SUCCEED WE MUST

DATABASE PROGRAMING BIT 2212

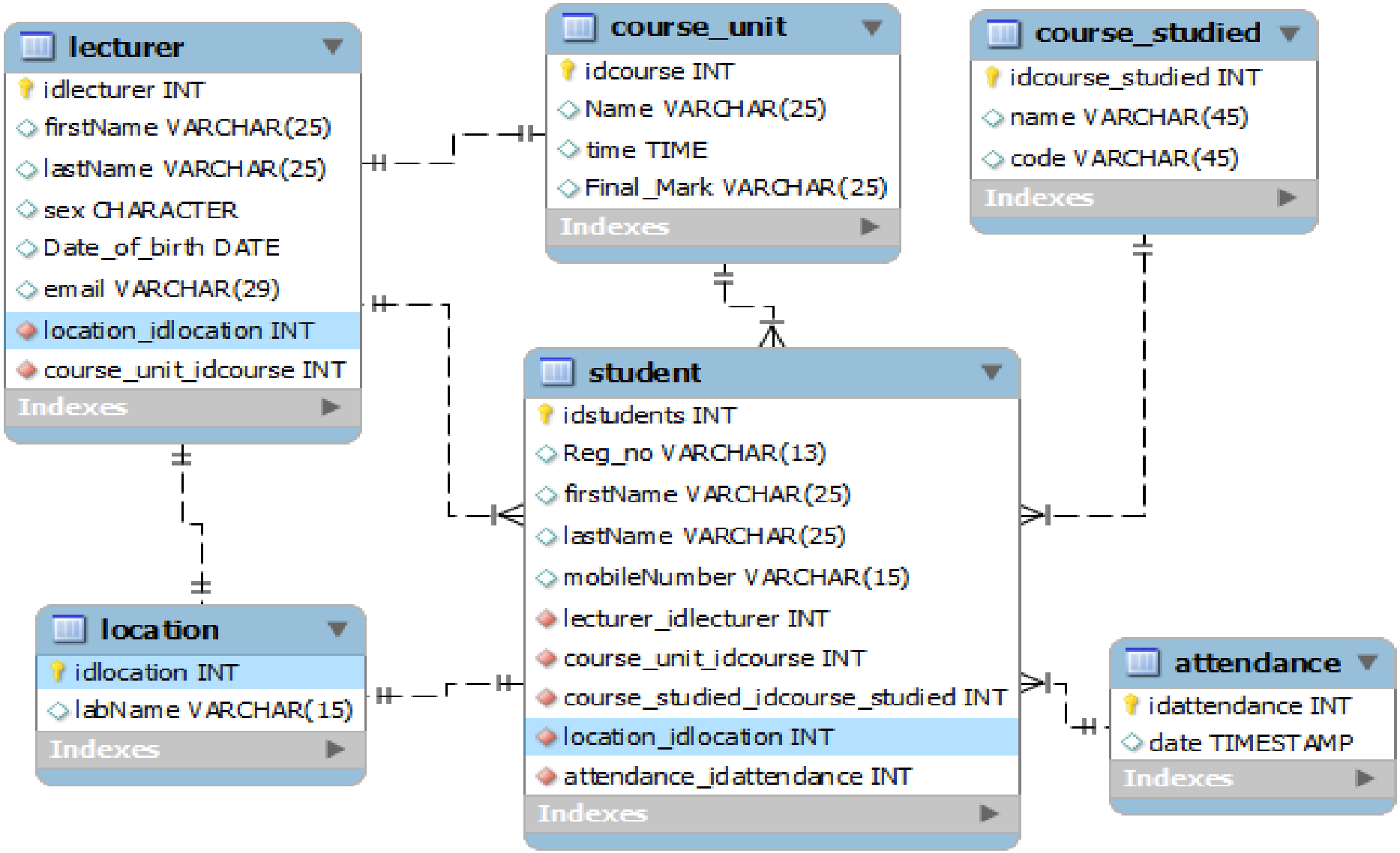
JOINS, UNIONS, SUB QUERRIES
Lecture 4

Mwavu Rogers
Email: mwavurogers@gmail.com
Phone 0773426328 and 0700497421
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF INFORMATION TECHNOLOGY

SUCCEED WE MUST

SUCCEED WE MUST

Using 20 minutes create a database having the following tables and make sure you insert data in the created tables



SUCCEED WE MUST

Joining Tables Together

- Joining tables together with SQL is one of its most important functions.

