

| | |
|-----------------------|----------------|
| Name | Adwait S Purao |
| UID no. | 2021300101 |
| Experiment No. | 6 |

| | |
|----------------------------|---|
| AIM: | To perform Sub-Queries in MySQL |
| Program 1 | |
| PROBLEM STATEMENT : | . Perform various subqueries on the Hotel database |
| Theory : | <p>What is subquery in SQL? A subquery is a SQL query nested inside a larger query.</p> <p>A subquery may occur in :</p> <ul style="list-style-type: none"> - A SELECT clause - A FROM clause - A WHERE clause <p>The subquery can be nested inside a SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery.</p> <p>A subquery is usually added within the WHERE Clause of another SQL SELECT statement.</p> <p>You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.</p> <p>A subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select.</p> <p>The inner query executes first before its parent query so that the results of an inner query can be passed to the outer query.</p> <p>You can use a subquery in a SELECT, INSERT, DELETE, or UPDATE statement to perform the following tasks:</p> <p>Compare an expression to the result of the query. Determine if an expression is included in the results of the query. Check whether the query selects any rows.</p> <p>Syntax:</p> |

```
SELECT    select_list
FROM      table
WHERE     expr operator
```

```
(SELECT    select_list
FROM      table);
```

The subquery (inner query) executes once before the main query (outer query) executes.

The main query (outer query) use the subquery result.

Queries

Subquery 1:

Statement:

Selects customer with minimum age

Code:

```
use hotel;
select * from customer
where c_age=(
select min(c_age)
from customer
);
```

Original table:

| | C_Id | C_Name | Reservation_no | C_Age | C_Address | C_contact | C_cin_time | C_cout_t | r_no |
|---|------|------------------|----------------|-------|-----------|-----------|------------|----------|------|
| ▶ | 1234 | Ramesh Verma | 1 | 34 | Dharavi | 123456 | 12:56:23 | 16:56:23 | 12 |
| | 1235 | Ram Sharma | 2 | 44 | Ghansoli | 123457 | 13:54:43 | 19:26:13 | 13 |
| | 1236 | Sachin Tendulkar | 3 | 50 | Colaba | 123458 | 11:24:41 | 20:55:53 | 14 |
| | 1237 | Virat Kohli | 4 | 30 | Dadar | 123459 | 22:21:45 | 16:25:33 | 15 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

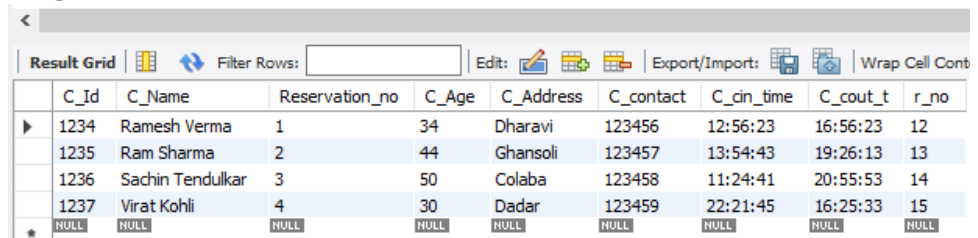
Output:

| | C_Id | C_Name | Reservation_no | C_Age | C_Address | C_contact | C_cin_time | C_cout_t | r_no |
|---|------|-------------|----------------|-------|-----------|-----------|------------|----------|------|
| ▶ | 1237 | Virat Kohli | 4 | 30 | Dadar | 123459 | 22:21:45 | 16:25:33 | 15 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

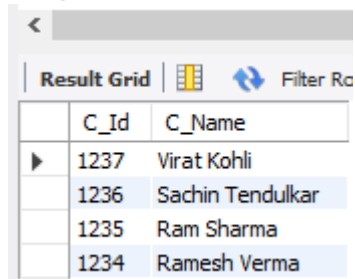
Subquery 2:SQL Subquery and Join

Code:

```
use hotel;
select distinct customer.C_Id,customer.C_Name
from customer
inner join reservation
on customer.Reservation_no=reservation.Reservation_no
order by customer.C_Id desc
```

