SUCCEED WE MUST

DATABASE PROGRAMING BIT-2212

Lecture 3 – February 2018
Data Manipulation Language and functions

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

SUCCEED WE MUST

Today's Content - Data Manipulation Language

- Data Manipulation Language
  - Insert statement
  - Select Statement
  - Update Statement
  - Where clause
- **FUNCTIONS**

Insert Statement syntax

- It is possible to write the INSERT INTO statement in two forms.
- The first form does not specify the column names where the data will be inserted, only their values:

INSERT INTO *table_name* VALUES(*value1*,*value2*,*value3*,...);

The second form specifies both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*,*column2*,*column3*,...)
VALUES (*value1*,*value2*,*value3*,...);

Insert Statement: Examples

- Example one: Inserting one record in a table using the first syntax;
  INSERT INTO *table_name* VALUES *value1*,*value2*,*value3*,...);

```
mysql> insert into students values(2,"Mabirizi","Vicent",22,"Tripple B");
Query OK, 1 row affected (0.19 sec)
```

Example two: Inserting multiple records in a table using the

```
mysql> insert into attendance (idattendance,date)  values (2,20130304030809),(3,201311030
90345);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Insert Statement: Examples

- Example 3: Inserting three records

```
mysql> insert into location (labName)  values ("Lab 1"),("Lab 2"),("Lab 3");
ERROR 1062 (23000): Duplicate entry '0' for key 'PRIMARY'
mysql> insert into location (idlocation,labName)  values (1,"Lab 1"),(2,"Lab 2"),(3,"Lab
3");
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select *from location;
+------------+---------+
| idlocation | labName |
+------------+---------+
|          1 | Lab 1   |
|          2 | Lab 2   |
|          3 | Lab 3   |
+------------+---------+
3 rows in set (0.00 sec)
```

Example 4. How to insert values on an auto increment Field

```
mysql> insert into attendance (date) values (201309090808096);
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Insert Statement: Examples

- Example 5: Inserting records with Foreign keys, you have to first insert data in the reference tables

```
mysql> insert into lecturer values (1,"Richard","Kimera","Male","1980-09-07","rkimera@mus
t.ac.ug",2,1),(2,"Richard","Kalungi","Male","1988-08-03","kaldixo@must.ac.ug",1,1);
Query OK, 2 rows affected, 2 warnings (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 2
```

- Basing on our database structure, information is to be stored in the format
  - First insert in all the following tables attendance, location, course_studied and course_unit
  - Next insert in the lecturer table and lastly student table

Select Statement syntax

SELECT [DISTINCT]*column_list*

FROM *table_list*

[**WHERE***conditions*]

[**GROUPBY***group*[HAVING*group_conditions*]]

[**ORDERBY***sort_columns*]

[LIMIT [*beginning_row*,]*number_retrieved*];

Select Statement syntax

- Selecting all records from a table;

```
mysql> select *From course_unit;
+----------+-------------------------+----------+-------------+
| idcourse | Name                    | time     | Final_Mark  |
+----------+-------------------------+----------+-------------+
|        1 | Database Programming    | 09:00:10 | 78          |
|        2 | Computer Programming    | 09:00:00 | 78          |
+----------+-------------------------+----------+-------------+
2 rows in set (0.00 sec)
```

Select specific records from a table

```
mysql> select Name,Final_mark FROM CouRSe_UniT;
+-------------------------+--------------+
| Name                    | Final_mark   |
+-------------------------+--------------+
| Database Programming    | 78           |
| Computer Programming    | 78           |
+-------------------------+--------------+
2 rows in set (0.00 sec)
```

Select Distinct

- In a table, some of the columns may contain duplicate values. This is not a problem, however, sometimes you will want to list only the different (distinct) values in a table.
- The DISTINCT keyword can be used to return only distinct (different) values.
- **SQL SELECT DISTINCT Syntax**
  - SELECT DISTINCT column_name(s) FROM table_name;

```
mysql> select * from students;
+------------+-----------+-----------+-----+-----------+
| student_id | firstname | lastname  | age | hostel    |
+------------+-----------+-----------+-----+-----------+
|          1 | Rebecca   | Nuwasiima |  24 | NULL      |
|          2 | Mabirizi  | Vicent    |  22 | Tripple B |
|          3 | Otema     | Patrick   |  45 | Tripple B |
|          4 | Otema     | Mystic    |  45 | Tripple B |
+------------+-----------+-----------+-----+-----------+
4 rows in set (0.00 sec)
```

```
mysql> select distinct firstname from students;
+-----------+
| firstname |
+-----------+
| Rebecca   |
| Mabirizi  |
| Otema     |
+-----------+
```

