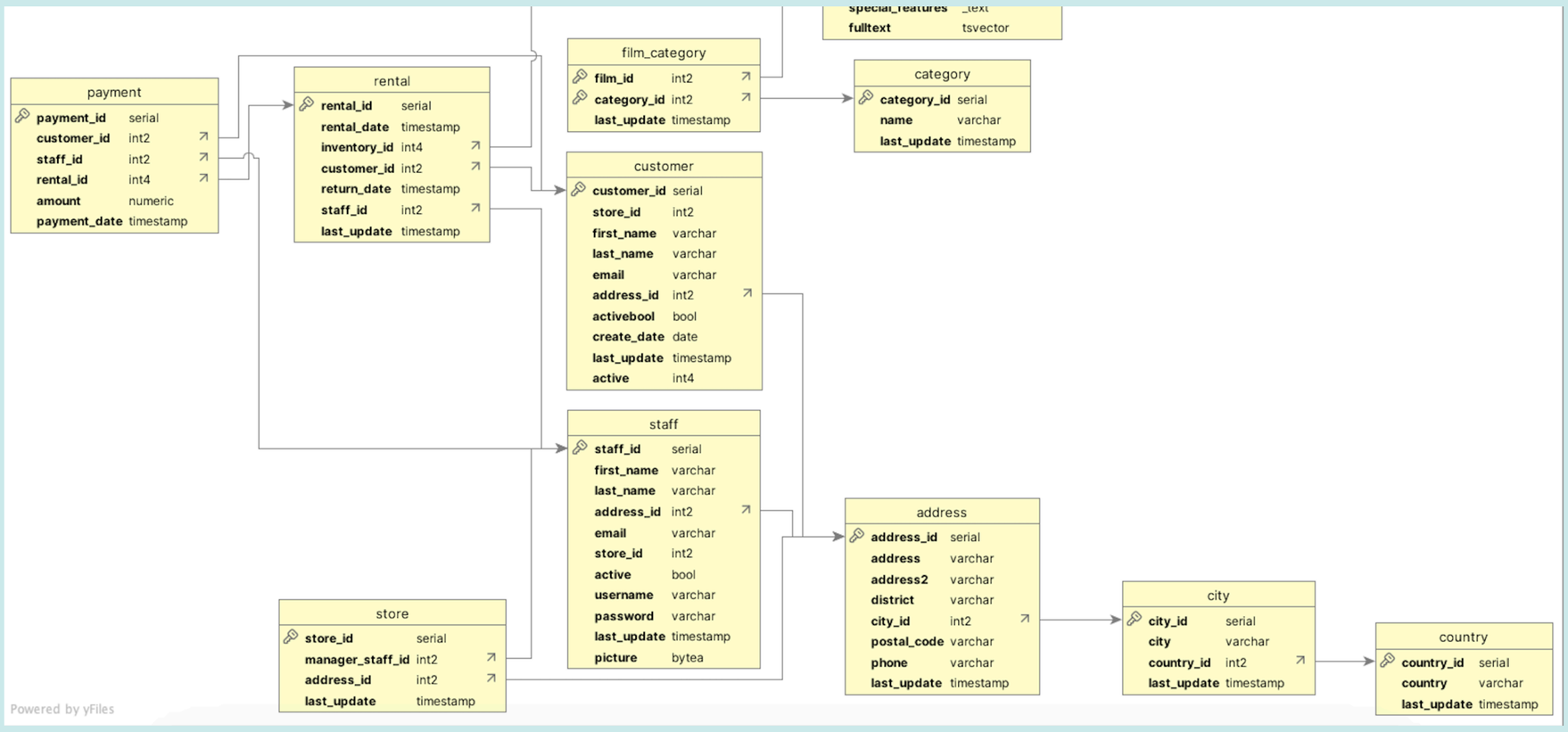
Data Immersion	Achievement III		
	SQL for Data Analysts		
	Task 3.8 By Ola Gaffarova	Table of content Cover	Made for Rockbuster Stealth

PERFORMING SUBQUERIES

Rockbuster Stealth

Step 1: Find the average amount paid by the top 5 customers.



Query Query History

```sql
SELECT AVG(total_amount_paid) as average_amount_paid
FROM
(SELECT
    SUM(p.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 P ON A.customer_id = P.customer_id
WHERE C.city IN (
    SELECT
        C.city
    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
    WHERE D.country IN (
        SELECT
            D.country
        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
        ORDER BY COUNT(A.customer_id) DESC
        LIMIT 10
    )
    GROUP BY D.country, C.city
    ORDER BY COUNT(A.customer_id) DESC
    LIMIT 10
)
GROUP BY A.customer_id, A.first_name, A.last_name, D.country, C.city
ORDER BY total_amount_paid DESC
LIMIT 5) AS average_amount_paid
```

Data Output Messages Explain Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	average_amount_paid numeric
1	105.5540000000000000

Step 2: Find out how many of the top 5 customers you identified in step 1 are based within each country.

Query | Query History

```sql
SELECT
    d.country,  -- the country of the customer
    COUNT(DISTINCT a.customer_id) AS all_customer_count,  -- total number of customers from this country
    -- count how many customers are "top customers" (those who paid more than average)
    COUNT(DISTINCT CASE
        WHEN (
            SELECT SUM(p2.amount)  -- total amount paid by this specific customer
            FROM payment p2
            WHERE p2.customer_id = a.customer_id
        ) > (
            SELECT AVG(total_sum)  -- average total payment across all customers
            FROM (
                SELECT SUM(p.amount) AS total_sum  -- total paid per customer
                FROM payment p
                GROUP BY p.customer_id
            ) AS customer_totals
        )
        THEN a.customer_id  -- count this customer if condition is true
        ELSE NULL  -- otherwise don't count
    END) AS top_customer_count
FROM customer a
JOIN address b ON a.address_id = b.address_id
JOIN city c ON b.city_id = c.city_id
JOIN country d ON c.country_id = d.country_id
GROUP BY d.country  -- group results by country
ORDER BY top_customer_count DESC  -- sort by number of top customers
LIMIT 10;  -- return only top 10 countries
```

Data Output | Messages | Explain | Notifications

Showing rows: 1 to 10 | Page No: 1 of 1

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	26
2	China	53	25
3	United States	36	16
4	Japan	31	14
5	Russian Federation	28	13
6	Brazil	28	12
7	Mexico	30	11
8	Philippines	20	11
9	Taiwan	10	7
10	Turkey	15	7

Step 3:

Steps 1 and 2 — calculating the total amount paid by each customer and comparing it to the average — would be difficult to do without subqueries. We could try using joins and grouping, but it would make the query much more complex and harder to understand. Subqueries let us focus on one part of the problem at a time, like calculating the total payment or the average, and then use those results in the main query.

Subqueries are especially useful when you need to perform a calculation that depends on a subset of data — like averages, maximums, or totals — and then use that result in a filter or comparison. They help keep the main query clean and readable by isolating logic that would otherwise clutter the main SELECT or WHERE clauses.