- If you have designed and created a good Database that is fully normalised, it will contain relationships between the tables via Primary and Foreign key links.

- Unless you know the correct SQL you will not be able to retrieve data as you desire.

SUCCEED WE MUST

What Is SQL join?

- SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.
- The most common type of join is: **SQL INNER JOIN (simple join)**. An SQL INNER JOIN returns all rows from multiple tables where the join condition is met.

Types of Joins

- Inner Join

- The INNER JOIN selects all rows from both tables as long as there is a match between the columns in both tables.

- Left outer Joins

The LEFT JOIN returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

- Natural join

Types of SQL Joins-cont……

- Right outer Joins

The RIGHT JOIN returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

- Full outer Joins

The FULL OUTER JOIN returns all rows from the left table (table1) and from the right table (table2).

TYPES OF JOINS- INNER JOIN

The Inner Join

This is the most common type of join. Inner joins combine records from two tables whenever there are matching values in a field common to both tables.

NOTE

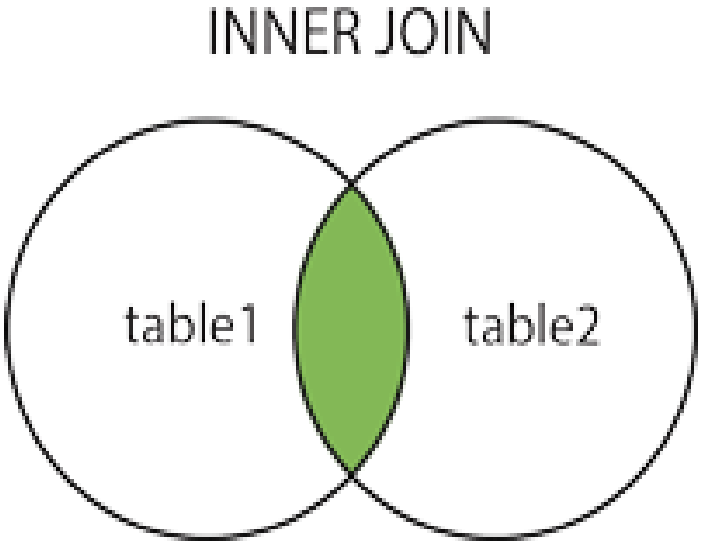
- If there are no matching records between two tables then the record is omitted.

SUCCEED WE MUST

INNER JOIN

The Inner Join combine records from two tables whenever there are matching values in a field common to both tables.

The diagram below illustrates how the inner join relates Tables.



Inner Join

- There are two way which an INNER JOIN can be written they are as follows:

```
SELECT table1.column1,table2.column1 FROM
table1,table2

WHERE table1.primary_key = table2.foreign_key;
```

```
SELECT table1.column1,table2.column1 FROM table1
INNER JOIN table 2 where table1.primary_key =
table2.foreign_key;
```

- NOTE: Both statements will return the exact same result set.

Example of two tables from a Database

```
mysql> select * from lecturers;
Empty set (0.10 sec)

mysql> select * from students;
+--------+----------+------------+-------+
| std_id | fname    | lname      | marks |
+--------+----------+------------+-------+
|      1 | Amai     | Clovis     | 69    |
|      2 | Namanya  | Beth       | 95    |
|      3 | deric    | Gumoshable | 90    |
|      4 | Mirembe  | Samuel     | 96    |
|      5 | Kamukama | Cathy      | 80    |
|      6 | Rukundo  | Agira      | 90    |
|      7 | Mustafa  | Nahabwe    | 20    |
|      8 | dickens  | sensa      | 90    |
|      9 | dickens  | tayebwa    | 89    |
+--------+----------+------------+-------+
9 rows in set (0.04 sec)

mysql> select * from courseunit;
+----------+------------------------+-------+-------+---------+
| courseid | CourseName             | Marks | Grade | std_id  |
+----------+------------------------+-------+-------+---------+
|        1 | database programming   |    80 | A     |       1 |
|        2 | OOP with java 2        |    70 | B     |       2 |
|        3 | Scripting Language P   |     2 | F     |       3 |
+----------+------------------------+-------+-------+---------+
3 rows in set (0.00 sec)
```

