M.A.N.U

3/7/2023

Mathematics Association of Nairobi University
isaak@students.uonbi.ac.ke

Lets create our first database

In [ ]:
```
CREATE DATABASE IF NOT EXISTS manu_sql;
```

We have successfully created database/schema using SQL command.

Show databases

In [ ]:
```
SHOW DATABASES;
```

Choose which database to use when evaluating commands

In [ ]:
```
USE manu_sql;
```

After selecting the database now we can query what tables are in the db

Show Tables

In [ ]:
```
SHOW TABLES;
```

Create a Table

The `CREATE TABLE` statement allows you to create a new table in a database.

1. First, you specify the name of the table that you want to create after the `CREATE TABLE` keywords. The table name must be unique within a database.
2. Second, you specify a list of columns of the table in the column_list section, columns are separated by commas.

MySQL CREATE TABLE Example

The following example creates a table called “Persons” that contains five columns: `PersonID`, `LastName`, `FirstName`, `Address`, and `City`:

In [ ]:
```
CREATE TABLE IF NOT EXISTS Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

The `PersonID` column is of type `int` and will hold an `integer`.

The `LastName`, `FirstName`, `Address`, and `City` columns are of type `varchar` and will hold `characters`, and the maximum length for these fields is `255` characters.

The empty “Persons” table will now look like this:

PersonID	LastName	FirstName	Address	City

Persons Table

See if table was created

In [ ]:
```
SHOW TABLES;
```

Describe a Table

We will use the `DESCRIBE` command to show the structure of our table, such as column names, constraints on column names, etc. The `DESC` command is a short form of the DESCRIBE command.

In [ ]:
```
DESCRIBE persons;
```

The empty " `Persons` " table can now be filled with data with the `SQL INSERT INTO` statement.

The `MySQL INSERT INTO` Statement

The `INSERT INTO` statement is used to insert new records in a table.

It is good practice to specify both the column names and the values to be inserted

`INSERT INTO` Example

In [ ]:
```
INSERT INTO persons (LastName, FirstName, Address, City)
VALUES("Muller","Thomas","00100","Munich");D
```

MySQL INSERT multiple rows statement:

1. First, specify the name of table that you want to insert after the INSERT INTO keywords.
2. Second, specify a comma-separated column list inside parentheses after the table name.
3. Third, specify a comma-separated list of row data in the VALUES clause. Each element of the list represents a row. The number of values in each element must be the same as the number of columns specified.

Insert Multiple rows example

Lets insert 5 records into our persons table

In [ ]:
```
INSERT INTO persons (PersonID, LastName, FirstName, Address, City)
VALUES(1,"Doe","Jane","0234","Nairobi"),
(2,"Einstein","Albert","1329", "Munich"),
(3,"Man","Bat","00001","New York"),
(4,"Margaret","Mitchelle","23344","Atlanta"),
(5,"Teresa","Mother","001324","Calcutta");
```

Lets see the number of rows our table contains

In [ ]:
```
SELECT COUNT(*) FROM persons;
```

MySQL `SELECT` Statement

The `SELECT` statement is used to select data from a database.

The `SELECT` and `FROM` are the keywords. By convention, you write the `SQL` keywords in uppercase. However, it's not mandatory. Because `SQL` is case-insensitive, you can write the `SQL` statement in lowercase, uppercase, etc

If you want to select all the fields available in the table, use the "*" wildcard syntax:

In [ ]:
```
SELECT * FROM persons;
```

`SELECT` Columns Example

The following `SQL` statement selects the " `LastName` ", " `FirstName` " and " `City` " from " `persons` " table

In [ ]:
```
SELECT
    LastName,
    FirstName,
    City
FROM
    persons;
```

LastName	FirstName	City
Doe	Jane	Kisumu
Einstein	Albert	Munich
Man	Bat	New York
Margaret	Mitchelle	Atlanta
Teresa	Mother	Calcutta
Muller	Thomas	Munich

The MySQL `SELECT DISTINCT` Statement

The `SELECT DISTINCT` statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

`SELECT` Example Without `DISTINCT`

In [ ]:
```
SELECT City FROM persons;
```

`SELECT DISTINCT` Example

In [ ]:
```
SELECT DISTINCT City from persons;
```

`CREATE` a Sample Table

Let's create a new table `notified_births_census` that records birth data for counties in Kenya.

In [ ]:
```
CREATE TABLE notified_births_census(
    id INT NOT NULL AUTO_INCREMENT,
    county_name VARCHAR(155),
    total_births INT,
    notified_births INT,
    not_notified_births INT,
    dont_know INT,
    not_stated INT,
    percent_notified DECIMAL(5,2),
    PRIMARY KEY(id)
);
```

In [ ]:
```
DESCRIBE notified_births_census;
```

8 records

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NA	auto_increment
county_name	varchar(155)	YES		NA	
total_births	int	YES		NA	
notified_births	int	YES		NA	
not_notified_births	int	YES		NA	
dont_know	int	YES		NA	
not_stated	int	YES		NA	
percent_notified	decimal(5,2)	YES		NA	

Let's insert some records.

In [ ]:
```
INSERT INTO notified_births_census (county_name, total_births, notified_births, not_notified_births, dont_know,
 VALUES
('KENYA', 1340468, 1212142, 125714, 2609, 3, 90.4),
('RURAL', 888039, 777343, 108563, 2131, 2, 87.5),
('URBAN', 452429, 434799, 17151, 478, 1, 96.1),
('MOMBASA', 37249, 35201, 2026, 22, NULL, 94.5),
('KWALE', 29226, 26455, 2719, 52, NULL, 90.5),
('KILIFI', 44519, 41950, 2509, 60, NULL, 94.2),
('TANA RIVER', 11683, 8541, 3106, 36, NULL, 73.1),
('LAMU', 4235, 3909, 324, 2, NULL, 92.3),
('TAITA/TAVETA', 9110, 8674, 435, 1, NULL, 95.2),
('GARISSA', 16414, 12198, 3986, 230, NULL, 74.3),
('WAJIR', 16767, 10777, 5921, 69, NULL, 64.3),
('MANDERA', 26639, 17395, 9027, 217, NULL, 65.3),
('MARSABIT', 13679, 9971, 3679, 29, NULL, 72.9),
('ISIOLO', 8037, 6518, 1496, 23, NULL, 81.1),
('MERU', 38222, 36649, 1532, 41, NULL, 95.9),
('THARAKA-NITHI', 9109, 8681, 417, 11, NULL, 95.3),
('EMBU', 14556, 14206, 345, 5, NULL, 97.6),
('KITUI', 27650, 24459, 3115, 75, 1, 88.5),
('MACHAKOS', 33548, 31726, 1783, 39, NULL, 94.6),
('MAKUENI', 20805, 19462, 1294, 49, NULL, 93.5),
('NYANDARUA', 16247, 15825, 417, 4, 1, 97.4),
('NYERI', 16831, 16614, 204, 13, NULL, 98.7),
('KIRINYAGA', 13638, 13459, 175, 4, NULL, 98.7),
('MURANGA', 24866, 24332, 529, 5, NULL, 97.9),
('KIAMBU', 69596, 67736, 1818, 42, NULL, 97.3),
('TURKANA', 24758, 17782, 6726, 250, NULL, 71.8),
('WEST POKOT', 24511, 16956, 7441, 114, NULL, 69.2),
('SAMBURU', 10665, 7561, 3080, 24, NULL, 70.9),
('TRANS NZOIA', 29005, 24817, 4125, 63, NULL, 85.6),
('UASIN GISHU', 32983, 30932, 1995, 56, NULL, 93.8),
('ELGEYO/MARAKWET', 13212, 12459, 742, 11, NULL, 94.3),
('NANDI', 23603, 21137, 2414, 52, NULL, 89.6),
('BARINGO', 19697, 16061, 3567, 69, NULL, 81.5),
('LAIKIPIA', 15383, 13400, 1969, 14, NULL, 87.1),
('NAKURU', 64797, 59771, 4923, 102, 1, 92.2),
('NAROK', 40643, 32520, 7980, 143, NULL, 80.0),
('KAJIADO', 36244, 32319, 3833, 92, NULL, 89.2),
('KERICHO', 24383, 22344, 2007, 32, NULL, 91.6),
('BOMET', 24647, 22848, 1752, 47, NULL, 92.7),
('KAKAMEGA', 49974, 46136, 3774, 64, NULL, 92.3),
('VIHIGA', 14329, 13581, 733, 15, NULL, 94.8),
('BUNGOMA', 47722, 43706, 3936, 80, NULL, 91.6),
('BUSIA', 25597, 23344, 2222, 31, NULL, 91.2),
('SIAYA', 28260, 26784, 1433, 43, NULL, 94.8),
('KISUMU', 34078, 32296, 1752, 30, NULL, 94.8),
('HOMABAY', 34833, 31723, 3069, 41, NULL, 91.1),
('MIGORI', 37118, 33827, 3228, 63, NULL, 91.1),
('KISII', 32057, 30419, 1609, 29, NULL, 94.9),
('NYAMIRA', 14114, 13406, 696, 12, NULL, 95.0),
('NAIROBI CITY', 135229, 131275, 3851, 103, NULL, 97.1);
```

Sorting Data

Now that we have some data in our table, let's go ahead and sort it.

The `ORDER BY` keyword is used to sort the result-set in ascending or descending order.

The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Using MySQL `ORDER BY` clause to sort the result set by one column example

Sort the counties by total number of births from county with lowest number of births

In [ ]:
```
SELECT county_name, total_births
FROM notified_births_census
ORDER BY total_births;
```

