SUBMITTED BY:
Grace Skelley on 11.7.2022

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

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

QUERY 1 *(from TASK 3.8)*

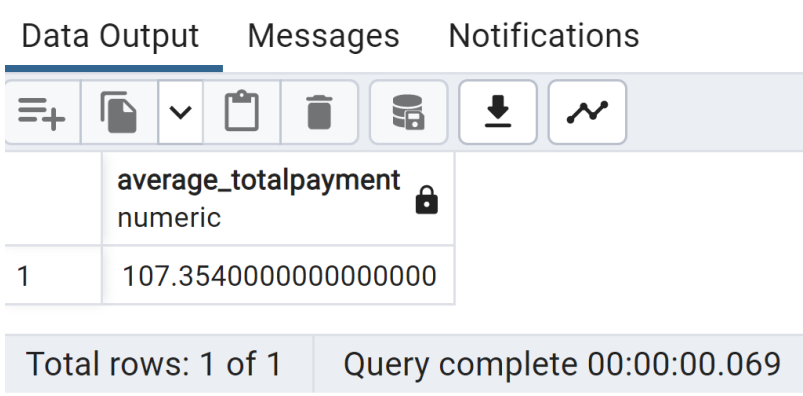
QUERY

```
WITH Average_TotalPayment_cte(customer_id ,first_name
                ,last_name ,city ,country ,payment) AS

(SELECT A.customer_id
      ,A.first_name
      ,A.last_name
      ,C.city
      ,D.country
      ,SUM(E.amount) AS 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 E.customer_id = A.customer_id
WHERE C. city IN ('Aurora' ,'Atlixco' ,'Xintai' ,'Adoni'
,'Dhule  (Dhulia)'  ,'Kurashiki'  ,'Pingxiang'  ,'Sivas'
,'Celaya' ,'So Leopoldo')
GROUP BY A.customer_id ,A.first_name ,A.last_name ,C.city
,D.country
ORDER BY SUM(E.amount) DESC
LIMIT 5)

SELECT AVG(payment) AS Average_TotalPayment
FROM Average_TotalPayment_cte
```

AVERAGE



QUERY 2 *(from TASK 3.8)*

QUERY

```
WITH total_paid_in_top_10_cities_cte(customer_id ,first_name ,last_name ,country ,city ,total_amount_paid) AS
                       (SELECT A.customer_id ,A.first_name ,A.last_name ,D.country ,C.city
                       ,SUM(E.amount) AS total_amount_paid
                        FROM payment E
                        INNER JOIN customer A ON A.customer_id = E.customer_id
                        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
                        WHERE C.city IN ('Aurora' ,'Atlixco' ,'Xintai' ,'Adoni' ,'Dhule (Dhulia)'
                                        ,'Kurashiki' ,'Pingxiang' ,'Sivas' ,'Celaya' ,'So Leopoldo')
                        GROUP BY A.customer_id ,first_name ,last_name ,city ,country
                        ORDER BY total_amount_paid DESC
                        LIMIT 5)



SELECT DISTINT(D.country) ,COUNT(DISTICNT A.customer_id) AS all_customer_count
                          ,COUNT(DISTICNT D.country_id) AS top_customer_count
FROM country D
INNER JOIN city C ON C.country_id = D.country_id
INNER JOIN address B ON C.city_id = B.city_id
INNER JOIN customer A ON A.address_id = B.address _id
LEFT JOIN total_paid_in_top_10_cities_cte ON D.country = total_paid_in_top_10_cities_cte.country
GROUP BY D.country
ORDER BY all_customer_count DESC
LIMIT 5;
```

TOP 5 CUSTOMER
(WITHIN EACH COUNTRY)

Data Output | Messages | Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1

Total rows: 5 of 5 | Query complete 00:00:00.214

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

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

While writing CTE is the most tedious work among the two, I am certain the CTE will produce better results in terms of performance as it already recognizes pre-existing query to reference its data output for rapid search.

2. *Compare the costs of all the queries by creating query plans for each one.*
3. *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.*
4. *Did the results surprise you? Write a few sentences to explain your answer.*

SUBQUEARIES

TASK 1

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=64.49..64.50 rows=1 width=32)
2	-> Limit (cost=64.41..64.43 rows=5 width=67)
3	-> Sort (cost=64.41..65.02 rows=244 width=67)
4	Sort Key: (sum(e.amount)) DESC
5	-> HashAggregate (cost=57.31..60.36 rows=244 width=67)
6	Group Key: a.customer_id, c.city, d.country
7	-> Nested Loop (cost=18.16..54.87 rows=244 width=41)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=35)
9	Hash Cond: (c.country_id = d.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: (b.city_id = c.city_id)
13	-> Seq Scan on address b (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 c (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 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..1.53 rows=24 width=8)
22	Index Cond: (customer_id = a.customer_id)

```
cost: 64.49..64.50 rows=1 width=32
runtime: 98 msec. 22 rows affected.
```

CTE

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=64.49..64.50 rows=1 width=32)
2	-> Limit (cost=64.41..64.43 rows=5 width=67)
3	-> Sort (cost=64.41..65.02 rows=244 width=67)
4	Sort Key: (sum(e.amount)) DESC
5	-> HashAggregate (cost=57.31..60.36 rows=244 width=67)
6	Group Key: a.customer_id, c.city, d.country
7	-> Nested Loop (cost=18.16..54.87 rows=244 width=41)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=35)
9	Hash Cond: (c.country_id = d.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: (b.city_id = c.city_id)
13	-> Seq Scan on address b (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 c (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 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..1.53 rows=24 width=8)
22	Index Cond: (customer_id = a.customer_id)

```
cost: 64.49..64.50 rows=1 width=32
runtime: 85 msec. 22 rows affected.
```

The cost results are unexpected as I would anticipate one of the scripts will be much lower than the other. However, the efficiency for executing of each differs with runtime which only proves that CTE provides result at high-speed.

