

DBMS LAB ASSIGNMENT – 5

NAME: DATLA TARUN ANJANEYA VARMA

REG NO: 19BCS036

GROUP NO: 2

DATABASE NAME: HOTEL

Q1) Illustrate logical ANY, ALL and LIKE operator- the queries should be relevant to your respective databases 3 queries for each operator. One query explaining the difference between ANY and ALL

QUERIES FOR “ANY”

The screenshot displays the SQL Server Enterprise Manager interface with a query window open. The query window contains three SQL queries using the ANY operator. The first query selects customers whose ID is less than or equal to any customer ID in the T2_Rooms table. The second query selects rooms where the number of beds is less than any number of guests in the T2_Reservation table. The third query selects services where the service ID is greater than or equal to any employee ID in the emp_info table where age is greater than 25. The Results pane shows the output of these queries, including customer details, room details, and service details.

```
USE HOTEL;
SELECT * FROM T2_Customer WHERE Customer_ID <= ANY(SELECT Customer_ID FROM T2_Rooms WHERE Customer_ID < 3)
SELECT * FROM T2_Rooms WHERE number_of_beds < ANY(SELECT Number_of_guests FROM T2_Reservation);
SELECT Service_name,Service_cost FROM T2_SERVICES WHERE Service_ID >= ANY(SELECT empid FROM emp_info WHERE age > 25);
```

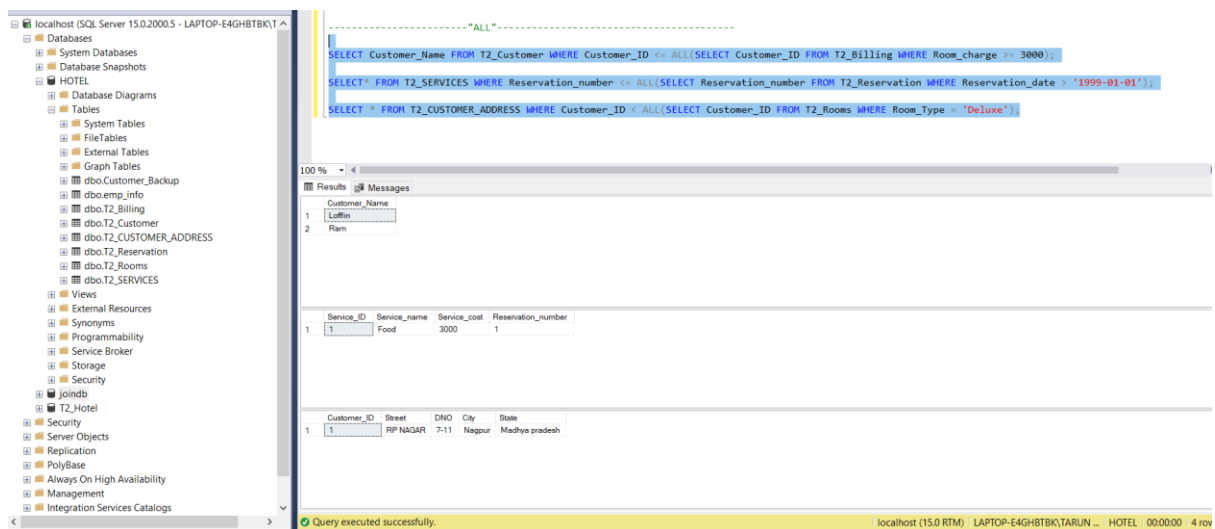
Customer_ID	Customer_Name	Phone_number	City	State	Zipcode	Email_ID
1	Loflin	868543748	Nagpur	MP	534201	lof@gmail.com
2	Ram	868543744	Hyderabad	TN	534204	Ram@gmail.com

Room_number	Room_Type	Room_location	number_of_beds	Customer_ID
1	Deluxe	block-2	1	2
2	Economic	block-1	3	1
3	Deluxe	block-2	1	4

Service_name	Service_cost
1 Transport	6000
2 Room	4000

Query executed successfully. localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN ... HOTEL 00:00:00 7 rows

