1

How to Retrieve Data From a Single Table

The Five Clauses of the SELECT statement

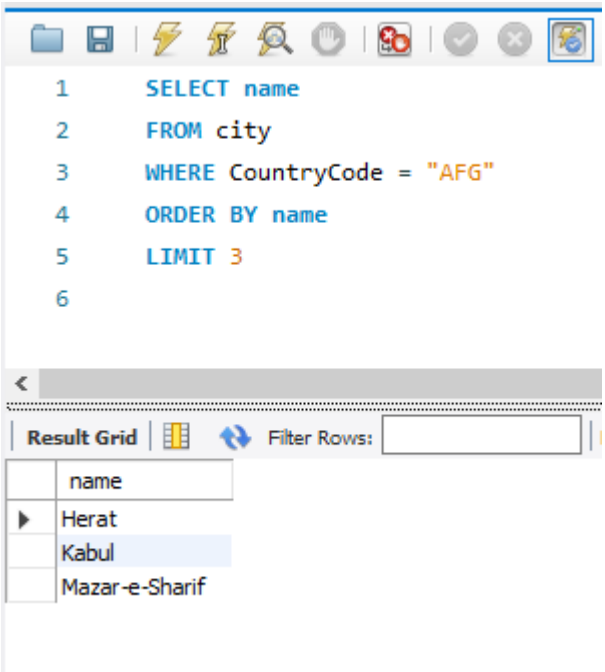
- SELECT – the columns in the result set
- FROM – names the base table(s) from which results will be retrieved
- WHERE – specifies any conditions for the results set (filter)
- ORDER BY – sets how the result set will be ordered
- LIMIT – sets the number of rows to be returned

The clauses MUST appear in the order shown above.

Code Example:

```
USE world;
SELECT name
FROM city
WHERE CountryCode = “AFG”
ORDER BY name
LIMIT 3
```

Results:



Let us break the statement line by line:

USE world;

- The **USE** clause sets the database that we will be querying. You typically have more than one database on your database server. You have to specify which database you are working in.
- The semicolon "**;**" indicates the end of a statement. You can execute multiple statements in sequence by defining each statement with a semicolon

SELECT name

- The **SELECT** clause defines the columns and column order that you want to retrieve in your results set. If you want to retrieve all of the columns from the base table you can simply use SELECT *
- You separate each column name with a comma "," ex., SELECT name, CountryCode
- There is no trailing comma at the end of a column list

FROM city

- The **FROM** clause specifies the table that the results will be coming from
- You can specify multiple tables by using a JOIN clause, but we will address that topic at a future time

ORDER BY name

- The **ORDER BY** clause is not required but when used it defines the sort order of the results
- By default, the sort order is ascending. This is *implicit* However, you can use *explicit* syntax of ASC. If you want the sort, order to be descending you can use the keyword DESC.
- You can specify more than one column in an Order By statement separated by commas. The sort order *DESC, ASC* applies to each column individually. Below IS some examples
  - **ORDER BY** population ASC, name DESC
  - **ORDER BY** population, name (ASC is always implied if not explicitly stated)

LIMIT 5;

- If you only want to return a specified number of rows from the result set, you can use the LIMIT clause. This can be helpful when you want to test a query for accuracy that could potentially bring back a very large number of rows.
- The semicolon ; defines the end of the statement

Table 1. Column Specifications

Source	Option	Syntax
Base Table Value	Show all columns	
Base Table Value	Column Name	Comma separated list of column names
Calculated Value	Calculation result	Arithmetic expression
Calculated Value	Calculation result	Functions

LIKE and REGEXP Operators

- The LIKE keyword is used with the WHERE clause.
- The LIKE keyword and can use two symbols as wildcards. The percent (%) symbol matches any number of characters and the underscore (_) matches a single character
- REGEXP keyword allows you to do more complex pattern matching than a LIKE keyword/
- Some version of REGEXP exists in many computer languages. Refer to the “LIKE and REGEXP” handout for a full list of examples.

