DBMS LAB ASSIGNMENT - 5

NAME: Divyansh Tripathi

REG NO: 19BCS039

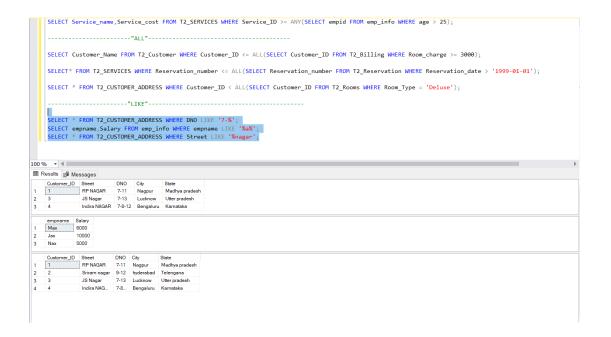
Q1) Illustrate logical ANY, ALL and LIKE operator- the queries should be relevant to your respective databases 3 queries for each operator. One query explaining the difference between ANY and ALL

QUERIES FOR "ANY"

	EUSE 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);						
100 % → ◀							
⊞ Results 8 Nessages							
1 2 3	Room_number 1 2 3	Customer_Name Lofflin Ram Room_Type Deluxe Economic Deluxe	8688543744 8688543744 Room_location block-2 block-1	Nagpur hyderabad		Email_I0 Inf@gmail.com Ram@gmail.com	
1 2	Service_name Transport Room	Service_cost 8000 4000					

QUERIES FOR "AII"

QUERIES FOR "Like"



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

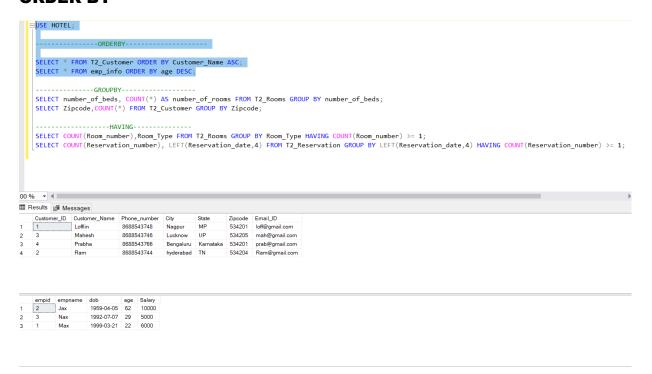
MAX() – returns the maximum value in set

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

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

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

ORDER BY



GROUP BY

HAVING CLAUSE

Q4) Use Aggregate function with group by and having

AVG():

```
EUECT AVG(number_of_beds) FROM T2_Rosms 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;

100-% - (Nocoumn name) 1 3
2 1
```

COUNT():

MIN():

MAX():

```
BUSE 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;
```

SUM():

```
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.

100 % 

Results Messages

(No column name)

1 4000
```

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

QUERY:

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

EXCEPT():

EXISTS():

```
-----FXCEPT-----
   SELECT Customer_ID FROM T2_Customer
    SELECT Customer_ID FROM T2_Reservation;
          ---EXISTS
   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
   SELECT Service_cost FROM T2_SERVICES;
100 % -
Customer_ID

1 1
2 2
3 4
```

NOT EXISTS():

```
-----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 FROM IZ_BIIIIng
ORDER BY Customer_ID ASC;
-----NOT EXISTS-----
SELECT * FROM T2_Customer
WHERE NOT EXISTS
     (SELECT Customer_ID FROM T2_Reservation);
    SELECT City FROM T2_CUSTOMER_ADDRESS
     UNION
     SELECT City FROM T2_Customer;
   SELECT Room_charge FROM T2_Billing
     INTERSECT
   SELECT Service_cost FROM T2_SERVICES;
100 % 🔻 🔻
Customer_ID Customer_Name Phone_number City State Zipcode Email_ID
```

UNION():