QUERIES FOR “All”



```
SELECT Customer_Name FROM T2_Customer WHERE Customer_ID <= ALL(SELECT Customer_ID FROM T2_Billing WHERE Room_charge >= 3000);
SELECT * FROM T2_SERVICES WHERE Reservation_number <= ALL(SELECT Reservation_number FROM T2_Reservation WHERE Reservation_date > '1999-01-01');
SELECT * FROM T2_CUSTOMER_ADDRESS WHERE Customer_ID < ALL(SELECT Customer_ID FROM T2_Rooms WHERE Room_Type = 'Deluxe');
```

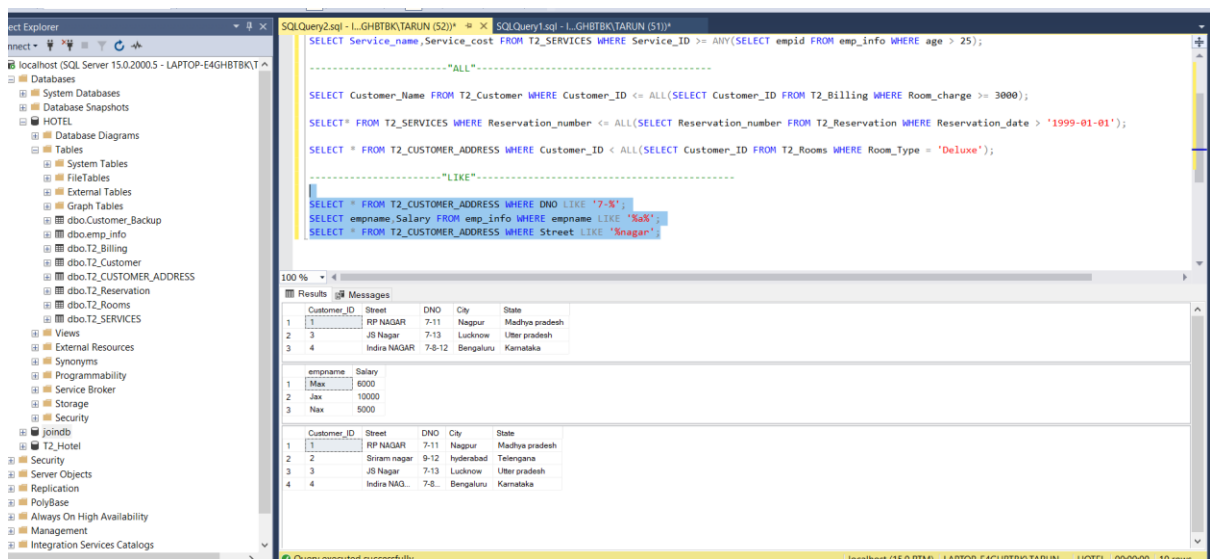
Customer_Name
1 Loflin
2 Ram

Service_ID	Service_name	Service_cost	Reservation_number
1	Food	3000	1

Customer_ID	Street	DNO	City	State
1	RP NAGAR	7-11	Nagpur	Madhya pradesh

Query executed successfully. localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN _ HOTEL 00:00:00 4 row

QUERIES FOR “Like”



```
SELECT Service_name,Service_cost FROM T2_SERVICES WHERE Service_ID >= ANY(SELECT empid FROM emp_info WHERE age > 25);
SELECT Customer_Name FROM T2_Customer WHERE Customer_ID <= ALL(SELECT Customer_ID FROM T2_Billing WHERE Room_charge >= 3000);
SELECT * FROM T2_SERVICES WHERE Reservation_number <= ALL(SELECT Reservation_number FROM T2_Reservation WHERE Reservation_date > '1999-01-01');
SELECT * FROM T2_CUSTOMER_ADDRESS WHERE Customer_ID < ALL(SELECT Customer_ID FROM T2_Rooms WHERE Room_Type = 'Deluxe');
```

```
SELECT * FROM T2_CUSTOMER_ADDRESS WHERE DNO LIKE '7-N';
SELECT empname,Salary FROM emp_info WHERE empname LIKE '%a%';
SELECT * FROM T2_CUSTOMER_ADDRESS WHERE Street LIKE '%nagar';
```

