MySQL SubQuery and CTE - 1

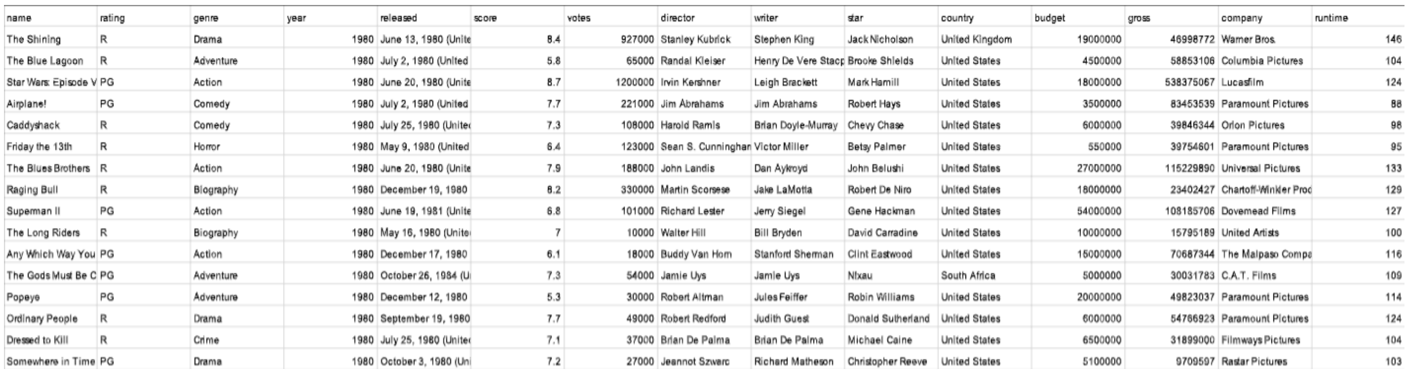
What is a Subquery ?

- In SQL, a subquery is a query within another query. It is a SELECT statement that is nested inside another SELECT, INSERT, UPDATE, or DELETE statement.
- The subquery is executed first, and its result is then used as a parameter or condition for the outer query.

Note - The topic is slightly difficult and needs a lot of practice

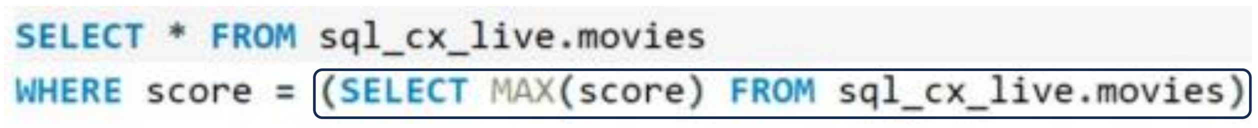
Example - Find the movie with highest rating

In this I'm Using Movies Dataset !



