC
4	-> HashAggregate (cost=175.02..180.47 rows=545 width=84)
5	Group Key: count(DISTINCT d.customer_id), a.country, count(DISTINCT a.coun...
6	-> GroupAggregate (cost=157.99..170.93 rows=545 width=84)
7	Group Key: a.country, top_5_customers.*
8	-> Sort (cost=157.99..159.49 rows=599 width=72)
9	Sort Key: a.country, top_5_customers.*
10	-> Hash Left Join (cost=108.06..130.36 rows=599 width=72)
11	Hash Cond: ((a.country)::text = (top_5_customers.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)
23	-> Hash (cost=2.09..2.09 rows=109 width=13)
24	-> Seq Scan on country a (cost=0.00..2.09 rows=109 width=13)
25	-> Hash (cost=64.48..64.48 rows=5 width=68)
26	-> Subquery Scan on top_5_customers (cost=64.41..64.48 rows=5 width=68)
27	-> Limit (cost=64.41..64.43 rows=5 width=67)
28	-> Sort (cost=64.41..65.02 rows=244 width=67)
29	Sort Key: (sum(c_1.amount)) DESC

- CTE cost: 166.83 .. 166.85



```
EXPLAIN WITH top_customer_count_cte (amount, customer_id, first_name, last_name, city, country,
          (SELECT A.amount, B.customer_id, B.first_name, B.last_name, D.city, E.country, SUM(amo
          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', 'Bhusawal', 'Shivapuri', 'Cianjur', 'Kuwana', 'Acua', 'Sain
          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 ( A.customer_id) AS all_customer_count, COUNT ( D.country) AS
          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 ( A.customer_id) AS all_customer_count, COUNT (DISTINCT top_customer_cou
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
```

	QUERY PLAN text
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.customer_id)) 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=17)
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)
16	-> Seq Scan on customer a (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 b (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 c (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 d (cost=0.00..2.09 rows=109 width=13)
23	-> Sort (cost=65.10..65.11 rows=5 width=13)
24	Sort Key: top_customer_count_cte.country
25	-> Subquery Scan on top_customer_count_cte (cost=64.98..65.04 rows=5 width=13)
26	-> Limit (cost=64.98..64.99 rows=5 width=73)
27	-> Sort (cost=64.98..65.59 rows=243 width=73)
28	Sort Key: (sum(a_1.amount)) DESC
29	-> HashAggregate (cost=57.91..60.95 rows=243 width=73)

Step 3:

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

- Finding the placement for the CTE was simple, however, finishing the statement so that a complete query can be made was difficult. Once additional commands, such as JOINs, were introduced, figuring out which type of joins correlate to which CTE's were a bit complicated to understand. Also, with queries that have multiple subqueries, things can get a bit complicated when it comes to converting them to CTE's, placing them within the command correctly, and referencing them at the correct time within the command.

Step 4:

Save your "Answers 3.9" document as a PDF and upload it here for your tutor to review.