IN605 W1L2 Introduction to MariaDB

This is taken pretty much straight from the MySQL reference document’s tutorial [http://dev.MySQL.com/doc/refman/5.1/en/tutorial.html](http://dev.mysql.com/doc/refman/5.1/en/tutorial.html) and adjusted for our environment.

Please try these examples yourself and fill in the blanks where indicated with this icon

We are using MariaDB a fork of MySQL which uses the same syntax and connection string.

# Getting Connected

1. Open Putty and log on to mariadb.ict.op.ac.nz, port 22 using ssh.
   1. Use your username (all lowercase) and your new password from the last lab
   2. Your account on Maria is not synchronised with your normal account. If you can’t log in then let someone know.
2. To connect interact with MariaDB, type the following command

mysql -u yourUserName -p

Enter password:\*\*\*\*\*\*\*\*\*\*\*

Replace “username” with your username, your password is your student ID unless you have already changed it.

Can you work out how to change your password?? Hint: the hostname is %

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The MariaDB [(none)]> prompt tells you that MariaDB is ready for you to enter commands. The (none) part indicates that you don’t currently have a database selected.

1. After you have connected successfully, you can disconnect any time by typing QUIT (**or \q**) at the prompt:

MariaDB [(none)]> QUIT

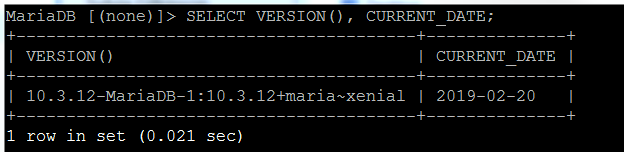
Bye

1. You can clear the Maria screen at any time using system clear; or the keystrokes Ctrl & L

# Entering Queries

Connecting to MariaDB does not in itself select any database to work with, but that is okay. At this point, it is more important to find out a little about how to issue queries than to jump right in creating tables, loading data into them, and retrieving data from them. This section describes the basic principles of entering commands, using several queries you can try out to familiarize yourself with how MariaDB works.

Here is a simple command that asks the server to tell you its version number and the current date. Type it in as shown here following the prompt and press Enter



What were your results?

Version:10.4.12-MariaDB-1:10.4.12+maria~xenial

Date:2020-02-21

This query illustrates several things about MariaDB:

* A command normally consists of an SQL statement followed by a semicolon. (There are some exceptions where a semicolon may be omitted. QUIT, mentioned earlier, is one of them. We'll get to others later.)
* When you issue a command, MariaDB sends it to the server for execution and displays the results, then prints another MariaDB> prompt to indicate that it is ready for another command.
* MariaDB displays query output in tabular form (rows and columns). The first row contains labels for the columns. The rows following are the query results. Normally, column labels are the names of the columns you fetch from database tables. If you're retrieving the value of an expression rather than a table column (as in the example just shown), MariaDB labels the column using the expression itself.
* MariaDB shows how many rows were returned and how long the query took to execute, which gives you a rough idea of server performance. These values are imprecise because they represent wall clock time (not CPU or machine time), and because they are affected by factors such as server load and network latency. (For brevity, the “rows in set” line is sometimes not shown in the remaining examples in this tutorial.)

Keywords may be entered in any lettercase. The following queries are equivalent:

mysql> SELECT VERSION(), CURRENT\_DATE;

mysql> select version(), current\_date;

mysql> SeLeCt vErSiOn(), current\_DATE;

Here is another query. It demonstrates that you can use mysql as a simple calculator. Enter this command then fill in the gaps:

mysql> **SELECT SIN(PI()/4), (4+1)\*5;**

+------------------+---------+

| | |

| | |

+------------------+---------+

| | |

| | |

+------------------+---------+

1 row in set (0.02 sec)

The queries shown thus far have been relatively short, single-line statements. You can even enter multiple statements on a single line. Just end each one with a semicolon:

MariaDB> SELECT VERSION(); SELECT NOW();

+-----------------+

| VERSION() |

+-----------------+

| |

+-----------------+

1 row in set (0.00 sec)

+---------------------+

| NOW() |

+---------------------+

| |

+---------------------+

1 row in set (0.00 sec)

A command need not be given all on a single line, so lengthy commands that require several lines are not a problem. MariaDB determines where your statement ends by looking for the terminating semicolon, not by looking for the end of the input line. (In other words, SQL accepts free-format input: it collects input lines but does not execute them until it sees the semicolon.)