Table 2. LIKE Keyword

LIKE Symbol	Description
%	Match any string of characters to the left of the symbol
_	Match a single character

Code Example:

```
USE world;
SELECT name
FROM country
WHERE name LIKE ‘A%’
```

Results:

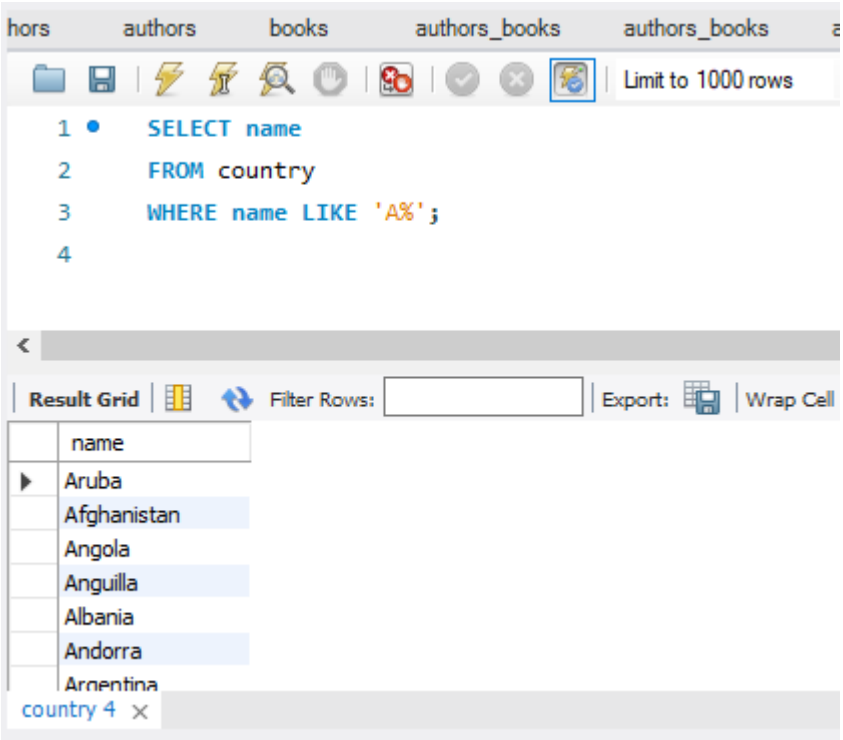


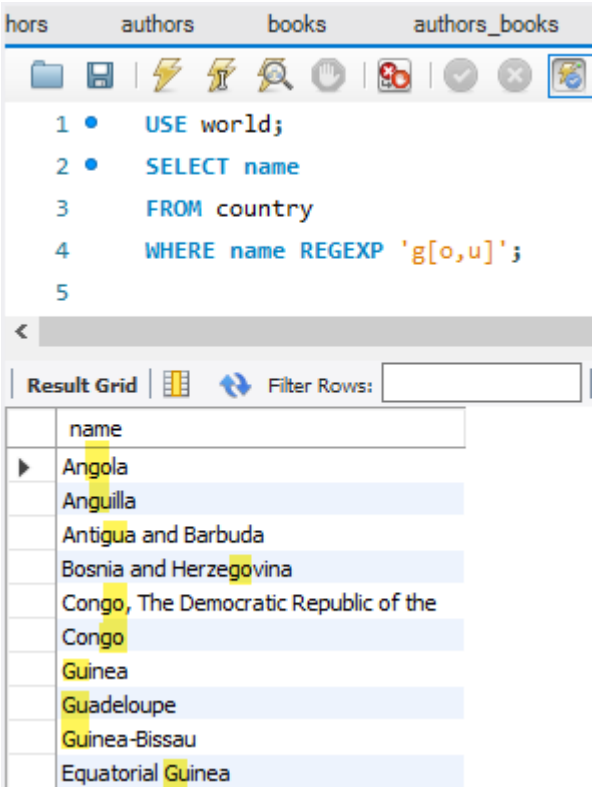
Table 3. REXEXP Keyword

REGEXP Characters	Description
^	Match the pattern to the beginning of the value being tested.
$	Match the pattern to the end of the value being tested.
.	Matches any single character.
[charlist]	Matches any single character listed within the brackets.
[char1 – char2]	Matches any single character within the given range.
|	Separates two string patterns and matches either one

Code Example:

```
USE world;
SELECT name
FROM country
WHERE name REGEXP 'g[o,u]';
```

Results:



Arithmetic Operators

- Arithmetic operators can be used in the SELECT, WHERE, and ORDER BY clauses.
- Operators are evaluated in the same way as arithmetic in other contexts.

