|  |  |
| --- | --- |
| **Name** | Sahil Shah |
| **UID No.** | 2021300115 |
| **Experiment No.** | 6 |

|  |  |
| --- | --- |
| **AIM:** | To perform subqueries in MySQL. |
| **Program 1** | |
| **PROBLEM STATEMENT:** | Write subqueries on the tables in the database on MySQL. |
| **THEORY:** | **What is an SQL subquery?**  A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.  A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.  Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.  There are a few rules that subqueries must follow −   * Subqueries must be enclosed within parentheses. * A subquery can have only one column in the SELECT clause unless multiple columns are in the main query for the subquery to compare its selected columns. * An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery. * Subqueries that return more than one row can only be used with multiple value operators such as the IN operator. * The SELECT list cannot include any references to values that evaluate a BLOB, ARRAY, CLOB, or NCLOB. * A subquery cannot be immediately enclosed in a set function. * The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.   **Subqueries with the SELECT Statement**  Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows –  SELECT column\_name [, column\_name ]  FROM table1 [, table2 ]  WHERE column\_name OPERATOR  (SELECT column\_name [, column\_name ]  FROM table1 [, table2 ]  [WHERE]) |
| **QUERIES:**  **Using Create, Insert Into, Select Commands:**    **Table Room**    **Using Create, Insert Into, Select Commands:**    **Table Customers**     1. Display all the items from the table Room where the room available is the minimum price for all room numbers greater than 300:   SELECT \*  FROM Room  WHERE RoomAvailability = (  SELECT DISTINCT RoomAvailability  FROM Room  WHERE RoomPrice = (  SELECT MIN(RoomPrice)  FROM Room  WHERE RoomNumber > 300  )  );     1. Display all the items from the table Room where the price of the room is either minimum or maximum:   SELECT \*  FROM Room  WHERE RoomPrice = (  SELECT MIN(RoomPrice) FROM Room  ) OR RoomPrice = (  SELECT MAX(RoomPrice) FROM Room  );     1. Display all the items from the table Room where the room number is of the customer who has the minimum contact number and has their address starting from ‘S’:   SELECT \*  FROM Room  WHERE RoomNumber = (  SELECT RoomNumber  FROM Customers  WHERE Contact = (  SELECT MIN(Contact)  FROM Customers  WHERE Address LIKE 'S%'  )  );     1. Display the room type, room price, and room availability from the table Room where the room number is the one of the customers from Delhi:   SELECT RoomType, RoomPrice, RoomAvailability  FROM Room  WHERE RoomNumber IN (  SELECT RoomNumber  FROM Customers  WHERE Address IN (  SELECT Address  FROM Customers  WHERE Address = 'Delhi'  )  );     1. Display all the items from the table Room where the room price is greater than the minimum room price:   SELECT \* FROM Room  WHERE RoomPrice > (  SELECT MIN(RoomPrice)  FROM Room  );     1. Display all the items from the table Room where the room number is greater than 300 with no availability:   SELECT \* FROM Room  WHERE RoomNumber = (  SELECT RoomNumber  FROM Room  WHERE RoomAvailability = 'NO' AND RoomNumber > 200  );     1. Display the customer name, date of birth, and Aadhar number from the table Customers whose room number is the one with a price greater than 4500:   SELECT CustomerName, DOB, Aadhar  FROM Customers  WHERE RoomNumber = (  SELECT RoomNumber  FROM Room  WHERE RoomPrice > 4500  );     1. Display the room number, size, availability, and type from the table Room whose price is greater than the average price of the rooms:   SELECT RoomNumber, RoomSize, RoomAvailability, RoomType  FROM Room  WHERE RoomPrice >= (  SELECT AVG(RoomPrice)  FROM Room  );     1. Display all the items from the table Room whose room number is the one with the capacity of two persons or three persons:   SELECT \* FROM Room  WHERE RoomNumber IN (  SELECT RoomNumber  FROM Room  WHERE RoomSize = '2 persons' OR RoomSize = '3 persons'  )  ORDER BY RoomAvailability;     1. Display the room number, type, size, and price from the table Room where the room number is the one where customers have their contact numbers greater than 777777777:   SELECT RoomNumber, RoomType, RoomSize, RoomPrice  FROM Room  WHERE RoomNumber IN (  SELECT RoomNumber  FROM Customers  WHERE Contact > 777777777  );     1. Display all the items from the table Room where the room size is the one with a price range between the average and the maximum prices of the rooms:   SELECT \* FROM Room  WHERE RoomSize IN (  SELECT RoomSize  FROM Room  HAVING RoomPrice >= AVG(RoomPrice) AND RoomPrice <= MAX(RoomPrice)  );     1. Update the address of the customers who have the room number as the one with Deluxe room type and of capacity 3 persons in the table Customers:   UPDATE Customers  SET Address = 'Delhi'  WHERE RoomNumber = (  SELECT RoomNumber  FROM Room  WHERE RoomType = 'Deluxe' AND RoomSize = '3 persons'  ); | |
| **CONCLUSION:**  In this experiment, I learned about subqueries and nested subqueries in SQL – their definitions and their generic syntaxes. I used the aggregate functions which I learned in the previous experiment in most of the subqueries. I also made a subquery with the update function. Subqueries help to divide the complex query into isolated parts so that a complex query can be broken down into a series of logical steps. | |