```
Displaying records 1 - 10

county_name                  total_births
LAMU                                 4235
ISIOLO                               8037
THARAKA-NITHI                        9109
TAITA/TAVETA                         9110
SAMBURU                             10665
TANA RIVER                          11683
ELGEYO/MARAKWET                     13212
KIRINYAGA                           13638
MARSABIT                            13679
NYAMIRA                             14114
```

Sort the counties by total number of births starting with county with highest number of births

In [ ]:
```
SELECT county_name, total_births
FROM notified_births_census
ORDER BY total_births DESC;
```

Displaying records 1 - 10

county_name	total_births
KENYA	1340468
RURAL	888039
URBAN	452429
NAIROBI CITY	135229
KIAMBU	69596
NAKURU	64797
KAKAMEGA	49974
BUNGOMA	47722
KILIFI	44519
NAROK	40643

We can also sort alphabetically

In [ ]:
```
SELECT county_name, total_births
FROM notified_births_census
ORDER BY county_name
```

Displaying records 1 - 10

county_name	total_births
BARINGO	19697
BOMET	24647
BUNGOMA	47722
BUSIA	25597
ELGEYO/MARAKWET	13212
EMBU	14556
GARISSA	16414
HOMABAY	34833
ISIOLO	8037
KAJIADO	36244