ORMEDIAN RESEARCH INSTITUTE

TOPIC:

DATABASE

SUBTOPIC: SQL FUNDAMENTALS (PART 4)

By:

ADEKOLA OLUWASEUN O.

DECEMBER 27, 2022

CONTENT COVERED

- ***REFERENCED TABLES***
- ***USE OF SQL ANY and ALL***
- ***SELECT INTO VS INSERT INTO***
- ***SQL CASES***
- ***NULL() AND IFNULL() FUNCTIONS IN SQL***
- ***COALESCE () FUNCTION***
- ***STORED PROCEDURES IN SQL***
- ***COMMENTS IN SQL***

IN THIS PRESNTATION, THE FOLLOWING TABLES WILL BE USED AS OUR REFERENCES AT EVERY LEVEL OF WRITING QUERIES

TABLENAME: ECE501

STUDENTID	SCORE	DEPARTMENT	LEVEL
160211012	65	ECE	500
160221014	89	MECH	500
160231044	89	CPE	500
160211059	89	ECE	500
160221088	53	MECH	500
160211014	88	ECE	500

TABLENAME: ECE507

STUDENTID	SCORE	CURRENT_CGPA	LEVEL	DEPARTMENT
160221015	89	4.67	500	ECE
160211012	89	4.67	500	ECE
160211014	83	4.6	500	MECH
160231044	81	4.6	500	CPE
160211059	89	4.63	500	ECE
160211088	89	4.6	500	ECE

THE USE OF SQL ANY and ALL OPERATORS

***ANY Operator**: it is used for comparing a value to values in a list or subquery results and evaluates to true if the result-set of inner query contains at least one row. ANY operator must also be preceded by comparison operator. Hence its result is of Boolean.*

***ALL Operator**: It is also used for comparing a value to any value in a subquery result-set. It returns a Boolean result if all the subqueries meet the condition. ALL operator must be preceded by comparison operator for the right result to be evaluated. It is also important to understand that ALL operator is used with SELECT, WHERE, HAVING statement.*

To summarize, both ANY and ALL operators give us the privilege to values within a single column or a range of values.

IMPLEMENTATION OF ANY Operator IN SQL

SCENARIO 1

For example, let's assume that we need to list STUDENTID from the ECE501 table if it finds ANY records in the ECE507 table that has SCORE equal to 89.

CODE EXAMPLE

```
SELECT STUDENTID FROM ECE501 WHERE STUDENTID=ANY (SELECT STUDENTID FROM ECE507 WHERE SCORE=89);
```

SCENARIO 2

Assume we need to list STUDENTID in ECE507 if it finds any records in ECE501 table where SCORE is equal to or greater that 75.

CODE EXAMPLE

```
SELECT STUDENTID FROM ECE507 WHERE STUDENTID=ANY(SELECT STUDENTID FROM ECE501 WHERE STUDENTID>=75);
```

IMPLEMENTATION OF ALL OPERATORS

SCENARIO

Let's say we need to list all the STUDENTID from ECE501 if all the records in the ECE507 has SCORE equal to 89.

CODE EXAMPLE:

```
SELECT STUDENTID FROM ECE501 WHERE STUDENTID=ALL (SELECT STUDENTID FROM ECE507 WHERE SCORE=89);
```

It is worth noting that the scenario above will return FALSE as the output because the SCORE column has many different values. But if the column has same values all through, the output is definitely going to be true

HOW "SELECT INTO " and ""INSERT INTO" STATEMENTS WORK

SELECT INTO: *The functionality of SELECT INTO statement is to copy data from one table to another table entirely.*

It can be used to copy one or more column data from one table to another table.

For example we may decide to copy the CURRENT_CGPA column existing in ECE507 into a new table called grades.

It is worth noting that this statement works on databases differently, SELECT INTO is supported in SQL server but not in MySQL. So, in this presentation, techniques to achieve results in both SQL and MySQL will be implemented.