name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime
The Shining	R	Drama	1980	June 13, 1980 (Unite	8.4	927000	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000	46998772	Warner Bros.	146
The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United	5.8	65000	Randal Kleiser	Henry De Vere Stacp	Brooke Shields	United States	4500000	58853106	Columbia Pictures	104
Star Wars: Episode V	PG	Action	1980	June 20, 1980 (Unite	8.7	1200000	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000	538375067	Lucasfilm	124
Airplane!	PG	Comedy	1980	July 2, 1980 (United	7.7	221000	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000	83453539	Paramount Pictures	88
Caddyshack	R	Comedy	1980	July 25, 1980 (Unite	7.3	108000	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000	39846344	Orion Pictures	98
Friday the 13th	R	Horror	1980	May 9, 1980 (United	6.4	123000	Sean S. Cunninghan	Victor Miller	Betsy Palmer	United States	550000	39754601	Paramount Pictures	95
The Blues Brothers	R	Action	1980	June 20, 1980 (Unite	7.9	188000	John Landis	Dan Aykroyd	John Belushi	United States	27000000	115229890	Universal Pictures	133
Raging Bull	R	Biography	1980	December 19, 1980	8.2	330000	Martin Scorsese	Jake LaMotta	Robert De Niro	United States	18000000	23402427	Chartoff-Winkler Proc	129
Superman II	PG	Action	1980	June 19, 1981 (Unite	6.8	101000	Richard Lester	Jerry Siegel	Gene Hackman	United States	54000000	108185706	Dovemead Films	127
The Long Riders	R	Biography	1980	May 16, 1980 (Unite	7	10000	Walter Hill	Bill Bryden	David Carradine	United States	10000000	15795189	United Artists	100
Any Which Way You	PG	Action	1980	December 17, 1980	6.1	18000	Buddy Van Horn	Stanford Sherman	Clint Eastwood	United States	15000000	70687344	The Malpaso Compa	116
The Gods Must Be C	PG	Adventure	1980	October 26, 1984 (U	7.3	54000	Jamie Uys	Jamie Uys	Nixau	South Africa	5000000	30031783	C.A.T. Films	109
Popeye	PG	Adventure	1980	December 12, 1980	5.3	30000	Robert Altman	Jules Feiffer	Robin Williams	United States	20000000	49823037	Paramount Pictures	114
Ordinary People	R	Drama	1980	September 19, 1980	7.7	49000	Robert Redford	Judith Guest	Donald Sutherland	United States	6000000	54766923	Paramount Pictures	124
Dressed to Kill	R	Crime	1980	July 25, 1980 (Unite	7.1	37000	Brian De Palma	Brian De Palma	Michael Caine	United States	6500000	31899000	Filmways Pictures	104
Somewhere in Time	PG	Drama	1980	October 3, 1980 (Uni	7.2	27000	Jeannot Szwarc	Richard Matheson	Christopher Reeve	United States	5100000	9709597	Rastar Pictures	103

Query :



```sql
SELECT * FROM sql_cx_live.movies
WHERE score = (SELECT MAX(score) FROM sql_cx_live.movies)
```

Outer query

Inner Query

@Bhoopendravishwakarma

Note : In the context of running/executing, the inner query is executed first. Then, the outer query fetches information from the result of the inner query and produces the final output.

Types of Subqueries

Based on:

1. The result it returns

2. Based on working

$$\begin{tikzcd}
 & \text{1. Based on the result it return} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Scaler subquery} & \text{Row Subquery} & \text{table subquery}
\end{tikzcd}$$

Scalar Subquery

- **Definition**: Returns a single value (single row and single column).
- **Usage**: Commonly used in WHERE, SELECT, or HAVING clauses.
- **Example**:

```
SELECT Name

FROM employees
```

```
WHERE salary > (SELECT AVG(salary) FROM employees);
```

In this example, the subquery calculates the average salary, which is then compared in the outer query.

Row Subquery

- **Definition**: Returns a single row but multiple columns.
- **Usage**: Typically used with operators like IN, EXISTS, or comparisons involving multiple values.
- **Example**:

```
SELECT *
FROM employees
WHERE (department_id, manager_id) = (SELECT department_id, manager_id
                                     FROM employees
                                     WHERE employee_id = 101);
```

Here, the subquery retrieves a specific department and manager ID for comparison.

Table Subquery

- **Definition**: Returns multiple rows and multiple columns (essentially a virtual table).

- **Usage**: Used in FROM clauses or with operators like IN or EXISTS.
- **Example**:

```
SELECT department_id, COUNT(*)
FROM (SELECT department_id FROM employees WHERE salary > 50000) sub
GROUP BY department_id;
```

2. Based on How It Works

a. Correlated Subquery

- **Definition**: A subquery that depends on the outer query for its values. The subquery is executed repeatedly, once for each row processed by the outer query.
- **Usage**: Used when filtering or calculations are based on the outer query's rows.
- **Example**:

```
SELECT name
FROM olympics o1
WHERE height > (SELECT AVG(height) FROM olympics o2 WHERE o1.sport = o2.sport);
```

b. Non-Correlated Subquery

- **Definition**: A subquery that is independent of the outer query. It is executed only once and its result is used by the outer query.
- **Usage**: Commonly used for static filtering or calculation.
- **Example**:

```
SELECT name
FROM olympics
WHERE height > (SELECT AVG(height) FROM olympics);
```