Here is a simple multiple-line statement:

MariaDB> SELECT

-> USER()

-> ,

-> CURRENT\_DATE;

+---------------+--------------+

| USER() | CURRENT\_DATE |

+---------------+--------------+

| | |

+---------------+--------------+

In this example, notice how the prompt changes from **MariaDB[(none)]>** to  after you enter the first line of a multiple-line query. This is how MariaDB indicates that it has not yet seen a complete statement and is waiting for the rest. The prompt is your friend, because it provides valuable feedback. If you use that feedback, you can always be aware of what MariaDB is waiting for.

If you decide you do not want to execute a command that you are in the process of entering, cancel it by typing \c (think “c” for cancel):

MariaDB> SELECT

-> USER()

-> \c

MariaDB>

Here, too, notice the prompt. It switches back to MariaDB> after you type \c, providing feedback to indicate that MariaDB is ready for a new command.

The following table shows each of the prompts you may see and summarizes what they mean about the state that MariaDB is in.

|  |  |
| --- | --- |
| **Prompt** | **Meaning** |
| MariaDB> | Ready for new command. |
| ‑> | Waiting for next line of multiple-line command. |
| '> | Waiting for next line, waiting for completion of a string that began with a single quote (“'”). |
| "> | Waiting for next line, waiting for completion of a string that began with a double quote (“"”). |
| `> | Waiting for next line, waiting for completion of an identifier that began with a backtick (“`”). |
| /\*> | Waiting for next line, waiting for completion of a comment that began with /\*. |

Multiple-line statements commonly occur by accident when you intend to issue a command on a single line, but forget the terminating semicolon. In this case, mysql waits for more input:

MariaDB> SELECT USER()

->

If this happens to you (you think you've entered a statement but the only response is a  prompt), most likely MariaDB is waiting for the semicolon. If you don't notice what the prompt is telling you, you might sit there for a while before realizing what you need to do. Enter a semicolon to complete the statement, and MariaDB executes it:

MariaDB> SELECT USER()

-> ;

+---------------+

| USER() |

+---------------+

| jon@localhost |

+---------------+

The '> and "> prompts occur during string collection (another way of saying that MariaDB is waiting for completion of a string). In MariaDB, you can write strings surrounded by either “'” or “"” characters (for example, 'hello' or "goodbye"), and MariaDB lets you enter strings that span multiple lines. When you see a '> or "> prompt, it means that you have entered a line containing a string that begins with a “'” or “"” quote character, but have not yet entered the matching quote that terminates the string. This often indicates that you have inadvertently left out a quote character.

Try this example:

MariaDB> SELECT \* FROM my\_table WHERE name = 'Smith AND age < 30;

'>

If you enter this SELECT statement, then press Enter and wait for the result, nothing happens. Instead of wondering why this query takes so long, notice the clue provided by the **'>** prompt. It tells you that MariaDB expects to see the rest of an unterminated string. (Do you see the error in the statement? The string 'Smith is missing the second single quote mark.)

At this point, what do you do? The simplest thing is to cancel the command. However, you cannot just type \c in this case, because MariaDB interprets it as part of the string that it is collecting. Instead, enter the closing quote character (so MariaDB knows you've finished the string), then type \c

MariaDB> SELECT \* FROM my\_table WHERE name = 'Smith AND age < 30;

'> '\c

MariaDB>

The prompt changes back to MariaDB>, indicating that MariaDB is ready for a new command.

The `> prompt is similar to the '> and "> prompts, but indicates that you have begun but not completed a backtick-quoted identifier.

