A.Y. 2020-2021
Class: SE-ITA/B, Semester: III

Subject: **Structured Query Lab**

Experiment – 6: Perform basic and complex queries using group by, nested and recursive queries on the chosen system.

1. **Aim:** To Perform basic and complex queries using group by, nested and recursive queries on the chosen system.

2. **Objective:**
   - After performing the experiment, the students will be able to formulate and use various complex SQL queries to manipulate database and retrieve data with help of nested, recursive and group by query.

3. **Outcome:** L304.3: To Write queries in SQL to retrieve any type of information from a database.

4. **Prerequisite:** Understanding of various SQL complex queries like group by, nested and recursive queries with notations and terminologies along with sample syntax.

5. **Requirements:** PC, Oracle 11g/SQL Server 2008 R2, Microsoft Word, Internet, My SQL workbench

6. **Pre-Experiment Exercise:**
   Brief Theory:(To be hand written/typed)
   Explain each of the following with 1 examples.
   - Group by clause

   The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
   The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.
   Example:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

   - Having clause

   The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.
   Example:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

   - Order By

   The ORDER BY keyword is used to sort the result-set in ascending or descending order.
   The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

• Nested queries

In nested queries, a query is written inside a query. The result of inner query is used in execution of outer query.

```
SELECT *
FROM CUSTOMERS
WHERE ID IN (SELECT ID
FROM CUSTOMERS
WHERE SALARY > 4500) ;
```

• Recursive queries

A recursive common table expression (CTE) is a CTE that references itself. By doing so, the CTE repeatedly executes, returns subsets of data, until it returns the complete result set.

```
WITH cte_numbers(n, weekday)
AS (
    SELECT
        0,
        DATENAME(DW, 0)
    UNION ALL
    SELECT
        n + 1,
        DATENAME(DW, n + 1)
    FROM
        cte_numbers
    WHERE n < 6
)
SELECT
    weekday
FROM
    cte_numbers;
```

7. Laboratory Exercise

A. Procedure:

i) Open SQL server 2008 using below login credentials:

Username: sa, Password: Lab301a or MySQL

ii) Use existing database created by you or

iii) Construct your own database

iv) Construct tables for any two to three entities from your chosen case study v) Insert at least 8 to 10 records for each tables

vi) Execute below queries:

Select query (Any 3)

```
SELECT * FROM table_name;
SELECT DISTINCT column1, column2, ...
FROM table_name;
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

String operations any 1

Use **Group by** using below syntax

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

Having Clause example

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

Order By

```
SELECT * FROM Customers
ORDER BY Country;
```

Example:

```
SELECT * FROM Customers
ORDER BY Country;
```

Nested Query example

```
SELECT *
 FROM CUSTOMERS
 WHERE ID IN (SELECT ID
 FROM CUSTOMERS
 WHERE SALARY > 4500) ;
```

Recursive Query Syntax

```
WITH expression_name (column_list)
AS
(
 -- Anchor member
 initial_query
 UNION ALL
 -- Recursive member that references expression_name.
 recursive_query
)
-- references expression name
SELECT *
FROM expression_name
```

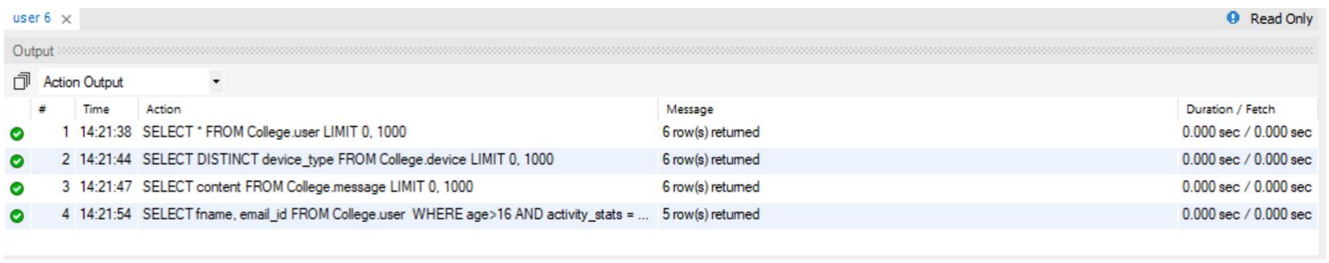
vi)Write/Print output for each query

B. **Result/Observation/Program code:** Attach all queries executed code with proper output

```
SELECT * FROM College.user;


SELECT DISTINCT device_type FROM College.device;


SELECT content FROM College.message;


SELECT fname, email_id FROM College.user
```

```sql
WHERE age>16 AND activity_stats = 1;

SELECT * FROM College.device
WHERE device_type LIKE '%PC-Windows%';

SELECT COUNT(device_id), device_type
FROM college.device
GROUP BY device_type;

SELECT COUNT(device_id), device_type
FROM college.device
GROUP BY device_type
HAVING COUNT(device_id)>=1;

SELECT * FROM College.user
ORDER BY DOB;

SELECT * FROM College.user WHERE email_id
IN( SELECT email_id FROM college.user WHERE age>16);