Original table:

| | C_Id | C_Name | Reservation_no | C_Age | C_Address | C_contact | C_cin_time | C_cout_t | r_no |
|---|------|------------------|----------------|-------|-----------|-----------|------------|----------|------|
| ▶ | 1234 | Ramesh Verma | 1 | 34 | Dharavi | 123456 | 12:56:23 | 16:56:23 | 12 |
| | 1235 | Ram Sharma | 2 | 44 | Ghansoli | 123457 | 13:54:43 | 19:26:13 | 13 |
| | 1236 | Sachin Tendulkar | 3 | 50 | Colaba | 123458 | 11:24:41 | 20:55:53 | 14 |
| | 1237 | Virat Kohli | 4 | 30 | Dadar | 123459 | 22:21:45 | 16:25:33 | 15 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Output:

| | C_Id | C_Name |
|---|------|------------------|
| ▶ | 1237 | Virat Kohli |
| | 1236 | Sachin Tendulkar |
| | 1235 | Ram Sharma |
| | 1234 | Ramesh Verma |

Subquery 3**Statement:**

Selects customer name and customer id with customer name starting with Ram

Code:

```
use hotel;
select c_name as Customer_Name,c_id as Customer_Id,c_contact
from customer
where c_id in(
select c_id from customer
where c_name like 'Ram%');
```

Original table:

| | C_Id | C_Name | Reservation_no | C_Age | C_Address | C_contact | C_cin_time | C_cout_t | r_no |
|---|------|------------------|----------------|-------|-----------|-----------|------------|----------|------|
| ▶ | 1234 | Ramesh Verma | 1 | 34 | Dharavi | 123456 | 12:56:23 | 16:56:23 | 12 |
| | 1235 | Ram Sharma | 2 | 44 | Ghansoli | 123457 | 13:54:43 | 19:26:13 | 13 |
| | 1236 | Sachin Tendulkar | 3 | 50 | Colaba | 123458 | 11:24:41 | 20:55:53 | 14 |
| | 1237 | Virat Kohli | 4 | 30 | Dadar | 123459 | 22:21:45 | 16:25:33 | 15 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Output:

| | Customer_Name | Customer_Id | c_contact |
|---|---------------|-------------|-----------|
| ▶ | Ramesh Verma | 1234 | 123456 |
| | Ram Sharma | 1235 | 123457 |

Subquery 4

Statement:

Selects customers with age greater than average age

Code:

use hotel;

select c_name as Customer_Name,c_id as Customer_Id,c_age as

Customer_age

from customer

where c_age>(

select avg(c_age)


from customer

);

Original table:

| | C_Id | C_Name | Reservation_no | C_Age | C_Address | C_contact | C_cin_time | C_cout_t | r_no |
|---|------|------------------|----------------|-------|-----------|-----------|------------|----------|------|
| ▶ | 1234 | Ramesh Verma | 1 | 34 | Dharavi | 123456 | 12:56:23 | 16:56:23 | 12 |
| | 1235 | Ram Sharma | 2 | 44 | Ghansoli | 123457 | 13:54:43 | 19:26:13 | 13 |
| | 1236 | Sachin Tendulkar | 3 | 50 | Colaba | 123458 | 11:24:41 | 20:55:53 | 14 |
| | 1237 | Virat Kohli | 4 | 30 | Dadar | 123459 | 22:21:45 | 16:25:33 | 15 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Output:

| < Result Grid  Filter Rows: <input type="text"/> | | | |
|---|------------------|-------------|--------------|
| | Customer_Name | Customer_Id | Customer_age |
| ▶ | Ram Sharma | 1235 | 44 |
| | Sachin Tendulkar | 1236 | 50 |

Subquery 5

Statement:

Select employees with salary greater than average salary

Code:

use hotel;

select E_Type,E_Name as Name_of_Employee,E_Salary as Salary

from employee








where E_Salary>(

select avg(E_Salary)


from employee

);

Original table:

| < Result Grid  Filter Rows: <input type="text"/> Edit:    Export/Import:   Wrap Cell Content:  | | | | | | | | | | |
|--|----------------|-----------|------|------|----------|-----------|---------------|-------------|-----------|----------|
| | E_Name | E_Type | E_ID | H_ID | LastName | FirstName | Address | City | E_Contact | E_Salary |
| ▶ | Adwait Purao | Permanent | 1 | 1234 | Purao | Adwait | Kurla | Mumbai | 12345 | 10000 |
| | Akshay Kumar | Temporary | 3 | 3456 | Kumar | Akshay | Ramgad | Bihar | 12347 | 30000 |
| | Ranbir Kapoor | Permanent | 4 | 2345 | Kapoor | Ranbir | Roopnagar | Agra | 12348 | 40000 |
| | Angelina Jolie | Permanent | 5 | 8970 | Jolie | Angelina | Beverly Hills | Los Angeles | 12349 | 50000 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Output:

| < Result Grid  Filter Rows: <input type="text"/> | | | |
|---|-----------|------------------|--------|
| | E_Type | Name_of_Employee | Salary |
| ▶ | Permanent | Ranbir Kapoor | 40000 |
| | Permanent | Angelina Jolie | 50000 |