SUCCEED WE MUST

Inner Join: Both syntax can yield the same results

- Selecting the student names, course names marks and grade

```
mysql> select students.fname,students.lname,
    -> courseunit.coursename,courseunit.marks,
    -> courseunit.grade from students,courseunit
    -> where students.std_id =courseunit.std_id;
+---------+-----------+----------------------+------+-------+
| fname   | lname     | coursename           | marks | grade |
+---------+-----------+----------------------+------+-------+
| Amai    | Clovis    | database programming |    80 | A     |
| Namanya | Beth      | OOP with java 2      |    70 | B     |
| deric   | Gumoshable | Scripting Language P |     2 | F     |
+---------+-----------+----------------------+------+-------+
3 rows in set (0.00 sec)

mysql> select students.fname,students.lname,
    -> courseunit.coursename,courseunit.marks,
    -> courseunit.grade from students inner join courseunit
    -> where students.std_id =courseunit.std_id;
+---------+-----------+----------------------+------+-------+
| fname   | lname     | coursename           | marks | grade |
+---------+-----------+----------------------+------+-------+
| Amai    | Clovis    | database programming |    80 | A     |
| Namanya | Beth      | OOP with java 2      |    70 | B     |
| deric   | Gumoshable | Scripting Language P |     2 | F     |
+---------+-----------+----------------------+------+-------+
3 rows in set (0.00 sec)
```

Inner Join – Alias Names

SUCCEED WE MUST

- Note

- SQL aliases are used to give a database table, or a column in a table, a temporary name.

That the syntax used is long when writing the table names repeatedly. We can therefore give the tables “Aliases” for efficiency. This is done in the FROM clause by typing the table name then the alias name:

- **SELECT t1.column1, t2.column1 FROM table1 t1, table2 t2**

- Select the students, courseunit,

```
mysql> select s.fname,s.lname,c.coursename,c.marks,c.grade
    -> from students s,courseunit c
    -> where s.std_id =c.std_id;
+---------+-----------+-----------------------+-------+-------+
| fname   | lname     | coursename            | marks | grade |
+---------+-----------+-----------------------+-------+-------+
| Amai    | Clovis    | database programming  |    80 | A     |
| Namanya | Beth      | OOP with java 2       |    70 | B     |
| deric   | Gumoshable| Scripting Language P  |     2 | F     |
+---------+-----------+-----------------------+-------+-------+
3 rows in set (0.00 sec)
```

More examples

- Select the Lecturer, course unit, time and total number of students who do a particular course

```
mysql> select l.firstname,l.lastname,c.name,c.time,
    -> count(student_id) as TotalstdNo
    -> from lecturer l,course_unit c, students s
    -> where l.lecturer= s.lecturer_id
    -> AND l.lecturer= c.course_id;
+-----------+----------+--------------------------+----------+------------+
| firstname | lastname | name                     | time     | TotalstdNo |
+-----------+----------+--------------------------+----------+------------+
| Kimera    | Richard  | Client Server Programing | 10:00:00 |         14 |
+-----------+----------+--------------------------+----------+------------+
1 row in set (0.00 sec)
```