WITH RECURSIVE message_send(sender_id, reciever_id, content) AS (
    SELECT sender_id, reciever_id, content
    FROM College.message
UNION
    SELECT message_send.sender_id, message_send.reciever_id, message_send.content
    FROM message_send, message
    WHERE message_send.sender_id = message.reciever_id
)
SELECT *
FROM message_send;
```

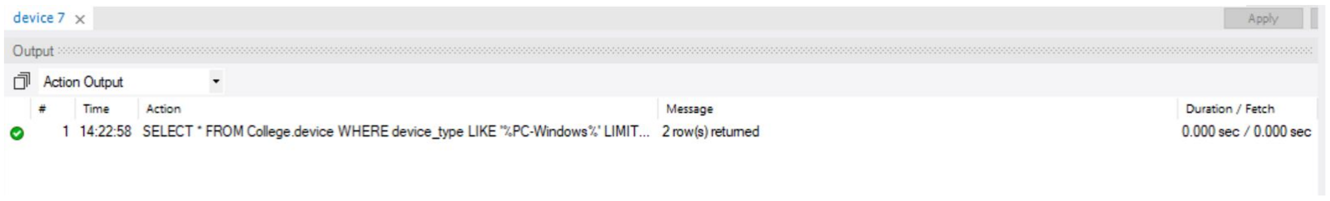
Select Output


user 6 × Read Only
Output
Action Output

	#	Time	Action	Message	Duration / Fetch
✓	1	14:21:38	SELECT * FROM College.user LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
✓	2	14:21:44	SELECT DISTINCT device_type FROM College.device LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
✓	3	14:21:47	SELECT content FROM College.message LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
✓	4	14:21:54	SELECT fname, email_id FROM College.user WHERE age>16 AND activity_stats = ...	5 row(s) returned	0.000 sec / 0.000 sec

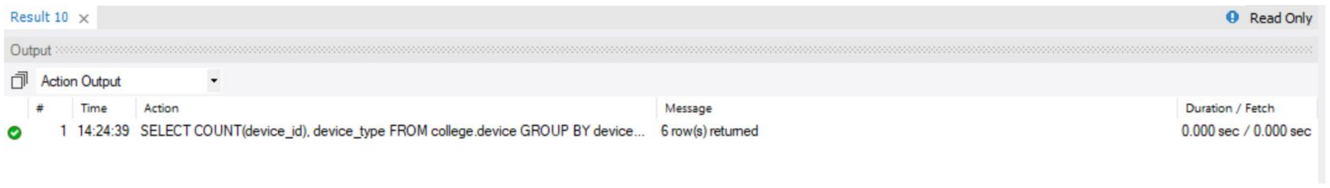
String Operation

Yash Mahajan SE IT B 04 191061



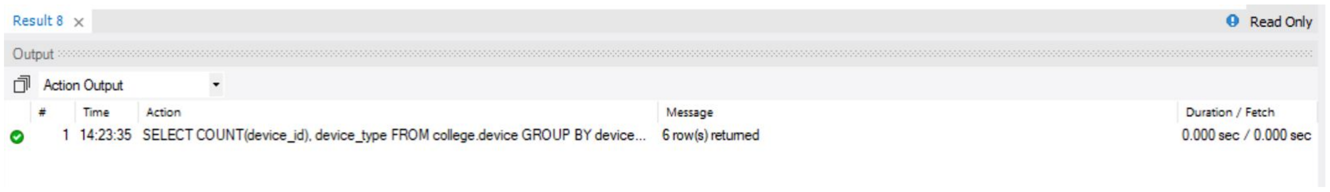
```
device 7 ×                                                                                   Apply
Output
Action Output
    #   Time      Action                                                                      Message              Duration / Fetch
✓   1   14:22:58  SELECT * FROM College.device WHERE device_type LIKE '%PC-Windows%' LIMIT...   2 row(s) returned    0.000 sec / 0.000 sec
```

Group by



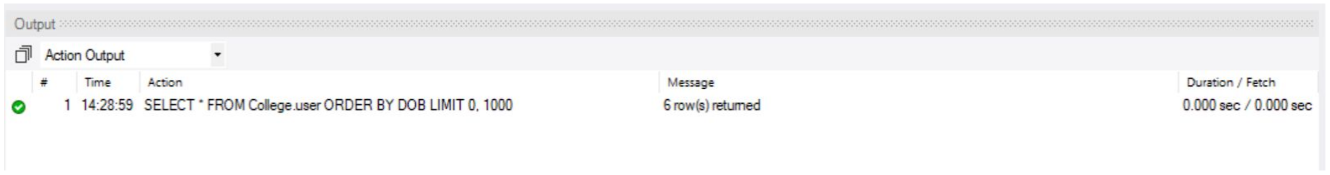
```
Result 10 ×                                                                                  Read Only
Output
Action Output
    #   Time      Action                                                                      Message              Duration / Fetch
✓   1   14:24:39  SELECT COUNT(device_id), device_type FROM college.device GROUP BY device...   6 row(s) returned    0.000 sec / 0.000 sec
```

Having Clause



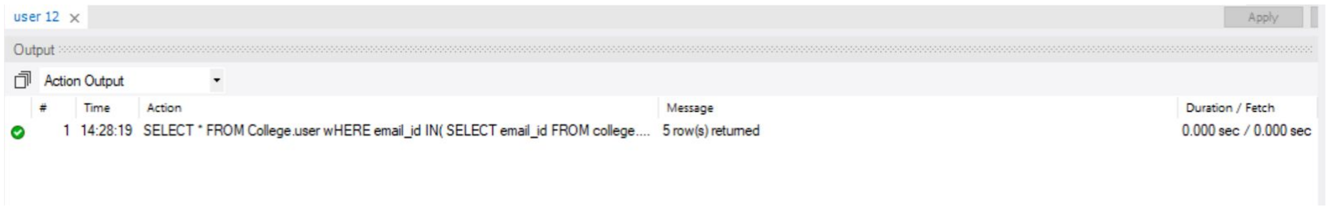
```
Result 8 ×                                                                                   Read Only
Output
Action Output
    #   Time      Action                                                                      Message              Duration / Fetch
✓   1   14:23:35  SELECT COUNT(device_id), device_type FROM college.device GROUP BY device...   6 row(s) returned    0.000 sec / 0.000 sec
```

Order By



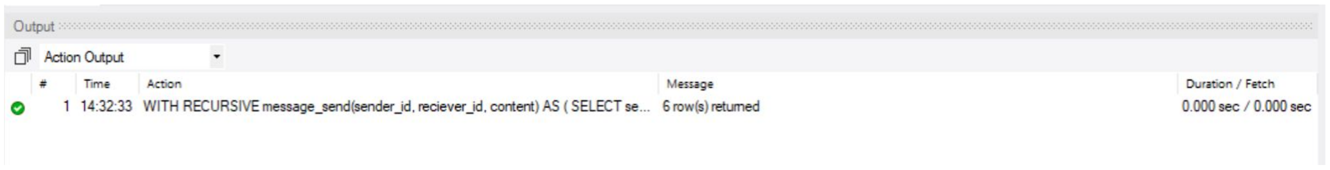
```
Output
Action Output
    #   Time      Action                                                                      Message              Duration / Fetch
✓   1   14:28:59  SELECT * FROM College.user ORDER BY DOB LIMIT 0, 1000                       6 row(s) returned    0.000 sec / 0.000 sec
```

Nested Query