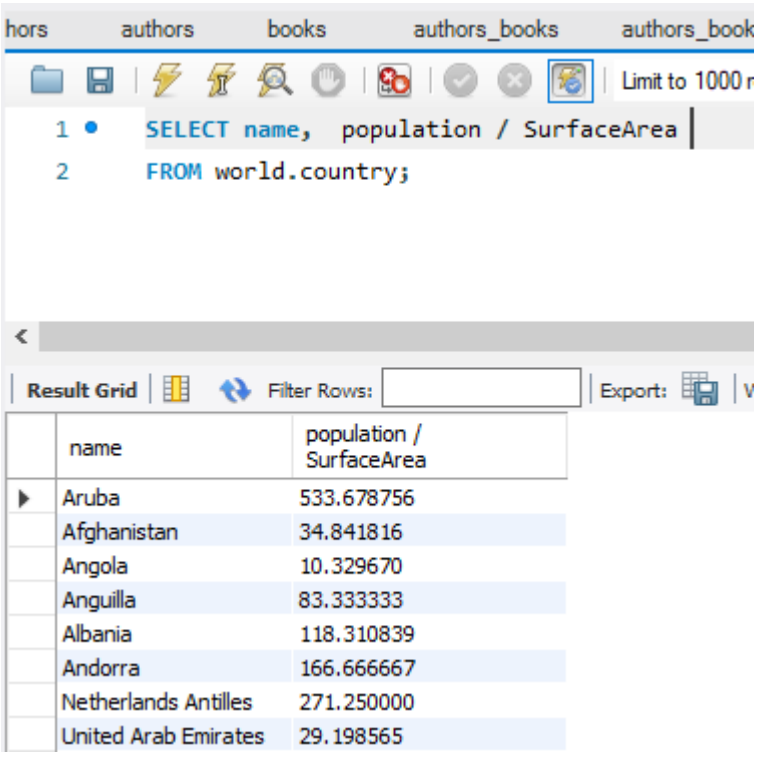
Table 4. Operators and precendence order

Operator	Name	Order of Precedence
*	Multiplication	1
/	Division	1
DIV	Integer Division	1
% (MOD)	Modulo (remainder)	1
+	Addition	2
-	Subtraction	2

Code Example:

```
USE world;
SELECT name, population / SurfaceArea
AS "People per square mile"
FROM country;
```

Results:



Column Aliases

- A column alias provides a way to create a clean or more descriptive header for a results set.
- A column alias **cannot** be used in a SELECT, WHERE, GROUP BY or HAVING clause due to the order of execution. You must refer to the original column name.

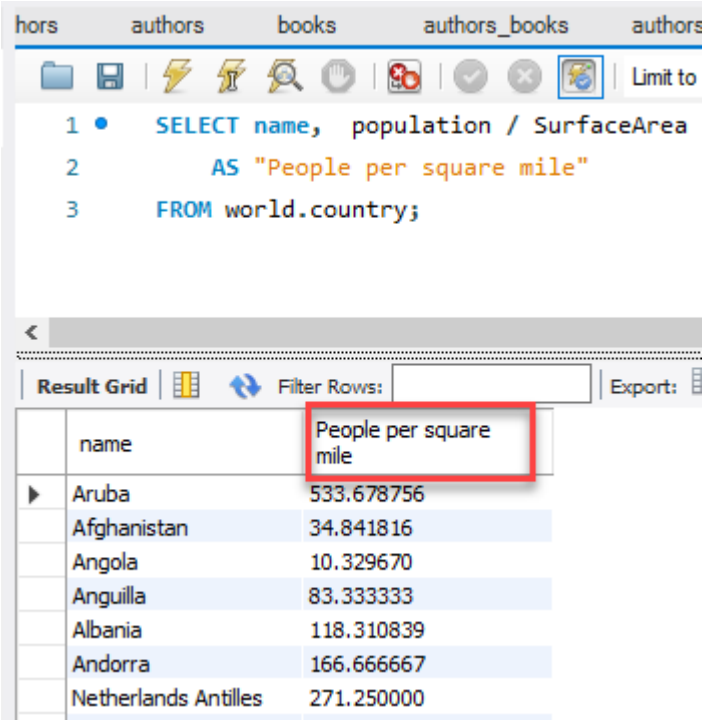
In the previous example, we created a new column that was a *calculated value*. The problem is that the column header is now population / SurfaceArea. However we can rename the column header to something cleaner be create a *column alias.* Look at the code snippet below.

