

# DATABASE MANAGEMENT SYSTEM

## LAB ASSIGNMENT - 05

NAME : NIKHITA B KARADI  
ROLL NO. : 19BCS077

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

The screenshot shows the Microsoft SQL Server Management Studio interface with the following details:

- File Path:** T11\_19bcs077.sql - localhost.master (DESKTOP-2EV6IDK\DELL (51))\*
- Object Explorer:** Shows the database structure for localhost.
- Query Editor:** Contains three SELECT statements under the comment /\* Question 1 \*/. The first statement uses 'ANY' in a WHERE clause. The second statement uses 'ANY' in a WHERE clause. The third statement uses 'ANY' in a WHERE clause.
- Results Grid:** Displays the results of the first query, which retrieves data from the instructor, course, and student tables where course\_id is less than 108. The results show 80 rows.
- Status Bar:** Shows "Query executed successfully." and other session details.

The screenshot shows the Microsoft SQL Server Management Studio interface with the following details:

- File Path:** T11\_19bcs077.sql - localhost.master (DESKTOP-2EV6IDK\DELL (51))\*
- Object Explorer:** Shows the database structure for localhost.
- Query Editor:** Contains three SELECT statements under the comment /\* Question 1 \*/. The first statement uses 'ALL' in a WHERE clause. The second statement uses 'ALL' in a WHERE clause. The third statement uses 'ALL' in a WHERE clause.
- Results Grid:** Displays the results of the first query, which retrieves data from the instructor, course, and student tables where course\_id is less than 108. The results show 61 rows.
- Status Bar:** Shows "Query executed successfully." and other session details.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'master' is selected. In the center pane, three queries are run against the 'student' table:

```

1. select * from student
   where last_name like '%s';
2. select * from student
   where first_name like 'A%';
3. select first_name , last_name , dept_name from student
   where age like 18;

```

The results pane displays the output for each query:

- Query 1: Returns 8 rows of student data.
- Query 2: Returns 4 rows of student data.
- Query 3: Returns 7 rows of student data.

At the bottom, a message indicates "Query executed successfully." and the status bar shows "localhost (15.0 RTM) DESKTOP-2EV6JDK\DELL (51) master 00:00:00 | 19 rows".

## Q2). One query for each Aggregate function

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'master' is selected. In the center pane, a single query is run using multiple aggregate functions:

```

where age like 18;

/* Question : 2 */

select avg(salary) as avg_salary from instructor

select count(s_id) as AGE_is_18 from student where age = 18

select max(age) from instructor

select min(salary) from instructor

select sum(salary) from instructor

```

The results pane displays the output for each aggregate function:

- avg\_salary: 60000
- AGE\_is\_18: 7
- (No column name): 43
- (No column name): 55000
- (No column name): 2400000

At the bottom, a message indicates "Query executed successfully." and the status bar shows "localhost (15.0 RTM) DESKTOP-2EV6JDK\DELL (63) master 00:00:00 | 5 rows".

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

```

select avg(salary),i_id from instructor
group by i_id having (i_id > 10)

```

```
select avg(age), s_id from student
group by s_id having (s_id > 1020)
```

```
/* Question 3 */
select * from instructor order by salary asc
select * from instructor order by age asc

select age, count(*) from student group by age
select duration , count(*) from course group by duration

select avg(salary), i_id from instructor
group by i_id having (i_id > 10)
select avg(age), s_id from student
```

	dept_name	first_name	last_name	salary	age
1	CSE	Ethan	Johnson	55000	32
2	CSE	Carter	Davis	55000	30
3	CSE	Benjamin	Garcia	55000	34
4	CSE	Ryan	Clark	55000	29

	i_id	dept_name	first_name	last_name	salary	age
1	15	CSE	Ryan	Clark	55000	29
2	20	CSE	Colob	Hill	55000	29
3	35	CSE	Can	Clark	55000	29
4	40	CSE	colon	Hill	55000	29

	age
1	18
2	19

	duration
1	29
2	1

	(No column name)	i_id
1	60000	11
2	60000	12
3	65000	13
4	60000	14

	(No column name)	s_id
1	19	1021
2	19	1022
3	19	1023
4	18	1024

Query executed successfully.

