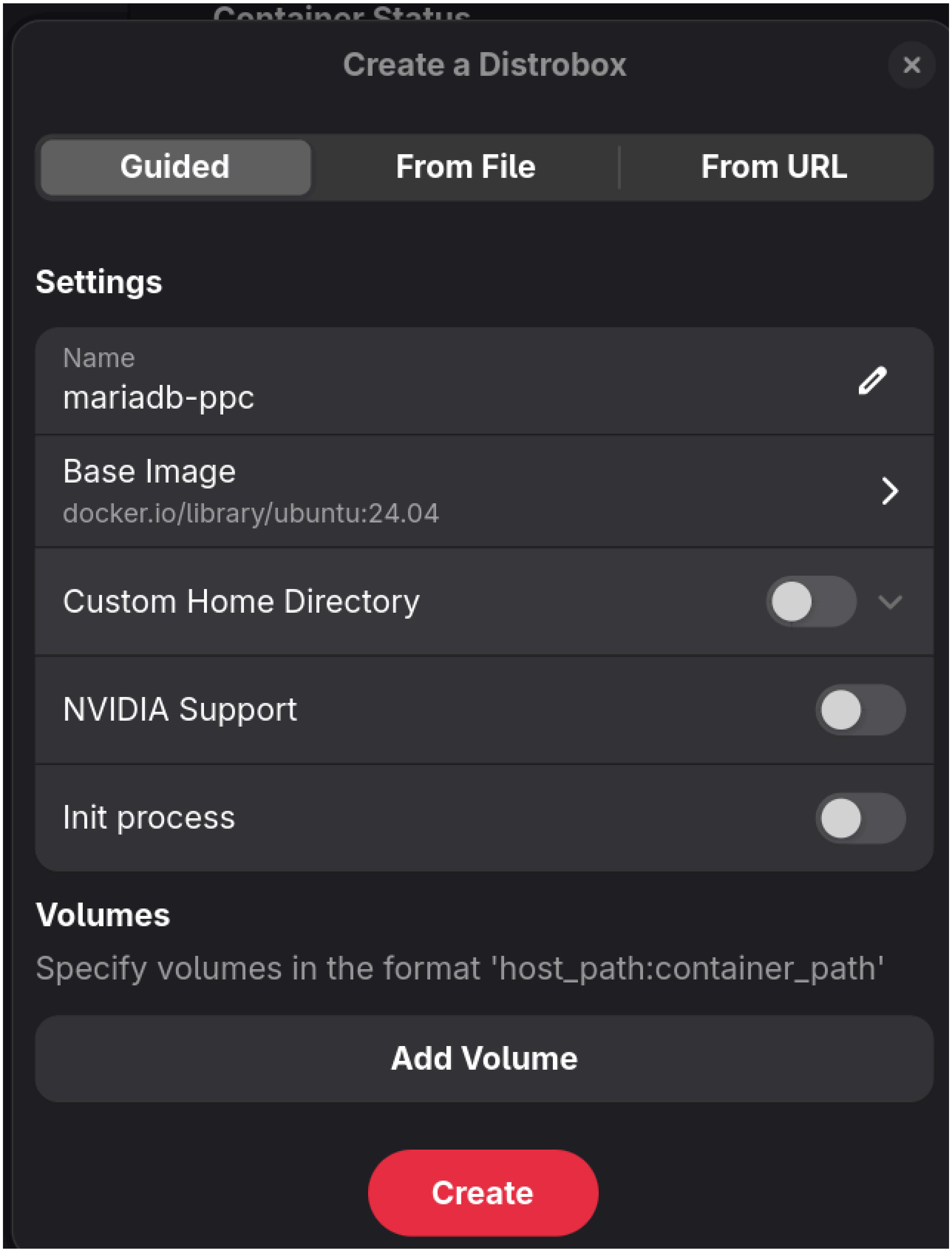
Introductory exercises with interacting with MariaDB.

Setup

Again, I'm completing this assignment on the Project Bluefin O.S. I'm going to create a new Ubuntu container for this assignment.