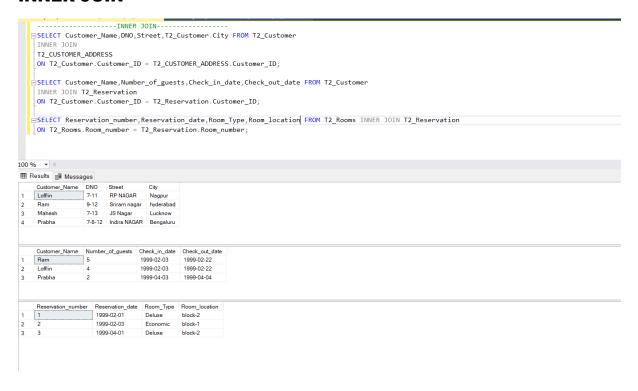
```
-----EXCEPT-----
  □SELECT Customer_ID FROM T2_Customer
    EXCEPT
    SELECT Customer_ID FROM T2_Reservation;
   SELECT Customer_ID FROM T2_Rooms
   WHERE EXISTS
(SELECT Customer_ID FROM T2_Billing)
    ORDER BY Customer_ID ASC;
   SELECT * FROM T2_Customer
    WHERE NOT EXISTS
    (SELECT Customer_ID FROM T2_Reservation);
    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;
.00 % 🔻 🔻
City
1 Bengaluru
   hyderabad
2 hyderaba
3 Lucknow
4 Nagpur
```

INTERSECT:

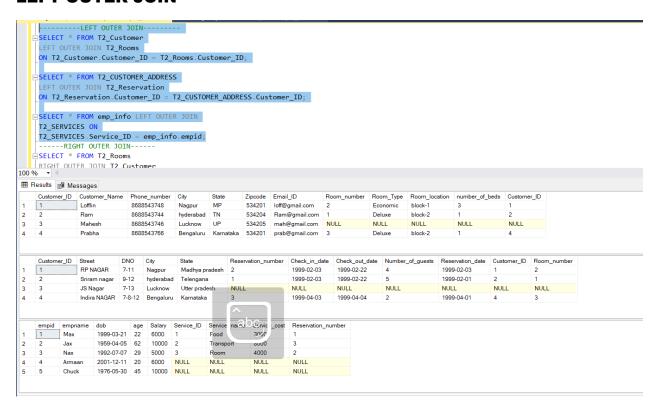
```
-----EXCEPT-----
    □SELECT Customer_ID FROM T2_Customer
     SELECT Customer_ID FROM T2_Reservation;
    SELECT Customer_ID FROM T2_Rooms
     WHERE EXISTS
     (SELECT Customer_ID FROM T2_Billing)
ORDER BY Customer_ID ASC;
    -----NOT EXISTS-----
SELECT * FROM T2_Customer
     WHERE NOT EXISTS
     (SELECT Customer_ID FROM T2_Reservation);
    SELECT City FROM T2_CUSTOMER_ADDRESS
     SELECT City FROM T2_Customer;
     SELECT CITY FROM 12_cdscomer,
------INTERSECTION------
SELECT Room_charge FROM 12_Billing
    SELECT Service_cost FROM T2_SERVICES;
100 % 🔻 🔻
Room_charge
1 3000
```

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

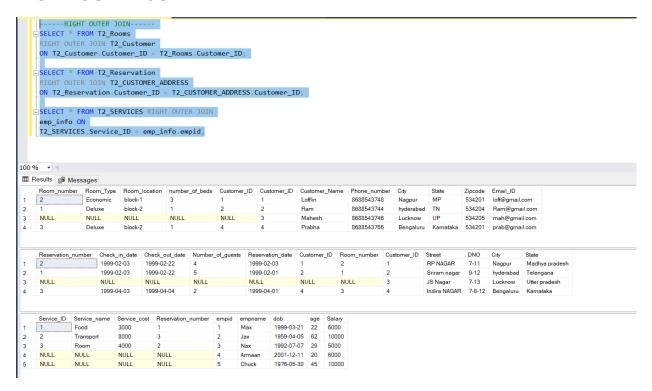
INNER JOIN



LEFT OUTER JOIN



RIGHT OUTER JOIN



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

QUERY:

```
ESELECT COUNT(*), Room_location FROM T2_Rooms
JOIN T2_Roservation
ON T2_Roservation
```