Where Subqueries Can Be Used

Subqueries are a versatile tool in SQL and can be used in various parts of a query to solve complex problems efficiently. Here's a breakdown of where subqueries can be used:

---

1. In the SELECT Clause

- Subqueries in the SELECT clause are used to compute values for each row in the result set.
- **Example**:

```
SELECT name,
    (SELECT AVG(height) FROM olympics) AS avg_height
FROM olympics;
```

Use Case: Add computed or aggregated values to each row in the result.

---

2. In the FROM Clause

- Subqueries in the FROM clause act as a derived table (temporary table) for the outer query.
- **Example**:

```
SELECT name
FROM (SELECT * FROM olympics WHERE medal = 'Gold') AS gold_medalists;
```

Use Case: Simplifies complex queries by breaking them into manageable parts.

---

3. In the WHERE Clause

- Subqueries in the WHERE clause filter rows based on the results of the subquery.
- **Example**:

```
SELECT name
FROM olympics
WHERE height > (SELECT AVG(height) FROM olympics);
```

Use Case: Filter data based on dynamic criteria.

4. In the HAVING Clause

- Subqueries in the HAVING clause filter aggregated results.
- **Example**:

```
SELECT sport, COUNT(*)

FROM olympics

GROUP BY sport

HAVING COUNT(*) > (SELECT AVG(count) FROM (SELECT
COUNT(*) AS count FROM olympics GROUP BY sport) AS
counts);
```

Use Case: Add dynamic filtering to aggregated data.

---

5. In the JOIN Clause

- Subqueries in JOIN conditions can be used to join derived tables or filtered data sets.
- **Example**:

```
SELECT o1.name, o2.medal

FROM olympics o1

JOIN (SELECT * FROM olympics WHERE medal = 'Gold') o2

ON o1.country = o2.country;
```

Use Case: Combine data from a filtered subquery with the main table.

---

6. In the INSERT Clause

- Subqueries in INSERT statements insert data into a table based on another table's results.
- **Example**:

```
INSERT INTO top_athletes (name, sport)

SELECT name, sport

FROM olympics

WHERE medal = 'Gold';
```

Use Case: Populate one table using data from another.

---

7. In the UPDATE Clause

- Subqueries in UPDATE statements update a table using values derived from another table or condition.
- **Example**:

```
UPDATE olympics

SET weight = (SELECT AVG(weight) FROM olympics WHERE sport = 'Swimming')

WHERE sport = 'Swimming';
```

Use Case: Modify data dynamically based on other table information.

---

8. In the DELETE Clause

- Subqueries in DELETE statements delete rows based on conditions defined in the subquery.
- **Example**:

```
DELETE FROM olympics
WHERE country NOT IN (SELECT country FROM medal_winners);
```

Use Case: Remove specific rows using criteria derived from another table.

Independent Subquery :

Summarized Subqueries (Scalar, Row, Table, and Correlated)

1. Scalar Subqueries

- **Definition**: Returns a single value (one row, one column). Commonly used in SELECT, WHERE, and HAVING clauses.

Examples and Derived Questions

1. **Find the movie with the highest score.**

```
SELECT *
```

FROM sql_live.movies

WHERE score = (SELECT MAX(score) FROM sql_live.movies);

2. **Find the movie with the highest profit.**

SELECT *

FROM movies

WHERE (gross - budget) = (SELECT MAX(gross - budget) FROM movies);

3. **Count movies with a score above the average.**

SELECT COUNT(*)

FROM movies

WHERE score > (SELECT AVG(score) FROM movies);

4. **Find the highest-rated movie of 2000.**

SELECT *

FROM movies

WHERE year = 2000 AND score = (SELECT MAX(score) FROM movies WHERE year = 2000);

5. **Find the highest-rated movie among movies with votes greater than the dataset's average.**

SELECT *

FROM movies

WHERE score = (SELECT MAX(score)

```
            FROM movies
            WHERE votes > (SELECT AVG(votes) FROM movies));
```

---

2. Row Subqueries

- **Definition**: Returns a single row with multiple columns. Used for comparison in conditions like WHERE or NOT IN.

