Subquery

Wednesday, October 4, 2023 7:51 PM

What is Subquery?

In SQL, a subquery is a query within another query.
SELECT statement is nested inside CURD.
The inside query can't run individually.

database →

```
use subquery;
SELECT * FROM movies
WHERE score=(SELECT MAX(score) FROM movies)
```

outer query

inner query

Types of Subqueris

Based on:
1. The results it return
   - → Scalar subquery (example) (i 9), horror
   - → Row subquery
   - → Table subquery
2. Based on working
   - Independent
   - correlated

Genres
horror comedy action

Genres	rating
horror	7.8
Comedy	9.4

Select - - - - where - - - = (- - - - -)

dependent

independent

Where can Subqueries be used?

- Insert
- select
  - where
  - select
  - from
  - having
- update
- Delete

Subquery Vs Order by

Order by → It is faster execution
→ it has indexing which make this

(It was [illegible]
much faster than subquery
on sorting.

--INDEPENDENT SUBQUERY - SCALAR SUBQUERY

-Find the highest rated movie among all the movies whose number of votes are >the dataset avg votes

```
SELECT * FROM movies
WHERE score=(SELECT MAX(score) FROM movies
          WHERE votes > (SELECT AVG(votes)
                    FROM movies))
```

--INDEPENDENT SUBQUERY - ROW SUBQUERY(ONE COLUMN MULTI ROWS)

-Find all movies of those actors who filmography avg rating >8.5(take 25000 votes as cutoff)

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

'IN' is used when rows are returned $\longrightarrow$

-- INDEPENDENT SUBQUERY - TABLE SUBQUERY(MULTI COL MULTI ROW)

- Find the highest grossing movies of top 5 actor/director combo in terms of total gross income

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

We can solve above subquery by using Common Table Expression

COMMON TABLE EXPRESSION

```
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)
```

CORRELATED SUBQUERY

- Find all the movies that have a rating higher than the average rating of movies in the same genre

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

USUAGE WITH SELECT

-Display all movie, genre, score and avg(score) of genre

```
SELECT name, genre, score,(SELECT AVG(score) FROM movies m2 where m2.genre = m1.genre)
 FROM movies m1
```

--USUAGE WITH FROM

-Display average rating of all the restaurants

```
 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
```

--USUAGE WITH HAVING

-Find genres having score > avg score of all the movies

```
SELECT genre, AVG(score)
FROM movies
GROUP BY genre
HAVING AVG(score) > (SELECT AVG(score) FROM movies)
```

--SUBQUERY IN INSERT

-Populate a already created loyal_customers table with records of only those customers who have ordered more than 3 times

```
INSERT INTO loyal_users
(user_id,name)
SELECT t1.user_id,name
FROM orders t1
JOIN users t2 ON t1.user_id = t2.user_id
GROUP BY user_id
HAVING COUNT(*) > 3
```

SUBQUERY IN UPDATE

-Populate the money col of loyal_customer table using the orders table. Provide a 10% app money to all the customers based on their order value

```
UPDATE loyal_users
SET money = (SELECT SUM(amount)*0.1
                  FROM orders
          WHERE orders.user_id = loyal_users.user_id)
```

SUBQUERY IN DELETE

-DELETE all the customer record who have never ordered.

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

Author : Nabin Adhikari