SQL Where Clause

- The where clause is used to specify a criteria
- **Syntax:** SELECT column_name(s) FROM table_name **WHERE** column_name operator value

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

The Logical Operators OR,AND and NOT

Logical Operators	Description
OR	For the row to be selected at least one of the conditions must be true.
AND	For a row to be selected all the specified conditions must be true.
NOT	For a row to be selected the specified condition must be false.

SUCCEED WE MUST

The SQL AND, OR and NOT Operators

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition:
- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.
- The NOT operator displays a record if the condition(s) are NOT TRUE.

SUCCEED WE MUST

The AND operator

- The AND operator are used to filter records based on more than one condition:
- Returns all records which have certain names. AND is used if both sides of the where clause are to hold true

```
mysql> select firstname from students where firstname="Bwaana"
 and lastname="Antony";
+-----------+
| firstname |
+-----------+
| Bwaana    |
+-----------+
1 row in set (0.07 sec)
```

Example Two with the AND operator

```
mysql> select firstname from students where firstname="Mabirizi" and lastname="Vicent";
+-----------+
| firstname |
+-----------+
| Mabirizi  |
+-----------+
1 row in set (0.06 sec)
```

SQL Where clause - BETWEEN

- BETWEEN is used to specify a range

```
mysql> select student_id,firstname,lastname from students where student_id between 2 and 6;
+------------+-------------+----------+
| student_id | firstname   | lastname |
+------------+-------------+----------+
|          2 | Mabirizi    | Vicent   |
|          3 | Otema       | Patrick  |
|          4 | Otema       | Mystic   |
|          5 | Nawasiima   | Rebecca  |
|          6 | Ainembabazi | Caroline |
+------------+-------------+----------+
5 rows in set (0.08 sec)
```

The logical operator, OR

- The OR operator displays a record if any of the conditions separated by OR is TRUE.
- So If you want to select rows that satisfy at least one of the given conditions, you can use the logical operator, OR.
- Example

```
mysql> select firstname from students where firstname="Bwaana" or lastname="Antony";
+-----------+
| firstname |
+-----------+
| Bwaana    |
+-----------+
1 row in set (0.01 sec)

mysql> select lastname from students where firstname="Samia" or lastname="Antony";
+----------+
| lastname |
+----------+
| Antony   |
+----------+
1 row in set (0.00 sec)
```

SUCCEED WE MUST

The NOT operator

- If you want to find rows that do not satisfy a condition, you can use the logical operator, NOT.
- That is, if a condition is satisfied, then the row is not returned.

```
mysql> select firstname, lastname from students where Not lastname="Antony";
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Tumusime  | Isaac    |
| Turyahabwe| Blessing |
| Nabimanya | Arnold   |
| Kasozi    | Francis  |
| Ndagire   | Mariat   |
| Asiimire  | Mark     |
+-----------+----------+
6 rows in set (0.00 sec)
```

The LIKE operator

- The LIKE operator is used to list all rows in a table whose column values match a specified pattern.
- It is useful when you want to search rows to match a specific pattern, or when you do not know the entire value. For this purpose we use a wildcard character '%'.
- The **%** operator can be used to specify wildcards (matches one or more characters in a pattern) both before and after the pattern.
- The _ operator can be used also to specify wildcards(matches one character.
- The like operator is used to search for specified format of information.