Update Statement

use hotel;

update employee set city='Mumbai' where e_id in (3,4);

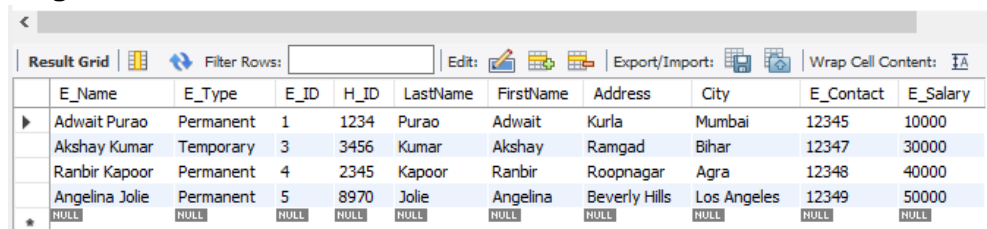
Subquery 6

Statement:

Selects employees who live in Mumbai

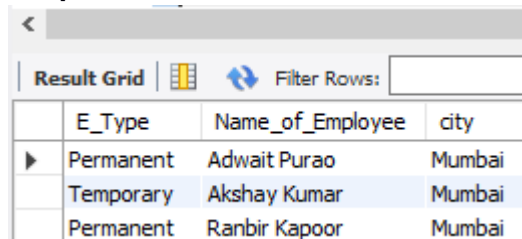
Code:

```
use hotel;
select E_Type,E_Name as Name_of_Employee,city
from employee
where city in(
select city
from employee
where city like 'Mum%'
);
```

Original table:

A screenshot of a database grid interface. At the top, there's a toolbar with icons for 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. Below the toolbar is a table with 10 columns: E_Name, E_Type, E_ID, H_ID, LastName, FirstName, Address, City, E_Contact, and E_Salary. The table contains 5 rows of data. The first row is Adwait Purao, Permanent, 1, 1234, Purao, Adwait, Kurla, Mumbai, 12345, 10000. The second row is Akshay Kumar, Temporary, 3, 3456, Kumar, Akshay, Ramgad, Bihar, 12347, 30000. The third row is Ranbir Kapoor, Permanent, 4, 2345, Kapoor, Ranbir, Roopnagar, Agra, 12348, 40000. The fourth row is Angelina Jolie, Permanent, 5, 8970, Jolie, Angelina, Beverly Hills, Los Angeles, 12349, 50000. The fifth row is a summary row with NULL values for all columns except E_ID, H_ID, LastName, and FirstName, which are also NULL.

| | E_Name | E_Type | E_ID | H_ID | LastName | FirstName | Address | City | E_Contact | E_Salary |
|---|----------------|-----------|------|------|----------|-----------|---------------|-------------|-----------|----------|
| ▶ | Adwait Purao | Permanent | 1 | 1234 | Purao | Adwait | Kurla | Mumbai | 12345 | 10000 |
| | Akshay Kumar | Temporary | 3 | 3456 | Kumar | Akshay | Ramgad | Bihar | 12347 | 30000 |
| | Ranbir Kapoor | Permanent | 4 | 2345 | Kapoor | Ranbir | Roopnagar | Agra | 12348 | 40000 |
| | Angelina Jolie | Permanent | 5 | 8970 | Jolie | Angelina | Beverly Hills | Los Angeles | 12349 | 50000 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Output:

A screenshot of a database grid interface showing the output of a query. The toolbar is the same as the previous screenshot. The table has 4 columns: E_Type, Name_of_Employee, and city. There are 3 rows of data. The first row is Permanent, Adwait Purao, Mumbai. The second row is Temporary, Akshay Kumar, Mumbai. The third row is Permanent, Ranbir Kapoor, Mumbai.

| | E_Type | Name_of_Employee | city |
|---|-----------|------------------|--------|
| ▶ | Permanent | Adwait Purao | Mumbai |
| | Temporary | Akshay Kumar | Mumbai |
| | Permanent | Ranbir Kapoor | Mumbai |