```
user 12 ×                                                                                    Apply
Output
Action Output
    #   Time      Action                                                                      Message              Duration / Fetch
✓   1   14:28:19  SELECT * FROM College.user wHERE email_id IN( SELECT email_id FROM college....  5 row(s) returned    0.000 sec / 0.000 sec
```

Recursive Query



```
Output
Action Output
    #   Time      Action                                                                      Message              Duration / Fetch
✓   1   14:32:33  WITH RECURSIVE message_send(sender_id, reciever_id, content) AS ( SELECT se...  6 row(s) returned    0.000 sec / 0.000 sec
```

8. Post Experimental Exercise

A. Questions:

Difference between having and group by clause. Etc.

Having	Group By
It is used for applying some extra condition to the query.	The group by clause is used to group the data according to particular column or row.
Having cannot be used without group by clause.	Groupby can be used without having clause with the select statement.
The having clause can contain aggregate functions.	It cannot contain aggregate functions.
It restricts the query output by using some conditions	It groups the output on basis of some rows or columns.

B. Conclusion:

In this experiment we have written SQL scripts to implement various complex queries for retrieving data from the database like SELECT, GROUP BY, HAVING, ORDER BY, string functions, nested queries and recursive queries.

The ORDER BY keyword is used to sort the result-set in ascending or descending order. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

In many cases we want to apply the aggregate functions to subgroups of tuples in a relation, where the subgroups are based on some attribute values. SQL has a GROUP BY clause for this purpose. The GROUP BY clause specifies the grouping attributes, which should also appear in the SELECT clause, so that the value resulting from applying each aggregate function to a group of tuples appears along with the value of the grouping attribute(s).

Sometimes we want to retrieve the values of these functions only for groups that satisfy certain conditions. SQL provides
a HAVING clause, which can appear in conjunction with a GROUP BY clause, for this purpose. HAVING provides a condition on the summary information regarding the group of tuples associated with each value of the grouping attributes. Only the groups that satisfy the condition are retrieved in the result of the query.

SQL provides a mechanism for nesting subqueries. A subquery is a select-from-where expression that is nested within another query. A common use of subqueries is to perform tests for set membership, make set comparisons, and determine set cardinality by nesting subqueries in the where clause.

A recursive CTE is useful in querying hierarchical data such as organization charts where one employee reports to a manager or multi-level bill of materials when a product consists of many components, and each component itself also consists of many other components.

C. **References:**

[1] Elmasri and Navathe, “Fundamentals of Database Systems”, 5th Edition, PEARSON Education.
[2] Korth, Silberchatz, Sudarshan, “Database System Concepts”, 6th Edition, McGraw – Hill
[3] https://www.w3schools.com/sql/sql_default.asp