DHA SUFFA UNIVERSITY

Department of Computer Science

CS-2003 Database Systems

Spring 2024

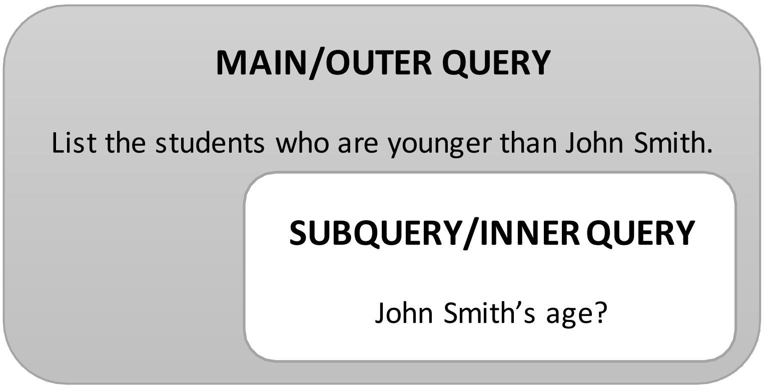
LAB 08 Subqueries I

OBJECTIVE(S)

- Learn about subqueries.

SQL SUBQUERIES

A subquery (inner/nested) query is a query within a query. It can be embedded within the SELECT, FROM, or WHERE clause but are usually added within the WHERE clause of another SELECT statement (also known as an outer query or outer select). Subqueries can be nested inside another SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery.



MAIN/OUTER QUERY

List the students who are younger than John Smith.

SUBQUERY/INNER QUERY

John Smith's age?

The subquery executes once before its parent query so that its result can be passed to the outer query. The outer query then uses that result for its execution.

SUBQUERIES WITH SELECT STATEMENTS

This is the one of the most common application of a subquery. The subquery is used in the WHERE clause of the main query such that the output of the subquery is used for comparison in the main query. If a subquery returns a NULL value to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.

```
• SELECT col_name(s) FROM tb_name
          WHERE expression operator
                    (SELECT col_name FROM tb_name
                              WHERE condition(s))
          AND/OR expression operator
                    (SELECT col_name FROM tb_name
                              WHERE condition(s));
```

-
```
SELECT col_name(s), aggregateFunc() FROM tb_name
        GROUP BY col_names()
        HAVING expression operator
                (SELECT col_name FROM tb_name
                        WHERE condition(s))
        AND/OR expression operator
                (SELECT col_name FROM tb_name
                        WHERE condition(s));
```

For example:

```
SELECT firstName, lastName, age FROM students
        WHERE age <
                (SELECT age FROM students
                        WHERE firstName = “John” AND lastName = “Smith”);
```

Rules for Writing a Subquery

- Subqueries must be enclosed in parentheses.
- Subqueries must be placed on the right side of the comparison operator.
- A subquery can have only one column in the SELECT clause unless there are multiple columns in the main query to compare the subquery columns with.
- Subqueries that return more than one row can only be used with multiple value operators such as IN, ANY, and ALL.
- The BETWEEN operator cannot be used with the subquery. However, it can be used within the subquery.

TYPES OF SUBQUERIES

Single Row Subqueries

Also called scalar subqueries, these subqueries return zero or one row to the outer query. Such subqueries can be placed in a WHERE clause, HAVING clause, or a FROM clause of a SELECT STATEMENT. Single-value comparison operators such as (=, <, >=, <> etc.) can be used to compare against the result of a single-row subquery. If the subquery attempts to pass multiple rows to these comparison operators, then an error will occur.

Multiple Row Subqueries

These subqueries return one or more rows to the outer query. Operators such as IN, ANY, and ALL are used in the outer query to handle subqueries that return multiple rows.

- **IN** operator is used to compare a value to a set of values.
```
        SELECT col_name(s) FROM tb_name
                WHERE col_name IN
                        (SELECT col_name FROM tb_name
                                WHERE condition(s));
```

- **ANY** operator is used to compare a value to any value in a list. True is returned when a column value matches any of the values returned by the subquery. Single-value comparison operators (=, <, >=, <> etc.) must be placed before **ANY** in the query.

  ```
  SELECT col_name(s) FROM tb_name
          WHERE col_name = ANY
                  (SELECT col_name FROM tb_name
                          WHERE condition(s));
  ```

- **ALL** operator is used to compare a value to all values in a list. True is returned only if the comparison is true for all of the values returned by the subquery. Like with ANY operator, single-value comparison operators (=, <, >=, <> etc.) must be placed before **ALL** in the query.

  ```
  SELECT col_name(s) FROM tb_name
          WHERE col_name = ALL
                  (SELECT col_name FROM tb_name
                          WHERE condition(s));
  ```

Note: The NOT operator can be used with all of the above operators to perform their logical opposite operations.

Multiple Column Subqueries

These subqueries return multiple columns to the outer query allowing us to combine duplicate WHERE conditions into a single WHERE clause. In such a case, we need to write a compound WHERE clause such that the number of columns for comparison in the WHERE clause should be equal to the number of columns returned by the subquery. The IN, ANY, and ALL operators can be used for comparison like in multiple row subqueries.

-
  ```
  SELECT col_name(s) FROM tb_name
          WHERE (col1_name, col2_name) IN
                  (SELECT col1_name, col2_name FROM tb_name
                          WHERE condition(s));
  ```

TASK

- Display the name, marks and department of the student with the highest marks in each department.

LAB ASSIGNMENT

1. Display the name, designation, and salary of the highest paid employee without using the ORDER BY clause.
2. Display the name and experience of the employee with the maximum experience.
3. Display the names and salaries of all the employees whose salary is greater than the average salary of the company.
4. Display the highest paid employee of each department.

SUBMISSION GUIDELINES

- Take a screenshot of each task. Ensure that all screenshots have a white background and black text. You can alter the background and text colors through the properties of the MySQL command line client.
- Place all the screenshots in a single word file labeled with Roll No and Lab No. e.g. **‘cs181xxx_Lab08’**
- Convert the file into PDF.
- Submit the PDF file at LMS
- **-100%** policies for plagiarism.