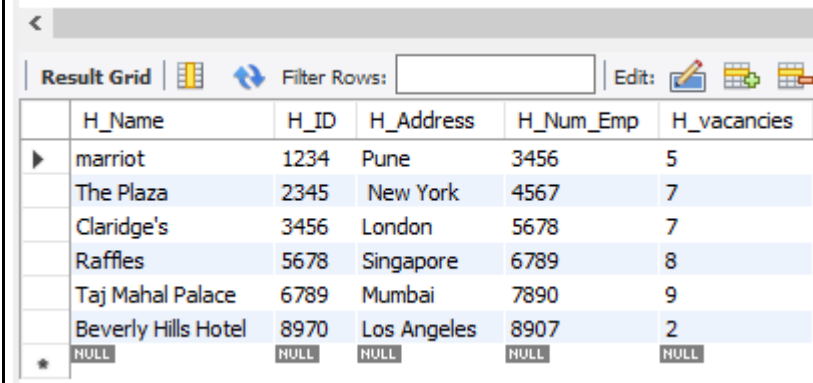
Subquery 7

Statement:

Select hotels with with employees greater than average number of employees

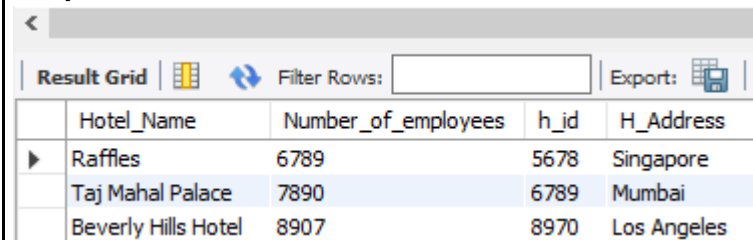
Code:

```
use hotel;
select h_name as Hotel_Name,h_num_emp as
Number_of_employees,h_id,H_Address
from hotel_info
where h_num_emp >(
select avg(h_num_emp)
from hotel_info
);
```

Original table:

A screenshot of a database grid interface. At the top, there is a toolbar with a 'Result Grid' button, a 'Filter Rows' input field, and an 'Edit' button. Below the toolbar is a table with the following columns: H_Name, H_ID, H_Address, H_Num_Emp, and H_vacancies. The table contains six rows of data, with the last row having NULL values for all columns except H_Name.

| | H_Name | H_ID | H_Address | H_Num_Emp | H_vacancies |
|---|---------------------|------|-------------|-----------|-------------|
| ▶ | marriot | 1234 | Pune | 3456 | 5 |
| | The Plaza | 2345 | New York | 4567 | 7 |
| | Claridge's | 3456 | London | 5678 | 7 |
| | Raffles | 5678 | Singapore | 6789 | 8 |
| | Taj Mahal Palace | 6789 | Mumbai | 7890 | 9 |
| | Beverly Hills Hotel | 8970 | Los Angeles | 8907 | 2 |
| ★ | NULL | NULL | NULL | NULL | NULL |

Output:

A screenshot of a database grid interface showing the output of a query. The toolbar includes a 'Result Grid' button, a 'Filter Rows' input field, and an 'Export' button. The table has the following columns: Hotel_Name, Number_of_employees, h_id, and H_Address. It contains three rows of data, all of which are highlighted in blue.

| | Hotel_Name | Number_of_employees | h_id | H_Address |
|---|---------------------|---------------------|------|-------------|
| ▶ | Raffles | 6789 | 5678 | Singapore |
| | Taj Mahal Palace | 7890 | 6789 | Mumbai |
| | Beverly Hills Hotel | 8907 | 8970 | Los Angeles |

Subquery 8**Statement:**

Select hotel name with name starting with ' T '

Code:

```
use hotel;
select h_name as Hotel_Name,h_id,H_Address
from hotel_info
where H_Name in(
select H_Name
from hotel_info
where H_Name like 'T%'
);
```

Original table:

| | H_Name | H_ID | H_Address | H_Num_Emp | H_vacancies |
|---|---------------------|------|-------------|-----------|-------------|
| ▶ | marriot | 1234 | Pune | 3456 | 5 |
| | The Plaza | 2345 | New York | 4567 | 7 |
| | Claridge's | 3456 | London | 5678 | 7 |
| | Raffles | 5678 | Singapore | 6789 | 8 |
| | Taj Mahal Palace | 6789 | Mumbai | 7890 | 9 |
| | Beverly Hills Hotel | 8970 | Los Angeles | 8907 | 2 |
| ★ | NULL | NULL | NULL | NULL | NULL |

Output:

| | Hotel_Name | h_id | H_Address |
|---|------------------|------|-----------|
| ▶ | The Plaza | 2345 | New York |
| | Taj Mahal Palace | 6789 | Mumbai |

Subquery 9

Statement:

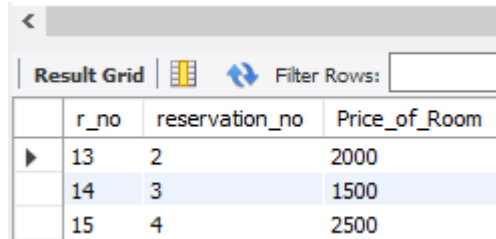
Select price of room with price greater than minimum amount

