M.A.N.U

3/8/2023

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

`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;
```

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;
```

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;
```

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
```

The `LIMIT` clause is used in the `SELECT` statement to constrain the number of rows to return. The `LIMIT` clause accepts one or two arguments. The values of both arguments must be zero or positive integers.

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

The following code will display the top 5 counties leading with number of births

In [ ]:
```
SELECT county_name, total_births
FROM notified_births_census
ORDER BY total_births
LIMIT 5;
```

Filtering Data

The SQL `WHERE` Clause

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition

WHERE Clause Example

SHOW total_births for county_name NAIROBI

In [ ]:
```
SELECT county_name, total_births FROM notified_births_census WHERE county_name = "NAIROBI CITY";
```

In [ ]:
```
SELECT county_name, total_births FROM notified_births_census WHERE not_stated = 1;
```

Population density by subcounty in kenya Data

Load in the data provided in the `population_density_county_subcounty.sql`

https://github.com/Isaakkamau/MANU-SQL/blob/main/population_density_county_subcounty.sql

Make sure the records are inserted correctly into your table.

We will proceed to use this table to learn more about the WHERE clause

Below is the description of the table:

In [ ]:
```
DESCRIBE subcounty_population_density;
```

Operators in The WHERE Clause

The following operators can be used in the `WHERE` clause

`Equal =` Select the sub counties that have 12 square kilometers

In [ ]:
```
SELECT county_name, subcounty_name,square_kms
FROM subcounty_population_density
WHERE square_kms = 12;
```

`Greater Than >` Select all the sub counties which have more than 500,000 total population

In [ ]:
```
SELECT county_name, subcounty_name, total
FROM subcounty_population_density
WHERE total > 500000;
```

`less Than <` Select all counties which have less than 10,000 total population

In [ ]:
```
SELECT county_name, subcounty_name, total
FROM subcounty_population_density
WHERE total < 10000;
```

`<= less than or equal to`

In [ ]:
```
SELECT county_name, subcounty_name, total
FROM subcounty_population_density
WHERE square_kms <= 12;
```

`<> Not equal`

> Note: In some versions of SQL this operator may be written as `!=`

In [ ]:
```
SELECT county_name, subcounty_name
FROM subcounty_population_density
WHERE county_name <> 'Nairobi';
```

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) is `NOT TRUE`.

`AND`

Example: Select sub counties in nairobi county which have less than 200000 total population

In [ ]:
```
SELECT county_name, subcounty_name, total
FROM subcounty_population_density
WHERE county_name = 'Nairobi' AND total < 200000;
```

`OR`

Example: The following SQL statement selects all fields from "subcounty_population_density" where county_name is "Nairobi" or "Mombasa"

In [ ]:
```
SELECT county_name, subcounty_name
FROM subcounty_population_density
WHERE county_name = 'Nairobi' OR county_name = 'Mombasa';
```

`NOT`

Example: The following SQL statement selects all fields from "subcounty_population_density" where the county_name is not Nairobi

In [ ]:
```
SELECT county_name, subcounty_name, total
FROM subcounty_population_density
WHERE NOT county_name='Nairobi';
```

Combining `AND` , `OR` and `NOT`

You can also combine the `AND` , `OR` and `NOT` operators.

The following statement selects all fields from `subcounty_population_density` that are in `Nairobi` or `Mombasa` county which have a population density of more than 10000

In [ ]:
```
SELECT county_name, subcounty_name, total, pop_density
FROM subcounty_population_density
WHERE (county_name = 'Nairobi' OR county_name = 'Mombasa') AND pop_density > 10000;
```

`BETWEEN`

MySQL "BETWEEN" operator to determine whether a value is in a range of values.

In [ ]:
```
SELECT county_name, subcounty_name, square_kms
FROM subcounty_population_density
WHERE square_kms BETWEEN 1000 AND 10000;
```

`IS NULL`

Show the number rows that are missing/NULL values for square_kms column

In [ ]:
```
SELECT county_name, subcounty_name, square_kms
FROM subcounty_population_density
WHERE square_kms IS NULL;
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js