SUBQUEARIES	CTE
TASK 2	

Data Output Messages Notifications

	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 a.customer_id)) DESC
4	-> HashAggregate (cost=175.02..180.47 rows=545 width=84)
5	Group Key: count(DISTINCT a.customer_id), d.country, count(DISTINCT d.country)
6	-> GroupAggregate (cost=157.99..170.93 rows=545 width=84)
7	Group Key: d.country, top_5_customers.*
8	-> Sort (cost=157.99..159.49 rows=599 width=72)
9	Sort Key: d.country, top_5_customers.*
10	-> Hash Left Join (cost=108.06..130.36 rows=599 width=72)
11	Hash Cond: ((d.country)::text = (top_5_customers.country)::text)
12	-> Hash Join (cost=43.52..63.30 rows=599 width=13)
13	Hash Cond: (c.country_id = d.country_id)
14	-> Hash Join (cost=40.07..58.22 rows=599 width=6)
15	Hash Cond: (b.city_id = c.city_id)
16	-> Hash Join (cost=21.57..38.14 rows=599 width=6)
17	Hash Cond: (a.address_id = b.address_id)
18	-> Seq Scan on customer a (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 b (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 c (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 d (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(e.amount)) DESC
30	-> HashAggregate (cost=57.31..60.36 rows=244 width=67)
31	Group Key: a_1.customer_id, c_1.city, d_1.country
32	-> Nested Loop (cost=18.16..54.87 rows=244 width=41)
33	-> Hash Join (cost=17.88..37.14 rows=10 width=35)

```
cost: 189.52..189.53 rows=5 width=84
runtime: 103 msec. 47 rows affected.
```

Data Output Messages Notifications

	QUERY PLAN text
1	Limit (cost=168.10..168.15 rows=5 width=25)
2	-> Unique (cost=168.10..169.19 rows=109 width=25)
3	-> Sort (cost=168.10..168.38 rows=109 width=25)
4	Sort Key: (count(DISTINCT a.customer_id)) DESC, d.country, (count(DISTINCT d.country_id))
5	-> GroupAggregate (cost=155.43..164.42 rows=109 width=25)
6	Group Key: d.country
7	-> Merge Left Join (cost=155.43..158.83 rows=599 width=17)
8	Merge Cond: ((d.country)::text = (total_paid_in_top_10_cities_cte.country)::text)
9	-> Sort (cost=90.94..92.44 rows=599 width=17)
10	Sort Key: d.country
11	-> Hash Join (cost=43.52..63.30 rows=599 width=17)
12	Hash Cond: (c.country_id = d.country_id)
13	-> Hash Join (cost=40.07..58.22 rows=599 width=6)
14	Hash Cond: (b.city_id = c.city_id)
15	-> Hash Join (cost=21.57..38.14 rows=599 width=6)
16	Hash Cond: (a.address_id = b.address_id)
17	-> Seq Scan on customer a (cost=0.00..14.99 rows=599 width=6)
18	-> Hash (cost=14.03..14.03 rows=603 width=6)
19	-> Seq Scan on address b (cost=0.00..14.03 rows=603 width=6)
20	-> Hash (cost=11.00..11.00 rows=600 width=6)
21	-> Seq Scan on city c (cost=0.00..11.00 rows=600 width=6)
22	-> Hash (cost=2.09..2.09 rows=109 width=13)
23	-> Seq Scan on country d (cost=0.00..2.09 rows=109 width=13)
24	-> Sort (cost=64.49..64.51 rows=5 width=9)
25	Sort Key: total_paid_in_top_10_cities_cte.country
26	-> Subquery Scan on total_paid_in_top_10_cities_cte (cost=64.37..64.44 rows=5 width=9)
27	-> Limit (cost=64.37..64.39 rows=5 width=67)
28	-> Sort (cost=64.37..64.98 rows=243 width=67)
29	Sort Key: (sum(e.amount)) DESC
30	-> HashAggregate (cost=57.30..60.34 rows=243 width=67)

```
cost: 168.10..168.15 rows=5 width=25
runtime: 85 msec. 47 rows affected.
```

The cost of CTE is as expected, which has straight target performance to produce data output. Whilst, similar to task 1 runtime results it generated much quicker outcome warranting cost report.

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

Revising the queries to CTEs is tougher than it seems for learning experience. The first query was not as intricate since it's up-front, although what presented more of a challenge was in query 2. Which has several built-in complex queries. Ensuring all statements are linked with each other, it was not easy task to produce a script at one try. There were a lot of trial-and-error along the path of studying this approach. Though, the data output is evidently efficient and costs much less. There is undeniably a need of strong attention to details to formulate the syntax accurately, which necessitates a lot of exercise for proper execution. I have a lot to learn and most definitely a demand for more practice into mastering SQL.