SQL Where clause - LIKE

- Syntax: **SELECT** column_name(s) **FROM** table_name **WHERE** column_name LIKE pattern
- **EXAMPLES:** Select all first names that start with **letter A**

```
mysql> select firstname from students where firstname like "A%";
+-------------+
| firstname   |
+-------------+
| Ainembabazi |
+-------------+
1 row in set (0.00 sec)
```

EXAMPLES: Select all first names that start with letter T

- **EXAMPLES:** Select all first names that start with **letter T**

```
mysql> select firstname from students where firstname like 't%';
+-----------+
| firstname |
+-----------+
| Tumusime  |
| Turyahabwe |
+-----------+
2 rows in set (0.06 sec)
```

SQL Where clause - LIKE

- **EXAMPLES:** Select all first names that End with **letter O**

```
mysql> select fname from students where fname LIKE '%o';
+-------+
| fname |
+-------+
| Abaho |
| Abaho |
+-------+
2 rows in set (0.00 sec)
```

Where clause - LIKE

- **EXAMPLES:** Select all first names that Start with letter **A** and End with **letter O**

```
mysql> select fname from students where fname LIKE 'A_%o';
+-------+
| fname |
+-------+
| Abaho |
| Abaho |
+-------+
2 rows in set (0.00 sec)
```

SUCCEED WE MUST

Where Clause and the IN operator

- The IN operator allows you to specify multiple values in a WHERE clause.
- **Syntax:** SELECT column_name(s) FROM table_name WHERE column_name IN (value1,value2,...)
- Example 1

```
mysql> select fname,lname from students where lname IN('Ivan','Job');
+-------+-------+
| fname | lname |
+-------+-------+
| Epou  | Ivan  |
| Chebet | Job  |
+-------+-------+
2 rows in set (0.00 sec)
```

Example 2 with The IN-operator

```
mysql> select firstname from students where firstname IN("nabimanya","Ndagire");
+-----------+
| firstname |
+-----------+
| Nabimanya |
| Ndagire   |
+-----------+
2 rows in set (0.04 sec)

mysql> select firstname,lastname from students where firstname IN("nabimanya","Ndagire");
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Nabimanya | Arnold   |
| Ndagire   | Mariat   |
+-----------+----------+
2 rows in set (0.00 sec)
```

Where Clause: IN operator and LIMIT clause

- The LIMIT clause can request the **first "n" rows**, the first row is **zero**, not **one** WHERE and ORDER BY clauses happen *before* the LIMIT is applied.
- The LIMIT clause can request the **first "n" rows**, the first row is **zero**, not **one** WHERE and ORDER BY clauses happen *before* the LIMIT is applied

```
mysql> select fname,lname from students where lname IN('Ivan','Job')LIMIT 1;
+-------+-------+
| fname | lname |
+-------+-------+
| Epou  | Ivan  |
+-------+-------+
1 row in set (0.00 sec)

mysql> select fname,lname from students where lname IN('Ivan','Job')LIMIT 2;
+--------+-------+
| fname  | lname |
+--------+-------+
| Epou   | Ivan  |
| Chebet | Job   |
+--------+-------+
2 rows in set (0.00 sec)
```

Example 2

```
mysql> select firstname,lastname from students where firstname IN("nabimanya","Ndagire") limit 1;
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Nabimanya | Arnold   |
+-----------+----------+
1 row in set (0.00 sec)

mysql> select firstname,lastname from students where firstname IN("nabimanya","Ndagire") limit 2;
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Nabimanya | Arnold   |
| Ndagire   | Mariat   |
+-----------+----------+
2 rows in set (0.00 sec)
```

Order by Clause

- Order by is used to sort information in a table, it could be either in Ascending (ASC) order or descending (DESC) order.
- An example

```
mysql> select fname,lname from students order by lname DESC;
+------------+-----------+
| fname      | lname     |
+------------+-----------+
| Nuwagaba   | Ronald    |
| Ayebare    | Prossy    |
| Abaho      | Patience  |
| Ayebare    | Merab     |
| Tebandeke  | Lawrence  |
| Chebet     | Job       |
| Epou       | Joan      |
| Epou       | Ivan      |
| Ninsiima   | Eunice    |
| Abaho      | Dickson   |
+------------+-----------+
10 rows in set (0.00 sec)

mysql> select fname,lname from students order by fname desc;
+------------+-----------+
| fname      | lname     |
+------------+-----------+
| Tebandeke  | Lawrence  |
| Nuwagaba   | Ronald    |
| Ninsiima   | Eunice    |
| Epou       | Joan      |
| Epou       | Ivan      |
| Chebet     | Job       |
| Ayebare    | Prossy    |
| Ayebare    | Merab     |
| Abaho      | Dickson   |
| Abaho      | Patience  |
+------------+-----------+
10 rows in set (0.00 sec)
```