#### Q4).Use Aggregate function with group by and having

```
/* Question 4 */
select avg(salary) from instructor group by first_name having first_name like 'A%'

select count(s_id) from student group by age having age < 19

select min(age) from instructor group by salary having salary > 55000

select max(age)from instructor group by salary having salary < 60000

select sum(salary), count(*) from instructor group by salary having salary between 45000 and 55000
```

	(No column name)
1	60000

	(No column name)
1	7

	(No column name)
1	35
2	40

	(No column name)
1	34

	(No column name)	(No column name)
1	55000	10

Query executed successfully.

#### Q5).Write atleast three nested queries using order by, group by and having clause

T11\_19bc077.sql - localhost.master (DESKTOP-2EV6JDK\DELL (63)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Execute

master

Object Explorer

Databases

Tables

Views

Security

Server Objects

Replication

PolyBase

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

T11\_19bc077.sql - 2EV6JDK\DELL (63)\*

```
select sum(salary), count(*) from instructor group by salary having salary between 45000 and 55000

/* Question 5 */

select last_name, count(*) from instructor
where i_id = any(select i_id from course where dept_name = 'CSE')
group by last_name having last_name like '%so%'
order by last_name desc

/* Question 6 */

Results Messages
```

last_name (No column name)
1 Brown

Query executed successfully.

localhost (15.0 RTM) | DESKTOP-2EV6JDK\DELL (63) | master | 00:00:00 | 1 rows

Ln 320 Col 1 Ch 1 INS

Ready

Type here to search

4:31 PM 3/23/2021

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

```
/* Question 6 */
select dept_name from department
EXCEPT
select dept_name from instructor

select i_id from instructor
where EXISTS
(select i_id from course)
order by i_id DESC

select * from course
where NOT EXISTS
(select dept_name from instructor)

select first_name from instructor
UNION
select first_name from student

select i_id from instructor
intersect
select age from student
```

T11\_19bcs077.sql - localhost.master (DESKTOP-2EV6IDK\DELL (51)) - Microsoft SQL Server Management Studio

```
/* Question 6 */
select dept_name from department
EXCEPT
select dept_name from instructor

select i_id from instructor
where EXISTS
(select i_id from course)
order by i_id DESC

select * from course
where NOT EXISTS
(select dept_name from instructor)
```

Results Messages

dept_name
ECE
EEE

i_id
40
39
38
37
36
35

course_id	dept_name	i_id	duration	course_name

first_name
Aidan
Alexander
Alexis
Alice
Ashley
Bella

i_id
18
19

Query executed successfully.

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

T11\_19bcs077.sql - localhost.master (DESKTOP-2EV6IDK\DELL (51)) - Microsoft SQL Server Management Studio

```
/* Question 7 */
select course_name,duration from course
inner join instructor
on course.course_id = instructor.i_id

select s_id from student
INNER JOIN instructor
ON student.age = instructor.i_id

select i_id,salary from instructor
INNER JOIN student
ON instructor.first_name=student.first_name
```

Results Messages

course_name	duration

s_id
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010

i_id	salary

Query executed successfully.

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to 'T11\_19bcs077.sql - localhost.master (DESKTOP-2EV6JDK\Dell (51))'.

The Object Explorer pane on the left shows the database structure for 'localhost (SQL Server 15.0.2080.9 - DESK)'. The 'master' database is selected.