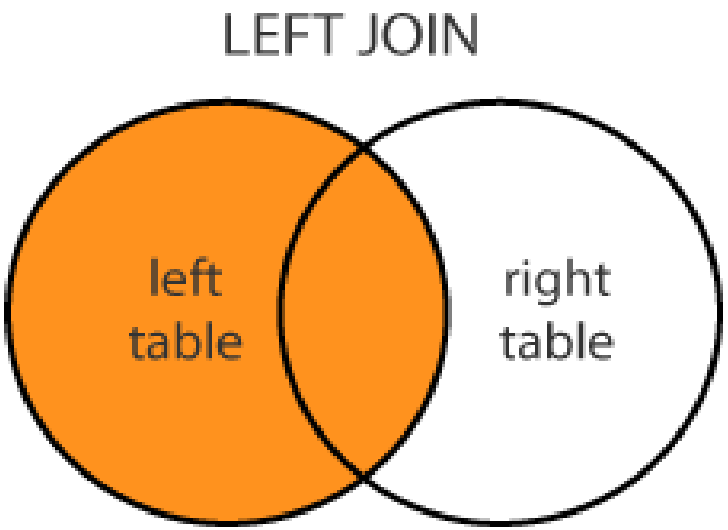
Inner Join – Joining multiple tables

- select l.firstname, l.lastname, l.email, ll.labname,cu.name
- from lecturerlocation ll inner join lecturer l inner join
- course_unit cu
- on l.location_idlocation=ll.idlocation
- and l.course_unit_idcourse = cu.idcourse;

```
mysql> select l.firstname, l.lastname, l.email, ll.labname,cu.name
    -> from lecturerlocation ll inner join lecturer l inner join
    -> course_unit cu
    -> on l.location_idlocation=ll.idlocation
    -> and l.course_unit_idcourse = cu.idcourse;
+-----------+-----------+----------------+---------+---------------+
| firstname | lastname  | email          | labname | name          |
+-----------+-----------+----------------+---------+---------------+
| Ronah     | Emma      | re@gmail.com   | lab2    | DBprogramming |
| Aidah     | Tumusiime | at@gmail.com   | lab2    | DBprogramming |
| Gloria    | Ninsiima  | ivan@gmail.com | lab1    | DBprogramming |
+-----------+-----------+----------------+---------+---------------+
3 rows in set (0.01 sec)
```

SQL LEFT JOIN

- LEFT JOIN performs a join starting with the first (left-most) table and then any matching second (right-most) table records.
- LEFT JOIN and LEFT OUTER JOIN are the same.



SUCCEED WE MUST

LEFT OUTER JOIN

- The **LEFT OUTER JOIN** returns all records from the Left Hand table (first table) even if there is no record in the second table (RIGHT), that fulfils the criteria. If there are no matching records in the second table it returns the row that are not matching as NULL

```
mysql> select students.fname,students.lname,courseunit.coursename
    -> ,courseunit.marks,courseunit.grade from students left join courseunit
 on students.std_id=courseunit.std_id;
+----------+------------+----------------------+-------+-------+
| fname    | lname      | coursename           | marks | grade |
+----------+------------+----------------------+-------+-------+
| Amai     | Clovis     | database programming |    80 | A     |
| Namanya  | Beth       | OOP with java 2      |    70 | B     |
| deric    | Gumoshable | Scripting Language P |     2 | F     |
| Mirembe  | Samuel     | NULL                 |  NULL | NULL  |
| Kamukama | Cathy      | NULL                 |  NULL | NULL  |
| Rukundo  | Agira      | NULL                 |  NULL | NULL  |
| Mustafa  | Nahabwe    | NULL                 |  NULL | NULL  |
| dickens  | sensa      | NULL                 |  NULL | NULL  |
| dickens  | tayebwa    | NULL                 |  NULL | NULL  |
+----------+------------+----------------------+-------+-------+
9 rows in set (0.00 sec)
```