INSERT INTO: *This statement will be used with SELECT statement in this section to achieve the same result that SQL has achieved with only SELECT INTO. The use of INSERT INTO and SELECT statement can be used together to copy one or more column from one table to another.*

IMPLEMENTATION OF SELECT INTO

SCENARIO 1

Let's say we need to copy CURRENT_CGPA and STUDENTID column records of ECE507 table into the new table called GRADES.

CODE EXAMPLE:

```
SELECT  CURRENT_CGPA, STUDENTID INTO GRADES FROM ECE507;
```

This command will automatically create a new table called grade and then copy those selected into the GRADES table.

This approach can be used to copy as many columns values as possible from one table to another. This works perfectly well in SQL.

We may also specify certain range from which values should be copied from one table to another.

NOTE: *the approach implemented above is not supported in MySQL but another approach is required instead.*

IMPLEMENTATION OF INSERT INTO and SELECT FOR copying data from one table to another

Another technique of implementing the data copying from one table to another is the approach that involves using INSERT INTO together with SELECT statement in SQL. This approach works fine on MySQL as well.

To achieve, we need to first create a table whose name can later be called in the query for copying data.

For example:

CREATE TABLE GRADES LIKE ECE5O7;

This will enable the new table to have the structure of the existing table from which we want to copy data from.

This step can then be followed by further queries.

INSERT INTO GRADES (CURRENT_CGPA, STUDENTID) SELECT CURRENT_CGPA, STUDENTID FROM ECE507;

This query will return the result that will copy those two fields into the GRADES table whereas other columns which were not mentioned will have NULL values.

SQL CASES Expression

SQL CASES is like an if-then-else statement which works in such a way that if a CASE goes through conditions, it will return a value if such condition is met but if not, it will return the value in the ELSE condition. But NULL is returned if ELSE condition is not part of the of the specified conditions.

To understand SQL CASES better, we will use ECE501 table to implement SQL Cases.

```
SELECT STUDENTID, SCORE,
 CASE
 WHEN SCORE >=70 THEN "GRADE A ACHIEVED"
WHEN SCORE >=60 THEN "GRADE B ACHIEVED"
WHEN SCORE >=50 THEN "GRADE C ACHIEVED"
WHEN SCORE >=45 THEN "GRADE C ACHIEVED"
ELSE "GRADE ACHIEVED IS D"
END AS GradeCategory FROM ECE501;
```

IMPLEMENTATION OF SQL CASES

SCENARIO

We already know that values in the score field are not the same in ECE501, but we want to use SQL CASES to specify the scores into categories like A, B, C etc.

To understand SQL CASES better, we will use ECE501 table to implement SQL Cases.

```
SELECT STUDENTID, SCORE,
 CASE
 WHEN SCORE >=70 THEN "GRADE A ACHIEVED"
WHEN SCORE >=60 THEN "GRADE B ACHIEVED"
WHEN SCORE >=50 THEN "GRADE C ACHIEVED"
WHEN SCORE >=45 THEN "GRADE C ACHIEVED"
ELSE "GRADE ACHIEVED IS D"
END AS GradeCategory FROM ECE501;
```

SQL NULL AND IFNULL FUNCTIONS

SQL NULL FUNCTION *it is used to allow for optional value to be inserted into a field. If the field is NULL, it means it can be left blank and updated later. But* ***IFNULL*** *function is basically used to return an alternative value in the place where there is NULL value in a given field. IF NULL () function becomes very essential when field values are to be used for mathematical computation. The advantage of IFNULL() function is that it helps to ensure that output of mathematical computation using the field values is never NULL but a value is returned instead.*

Let's say we need to convert the individual score of each student to be 75% of the actual score, this means that we need to multiply each score by 0.75. What if one of the students score has not been computed but left as NULL, if we try automatic conversion for all these fields values in this case, there is possibility of having NULL being returned where students scores are missing.