It is important to know what the '>, ">, and `> prompts signify, because if you mistakenly enter an unterminated string, any further lines you type appear to be ignored by MariaDB — including a line containing QUIT. This can be quite confusing, especially if you do not know that you need to supply the terminating quote before you can cancel the current command.

**3.3. Creating and Using a Database**

Once you know how to enter commands, you are ready to access a database.

Suppose that you have several pets in your home (your menagerie) and you would like to keep track of various types of information about them. You can do so by creating tables to hold your data and loading them with the desired information. Then you can answer different sorts of questions about your animals by retrieving data from the tables. This section shows you how to perform the following operations:

* Create a database
* Create a table
* Load data into the table
* Retrieve data from the table in various ways
* Use multiple tables

The menagerie database is simple (deliberately), but it is not difficult to think of real-world situations in which a similar type of database might be used. For example, a database like this could be used by a farmer to keep track of livestock, or by a veterinarian to keep track of patient records.

Use the SHOW statement to find out what databases currently exist on the server:

MariaDB> SHOW DATABASES;

+----------+

| Database |

+----------+

| mysql |

| test |

| tmp |

+----------+

**NOTE: This may be a huge list**

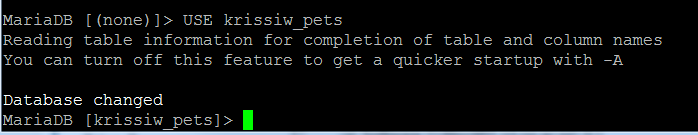
## Creating and Selecting a Database

Create a database to store your pet information. Use **your username as a prefix** and an underscore. You should have permission to create any database prefixed with **your** username.

MariaDB> CREATE DATABASE krissiw\_pets;

Under Unix, database names are case sensitive (unlike SQL keywords), so I must always refer to my database as krissiw\_pets, not as KrissiW\_Pets, KRISSIW\_PETS, or some other variant. This is also true for table names.

Creating a database does not select it for use; you must do that explicitly. To make krissiw\_pets the current database, use this command (change the name to use your own database which you just created)



Your database needs to be created only once, but you must select it for use each time you begin a MariaDB session. You can do this by issuing a USE statement as shown in the example. Alternatively, you can select the database on the command line when you invoke MariaDB. Just specify its name after any connection parameters that you might need to provide. For example:

shell> mysql -u user -p krissiw\_pets

Enter password: \*\*\*\*\*\*\*\*

**Important**

krissiw\_pets in the command just shown is not my password. If you want to supply your password on the command line after the -p option, you must do so with no intervening space (for example, as -pmypassword, not as -p mypassword). However, putting your password on the command line is ***not*** recommended, because doing so exposes it to snooping by other users logged in on your machine.

**Note**

You can see at any time which database is currently selected using

SELECT DATABASE();

## Creating a Table

Creating the database is the easy part, but at this point it is empty, as SHOW TABLES tells you:

MariaDB> SHOW TABLES;

Empty set (0.00 sec)

The harder part is deciding what the structure of your database should be: what tables you need and what columns should be in each of them.

You want a table that contains a record for each of your pets. This can be called the pet table, and it should contain, as a bare minimum, each animal's name. Because the name by itself is not very interesting, the table should contain other information. For example, if more than one person in your family keeps pets, you might want to list each animal's owner. You might also want to record some basic descriptive information such as species and sex.

How about age? That might be of interest, but it is not a good thing to store in a database. Age changes as time passes, which means you'd have to update your records often. Instead, it is better to store a fixed value such as date of birth. Then, whenever you need age, you can calculate it as the difference between the current date and the birth date. MariaDB provides functions for doing date arithmetic, so this is not difficult. Storing birth date rather than age has other advantages, too:

* You can use the database for tasks such as generating reminders for upcoming pet birthdays. (If you think this type of query is somewhat silly, note that it is the same question you might ask in the context of a business database to identify clients to whom you need to send out birthday greetings in the current week or month, for that computer-assisted personal touch.)
* You can calculate age in relation to dates other than the current date. For example, if you store death date in the database, you can easily calculate how old a pet was when it died.