Left Outer Joins

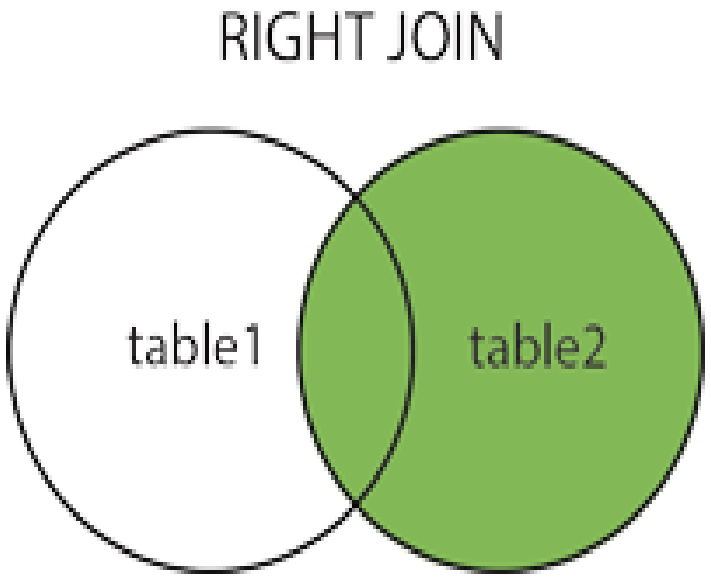
- The **LEFT OUTER JOIN** returns all records from the Left Hand table (first table) even if there is no record in the second table (RIGHT), that fulfils the criteria.

```
mysql> select l.firstname,l.lastname,l.sex,c.name,c.time
    -> from lecturer l
    -> LEFT JOIN courseunit c
    -> ON l.idlecturer=c.idcourseunit;
+-----------+----------+--------+-----------------------------+----------+
| firstname | lastname | sex    | name                        | time     |
+-----------+----------+--------+-----------------------------+----------+
| Mwavu     | Rogers   | Male   | Database Programming        | 08:00:00 |
| kimera    | Richard  | Male   | Computer Programming        | 11:00:00 |
| Karungi   | Dickson  | male   | Object Oriented Programming | 08:00:00 |
| kabarungi | moreen   | female | User interface              | 09:00:00 |
+-----------+----------+--------+-----------------------------+----------+
4 rows in set (0.00 sec)
```

SUCCEED WE MUST

Right Outer Joins

- A RIGHT OUTER JOIN returns all records from the right hand table (second) even if there in no reference to it in the left hand table (first).



SUCCEED WE MUST

THE RIGHT JOIN

- The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

```
mysql> select lecturer.firstname,lecturer.lastname,course_unit.name,
    -> course_unit.time from lecturer right outer join course_unit
    -> on lecturer.lecturer=course_unit.course_id;
+-----------+----------+--------------------------+----------+
| firstname | lastname | name                     | time     |
+-----------+----------+--------------------------+----------+
| Mwavu     | Roger    | Computer Graphics        | 08:00:00 |
| Kimera    | Richard  | Client Server Programing | 10:00:00 |
| Baguma    |  Kenneth | Database Programming     | 02:00:00 |
| Kabarungi |  Moreen  | OO Programming-Java      | 09:00:00 |
| Natumanya |  Deborah | GIS                      | 07:00:00 |
| NULL      | NULL     | Computer Programing      | 08:30:00 |
+-----------+----------+--------------------------+----------+
6 rows in set (0.00 sec)
```

Right Outer Joins Cont..

- Compare the output on the previous slide and the out put on this slide

```
mysql> select course_unit.name,course_unit.time,lecturer.firstname
    -> ,lecturer.lastname from course_unit right outer join lecturer
    -> on course_unit.course_id=lecturer.lecturer;
+--------------------------+----------+-----------+----------+
| name                     | time     | firstname | lastname |
+--------------------------+----------+-----------+----------+
| Computer Graphics        | 08:00:00 | Mwavu     | Roger    |
| Client Server Programing | 10:00:00 | Kimera    | Richard  |
| Database Programming     | 02:00:00 | Baguma    |  Kenneth |
| OO Programming-Java      | 09:00:00 | Kabarungi |  Moreen  |
| GIS                      | 07:00:00 | Natumanya |  Deborah |
+--------------------------+----------+-----------+----------+
5 rows in set (0.00 sec)
```

Right Outer Joins

- A RIGHT OUTER JOIN returns all records from the right hand table (second) even if there in no reference to it in the left hand table (first).

