Task 3.8 Chris Arnold

Q1

Rockbuster/postgres@PostgreSQL 13

Query Editor | Query History

```
SELECT ROUND(AVG(total_paid),2) AS average
FROM (SELECT A.Customer_id, A.first_name, A.last_name, E.country, B.city, SUM(C.amount)AS Total_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', 'Atlixco', 'Xintai', 'Adoni', 'Dhule', 'Kurashiki', 'Pingxian', 'Sivas', 'Celaya', 'So Leopoldo')
    GROUP BY A.customer_id, E.country, B.city
    ORDER BY Total_Paid DESC
    LIMIT 5) AS total_amount_paid;
```

Data Output | Explain | Messages | Notifications

	average numeric
1	107.35

Q2

Rockbuster/postgres@PostgreSQL 13

Query Editor | Query History

```
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, E.country, B.city, SUM(C.amount)AS Total_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 E.country IN
            ('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil', 'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
        AND B.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule', 'Kurashiki', 'Pingxian', 'Sivas', 'Celaya', 'So Leopoldo')
        GROUP BY A.customer_id, E.country, B.city
        ORDER BY Total_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;
```

Data Output | Explain | 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

Q3

- Do you think steps 1 and 2 could be done without using subqueries?

  You definitely could have done step one without a sub query through the use of HAVE with aggregation functions. This would have taken just as much code just in a different way though, so it isnt necessarily more efficient or a better choice in my eyes.

- When do you think subqueries are useful?

  As stated in our reading today, the biggest place that you are at the advantage using a sub query is in a situation where you need to adjust answers to previous questions (through queries) that will change frequently in the business you are working for. Averages will change over time, popular trends will change over time, and many other things on top of that. You can run the previous query with new parameters that may have changed (two steps in one).