Use a CREATE TABLE statement to specify the layout of your table:

MariaDB> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),

-> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);

VARCHAR is a good choice for the name, owner, and species columns because the column values vary in length. The lengths in those column definitions need not all be the same, and need not be 20. You can normally pick any length from 1 to 65535, whatever seems most reasonable to you. If you make a poor choice and it turns out later that you need a longer field, MariaDB provides an ALTER TABLE statement.

Several types of values can be chosen to represent sex in animal records, such as 'm' and 'f', or perhaps 'male' and 'female'. It is simplest to use the single characters 'm' and 'f'.

The use of the DATE data type for the birth and death columns is a fairly obvious choice.

Once you have created a table, SHOW TABLES should produce some output:

MariaDB> SHOW TABLES;

+----------------------+

| Tables in krissiw\_pets |

+----------------------+

| pet |

+--------------------- +

To verify that your table was created the way you expected, use a DESCRIBE statement:

MariaDB> DESCRIBE pet;

+---------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+-------------+------+-----+---------+-------+

| name | varchar(20) | YES | | NULL | |

| owner | varchar(20) | YES | | NULL | |

| species | varchar(20) | YES | | NULL | |

| sex | char(1) | YES | | NULL | |

| birth | date | YES | | NULL | |

| death | date | YES | | NULL | |

+---------+-------------+------+-----+---------+-------+

You can use DESCRIBE (or the shortened command DESC) any time, for example, if you forget the names of the columns in your table or what types they have.

## Loading Data into a Table

After creating your table, you need to populate it. The LOAD DATA and INSERT statements are useful for this.

Suppose that your pet records can be described as shown here. (Observe that MariaDB expects dates in 'YYYY-MM-DD' format; this may be different from what you are used to.)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **name** | **owner** | **species** | **sex** | **birth** | **death** |
| Puffball | Diane | hamster | f | 1999-03-30 |  |
| Fluffy | Harold | cat | f | 1993-02-04 |  |
| Claws | Gwen | cat | m | 1994-03-17 |  |
| Buffy | Harold | dog | f | 1989-05-13 |  |
| Fang | Benny | dog | m | 1990-08-27 |  |
| Bowser | Diane | dog | m | 1979-08-31 | 1995-07-29 |
| Chirpy | Gwen | bird | f | 1998-09-11 |  |
| Whistler | Gwen | bird |  | 1997-12-09 |  |
| Slim | Benny | snake | m | 1996-04-29 |  |

When you want to add new records one at a time, the INSERT statement is useful. In its simplest form, you supply values for each column, in the order in which the columns were listed in the CREATE TABLE statement. Suppose that Diane gets a new hamster named “Puffball.” You could add a new record using an INSERT statement like this:

MariaDB> INSERT INTO pet

-> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);

String and date values are specified as quoted strings here. Also, with INSERT, you can insert NULL directly to represent a missing value.

But that’s a lot of work. Surely there must be a better way, and there is.

An easy way to populate it is to create a text file containing a row for each of your animals, then load the contents of the file into the table with a single statement.

You could create a text file *pets.txt* containing one record per line, with values separated by tabs, and given in the order in which the columns were listed in the CREATE TABLE statement. For missing values (such as unknown sexes or death dates for animals that are still living), you can use NULL values. To represent these in your text file, use \N (backslash, capital-N). For example, the record for Whistler the bird would look like this (where a comma is used as a delimiter between values):

Whistler,Gwen,bird,\N,1997-12-09,\N

Copy the file pets.txt using the following command on Maria. Note that you must execute this from the normal shell prompt, not the MariaDB command interface, and **yes** the trailing dot is needed, it signifies your home directory. This is a standard *command – source – destination* approach. The source is my home directory so you will need to specify krissiw.

username@fthictmaria01:~$ cp /home/krissiw/pets.txt .