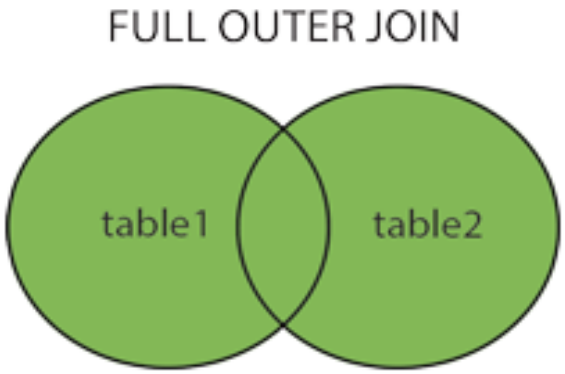
- Selecting students who have marks for all the taught course units

```
mysql> SELECT S.FirstName, S.LastName, C.name, C.Final_Mark
    -> FROM Student S
    -> RIGHT OUTER JOIN course_unit C
    -> ON S.idstudents=C.idcourse;
+-----------+----------+-----------------------+------------+
| FirstName | LastName | name                  | Final_Mark |
+-----------+----------+-----------------------+------------+
| Mukasa    | stephen  | Database Programming | 78         |
| Mukwaya   | Ambrose  | Computer Programming | 78         |
+-----------+----------+-----------------------+------------+
2 rows in set (0.00 sec)
```

SUCCEED WE MUST

FULL OUTER JOIN

- The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).



SQL FULL OUTER JOIN Syntax

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

Natural Join : Guidelines

- A **NATURAL JOIN** is a **JOIN** operation that creates an implicit **join** clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables.

- **Natural Join : Guidelines**

  - The associated tables have one or more pairs of identically named columns
  - The columns must be the same data type.
  - Don't use ON clause in a natural join.

Syntax

```
SELECT * FROM table1 NATURAL JOIN table2;
```

Example

Sample table : foods

ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18

Sample table : company

COMPANY_ID	COMPANY_NAME	COMPANY_CITY
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

an example of SQL natural join between two tables

- **SELECT** * **FROM** foods NATURAL JOIN company;

Output

COMPANY_ID	ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_NAME	COMPANY_CITY
16	1	Chex Mix	Pcs	Akas Foods	Delhi
15	6	Cheez-It	Pcs	Jack Hill Ltd	London
15	2	BN Biscuit	Pcs	Jack Hill Ltd	London
17	3	Mighty Munch	Pcs	Foodies.	London
15	4	Pot Rice	Pcs	Jack Hill Ltd	London
18	5	Jaffa Cakes	Pcs	Order All	Boston

SUCCEED WE MUST

Natural Join

```
+-----------+-----------+--------+-----------------------------+----------+
| firstname | lastname  | sex    | name                        | time     |
+-----------+-----------+--------+-----------------------------+----------+
| Mwavu     | Rogers    | Male   | Database Programming        | 08:00:00 |
| kimera    | Richard   | Male   | Database Programming        | 08:00:00 |
| Karungi   | Dickson   | male   | Database Programming        | 08:00:00 |
| kabarungi | moreen    | female | Database Programming        | 08:00:00 |
| Mwavu     | Rogers    | Male   | Computer Programming        | 11:00:00 |
| kimera    | Richard   | Male   | Computer Programming        | 11:00:00 |
| Karungi   | Dickson   | male   | Computer Programming        | 11:00:00 |
| kabarungi | moreen    | female | Computer Programming        | 11:00:00 |
| Mwavu     | Rogers    | Male   | Object Oriented Programming | 08:00:00 |
| kimera    | Richard   | Male   | Object Oriented Programming | 08:00:00 |
| Karungi   | Dickson   | male   | Object Oriented Programming | 08:00:00 |
| kabarungi | moreen    | female | Object Oriented Programming | 08:00:00 |
| Mwavu     | Rogers    | Male   | User interface              | 09:00:00 |
| kimera    | Richard   | Male   | User interface              | 09:00:00 |
| Karungi   | Dickson   | male   | User interface              | 09:00:00 |
| kabarungi | moreen    | female | User interface              | 09:00:00 |
+-----------+-----------+--------+-----------------------------+----------+
16 rows in set (0.00 sec)
```

Unions

- The UNION command is used to select related information from two queries, much like the JOIN command. However, when using the UNION command all selected columns need to be of the same number and same **datatype**.
- The UNION command will return all distinct rows retrieved by either query.