Customer_ID	Street	DNO	City	State
1	RP NAGAR	7-11	Nagpur	Madhya pradesh
2	J8 Nagar	7-13	Lucknow	Uttar pradesh
3	Indira NAGAR	7-9-12	Bengaluru	Karnataka

empname	Salary
Max	6000
Jax	10000
Nax	5000

Customer_ID	Street	DNO	City	State
1	RP NAGAR	7-11	Nagpur	Madhya pradesh
2	Sriram nagar	9-12	hyderabad	Telangana
3	J8 Nagar	7-13	Lucknow	Uttar pradesh
4	Indira NAG	7-8	Bengaluru	Karnataka

Query executed successfully. localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN _ HOTEL 00:00:00 10 row

QUERY FOR DIFFERENCE BETWEEN ANY AND ALL

The screenshot shows the SQL Server Management Studio interface. The query editor contains the following SQL code:

```
SELECT * FROM T2_CUSTOMER_ADDRESS WHERE DNO LIKE '7-8';
SELECT empname,Salary FROM emp_info WHERE empname LIKE '%a%';
SELECT * FROM T2_CUSTOMER_ADDRESS WHERE Street LIKE '%nagan';

-----DIFFERENCE BETWEEN ANY AND ALL-----
SELECT Customer_Name FROM T2_Customer WHERE Customer_ID <= ALL(SELECT Customer_ID FROM T2_Billing WHERE Room_charge >= 3000);
SELECT Customer_Name FROM T2_Customer WHERE Customer_ID <= ANY(SELECT Customer_ID FROM T2_Billing WHERE Room_charge >= 3000);
```

The Results pane shows the output of the query, displaying two tables of results. The first table shows Customer_Name values: Loflin and Ram. The second table shows Customer_Name values: Loflin, Ram, Mahesh, and Prabha.

Q2) One query for each Aggregate function.

The aggregate functions are MIN(), MAX(), COUNT(), AVG(), SUM()

AVG() – return the average of the set

MIN() – returns the minimum value in a set

MAX() – returns the maximum value in set

SUM() – returns the sum of all distinct values of a set

COUNT() – returns the number of items in a set

The screenshot shows the SQL Server Management Studio interface. The query editor contains the following SQL code:

```
USE HOTEL;

SELECT AVG(age) AS average_age FROM emp_info;
SELECT MAX(number_of_beds) AS Max_numofbeds FROM T2_Rooms;
SELECT MIN(Service_cost) AS min_service_charge FROM T2_SERVICES;
SELECT COUNT(Customer_ID) FROM T2_Customer WHERE Customer_Name LIKE '%a%';
SELECT SUM(Room_charge) AS total_charges FROM T2_Billing;
```

The Results pane shows the output of the query, displaying five rows of results:

average_age
37

Max_numofbeds
3

min_service_charge
3000

(No column name)
3

total_charges
18000

Q3) Illustrate the usage of order by, group by and having clause (2 queries for each case)

ORDER BY

The screenshot shows the Microsoft SQL Server Management Studio interface. The SQL query editor contains the following queries:

```
USE HOTEL;

-----ORDERBY-----
SELECT * FROM T2_Customer ORDER BY Customer_Name ASC;
SELECT * FROM emp_info ORDER BY age DESC;

-----GROUPBY-----
SELECT number_of_beds, COUNT(*) AS number_of_rooms FROM T2_Rooms GROUP BY number_of_beds;
SELECT Zipcode, COUNT(*) FROM T2_Customer GROUP BY Zipcode;

-----HAVING-----
SELECT COUNT(Room_number), Room_Type FROM T2_Rooms GROUP BY Room_Type HAVING COUNT(Room_number) >= 1;
SELECT COUNT(Reservation_number), LEFT(Reservation_date, 4) FROM T2_Reservation GROUP BY LEFT(Reservation_date, 4) HAVING COUNT(Reservation_number) >= 1;
```

The Results pane displays the output of the first two queries:

Customer_ID	Customer_Name	Phone_number	City	State	Zipcode	Email_ID
1	Luffy	868543742	Nagpur	MP	534201	luff@gmail.com
2	Mahesh	868543746	Lucknow	UP	534205	mah@gmail.com
3	Prabha	868543766	Bengaluru	Karnataka	534201	prab@gmail.com
4	Ram	868543744	Hyderabad	TN	534204	Ram@gmail.com

empid	empname	dob	age	Salary
1	Jax	1959-04-05	62	10000
2	Nax	1992-07-07	29	5000
3	Max	1999-03-21	22	6000

GROUP BY

The screenshot shows the Microsoft SQL Server Management Studio interface. The SQL query editor contains the following queries:

```
USE HOTEL;

-----ORDERBY-----
SELECT * FROM T2_Customer ORDER BY Customer_Name ASC;
SELECT * FROM emp_info ORDER BY age DESC;

-----GROUPBY-----
SELECT number_of_beds, COUNT(*) AS number_of_rooms FROM T2_Rooms GROUP BY number_of_beds;
SELECT Zipcode, COUNT(*) FROM T2_Customer GROUP BY Zipcode;

-----HAVING-----
SELECT COUNT(Room_number), Room_Type FROM T2_Rooms GROUP BY Room_Type HAVING COUNT(Room_number) >= 1;
SELECT COUNT(Reservation_number), LEFT(Reservation_date, 4) FROM T2_Reservation GROUP BY LEFT(Reservation_date, 4) HAVING COUNT(Reservation_number) >= 1;
```

The Results pane displays the output of the third and fourth queries:

number_of_beds	number_of_rooms
1	2
2	1

Zipcode	(No column name)
534201	2
534204	1
534205	1

HAVING CLAUSE

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
USE HOTEL;

--ORDERBY--
SELECT * FROM T2_Customer ORDER BY Customer_Name ASC;
SELECT * FROM emp_info ORDER BY age DESC;

--GROUPBY--
SELECT number_of_beds, COUNT(*) AS number_of_rooms FROM T2_Rooms GROUP BY number_of_beds;
SELECT Zipcode, COUNT(*) FROM T2_Customer GROUP BY Zipcode;

--HAVING--
SELECT COUNT(Room_number), Room_Type FROM T2_Rooms GROUP BY Room_Type HAVING COUNT(Room_number) >= 1;
SELECT COUNT(Reservation_number), LEFT(Reservation_date, 4) FROM T2_Reservation GROUP BY LEFT(Reservation_date, 4) HAVING COUNT(Reservation_number) >= 1;
```

The Results pane shows the output of the last query:

(No column name)	Room_Type
1	2
2	1

The Messages pane shows the status: 'Query executed successfully.'

Q4) Use Aggregate function with group by and having

AVG():

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
USE HOTEL;

SELECT AVG(number_of_beds) FROM T2_Rooms GROUP BY Room_location HAVING Room_location LIKE 'blockB';

SELECT COUNT(Customer_ID) FROM T2_Reservation GROUP BY Check_in_date HAVING Check_in_date >= '1992-02-03';

SELECT MIN(Salary) FROM emp_info GROUP BY age HAVING age > 25;

SELECT MAX(Room_charge) FROM T2_Billing GROUP BY LEFT(Payment_date, 7) HAVING LEFT(Payment_date, 7) LIKE '2021-X';

SELECT SUM(Service_cost) FROM T2_SERVICES GROUP BY Service_cost HAVING Service_cost BETWEEN 4000 AND 6000;
```

The Results pane shows the output of the first query:

(No column name)
1
2

The Messages pane shows the status: 'Query executed successfully.'

COUNT():

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'HOTEL' database selected. The right pane shows a SQL query window with the following code:

```
USE HOTEL;  
  
SELECT AVG(number_of_beds) FROM T2_Rooms GROUP BY Room_location HAVING Room_location LIKE 'block%';  
  
SELECT COUNT(Customer_ID) FROM T2_Reservation GROUP BY Check_in_date HAVING Check_in_date >= '1992-02-03';  
  
SELECT MIN(Salary) FROM emp_info GROUP BY age HAVING age > 25;  
  
SELECT MAX(Room_charge) FROM T2_Billing GROUP BY LEFT(Payment_date,7) HAVING LEFT(Payment_date,7) LIKE '2021-%';  
  
SELECT SUM(Service_cost) FROM T2_SERVICES GROUP BY Service_cost HAVING Service_cost BETWEEN 4000 AND 6000;
```

The 'Results' pane shows the output of the second query:

(No column name)
2
1

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN ... HOTEL 00:00:00 2 rows'.