To load the text file pets.txt into the pet table, log back into MariaDB server and use this statement:

MariaDB> LOAD DATA LOCAL INFILE 'pets.txt' INTO TABLE pet

-> FIELDS TERMINATED BY ',';

NB: On a linux system the tilde character ‘~’ is used as an alias for your home directory. In my case this expands to ‘/home/krissiw’. If you created the file on Windows with an editor that uses \r\n as a line terminator, you should use this statement instead:

MariaDB> LOAD DATA LOCAL INFILE 'pets.txt' INTO TABLE pet

-> FIELDS TERMINATED BY ','

-> LINES TERMINATED BY '\r\n';

## Retrieving Information from a Table

The SELECT statement is used to pull information from a table. The general form of the statement is:

SELECT what\_to\_select

FROM which\_table

WHERE conditions\_to\_satisfy;

* what\_to\_select indicates what you want to see. This can be a list of columns, or \* to indicate “all columns.”
* which\_table indicates the table from which you want to retrieve data.
* The WHERE clause is optional. If it is present, conditions\_to\_satisfy specifies one or more conditions that rows must satisfy to qualify for retrieval.

The simplest form of SELECT retrieves everything from a table:

MariaDB> SELECT \* FROM pet;

+----------+--------+---------+------+------------+------------+

| name | owner | species | sex | birth | death |

+----------+--------+---------+------+------------+------------+

| Fluffy | Harold | cat | f | 1993-02-04 | NULL |

| Claws | Gwen | cat | m | 1994-03-17 | NULL |

| Buffy | Harold | dog | f | 1989-05-13 | NULL |

| Fang | Benny | dog | m | 1990-08-27 | NULL |

| Bowser | Diane | dog | m | 1979-08-31 | 1995-07-29 |

| Chirpy | Gwen | bird | f | 1998-09-11 | NULL |

| Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |

| Slim | Benny | snake | m | 1996-04-29 | NULL |

| Puffball | Diane | hamster | f | 1999-03-30 | NULL |

+----------+--------+---------+------+------------+------------+

This form of SELECT is useful if you want to review your entire table, for example, after you've just loaded it with your initial data set. For example, you may happen to think that the birth date for Bowser doesn't seem quite right. Consulting your original pedigree papers, you find that the correct birth year should be 1989, not 1979.

There are at least two ways to fix this:

1. Edit the file pet.txt to correct the error, then empty the table and reload it using DELETE and LOAD DATA:

MariaDB> DELETE FROM pet;

MariaDB> LOAD DATA LOCAL INFILE 'pets.txt' INTO TABLE pet

-> FIELDS TERMINATED BY ',';

However, if you do this, you must also re-enter the record for Puffball.

1. Fix only the erroneous record with an UPDATE statement:

MariaDB> UPDATE pet SET birth = '1989-08-31' WHERE name = 'Bowser';

The UPDATE changes only the record in question and does not require you to reload the table.

Perform the UPDATE to change the record so that the DoB is 31/8/1989.

## Selecting Particular Rows

As shown in the preceding section, it is easy to retrieve an entire table. Just omit the WHERE clause from the SELECT statement. But typically you don't want to see the entire table, particularly when it becomes large. Instead, you're usually more interested in answering a particular question, in which case you specify some constraints on the information you want. Let's look at some selection queries in terms of questions about your pets that they answer.

You can select only particular rows from your table. For example, if you want to verify the change that you made to Bowser's birth date, select Bowser's record like this:

MariaDB> **SELECT \* FROM pet WHERE name = 'Bowser';**

+--------+-------+---------+------+------------+------------+

| name | owner | species | sex | birth | death |

+--------+-------+---------+------+------------+------------+

| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |

+--------+-------+---------+------+------------+------------+

The output confirms that the year is correctly recorded as 1989, not 1979.

String comparisons normally are case-insensitive, so you can specify the name as 'bowser', 'BOWSER', and so forth. The query result is the same.