The main window displays a T-SQL query:

```
select course_name, duration from course  
LEFT OUTER JOIN instructor  
ON course.course_id = instructor.i_id  
  
select s_id from student  
LEFT OUTER JOIN instructor  
ON student.age = instructor.i_id  
  
select i_id,salary from instructor  
LEFT OUTER JOIN student  
ON instructor.first_name=student.first_name
```

The results pane shows three tables:

- course\_name, duration**

	course_name	duration
1	Programming	1
2	DataStructure	1
3	OS	1
4	CircuitAnalysis	2
5	DigitalLogic	1
6	DBMS	1

- s\_id**

	s_id
1	1000
2	1001
3	1002
4	1003
5	1004
6	1005

- i\_id, salary**

	i_id	salary
1	1	60000
2	2	55000
3	3	65000
4	4	60000
5	5	55000
6	6	60000
7	7	55000
8	8	60000

The status bar at the bottom shows 'Query executed successfully.' and the connection details: 'localhost (15.0 RTM) | DESKTOP-2EV6JDK\Dell (51) | master | 00:00:00 | 100 rows'.

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to 'T11\_19bcs077.sql - localhost.master (DESKTOP-2EV6JDK\DELL (51))'.

The Object Explorer on the left shows the database structure, including 'localhost (SQL Server 15.0.2080.9 - DESKTOP-2EV6JDK\DELL (51))' with its databases, security, server objects, replication, polybase, and management components.

The main window displays three queries in the 'T11\_19bcs077.sql' file:

```
select * from course
RIGHT OUTER JOIN instructor
ON course.course_id = instructor.i_id

select * from student
RIGHT OUTER JOIN instructor
ON student.age = instructor.i_id

select * from instructor
RIGHT OUTER JOIN student
ON instructor.first_name = student.first_name
```

The 'Results' tab shows the output of the first query:

	course_id	dept_name	i_id	duration	course_name	i_id	dept_name	first_name	last_name	salary	age
1	NULL	NULL	NULL	NULL	CSE	1	CSE	Liam	Smith	60000	36
2	NULL	NULL	NULL	NULL	CSE	2	CSE	Ethan	Johnson	55000	32
3	NULL	NULL	NULL	NULL	CSE	3	CSE	Jacob	Brown	65000	40
4	NULL	NULL	NULL	NULL	CSE	4	CSE	Michal	Jones	60000	35
5	NULL	NULL	NULL	NULL	CSE	5	CSE	Benjamin	Garcia	55000	34
6	NULL	NULL	NULL	NULL	CSE	6	CSE	Daniel	Miller	60000	38

The 'Messages' tab shows the output of the second query:

	s_id	first_name	last_name	age	dept_name	i_id	dept_name	first_name	last_name	salary	age
1	NULL	NULL	NULL	NULL	CSE	1	CSE	Liam	Smith	60000	36
2	NULL	NULL	NULL	NULL	CSE	2	CSE	Ethan	Johnson	55000	32
3	NULL	NULL	NULL	NULL	CSE	3	CSE	Jacob	Brown	65000	40
4	NULL	NULL	NULL	NULL	CSE	4	CSE	Michal	Jones	60000	35
5	NULL	NULL	NULL	NULL	CSE	5	CSE	Benjamin	Garcia	55000	34
6	NULL	NULL	NULL	NULL	CSE	6	CSE	Daniel	Miller	60000	38

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

	i_id	dept_name	first_name	last_name	salary	age	s_id	first_name	last_name	age	dept_name
1	NULL	NULL	NULL	NULL	1000	19	1000	Aidan	Buttle	19	CSE
2	NULL	NULL	NULL	NULL	1001	18	1001	Teresa	Simmons	18	CSE
3	NULL	NULL	NULL	NULL	1002	19	1002	Gabriela	Flores	19	CSE
4	NULL	NULL	NULL	NULL	1003	19	1003	Harold	Bennett	19	CSE
5	NULL	NULL	NULL	NULL	1004	18	1004	Conner	Sanders	18	CSE
6	NULL	NULL	NULL	NULL	1005	18	1005	Peter	Hughes	18	CSE
7	NULL	NULL	NULL	NULL	1006	18	1006	Ashley	Bryant	18	CSE
8	NULL	NULL	NULL	NULL	1007	19	1007	Nicole	Patterson	19	CSE

The status bar at the bottom indicates 'Query executed successfully.' and the execution time '5:22 PM 3/24/2021'.