Settings

Name
mariadb-ppc

Base Image
docker.io/library/ubuntu:24.04

Custom Home Directory

NVIDIA Support

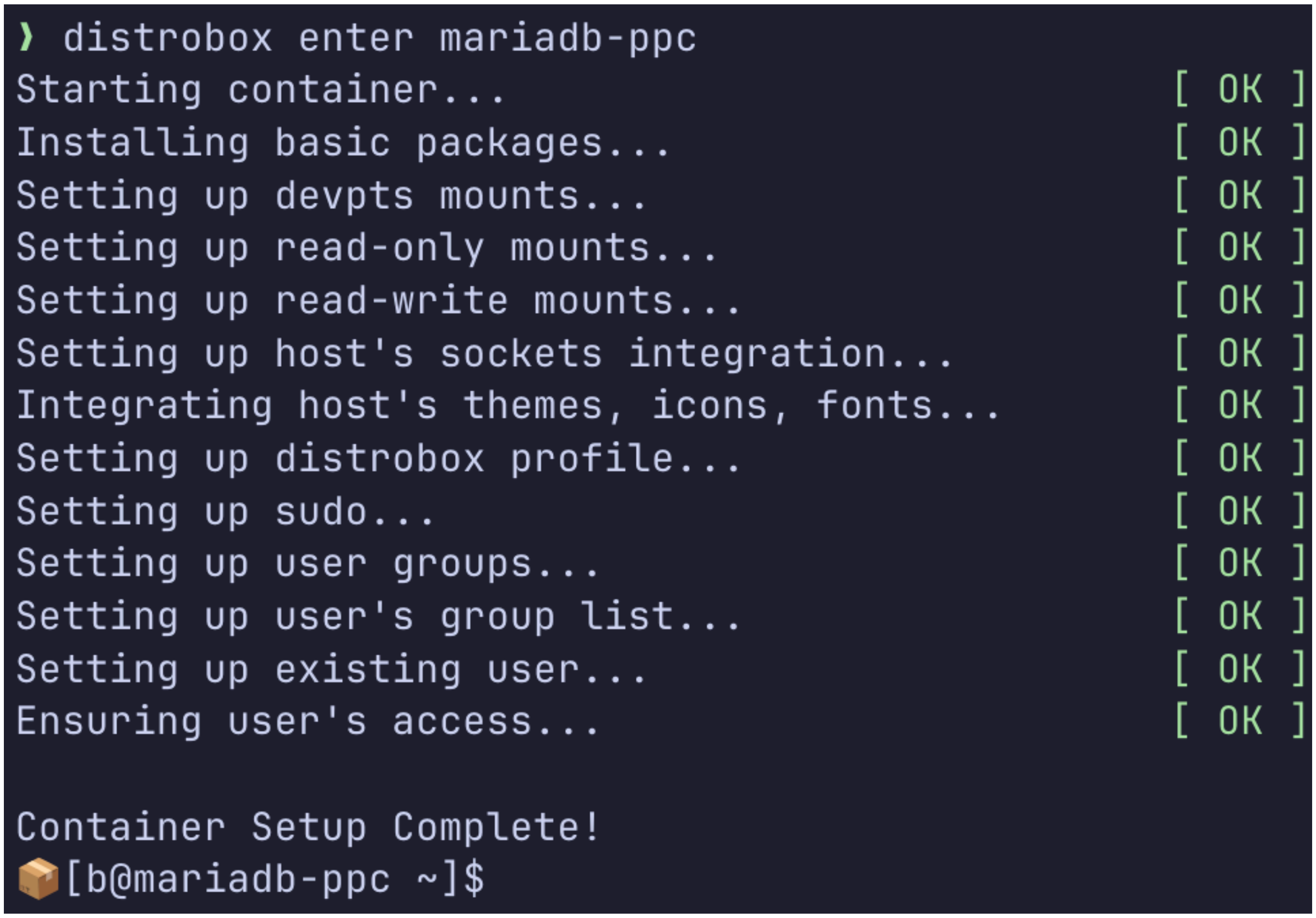
Init process

Volumes

Specify volumes in the format 'host_path:container_path'

Add Volume

I've selected Ubuntu 24.04, because I already have the image downloaded, and it is stable.