MIN():

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'HOTEL' database selected. The right pane shows a SQL query window with the following code:

```
USE HOTEL;  
  
SELECT AVG(number_of_beds) FROM T2_Rooms GROUP BY Room_location HAVING Room_location LIKE 'block%';  
  
SELECT COUNT(Customer_ID) FROM T2_Reservation GROUP BY Check_in_date HAVING Check_in_date >= '1992-02-03';  
  
SELECT MIN(Salary) FROM emp_info GROUP BY age HAVING age > 25;  
  
SELECT MAX(Room_charge) FROM T2_Billing GROUP BY LEFT(Payment_date,7) HAVING LEFT(Payment_date,7) LIKE '2021-%';  
  
SELECT SUM(Service_cost) FROM T2_SERVICES GROUP BY Service_cost HAVING Service_cost BETWEEN 4000 AND 6000;
```

The 'Results' pane shows the output of the third query:

(No column name)
5000
10000
10000

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN ... HOTEL 00:00:00 3 rows'.

MAX():

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure for 'HOTEL'. The central pane shows a SQL query in 'SQLQuery2.sql' using the 'MAX()' function. The query is as follows:

```
USE HOTEL;

SELECT AVG(number_of_beds) FROM T2_Rooms GROUP BY Room_location HAVING Room_location LIKE 'block$';

SELECT COUNT(Customer_ID) FROM T2_Reservation GROUP BY Check_in_date HAVING Check_in_date >= '1992-02-03';

SELECT MIN(Salary) FROM emp_info GROUP BY age HAVING age > 25;

SELECT MAX(Room_charge) FROM T2_Billing GROUP BY LEFT(Payment_date,7) HAVING LEFT(Payment_date,7) LIKE '2021-';

SELECT SUM(Service_cost) FROM T2_SERVICES GROUP BY Service_cost HAVING Service_cost BETWEEN 4000 AND 6000;
```

The 'Results' pane shows the output of the last query, which is a single value: 4000.

(No column name)
4000

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK(TARUN) HOTEL 00:00:00 1 rows'.

SUM():

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure for 'HOTEL'. The central pane shows a SQL query in 'SQLQuery2.sql' using the 'SUM()' function. The query is as follows:

```
USE HOTEL;

SELECT AVG(number_of_beds) FROM T2_Rooms GROUP BY Room_location HAVING Room_location LIKE 'block$';

SELECT COUNT(Customer_ID) FROM T2_Reservation GROUP BY Check_in_date HAVING Check_in_date >= '1992-02-03';

SELECT MIN(Salary) FROM emp_info GROUP BY age HAVING age > 25;

SELECT MAX(Room_charge) FROM T2_Billing GROUP BY LEFT(Payment_date,7) HAVING LEFT(Payment_date,7) LIKE '2021-';

SELECT SUM(Service_cost) FROM T2_SERVICES GROUP BY Service_cost HAVING Service_cost BETWEEN 4000 AND 6000;
```

The 'Results' pane shows the output of the last query, which is a single value: 4000.

(No column name)
4000

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK(TARUN) HOTEL 00:00:00 1 rows'.

Q5) Write at least 3 nested queries using order by, group by and having clause.