Syntax:

```
Select column1, column2,
FROM table1
UNION
Select column1, column2
FROM table2
```

Union

```
mysql> select firstname,lastname from lecturer
    -> union
    -> select firstname,lastname from students;
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Mwavu     | Rogers   |
| kimera    | Richard  |
| Karungi   | Dickson  |
| kabarungi | moreen   |
| Mwavu     | Henry    |
| Karungi   | Dorcus   |
| Tebandeke | Lawrence |
| okello    | sam      |
| kiwuka    | Ritah    |
| Mumbere   | Nobert   |
| Atwine    | Evans    |
| Niwamanya | Javira   |
| Tebandeke | Musa     |
| Ahumuza   | Gladys   |
| Tebandeke | Amon     |
| Ahumuza   | Claire   |
+-----------+----------+
16 rows in set (0.00 sec)
```

```
mysql> select firstname from students
    -> union
    -> select firstname from lecturer;
+-----------+
| firstname |
+-----------+
| Tebandeke |
| okello    |
| kiwuka    |
| Mumbere   |
| Atwine    |
| Niwamanya |
| Ahumuza   |
| Mwavu     |
| kimera    |
| Karungi   |
| kabarungi |
+-----------+
11 rows in set (0.00 sec)
```

UNION ALL

- The UNION ALL command is used much the same way like the UNION command except that it returns all records including duplicates:

- **Syntax:** Select column1, column2, FROM table

- UNION ALL Select column1, column2 FROM table2

UNION ALL

```
mysql> select firstname from students
    -> union all
    -> select firstname from lecturer;
+-------------+
| firstname   |
+-------------+
| Tebandeke   |
| okello      |
| kiwuka      |
| Mumbere     |
| Atwine      |
| Niwamanya   |
| Tebandeke   |
| Ahumuza     |
| Tebandeke   |
| Ahumuza     |
| Mwavu       |
| kimera      |
| Karungi     |
| kabarungi   |
| Mwavu       |
| Karungi     |
+-------------+
16 rows in set (0.00 sec)
```

Sub Queries

- A Nested / Sub query is a SELECT statement nested inside a SELECT, SELECT...INTO, INSERT...INTO, DELETE, or UPDATE statement or inside another query.

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

- **Note**

- 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.

Sub Queries Cont.……

- A single row sub query is one that returns a single row to the outer SELECT statement.
- A multiple row sub query is one that returns more than one row to the outer SELECT statement.
- **Example 1 – Single row sub query: To return information for a lecturer teaching a certain course**

```
mysql> select firstname,lastname from lecturer where
    -> idlecturer IN
    -> (select idcourseunit from courseunit where name='Database Programming');
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Mwavu     | Rogers   |
+-----------+----------+
1 row in set (0.00 sec)
```

Sub Queries

- Example 2: Multiple Row Sub query

- To return, all students studying a specific course

```
mysql> select c.name, s.firstname,s.lastname from course_studied c, student s
    -> where c.idcourse_studied
    -> IN (SELECT course_unit_idcourse from student where course_unit_idcourse=2);
+------+------------+-------------+
| name | firstname  | lastname    |
+------+------------+-------------+
| BCE  | Mukasa     | stephen     |
| BCE  | Mukwaya    | Ambrose     |
| BCE  | Jenifer    | Tukamushabe |
| BCE  | Ziyal      | Ambrose     |
| BCE  | Byaruhanga | Gordon      |
| BCE  | Allen      | Nayiga      |
| BCE  | Ayebare    | Sheira      |
+------+------------+-------------+
7 rows in set (0.00 sec)
```

More examples

An example

Consider a table CUSTOMERS_BackUP with similar structure as CUSTOMERS table. Now lets copy the complete CUSTOMERS table into the CUSTOMERS_BackUP table

```
mysql> insert into customer_backup
    -> select * from customer where id in(select id from customer where salary>90000);
Query OK, 2 rows affected (0.12 sec)
Enregistrements: 2  Doublons: 0  Avertissements: 0
```

The result is;

```
mysql> select * from customer_backup;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
| 11 | COSTA    |  24 | NTUNGAMO  |  95000   |
| 12 | MORATA   |  24 | KABALE    | 100000   |
+----+----------+-----+-----------+----------+
2 rows in set (0.00 sec)
```