You can specify conditions on any column, not just name. For example, if you want to know which animals were born during or after 1998, test the birth column:

Fill in the names of the animals to show you have identified them

MariaDB> SELECT \* FROM pet WHERE birth >= '1998-1-1';

+----------+-------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+----------+-------+---------+------+------------+-------+

|Chirpy | Gwen | bird | f | 1998-09-11 | NULL |

|Puffball | Diane | hamster | f | 1999-03-30 | NULL |

+----------+-------+---------+------+------------+-------+

You can combine conditions, for example, to locate female dogs:

MariaDB> SELECT \* FROM pet WHERE species = 'dog' AND sex = 'f';

+-------+--------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+-------+--------+---------+------+------------+-------+

|Buffy | Harold | dog | f | 1989-05-13 | NULL |

+-------+--------+---------+------+------------+-------+

The preceding query uses the AND logical operator. There is also an OR operator:

MariaDB> SELECT \* FROM pet WHERE species = 'snake' OR species = 'bird';

+----------+-------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+----------+-------+---------+------+------------+-------+

|Chirpy | Gwen | bird | f | 1998-09-11 | NULL |

|Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |

|Slim | Benny | snake | m | 1996-04-29 | NULL |

+----------+-------+---------+------+------------+-------+

AND and OR may be intermixed, although AND has higher precedence than OR. If you use both operators, it is a good idea to use parentheses to indicate explicitly how conditions should be grouped:

MariaDB> SELECT \* FROM pet WHERE (species = 'cat' AND sex = 'm')

-> OR (species = 'dog' AND sex = 'f');

+-------+--------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+-------+--------+---------+------+------------+-------+

|Claws | Gwen | cat | m | 1994-03-17 | NULL |

|Buffy | Harold | dog | f | 1989-05-13 | NULL |

+-------+--------+---------+------+------------+-------+

## Selecting Particular Columns

If you do not want to see entire rows from your table, just name the columns in which you are interested, separated by commas. For example, if you want to know when your animals were born, select the name and birth columns:

MariaDB> SELECT name, birth FROM pet;

+----------+------------+

| name | birth |

+----------+------------+

| Fluffy | 1993-02-04 |

| Claws | 1994-03-17 |

| Buffy | 1989-05-13 |

| Fang | 1990-08-27 |

| Bowser | 1989-08-31 |

| Chirpy | 1998-09-11 |

| Whistler | 1997-12-09 |

| Slim | 1996-04-29 |

| Puffball | 1999-03-30 |

+----------+------------+

To find out who owns pets, use this query:

MariaDB> SELECT owner FROM pet;

+--------+

| owner |

+--------+

| Harold |

| Gwen |

| Harold |

| Benny |

| Diane |

| Gwen |

| Gwen |

| Benny |

| Diane |

+--------+

Notice that the query simply retrieves the owner column from each record, and some of them appear more than once. To minimize the output, retrieve each unique output record just once by adding the keyword DISTINCT:

MariaDB> SELECT DISTINCT owner FROM pet;

+--------+

| owner |

+--------+

| |

| |

| |

| |

+--------+

You can use a WHERE clause to combine row selection with column selection. For example, to get birth dates for dogs and cats only, use this query:

MariaDB> SELECT name, species, birth FROM pet

-> WHERE species = 'dog' OR species = 'cat';

+--------+---------+------------+

| name | species | birth |

+--------+---------+------------+

| Fluffy | cat | 1993-02-04 |

| Claws | cat | 1994-03-17 |

| Buffy | dog | 1989-05-13 |

| Fang | dog | 1990-08-27 |

| Bowser | dog | 1989-08-31 |

+--------+---------+------------+

## Sorting Rows

You may have noticed in the preceding examples that the result rows are displayed in no particular order. It is often easier to examine query output when the rows are sorted in some meaningful way. To sort a result, use an ORDER BY clause.

Here are animal birthdays, sorted by date:

MariaDB> SELECT name, birth FROM pet ORDER BY birth;

+----------+------------+

| name | birth |

+----------+------------+

| Buffy | 1989-05-13 |

| Bowser | 1989-08-31 |

| Fang | 1990-08-27 |

| Fluffy | 1993-02-04 |

| Claws | 1994-03-17 |

| Slim | 1996-04-29 |

| Whistler | 1997-12-09 |

| Chirpy | 1998-09-11 |

| Puffball | 1999-03-30 |

+----------+------------+

On character type columns, sorting — like all other comparison operations — is normally performed in a case-insensitive fashion. This means that the order is undefined for columns that are identical except for their case. You can force a case-sensitive sort for a column by using BINARY like so: ORDER BY BINARY col\_name.