QUERY:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
SELECT Customer_Name, COUNT(*) FROM T2_Customer
WHERE Customer_ID = ANY(
    SELECT Customer_ID FROM T2_Reservation
    WHERE Reservation_number = ANY(
        SELECT Reservation_number FROM T2_SERVICES
        WHERE Service_cost >= 4000
    )
)
GROUP BY Customer_Name HAVING Customer_Name LIKE '%a%'
ORDER BY Customer_Name desc;
```

The Results pane shows the output of the query:

Customer_Name	(No column name)
1	Prabha

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN HOTEL 00:00:00 1 rows'.

Q6) Illustrate the Usage of Except, Exists, Not Exists, Union, Intersection

EXCEPT():

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
SELECT Customer_ID FROM T2_Customer
EXCEPT
SELECT Customer_ID FROM T2_Reservation;
-----EXISTS-----
SELECT Customer_ID FROM T2_Rooms
WHERE EXISTS
(SELECT Customer_ID FROM T2_Billing)
ORDER BY Customer_ID ASC;
-----NOT EXISTS-----
SELECT * FROM T2_Customer
WHERE NOT EXISTS
(SELECT Customer_ID FROM T2_Reservation);
-----UNION-----
SELECT City FROM T2_CUSTOMER_ADDRESS
UNION
SELECT City FROM T2_Customer;
-----INTERSECTION-----
SELECT Room_charge FROM T2_Billing
INTERSECT
SELECT Service_cost FROM T2_SERVICES;
```

The Results pane shows the output of the query:

Customer_ID
1

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN HOTEL 00:00:00 1 rows'.

EXISTS():

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
-----EXCEPT-----
SELECT Customer_ID FROM T2_Customer
EXCEPT
SELECT Customer_ID FROM T2_Reservation;
-----EXISTS-----
SELECT Customer_ID FROM T2_Rooms
WHERE EXISTS
(SELECT Customer_ID FROM T2_Billing
ORDER BY Customer_ID ASC;
-----NOT EXISTS-----
SELECT * FROM T2_Customer
WHERE NOT EXISTS
(SELECT Customer_ID FROM T2_Reservation);
-----UNION-----
SELECT City FROM T2_CUSTOMER_ADDRESS
UNION
SELECT City FROM T2_Customer;
-----INTERSECTION-----
SELECT Room_charge FROM T2_Billing
INTERSECT
SELECT Service_cost FROM T2_SERVICES;
```

The Results pane at the bottom shows the output of the query, which is a single column 'Customer_ID' with three rows:

Customer_ID
1
2
4

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN - HOTEL 00:00:00 3 rows'.

NOT EXISTS():

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
-----EXCEPT-----
SELECT Customer_ID FROM T2_Customer
EXCEPT
SELECT Customer_ID FROM T2_Reservation;
-----EXISTS-----
SELECT Customer_ID FROM T2_Rooms
WHERE EXISTS
(SELECT Customer_ID FROM T2_Billing
ORDER BY Customer_ID ASC;
-----NOT EXISTS-----
SELECT * FROM T2_Customer
WHERE NOT EXISTS
(SELECT Customer_ID FROM T2_Reservation);
-----UNION-----
SELECT City FROM T2_CUSTOMER_ADDRESS
UNION
SELECT City FROM T2_Customer;
-----INTERSECTION-----
SELECT Room_charge FROM T2_Billing
INTERSECT
SELECT Service_cost FROM T2_SERVICES;
```

The Results pane at the bottom shows the output of the query, which is a single column 'Customer_ID' with three rows:

Customer_ID
1
2
4

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN - HOTEL 00:00:00 0 rows'.

UNION():

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
-----EXCEPT-----
SELECT Customer_ID FROM T2_Customer
EXCEPT
SELECT Customer_ID FROM T2_Reservation;
-----EXISTS-----
SELECT Customer_ID FROM T2_Rooms
WHERE EXISTS
(SELECT Customer_ID FROM T2_Billing)
ORDER BY Customer_ID ASC;
-----NOT EXISTS-----
SELECT * FROM T2_Customer
WHERE NOT EXISTS
(SELECT Customer_ID FROM T2_Reservation);
-----UNION-----
SELECT City FROM T2_CUSTOMER_ADDRESS
UNION
SELECT City FROM T2_Customer;
-----INTERSECTION-----
SELECT Room_charge FROM T2_Billing
INTERSECT
SELECT Service_cost FROM T2_SERVICES;
```

The Results pane at the bottom shows the output of the query, which is a list of cities:

City
Bengaluru
Hyderabad
Lucknow
Nagpur

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN... HOTEL 00:00:00 4 rows'.

INTERSECT:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains the following SQL code:

```
-----EXCEPT-----
SELECT Customer_ID FROM T2_Customer
EXCEPT
SELECT Customer_ID FROM T2_Reservation;
-----EXISTS-----
SELECT Customer_ID FROM T2_Rooms
WHERE EXISTS
(SELECT Customer_ID FROM T2_Billing)
ORDER BY Customer_ID ASC;
-----NOT EXISTS-----
SELECT * FROM T2_Customer
WHERE NOT EXISTS
(SELECT Customer_ID FROM T2_Reservation);
-----UNION-----
SELECT City FROM T2_CUSTOMER_ADDRESS
UNION
SELECT City FROM T2_Customer;
-----INTERSECTION-----
SELECT Room_charge FROM T2_Billing
INTERSECT
SELECT Service_cost FROM T2_SERVICES;
```

The Results pane at the bottom shows the output of the query, which is a single row representing the intersection of room charges and service costs:

Room_charge
3000

The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK\TARUN... HOTEL 00:00:00 1 rows'.

Q7) INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN- 3 queries for each instance

INNER JOIN

The screenshot shows the Microsoft SQL Server Management Studio interface with three queries executed in the SQL Query window. The Object Explorer on the left shows the database structure for 'HOTEL'.

Query 1: INNER JOIN T2_CUSTOMER

```
SELECT Customer_Name, DNO, Street, T2_Customer.City FROM T2_Customer
INNER JOIN
T2_CUSTOMER_ADDRESS
ON T2_Customer.Customer_ID = T2_CUSTOMER_ADDRESS.Customer_ID;
```

Customer_Name	DNO	Street	City
Loftin	7-11	RP NAGAR	Nagpur
Ram	9-12	Sram nagar	hyderabad
Mahesh	7-13	JS Nagar	Lucknow
Prabha	7-8-12	Indira NAGAR	Bengaluru

Query 2: INNER JOIN T2_Reservations

```
SELECT Customer_Name, Number_of_guests, Check_in_date, Check_out_date FROM T2_Customer
INNER JOIN T2_Reservations
ON T2_Customer.Customer_ID = T2_Reservations.Customer_ID;
```

Customer_Name	Number_of_guests	Check_in_date	Check_out_date
Ram	5	1999-02-03	1999-02-22
Loftin	4	1999-02-03	1999-02-22
Prabha	2	1999-04-03	1999-04-04

Query 3: INNER JOIN T2_Rooms

```
SELECT Reservation_number, Reservation_date, Room_Type, Room_location FROM T2_Rooms INNER JOIN T2_Reservations
ON T2_Rooms.Room_number = T2_Reservations.Room_number;
```

Reservation_number	Reservation_date	Room_Type	Room_location
1	1999-02-01	Deluxe	block-2
2	1999-02-03	Economic	block-1
3	1999-04-01	Deluxe	block-2

LEFT OUTER JOIN

The screenshot shows the Microsoft SQL Server Management Studio interface with three queries executed in the SQL Query window. The Object Explorer on the left shows the database structure for 'HOTEL'.

Query 1: LEFT OUTER JOIN T2_Rooms

```
SELECT * FROM T2_Customer
LEFT OUTER JOIN T2_Rooms
ON T2_Customer.Customer_ID = T2_Rooms.Customer_ID;
```

Customer_ID	Customer_Name	Phone_number	City	State	Zipcode	Email_ID	Room_number	Room_Type	Room_location	number_of_beds	Customer_ID
1	Loftin	868543748	Nagpur	MP	534201	loft@gmail.com	2	Economic	block-1	3	1
2	Ram	868543744	hyderabad	TN	534204	Ram@gmail.com	1	Deluxe	block-2	1	2
3	Mahesh	868543746	Lucknow	UP	534205	mah@gmail.com	NULL	NULL	NULL	NULL	NULL
4	Prabha	868543766	Bengaluru	Karnataka	534201	prab@gmail.com	3	Deluxe	block-2	1	4

Query 2: LEFT OUTER JOIN T2_CUSTOMER_ADDRESS

