Ex. No: 05	SUB QUERIES AND CORRELATED SUB QUERIES
Date	13.02.2024

Objective:

To execute the given queries using sub queries and correlated sub queries.

Description:

Sub Queries:

A subquery, also known as an inner query or nested query, is a query that is embedded within another query. It allows you to retrieve data from one table or set of tables based on the results of another query. The subquery is executed first, and its results are then used by the outer query to perform further operations.

Subqueries can be used in various scenarios, such as filtering, joining, or aggregating data. They provide a powerful way to write complex queries and achieve more precise and targeted results. With subqueries, you can break down a problem into smaller, more manageable parts and solve them step by step.

Example of a subquery:

```
SELECT name FROM Customers
WHERE customer_id IN (SELECT customer_id FROM Orders WHERE YEAR(order_date) = 2023)
```

Correlated Subqueries:

A correlated subquery is a special type of subquery where the inner query references the outer query. In other words, the inner query depends on the outer query for its values. This correlation allows the subquery to be evaluated once for each row processed by the outer query. Correlated subqueries are useful when you need to perform calculations or filtering based on values from the outer query. They enable you to compare values between the inner and outer queries and make decisions based on the results.

One common use case for correlated subqueries is finding records that meet specific conditions relative to other records in the same table.

For example, you might use a correlated subquery to find all employees whose salary is higher than the average salary in their department.

Correlated subqueries can be a powerful tool, but they can also impact performance if not used carefully. The database engine needs to execute the subquery multiple times, which can lead to increased resource usage. It's important to optimize and evaluate the performance implications when using correlated subqueries in your queries.

Example of a correlated subquery:

```
SELECT product_id, priceFROM Products p1
WHERE price > (SELECT AVG(price) FROM Products p2)
```

Questions:

Sub Query

1. List all users who have made reservations for events that are taking place in a specific venue
(e.g.,”USA”).

```
SQL> SELECT UserID, Name
  2  FROM user_41
  3  WHERE UserID IN (
  4      SELECT UserID  FROM ticket_41 WHERE EventID IN
  5      (
  6          SELECT EventID
  7          FROM event_41
  8          WHERE VenueID IN (
  9              SELECT VenueID
 10              FROM venue_41
 11              WHERE Country = 'USA'
 12          )
 13      )
 14  );

    USERID NAME
---------- ------------------------------
         1 John Smith
         2 Jane Doe
         3 Michael Lee
         4 Sarah Adams
         5 David Wang
         6 Emily Chen
         7 Alex Kim
         8 Lisa Lopez

8 rows selected.
```

2. Find the events with the highest ticket prices.

```
SQL> SELECT EventID, Name
  2  FROM event_41
  3  WHERE EventID IN (
  4      SELECT EventID FROM ticket_41 WHERE Price = (
  5          SELECT MAX(Price)
  6          FROM ticket_41
  7      )
  8  );

   EVENTID NAME
---------- ------------------------------
         1 Concert in Park
```

3. Find the total number of tickets reserved for a specific event.

```
SQL> SELECT e.EventID, e.Name, (
  2      SELECT COUNT(TicketID)
  3      FROM ticket_41 t
  4      WHERE t.EventID = e.EventID
  5  ) AS TotalTicketsReserved
  6  FROM event_41 e;

   EVENTID NAME                           TOTALTICKETSRESERVED
---------- ------------------------------ --------------------
         7 Tech Conference                                   0
         1 Concert in Park                                   2
         2 Movie Night                                       2
         3 Sports Tournament                                 2
         4 Art Exhibition                                    2
         5 Food Festival                                     0
         6 Comedy Show                                       0
         8 Dance Workshop                                    0

8 rows selected.
```

4. List the users who have made reservations with a total cost exceeding a certain amount (e.g.,50)

```
SQL> SELECT u.UserID, u.Name
  2  FROM user_41 u
  3  WHERE u.UserID IN (
  4      SELECT t.UserID
  5      FROM ticket_41 t
  6      GROUP BY t.UserID
  7      HAVING SUM(t.Price) > 20
  8  );

    USERID NAME
---------- ------------------------------
         1 John Smith
         2 Jane Doe
```

5. Retrieve the events where the number of reservations exceeds a certain threshold.

```
SQL> SELECT e.EventID, e.Name
  2  FROM event_41  e
  3  WHERE e.EventID IN (
  4      SELECT t.EventID
  5      FROM ticket_41  t
  6      GROUP BY t.EventID
  7      HAVING COUNT(t.TicketID) > 1
  8  );

   EVENTID NAME
---------- ------------------------------
         1 Concert in Park
         2 Movie Night
         4 Art Exhibition
         3 Sports Tournament
```

