# MYSQL Session

- Agenda
  - Correlated Subqueries
  - Other Schema Objects  - views, Index & synonyms

# Correlated Subqueries

- Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query.

- A correlated subquery is evaluated once for each row processed by the parent statement. The parent statement can be a **SELECT**, **UPDATE**, or **DELETE** statement.

- A correlated subquery is one way of reading every row in a table and comparing values in each row against related data.

- It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query.

- you can use a correlated subquery to answer a multipart question whose answer depends on the value in each row processed by the parent statement.

- A correlated subquery in MySQL is a subquery that depends on the outer query. It uses the data from the outer query or contains a reference to a parent query that also appears in the outer query. MySQL evaluates it once from each row in the outer query.

- A correlated subquery is a subquery that uses values from the outer query, requiring the inner query to execute once for each outer query.

- A correlated subquery is related to a regular subquery in that it uses an inner query to feed result values to the outer query. A correlated subquery executes the outer query multiple times, once for each row returned by the inner query; it is processed by joining a column in the subquery to a column in the parent query.

- A subquery in MySQL is a query, which is nested into another SQL query and embedded with SELECT, INSERT, UPDATE or DELETE statement along with the various operators.

- We can also nest the subquery with another subquery. A subquery is known as the **inner query**, and the query that contains subquery is known as the **outer query**.

- The inner query executed first gives the result to the outer query, and then the main/outer query will be performed. MySQL allows us to use subquery anywhere, but it must be closed within parenthesis.

- All subquery forms and operations supported by the SQL standard will be supported in MySQL also.

## The following are the rules to use subqueries:

- Subqueries should always use in **parentheses.**
- If the main query does not have multiple columns for subquery, then a subquery can have only one column in the SELECT command.
- We can use various comparison operators with the subquery, such as >, <, =, IN, ANY, SOME, and ALL. A multiple-row operator is very useful when the subquery returns more than one row.
- We cannot use the **ORDER BY** clause in a subquery, although it can be used inside the main query.
- If we use a subquery in a **set function**, it cannot be immediately enclosed in a set function.

## The following are the advantages of using subqueries:

- The subqueries make the queries in a structured form that allows us to isolate each part of a statement.
- The subqueries provide alternative ways to query the data from the table; otherwise, we need to use complex joins and unions.
- The subqueries are more readable than complex join or union statements.

# MySQL Subquery Syntax

Syntax:-

**SELECT** column_list (s) **FROM** table_name

**WHERE** column_name OPERATOR

(**SELECT** column_list (s) **FROM** table_name [**WHERE**])

Example:-

**SELECT** emp_name, city, income **FROM** employees

**WHERE** emp_id IN (**SELECT** emp_id **FROM** employees);

# Other Schema Objects - views, Index & synonyms

- Apart from tables, other essential schema objects are view, sequences, indexes and synonyms.
- A **view** is a logical or virtual table. Synonyms are simply alias names for database objects.
- **Synonyms** also simplify query writing and provide an element of system security by disguising the actual name of a database object.
- **Sequences** are special database objects that support the automatic generation of integer values, and are often used to generate primary key values for tables.
- **Indexes** are created on table columns to facilitate the rapid retrieval of information from tables.

# Views

- A database view is a logical or virtual table based on a query. Views are queried just like tables.
- This means that from your perspective as a developer or from a database system user's perspective, a view looks like a table.
- The definition of a view as an object is stored within a database's data dictionary; however, a view stores no data itself.
- A database also stores the execution plan for creating a view-this means that data can be retrieved rapidly through use of a view even though the actual data presented by a SELECT query of a view is not stored as part of a view.
- Rather, the data is "gathered together" each time that a view is queried from the database tables for which a view is defined-these are termed base tables.

- The general syntax is given below.

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW
[ViewName]
[(Column Alias Name...)]
AS [Query]
[WITH [CHECK OPTION] [READ ONLY] [CONSTRAINT]]
```

- Types of Views
  - Simple View
  - Complex view