Update Statement syntax

- The UPDATE command works the same way as the corresponding Oracle command, but users can specify an ordering by using ORDER BY, and they can also specify a limit.
- SYNTAX: **UPDATE <table name> SET <attribute> = <expression> WHERE <condition>;**
- **Example 1.** Changing the first and last names of a specific record

```
mysql> Update student set FirstName="Mukasa", LastName="stephen" where idstudents=1;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select FirstName, LastName from student;
+-----------+-------------+
| FirstName | LastName    |
+-----------+-------------+
| Mukasa    | stephen     |
| Mukwaya   | Abdul       |
| Jenifer   | Tukamushabe |
| Ziyal     | Ambrose     |
+-----------+-------------+
```

Update Statement syntax

- MySQL also has a REPLACE command that can either update an old record or insert a new one, depending on whether the record exists already. The form is similar to the INSERT form.
- REPLACE INTO *tablename*(*colnames*) VALUES (*column_values*);

```
mysql> Replace into attendance (date) values (20130909040829);
Query OK, 1 row affected (0.00 sec)

mysql> select *from attendance;
+--------------+---------------------+
| idattendance | date                |
+--------------+---------------------+
|            1 | 2013-03-04 03:08:09 |
|            2 | 2013-03-04 03:08:09 |
|            3 | 2013-11-03 09:03:45 |
|            4 | 0000-00-00 00:00:00 |
|            5 | 2013-09-09 04:08:27 |
|            6 | 2013-09-09 04:08:29 |
|            7 | 2013-09-09 04:08:29 |
+--------------+---------------------+
7 rows in set (0.00 sec)
```

Functions

- Functions make the basic query far more powerful and are used to :
  - Perform calculations on data
  - Modify data items
  - Manipulate output for rows
  - Alter date formats for display
  - Convert column data types

Functions

SQL aggregate functions return a single value, calculated from values in a column. Some of these functions include

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

Functions

- **SQL scalar functions** return a single value, based on the input value. Some of these include:
- UCASE() - Converts a field to upper case
- LCASE() - Converts a field to lower case
- MID() - Extract characters from a text field
- LEN() - Returns the length of a text field
- ROUND() - Rounds a numeric field to the number of decimals specified
- NOW() - Returns the current system date and time
- FORMAT() - Formats how a field is to be displayed

Aggregate Functions: Examples

- Syntax for average:
  SELECT AVG(column_name) FROM table_name

```
mysql> select * from course_unit;
+----------+-------------------------+----------+-------------+
| idcourse | Name                    | time     | Final_Mark  |
+----------+-------------------------+----------+-------------+
|        1 | Database Programming    | 09:00:10 | 78          |
|        2 | Computer Programming    | 09:00:00 | 78          |
+----------+-------------------------+----------+-------------+
2 rows in set (0.00 sec)

mysql> select avg(Final_Mark) From course_unit;
+-----------------+
| avg(Final_Mark) |
+-----------------+
|              78 |
+-----------------+
1 row in set (0.04 sec)
```

counting Functions: Examples

- counting distinct columns from a table

```
mysql> select count(distinct FirstName) FROM student;
+-----------------------------+
| count(distinct FirstName) |
+-----------------------------+
|                           4 |
+-----------------------------+
1 row in set (0.00 sec)
```

The LENGTH() Function

- The LENGTH() function returns the length of the value in a text field.
- Syntax: SELECT LENGTH(column_name) FROM table_name;

```
mysql> select LENGTH(Name) FROM course_unit;
+-----------------+
| LENGTH(Name) |
+-----------------+
|              20 |
|              20 |
+-----------------+
2 rows in set (0.00 sec)
```

The FORMAT() Function

- The FORMAT() function is used to format how a field is to be displayed.
- Syntax: SELECT FORMAT(column_name,format) FROM table_name;
- column_name - The field to be formatted.
- Format - Specifies the format.

```
mysql> select name, Format(Final_Mark, 3) from course_unit;
+-----------------------+-------------------------+
| name                  | Format(Final_Mark, 3) |
+-----------------------+-------------------------+
| Database Programming | 78.000                  |
| Computer Programming | 78.000                  |
+-----------------------+-------------------------+
2 rows in set (0.00 sec)
```