IMPLEMENTATION OF SQL IFNULL() FUNCTION

*In this implementation, we will use this table named "**Evaluator**" to see how IFNULL() Function works. We have three columns STUDENTID, SCORE, and FACTOR.*

STUDENTID	SCORE	FACTOR
160211014	93	0.75
160211067	*NULL*	0.75
160211068	88	0.75
160211069	75	0.75
160211070	*NULL*	0.75

IMPLEMENTATION OF SQL IFNULL() FUNCTION

.....Continuation

From the table in the previous slide, if our decision is to obtain 75% equivalent conversion of the individual SCORE in the Evaluator table such that NULL is not returned in the final evaluation, then we need to implement it as shown below:

CODE EXAMPLE

```
SELECT STUDENTID, (FACTOR*IFNULL(SCORE, 0)) FROM Evaluator;
```

COALESCE() FUNCTION

COALESCE() FUNCTION: *this is a function in SQL that performs the same function as that of IFNULL function. It gives alternative value to NULL value in a given field within a table.*

The implementation code sample for solving the previous example as shown in the previous slide alternative to the use of IFNULL():

CODE EXAMPLE

```
SELECT STUDENTID, (FACTOR*COALESCE(SCORE, 0)) FROM Evaluator;
```

RESULT FROM THE IMPLEMENTATION OF IFNULL()/COALSECE() FUNCTION

STUDENTID	(FACTOR*IFNULL(SCORE, 0))
160211014	69.75
160211067	0
160211068	66
160211069	56.25
160211070	0

SQL STORED PROCEDURES

This is simply prepared with SQL and can be saved for reusability purpose. There are some cases where we might need a particular code to be used over and over again. Instead of writing such code repeatedly, Stored procedures can save us the stress of doing that. Once we have saved such code into stored procedures, then we can just call it to execute it. Hence it is simply a set of statement that performs some defined actions.

We have to do two things to achieve this, first wee need to create procedure through which we can pass our code (action to be performed). Secondly, we need to write a query to execute the code.

IMPLEMENTATION OF STORED PROCEDURE

Implementation of this stored procedure works differently on different databases.

Assuming we need to create a reusable program that will always return the records of students that scored 89 in ECE501.

On SQL Server, it could be written as;

```
CREATE PROCEDURE
selectrecords
AS
SELECT *FROM ECE501 WHERE SCORE=89;
GO
EXEC Selectallrecords;
GO
```

On MySQL, it could be written as:

```
DELIMITER //

CREATE PROCEDURE selectrecords()
BEGIN
                SELECT *FROM ECE501 WHERE
SCORE=89;
END //

DELIMITER ;

CALL selectrecords;
```

UNDERSTANDING WHAT COMMENT IS IN SQL

Comment in SQL is used to achieve two things:

(i) To explain sections of SQL Statement

(ii) To prevent execution of SQL statements

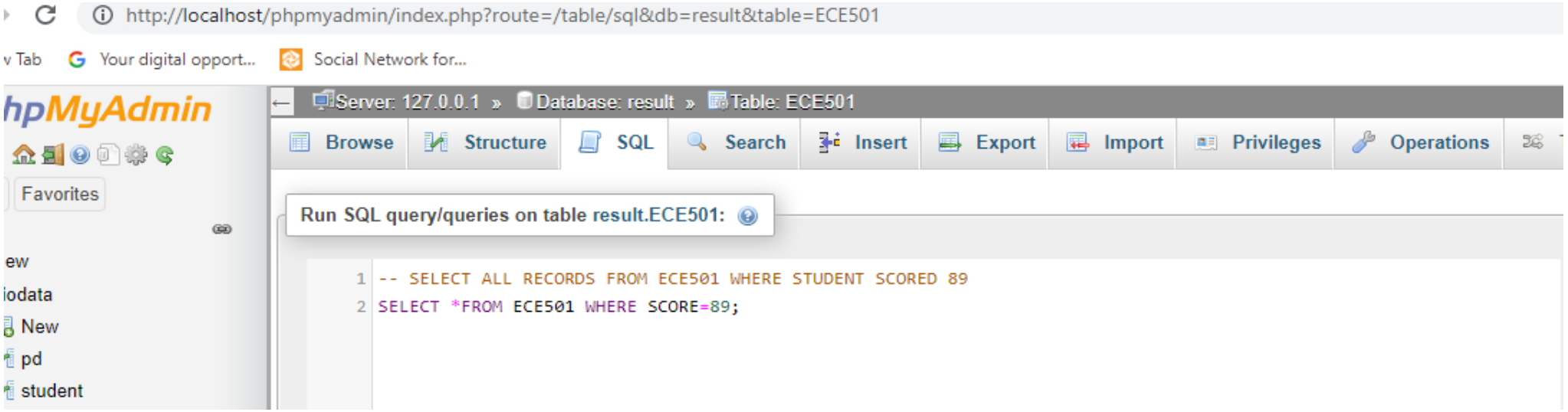
To explain sections of codes, the comment is prefixed with two hyphens as demonstrated below.

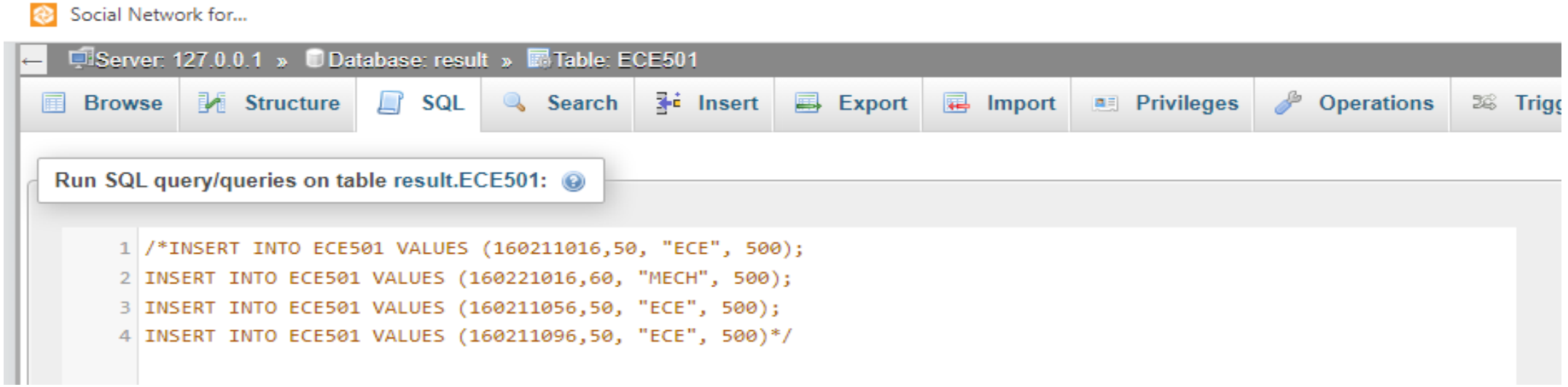
-- SELECT ALL RECORDS FROM ECE501 WHERE STUDENT SCORED 89

In the comment above, we only stated that only records for where scores = 89 should be outputted

To prevent code execution, the previous approach is still applicable but in the case where we have multiline codes which we need to prevent from getting executed, we have to use multiline comment which starts any line with / and ends it with */*

WHAT COMMENT LOOKS LIKE UPON EXECUTION





THANKS FOR VIEWING

NOTE: ***Subsequent topics under SQL will be discussed in Part 5***

Also, all Implementation as far as this presentation is concerned are in line with MySQL Syntax. All have been tested on XAMPP Server.

THE NEXT PART WILL REVOLVE AROUND DEEPER UNDERSTANDING OF HOW TO BUILD SQL DATABASES