Subqueries with the UPDATE Statement

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

The basic syntax is;

```
UPDATE table
SET column_name = new_value
WHERE column_name OPERATOR VALUE
(SELECT COLUMN_NAME
FROM TABLE_NAME)
WHERE condition(s));
```

SUCCEED WE MUST

An example;

Using the table CUSTOMER_BackUP which is backup of CUSTOMER table. We can update it for example update SALARY by 0.25 times in the CUSTOMERS table for all the customers whose AGE is greater than or equal to 23.

```
mysql> SELECT * FROM CUSTOMER_BACKUP;
+----+--------+------+----------+--------+
| ID | NAME   | AGE  | ADDRESS  | SALARY |
+----+--------+------+----------+--------+
| 11 | COSTA  |   48 | NTUNGAMO |  47500 |
| 12 | MORATA |   96 | KABALE   |  50000 |
+----+--------+------+----------+--------+
2 rows in set (0.00 sec)
```

```
mysql> select * from customer;
+----+--------+------+-----------+--------+
| ID | NAME   | AGE  | ADDRESS   | SALARY |
+----+--------+------+-----------+--------+
| 10 | DIEGO  |   22 | RUKUNGIRI |  90000 |
| 11 | COSTA  |   48 | NTUNGAMO  |  47500 |
| 12 | MORATA |   96 | KABALE    |  50000 |
| 13 | LUKAKU |   26 | KAMPALA   |   5000 |
+----+--------+------+-----------+--------+
4 rows in set (0.00 sec)
```

```
mysql> UPDATE CUSTOMER SET SALARY=SALARY*0.25 WHERE ID IN(SELECT ID FROM CUSTOMER_BACKUP WHERE AGE>=23);
Query OK, 2 rows affected (0.25 sec)
Enregistrements correspondants: 2  Modifi├®s: 2  Warnings: 0

mysql> SELECT * FROM CUSTOMER;
+----+--------+------+-----------+--------+
| ID | NAME   | AGE  | ADDRESS   | SALARY |
+----+--------+------+-----------+--------+
| 10 | DIEGO  |   22 | RUKUNGIRI |  90000 |
| 11 | COSTA  |   48 | NTUNGAMO  |  11875 |
| 12 | MORATA |   96 | KABALE    |  12500 |
| 13 | LUKAKU |   26 | KAMPALA   |   5000 |
+----+--------+------+-----------+--------+
4 rows in set (0.00 sec)
```

Subqueries with the DELETE Statement

The basic syntax is as follows.

```
DELETE FROM TABLE_NAME
WHERE column_name OPERATOR
(SELECT COLUMN_NAME
FROM TABLE_NAME)
WHERE condition(s);
```

An example

Using the table CUSTOMER_BackUP which is backup of CUSTOMER table .The following example deletes the records from the CUSTOMERS table for all the customers whose AGE is greater than or equal to 48.

```
mysql> select * from customer;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
| 10 | DIEGO    |  22 | RUKUNGIRI |  90000   |
| 11 | COSTA    |  48 | NTUNGAMO  |  47500   |
| 12 | MORATA   |  96 | KABALE    |  50000   |
| 13 | LUKAKU   |  26 | KAMPALA   |   5000   |
+----+----------+-----+-----------+----------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM CUSTOMER_BACKUP;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
| 11 | COSTA    |  48 | NTUNGAMO  |  47500   |
| 12 | MORATA   |  96 | KABALE    |  50000   |
+----+----------+-----+-----------+----------+
2 rows in set (0.00 sec)
```

```
mysql> DELETE FROM CUSTOMER WHERE ID IN(SELECT ID FROM CUSTOMER_BACKUP WHERE AGE>=48)
Query OK, 2 rows affected (0.05 sec)

mysql> SELECT * FROM CUSTOMER;
+----+--------+-----+-----------+--------+
| ID | NAME   | AGE | ADDRESS   | SALARY |
+----+--------+-----+-----------+--------+
| 10 | DIEGO  |  22 | RUKUNGIRI |  90000 |
| 13 | LUKAKU |  26 | KAMPALA   |   5000 |
+----+--------+-----+-----------+--------+
2 rows in set (0.00 sec)
```

SUCCEED WE MUST

Next Lecture

Views and Triggers