The default sort order is ascending, with smallest values first. To sort in reverse (descending) order, add the DESC keyword to the name of the column you are sorting by:

MariaDB> SELECT name, birth FROM pet ORDER BY birth DESC;

+----------+------------+

| name | birth |

+----------+------------+

| Puffball | 1999-03-30 |

| Chirpy | 1998-09-11 |

| Whistler | 1997-12-09 |

| Slim | 1996-04-29 |

| Claws | 1994-03-17 |

| Fluffy | 1993-02-04 |

| Fang | 1990-08-27 |

| Bowser | 1989-08-31 |

| Buffy | 1989-05-13 |

+----------+------------+

You can sort on multiple columns, and you can sort different columns in different directions. For example, to sort by type of animal in ascending order, then by birth date within animal type in descending order (youngest animals first), use the following query:

MariaDB> SELECT name, species, birth FROM pet

-> ORDER BY species, birth DESC;

+----------+---------+------------+

| name | species | birth |

+----------+---------+------------+

| Chirpy | bird | 1998-09-11 |

| Whistler | bird | 1997-12-09 |

| Claws | cat | 1994-03-17 |

| Fluffy | cat | 1993-02-04 |

| Fang | dog | 1990-08-27 |

| Bowser | dog | 1989-08-31 |

| Buffy | dog | 1989-05-13 |

| Puffball | hamster | 1999-03-30 |

| Slim | snake | 1996-04-29 |

+----------+---------+------------+

The DESC keyword applies only to the column name immediately preceding it (birth); it does not affect the species column sort order.

## Pattern Matching

MariaDB provides standard SQL pattern matching as well as a form of pattern matching based on extended regular expressions similar to those used by Unix utilities such as vi, grep, and sed.

SQL pattern matching allows you to use “\_” to match any single character and “%” to match an arbitrary number of characters (including zero characters). In MariaDB, SQL patterns are case-insensitive by default. Some examples are shown here. You do not use = or <> when you use SQL patterns; use the LIKE or NOT LIKE comparison operators instead.

To find names beginning with “b”:

MariaDB> SELECT \* FROM pet WHERE name LIKE 'b%';

+--------+--------+---------+------+------------+------------+

| name | owner | species | sex | birth | death |

+--------+--------+---------+------+------------+------------+

|Buffy | Harold | dog | f | 1989-05-13 | NULL |

|Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |

+--------+--------+---------+------+------------+------------+

To find names ending with “fy”:

MariaDB> SELECT \* FROM pet WHERE name LIKE '%fy';

+--------+--------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+--------+--------+---------+------+------------+-------+

| Fluffy | Harold | cat | f | 1993-02-04 | NULL |

| Buffy | Harold | dog | f | 1989-05-13 | NULL |

+--------+--------+---------+------+------------+-------+

To find names containing a “w”:

MariaDB> SELECT \* FROM pet WHERE name LIKE '%w%';

+----------+-------+---------+------+------------+------------+

| name | owner | species | sex | birth | death |

+----------+-------+---------+------+------------+------------+

| Claws | Gwen | cat | m | 1994-03-17 | NULL |

| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |

| Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |

+----------+-------+---------+------+------------+------------+

To find names containing exactly five characters, use five instances of the “\_” pattern character:

MariaDB> SELECT \* FROM pet WHERE name LIKE '\_\_\_\_\_';

+-------+--------+---------+------+------------+-------+

| name | owner | species | sex | birth | death |

+-------+--------+---------+------+------------+-------+

|Claws | Gwen | cat | m | 1994-03-17 | NULL |

|buffy | Harold | dog | f | 1989-05-13 | NULL |

+-------+--------+---------+------+------------+-------+