Birla Institute of Technology & Science, Pilani, K. K. BIRLA Goa campus
Database Systems (CS F212) Second Semester 2019-2020
Lab-4: To study Group by and Having clause and Nested queries

Group by and Having clause

Group by and having clauses facilitate selective retrieval of rows. They act on record sets and not on individual records.

Syntax: SELECT <column name> FROM <table name> WHERE <condition> **GROUP BY** <column name>

Syntax: SELECT <column name> FROM <table name> WHERE <condition> GROUP BY <column name> **HAVING** <condition>

Examples: //to find out how many students each hostel has in year 2008. mysql> SELECT hostelno, COUNT(*) FROM students where year = 2008 GROUP BY hostelno;

//to find hostel numbers having more than 100 students in year 2008. mysql> SELECT hostelno, COUNT(*) FROM students where year = 2008 GROUP BY hostelno HAVING COUNT(*)>100;

• Observe which condition has put in where clause and which in having clause.

• The GROUP BY clause creates a data set, containing several sets of records grouped together based on condition.

• The HAVING clause can be used in conjunction with the GROUP BY. It imposes a condition on the GROUP BY clause, which further filters the groups created by the GROUP BY clause.

• The columns in group by and having clause must appear in select clause.

Nested Queries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow −

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Subqueries with the SELECT Statement

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows −

```
SELECT column_name [, column_name ]
FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
   (SELECT column_name [, column_name ]
   FROM table1 [, table2 ]
   [WHERE])
```

Example:
Consider the CUSTOMERS table having the following records −

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  35 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Now, let us check the following subquery with a SELECT statement.

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

This would produce the following result.

```
+----+----------+-----+---------+----------+
| ID | NAME     | AGE | ADDRESS | SALARY   |
+----+----------+-----+---------+----------+
|  4 | Chaitali |  25 | Mumbai  |  6500.00 |
|  5 | Hardik   |  27 | Bhopal  |  8500.00 |
|  7 | Muffy    |  24 | Indore  | 10000.00 |
+----+----------+-----+---------+----------+
```

IN operator

The IN operator allows you to specify multiple values in a WHERE clause. INoperator allows you to determine if a specified value matches any value in aset of values returned by a subquery.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

OR

```
SELECT column_name(s)
FROM table_name
```

```
WHERE column_name IN (SELECT STATEMENT);
```

Example:

```
mysql> SELECT 5 IN(10,5,21);
+-----------------+
| 5 IN(10,5,21) |
+-----------------+
|               5 |
+-----------------+
1 row in set (0.00 sec)
```

NOT IN Operator

Example:

```
mysql> SELECT isbn FROM Book AS B1 WHERE ('C', 'Koffman') NOT IN (SELECT title,
author FROM Book AS B2 WHERE B1.isbn=B2.is
bn );
+-------+
| isbn  |
+-------+
| A1235 |
| A1236 |
| A1238 |
+-------+
3 rows in set (0.00 sec)
```

ANY operator

● ANY operator returns TRUE if the comparison is TRUE for ANY of the values returned by the subquery.

ALL operator

● ALL operator returns TRUE if the comparison is TRUE for ALL of the values returned by the subquery.

Example: print details of the customer who is older than some other customer.

```
mysql> select * from Customers where age > any (select age from
Customers);
+-----+--------+------------------------------+------+
| cid | cname  | address                      | age  |
+-----+--------+------------------------------+------+
| c1  | Amar   | 23, M.G. road, Ahmadabad     |   20 |
| c2  | Akbar  | D-20, Sainivas, Mumbai       |   19 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 |
| c4  | Saloni | Hyderabad                    |   22 |
+-----+--------+------------------------------+------+
4 rows in set (0.00 sec)
```

Example: print details of the customer whose age is greater than or equal to all other customers.

```
mysql> select * from Customers where age >= all( select age from Customers);
+-----+-------+------------------------------+------+
| cid | cname | address                      | age  |
+-----+-------+------------------------------+------+
| c3  | Pooja | sector no. 23, Vashi, Mumbai |   24 |
+-----+-------+------------------------------+------+
1 row in set (0.00 sec)
```

Correlated Subqueries

● A correlated subquery is a subquery that uses the data from the outer query. In other words, a correlated subquery depends on the outer query.
● A correlated subquery is evaluated once for each row in the outer query.

EXISTS operator

The EXISTS operator is used to test for the existence of any record in a subquery. It returns true if the subquery returns one or more records.

Example: print details of customers who have ordered atleast one book.

```
mysql> SELECT * from Customers AS T1 WHERE EXISTS (SELECT * FROM OrderBook
AS T2 WHERE T1.cid=T2.ocid);
+-----+--------+-------------------------------+------+
| cid | cname  | address                       | age  |
+-----+--------+-------------------------------+------+
| c1  | Amar   | 23, M.G. road, Ahmadabad      |   20 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 |
| c4  | Saloni | Hyderabad                     |   22 |
| c5  | John   | Pune, Shivajinagar            |   18 |
+-----+--------+-------------------------------+------+
4 rows in set (0.00 sec)
```

<u>NOT EXISTS operator</u>

Example:

```
mysql> SELECT * from Customers AS T1 WHERE NOT EXISTS (SELECT * FROM
OrderBook AS T2 WHERE T1.cid=T2.ocid);
+-----+-------+------------------------+------+
| cid | cname | address                | age  |
+-----+-------+------------------------+------+
| c2  | Akbar | D-20, Sainivas, Mumbai |   19 |
+-----+-------+------------------------+------+
1 row in set (0.00 sec)
```

The above query returns all the records of customers whose details are not present in OrderBook or in other words who didn’t order any books.