Code Example:

```
SELECT name, population / SurfaceArea
     AS “People per square mile”
FROM country;
```

We used the AS keyword then in quotes we put the new column alias of “People per square mile.” Which changes the column header as seen show below.

Results:



Comparison Operators

- Comparison operators compare two expressions.
- The result of a comparison results to true or false.
- Comparison operators are not case sensitive and are used with text and dates as well as numbers.

Table 5. Comparison Operators

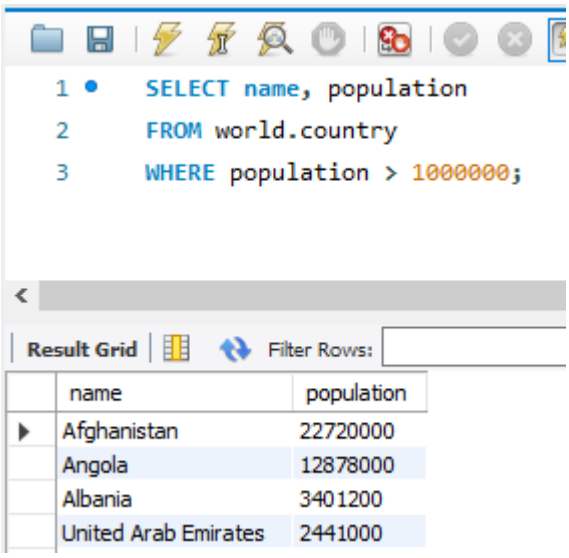
Operator	Description
=	Equal
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal

Operator	Description
!=	Not equal

Code Example:

```
USE world;
SELECT name, population
FROM country
WHERE population > 1000000;
```

Results:



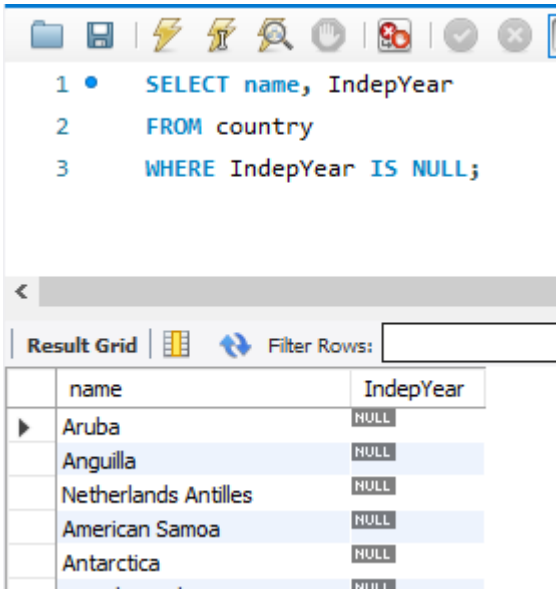
IS NULL

- *Null values* indicate an unknown or non-existent value and is different from an empty string (' ').
- To test for a *null value* you use the IS NULL clause
- The test for a value use IS NOT NULL clause

Code Example:

```
SELECT name, IndepYear
FROM country
WHERE IndepYear IS NULL;
```

Results:



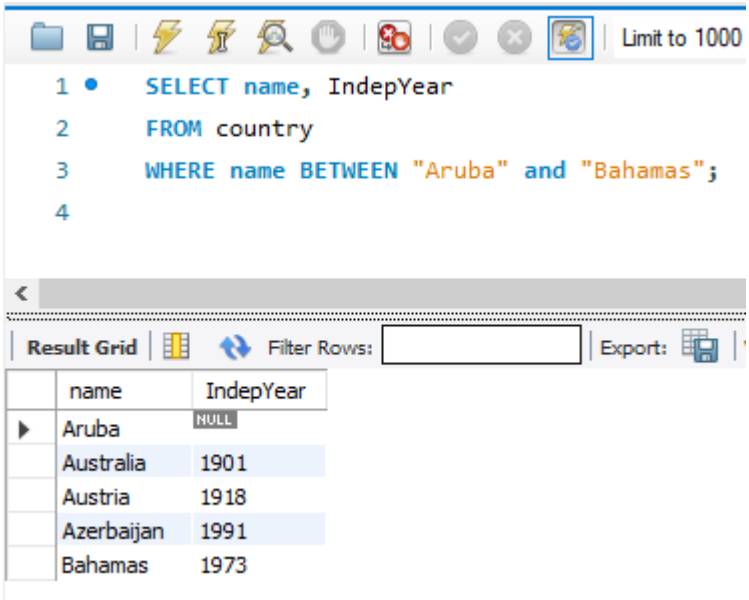
BETWEEN Operators

- The BETWEEN operator is similar to >= and <=.
- BETWEEN includes everything between the two values indicated.
- BETWEEN works with both text and number.

Code Example:

```
USE world;
SELECT name, IndepYear
FROM country
WHERE name BETWEEN "Aruba" and "Bahamas";
```

Results:



The IN Keyword

- The IN clause tests whether an expression is equal to a value or values in a list of expressions.
- The order of the items in the list does not matter.
- You can use the NOT operator to test for items not in the list.
- The IN clause may be used with a subquery.

Code Example:

```
USE world;
SELECT name
FROM country
WHERE name IN ('Aruba', 'Barbados', 'Cuba', 'Bahamas')
ORDER BY population ASC;
```

Results:

```
USE world;
SELECT name
FROM country
WHERE name IN ('Aruba', 'Barbados', 'Cuba', 'Bahamas')
ORDER BY population ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

name
Aruba
Barbados
Bahamas
Cuba

AND, OR, NOT Logical Operators

- *Logical operators* are used in the WHERE clause
- You may use multiple *logical operators* in a WHERE clause to create a *compound condition*. The order of evaluation when multiple operators are used is shown in the table above.

Table 6. Logical Operators

Operator	Description	Order of Evaluation
NOT	(a NOT b) – **a** must be present but **b** must NOT be present to be included	1
AND	(*a AND b*) –If both **a** and **b** are present, item is included	2
OR	(a OR b) – If either **a** OR **b** is present item is included	3

Example:

```
USE world;
SELECT name, population
FROM country
WHERE region = 'caribbean'
AND population > 100000
ORDER BY population ASC;
```

Results:

```
SELECT name, population
FROM world.country
WHERE region = 'caribbean' AND
    population > 100000
ORDER BY population ASC;
```

Result Grid | Filter Rows:

name	population
Aruba	103000
Saint Vincent and the Grenadines	114000
Saint Lucia	154000
Netherlands Antilles	217000
Barbados	270000
Bahamas	307000
Martinique	395000

DISTINCT Keyword

- DISTINCT appears directly after the SELECT clause.
- You can specify multiple columns, which means that the combination of columns must be unique.

Table 7. DISTINCT Keyword

Keyword	Description	Order of Evaluation
DISTINCT	Eliminates duplicate rows	1

Example:

```
SELECT DISTINCT continent, name
FROM country
ORDER BY continent;
```

Results:

```
SELECT DISTINCT continent
FROM country
ORDER BY continent;
```

Result Grid | Filter Rows:

continent
Asia
Europe
North America
Africa
Oceania
Antarctica
South America