Examples and Derived Questions

1. **Find users who never placed an order.**

```
SELECT *
FROM users
WHERE user_id NOT IN (SELECT DISTINCT user_id FROM
orders);
```

2. **Find the top directors based on gross income.**

```
WITH top_director AS (
   SELECT director
   FROM movies
   GROUP BY director
   ORDER BY SUM(gross) DESC
   LIMIT 3
)
SELECT *
```

```
FROM movies
WHERE director IN (SELECT * FROM top_director);
```

3. **Find actors with an average score greater than 8.5 and more than 25,000 votes.**

```
SELECT *
FROM movies
WHERE star IN (
   SELECT star
   FROM movies
   WHERE votes > 25000
   GROUP BY star
   HAVING AVG(score) > 8.5
);
```

---

3. Table Subqueries

- **Definition**: Returns multiple rows and columns, often used in FROM or IN clauses.

Examples and Derived Questions

1. **Find the highest profitable movie of each year.**

```
SELECT *
FROM movies
```

```
WHERE (year, gross - budget) IN (
   SELECT year, MAX(gross - budget)
   FROM movies
   GROUP BY year
)
ORDER BY (gross - budget) DESC;
```

2. **Find the highest-rated movies of each genre with at least 25,000 votes.**

```
SELECT *
FROM movies
WHERE votes > 25000
  AND (genre, score) IN (
      SELECT genre, MAX(score)
      FROM movies
      WHERE votes > 25000
      GROUP BY genre
  );
```

3. **Find the top 5 actor-director duos based on gross income.**

```
WITH top_duos AS (
   SELECT star, director, MAX(gross)
   FROM movies
   GROUP BY star, director
   ORDER BY SUM(gross) DESC
   LIMIT 5
)
SELECT *
FROM movies
WHERE (star, director, gross) IN (SELECT * FROM top_duos);
```

---

4. Correlated Subqueries

- **Definition**: The inner query depends on the outer query and executes once for each row of the outer query.

Examples and Derived Questions

1. **Find movies with a score above the average score of their genre.**

```
SELECT *
FROM movies m1
WHERE score > (
   SELECT AVG(score)
   FROM movies m2
   WHERE m2.genre = m1.genre
);
```

2. **Find the favorite food of each user based on frequency.**

```
WITH fav_food AS (
   SELECT t2.user_id, t1.name, t4.f_name, COUNT(*) AS freq
   FROM users t1
   JOIN orders t2 ON t1.user_id = t2.user_id
   JOIN order_details t3 ON t2.order_id = t3.order_id
   JOIN food t4 ON t3.f_id = t4.f_id
   GROUP BY t1.name, t2.user_id, t4.f_name
)
SELECT *
FROM fav_food f1
WHERE freq = (
   SELECT MAX(freq)
   FROM fav_food f2
   WHERE f1.user_id = f2.user_id
);
```

Usage Scenarios for Subqueries

1. **In SELECT**: To calculate values like percentages.

```
SELECT name, ROUND(votes / (SELECT SUM(votes) FROM
movies) * 100, 3)

FROM movies;
```

2. **In FROM**: For creating derived tables.

```
SELECT r_name, avg_rating

FROM (

   SELECT r_id, AVG(restaurant_rating) AS avg_rating

   FROM orders

   GROUP BY r_id

) t1

JOIN restaurants t2 ON t1.r_id = t2.r_id;
```

3. **In HAVING**: To filter groups based on aggregated subquery results.

```
SELECT genre, AVG(score)

FROM movies

GROUP BY genre

HAVING AVG(score) > (SELECT AVG(score) FROM movies);
```

→ This breakdown simplifies subqueries into understandable examples and questions, showcasing their practical use in SQL queries.

If this is Helping than connect for more such insights

For datasets & Projects : → Join GitHub