```
SELECT * FROM T2_CUSTOMER_ADDRESS
LEFT OUTER JOIN T2_Reservations
ON T2_Reservations.Customer_ID = T2_CUSTOMER_ADDRESS.Customer_ID;
```

Customer_ID	Street	DNO	City	State	Reservation_number	Check_in_date	Check_out_date	Number_of_guests	Reservation_date	Customer_ID	Room_number
1	RP NAGAR	7-11	Nagpur	Madhya pradesh	1	1999-02-03	1999-02-22	4	1999-02-03	1	2
2	Sram nagar	9-12	hyderabad	Telangana	1	1999-02-03	1999-02-22	5	1999-02-01	2	1
3	JS Nagar	7-13	Lucknow	Uttar pradesh	NULL	NULL	NULL	NULL	1999-04-01	NULL	NULL
4	Indira NAGAR	7-8-12	Bengaluru	Karnataka	3	1999-04-03	1999-04-04	2	1999-04-01	4	3

Query 3: LEFT OUTER JOIN empinfo

```
SELECT * FROM empinfo LEFT OUTER JOIN
T2_SERVICES ON
T2_SERVICES.Service_ID = empinfo.empid;
```

empid	empname	dob	age	Salary	Service_ID	Service_name	Service_cost	Reservation_number
1	Max	1999-03-21	22	6000	1	Food	3000	1
2	Jai	1999-04-05	42	10000	2	Transport	4000	2
3	Nax	1992-07-07	29	5000	3	Room	4000	2
4	Amaan	2001-12-11	20	6000	NULL	NULL	NULL	NULL
5	Chuck	1976-05-30	45	10000	NULL	NULL	NULL	NULL

RIGHT OUTER JOIN

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'HOTEL'. The main query window contains three SQL queries demonstrating RIGHT OUTER JOINs:

```

-- Query 1: T2_Rooms to T2_Customer
SELECT * FROM T2_Rooms
RIGHT OUTER JOIN T2_Customer
ON T2_Customer.Customer_ID = T2_Rooms.Customer_ID;

-- Query 2: T2_Reservation to T2_CUSTOMER_ADDRESS
SELECT * FROM T2_Reservation
RIGHT OUTER JOIN T2_CUSTOMER_ADDRESS
ON T2_Reservation.Customer_ID = T2_CUSTOMER_ADDRESS.Customer_ID;

-- Query 3: T2_SERVICES to emp_info
SELECT * FROM T2_SERVICES RIGHT OUTER JOIN
emp_info ON
T2_SERVICES.Service_ID = emp_info.empid;

```

The Results pane shows the output of the first query, displaying columns: Room_number, Room_Type, Room_location, number_of_beds, Customer_ID, Customer_Name, Phone_number, City, State, Zipcode, Email_ID. The results are as follows:

Room_number	Room_Type	Room_location	number_of_beds	Customer_ID	Customer_Name	Phone_number	City	State	Zipcode	Email_ID
2	Economic	block-1	3	1	Loflin	888543748	Nagpur	MP	534201	lof@gmail.com
1	Deluxe	block-2	1	2	Ram	888543744	Hyderabad	TN	534204	Ram@gmail.com
3	NULL	NULL	NULL	3	Maheesh	888543746	Lucknow	UP	534205	mahe@gmail.com
3	Deluxe	block-2	1	4	Prabha	888543766	Bengaluru	Karnataka	534201	prab@gmail.com

The bottom status bar indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK,TARUN ... HOTEL 00:00:00 13 rows'.

Q8) Use all the above condition in JOIN as well.

QUERY:

The screenshot shows the Microsoft SQL Server Management Studio interface. The main query window contains a complex SQL query using multiple JOINs and filters:

```

USE HOTEL;

SELECT COUNT(*) ,Room_location FROM T2_Rooms
JOIN T2_Reservation
ON T2_Rooms.Customer_ID = T2_Reservation.Customer_ID
JOIN T2_Customer
ON T2_Rooms.Customer_ID = T2_Customer.Customer_ID
GROUP BY Room_location
HAVING Room_location LIKE 'block%'
ORDER BY Room_location DESC;

```

The Results pane shows the output of the query, displaying columns: (No column name), Room_location. The results are as follows:

(No column name)	Room_location
2	block-2
1	block-1

The bottom status bar indicates 'Query executed successfully.' and 'localhost (15.0 RTM) LAPTOP-E4GHBTBK,TARUN ... HOTEL 00:00:00 2 rows'.