6. Find all users who have made more reservations than the average number of reservations across all users.

```
SQL> SELECT u.UserID, u.Name
  2  FROM user_41  u
  3  WHERE u.UserID IN (
  4      SELECT t.UserID FROM ticket_41  t GROUP BY t.UserID
  5      HAVING COUNT(t.TicketID) > (
  6          SELECT AVG(reservation_count)
  7          FROM (
  8              SELECT COUNT(TicketID)  reservation_count
  9              FROM ticket_41
 10              GROUP BY UserID
 11          )  avg_reservations
 12      )
 13  );

no rows selected
```

Correlated Sub Query

7. List all events where the total ticket price of reservations exceeds a certain amount.

```
SQL> SELECT e.EventID, e.Name
  2  FROM event_41 e
  3  WHERE (
  4    SELECT SUM(t.Price)
  5    FROM ticket_41 t
  6    WHERE t.EventID = e.EventID
  7  ) > 40;

   EVENTID NAME
---------- ------------------------------
         1 Concert in Park
```

8. Find the users who have made reservations for more than one event.

```
SQL> SELECT UserID, (
  2    SELECT COUNT(DISTINCT EventID)
  3    FROM ticket_41 t2
  4    WHERE t2.UserID = t1.UserID
  5  ) AS NumEvents
  6  FROM ticket_41 t1
  7  GROUP BY UserID
  8  HAVING (
  9    SELECT COUNT(DISTINCT EventID)
 10    FROM ticket_41 t2
 11    WHERE t2.UserID = t1.UserID
 12  ) > 1;

no rows selected
```

9. Retrieve the events with the highest number of reservations.

```
SQL> SELECT e.EventID, e.Name
  2  FROM event_41 e
  3  WHERE (
  4    SELECT COUNT(*)
  5    FROM ticket_41 t
  6    WHERE t.EventID = e.EventID
  7  ) = (
  8    SELECT MAX(CountReservations)
  9    FROM (
 10      SELECT COUNT(*) AS CountReservations
 11      FROM ticket_41
 12      GROUP BY EventID
 13    )
 14  );

   EVENTID NAME
---------- ------------------------------
         1 Concert in Park
         2 Movie Night
         3 Sports Tournament
         4 Art Exhibition
```

10. For each event, find the number of reservations made by users

```
SQL> SELECT e.EventID, e.Name,
  2    (SELECT COUNT(*)
  3     FROM ticket_41 t
  4     WHERE t.EventID = e.EventID) AS ReservationCount
  5  FROM event_41 e;

   EVENTID NAME                           RESERVATIONCOUNT
---------- ------------------------------ ----------------
         7 Tech Conference                               0
         1 Concert in Park                               2
         2 Movie Night                                   2
         3 Sports Tournament                             2
         4 Art Exhibition                                2
         5 Food Festival                                 0
         6 Comedy Show                                   0
         8 Dance Workshop                                0

8 rows selected.
```

11. Find the events for which the total ticket price of reservations exceeds the average total ticket price for all events.

```
SQL> SELECT e.EventID, e.Name
  2  FROM event_41 e
  3  WHERE (
  4    SELECT SUM(t.Price)
  5    FROM ticket_41 t
  6    WHERE t.EventID = e.EventID
  7  ) > (
  8    SELECT AVG(TotalPrice)
  9    FROM (
 10      SELECT EventID, SUM(Price) AS TotalPrice
 11      FROM ticket_41
 12      GROUP BY EventID
 13    )
 14  );

   EVENTID NAME
---------- ------------------------------
         1 Concert in Park
         2 Movie Night
```

12. List users who have made reservations for multiple events on the same day.

```
SQL> SELECT u.UserID, u.Name, COUNT(t.EventID) AS ReservationCount
  2  FROM User_41 u
  3  JOIN Ticket_41 t ON u.UserID = t.UserID
  4  JOIN Event_41 e ON t.EventID = e.EventID
  5  WHERE EXISTS (
  6      SELECT 1
  7      FROM Ticket_41 t2
  8      JOIN event_41 e2 ON t2.EventID = e2.EventID
  9      WHERE u.UserID = t2.UserID
 10      AND e.Dates = e2.Dates
 11      AND t.EventID <> t2.EventID
 12  )
 13  GROUP BY u.UserID, u.Name
 14  HAVING COUNT(t.EventID) > 1;

no rows selected
```

Result:

The given queries using sub queries and correlated sub queries were executed.