Code:

```
use hotel;
select r_no,reservation_no,Amount as Price_of_Room
from reservation
where amount>(
select min(amount)
from reservation
);
```

Original table:

| | Reservation_no | R_intime | R_outtime | Amount | R_no | C_ID |
|---|----------------|----------|-----------|--------|------|------|
| ▶ | 1 | 12:56:23 | 16:56:23 | 1000 | 12 | 1234 |
| | 2 | 13:54:43 | 19:26:13 | 2000 | 13 | 1235 |
| | 3 | 11:24:41 | 20:55:53 | 1500 | 14 | 1236 |
| | 4 | 22:21:45 | 16:25:33 | 2500 | 15 | 1237 |
| ★ | NULL | NULL | NULL | NULL | NULL | NULL |

Output:

The screenshot shows a database interface with a 'Result Grid' tab. It contains a table with three columns: 'r_no', 'reservation_no', and 'Price_of_Room'. The data is as follows:

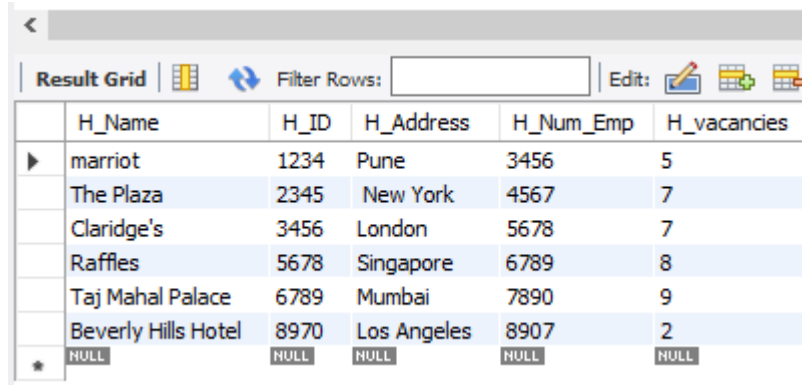
| r_no | reservation_no | Price_of_Room |
|------|----------------|---------------|
| 13 | 2 | 2000 |
| 14 | 3 | 1500 |
| 15 | 4 | 2500 |

Subquery 10**Statement:**

Select hotels with vacancies lesser than average vacancies

Code:

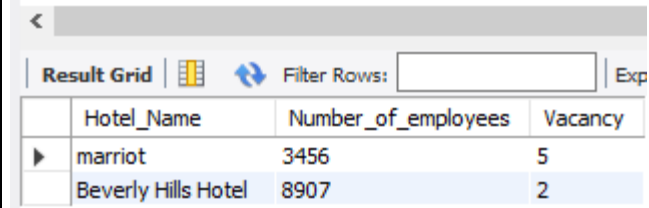
```
use hotel;
select h_name as Hotel_Name,h_num_emp as
Number_of_employees,H_vacancies as Vacancy
from hotel_info
where H_vacancies<(
select avg(H_vacancies)
from hotel_info
)
order by vacancy desc;
```

Original table:

The screenshot shows a database interface with a 'Result Grid' tab. It contains a table with six columns: 'H_Name', 'H_ID', 'H_Address', 'H_Num_Emp', and 'H_vacancies'. The data is as follows:

| H_Name | H_ID | H_Address | H_Num_Emp | H_vacancies |
|---------------------|------|-------------|-----------|-------------|
| marriot | 1234 | Pune | 3456 | 5 |
| The Plaza | 2345 | New York | 4567 | 7 |
| Claridge's | 3456 | London | 5678 | 7 |
| Raffles | 5678 | Singapore | 6789 | 8 |
| Taj Mahal Palace | 6789 | Mumbai | 7890 | 9 |
| Beverly Hills Hotel | 8970 | Los Angeles | 8907 | 2 |
| * | NULL | NULL | NULL | NULL |

Output:



The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with a back arrow, a 'Result Grid' button, a grid icon, a refresh icon, a 'Filter Rows:' text box, and an 'Exp' button. Below the toolbar is a table with the following data:

| | Hotel_Name | Number_of_employees | Vacancy |
|---|---------------------|---------------------|---------|
| ▶ | marriot | 3456 | 5 |
| | Beverly Hills Hotel | 8907 | 2 |

Conclusion: We learned and implemented subqueries in MySQL workbench. We saw how we can get any desired output by making use of subqueries. We especially saw how useful complex nested subqueries were in retrieving the results. We got our output by making use of subqueries and nesting them.