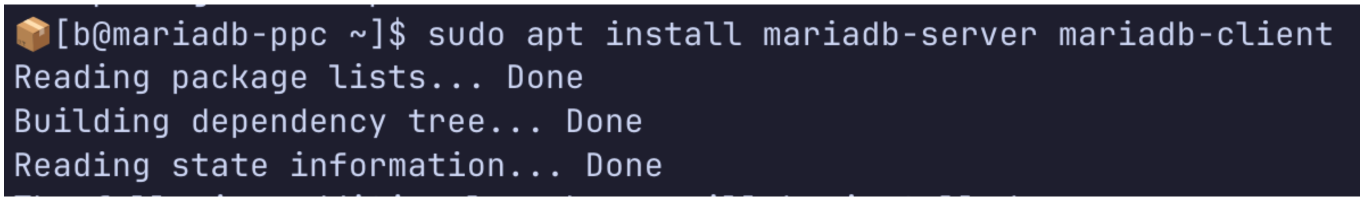


```
❯ distrobox enter mariadb-ppc
Starting container...                                   [ OK ]
Installing basic packages...                            [ OK ]
Setting up devpts mounts...                             [ OK ]
Setting up read-only mounts...                          [ OK ]
Setting up read-write mounts...                         [ OK ]
Setting up host's sockets integration...                [ OK ]
Integrating host's themes, icons, fonts...              [ OK ]
Setting up distrobox profile...                         [ OK ]
Setting up sudo...                                      [ OK ]
Setting up user groups...                               [ OK ]
Setting up user's group list...                         [ OK ]
Setting up existing user...                             [ OK ]
Ensuring user's access...                               [ OK ]

Container Setup Complete!
📦[b@mariadb-ppc ~]$
```

Here I've entered my container, and we can see that setup is successful. Let's enter it and get MariaDB installed.

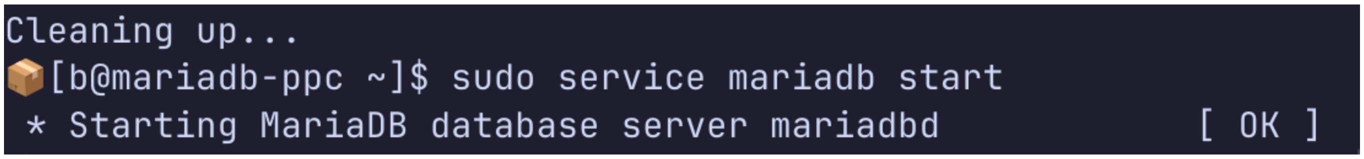
We'll install both the server and the client for MariaDB. The MariaDB installation guide recommends installing with `galera-4`, a cluster library that allows MariaDB to run consistently across multiple nodes. Because I'm just setting up a single instance for this one assignment, I'm going to skip installing galera.



```
📦[b@mariadb-ppc ~]$ sudo apt install mariadb-server mariadb-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

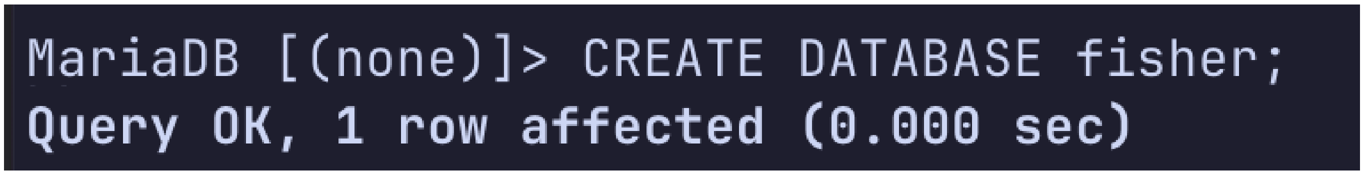
I've trimmed the screenshot because the output for this installation was far too long to include, but we successfully installed MariaDB client and server.

Now, let's start the server.



```
Cleaning up...
📦[b@mariadb-ppc ~]$ sudo service mariadb start
 * Starting MariaDB database server mariadbd                    [ OK ]
```

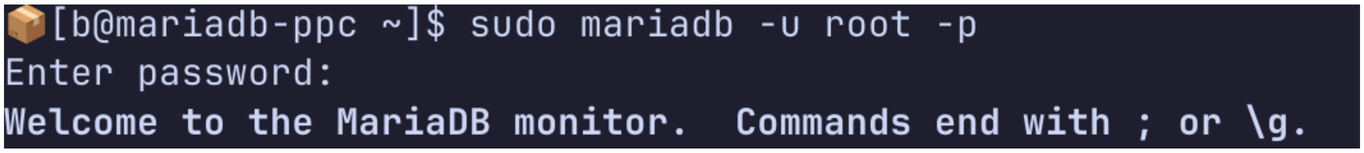
Next, we run the security script. I add a password to the root account,and disable anonymous users. After that, I create a database:



```
MariaDB [(none)]> CREATE DATABASE fisher;
Query OK, 1 row affected (0.000 sec)
```

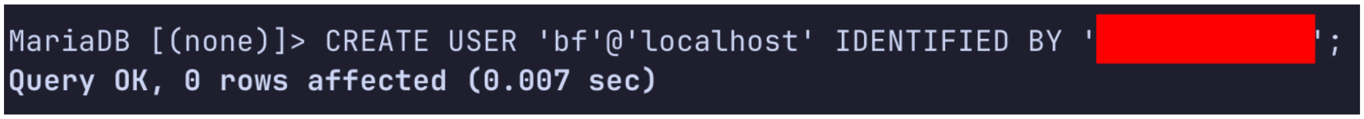
Next, I need to create an ordinary user account.

First, I'll access the root account.



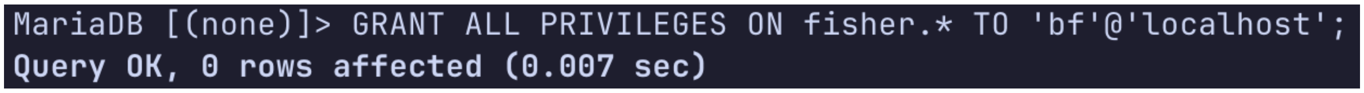
```
📦[b@mariadb-ppc ~]$ sudo mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
```

Now, I'll make a new user and assign a password.



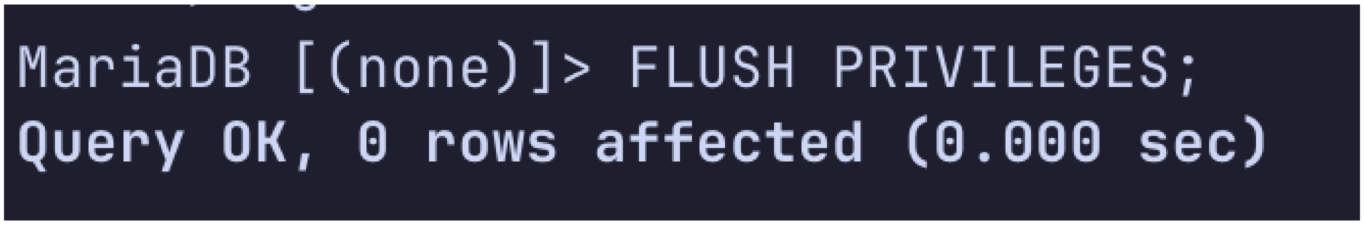
```
MariaDB [(none)]> CREATE USER 'bf'@'localhost' IDENTIFIED BY '[redacted]';
Query OK, 0 rows affected (0.007 sec)
```

The next step is to grant my new user permissions.



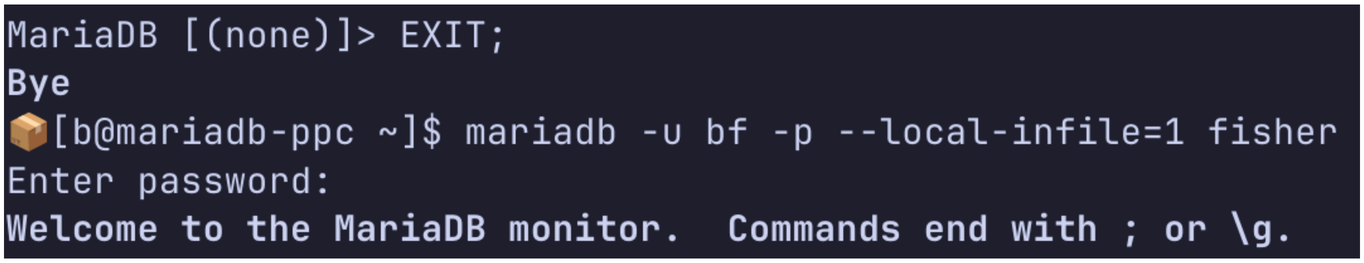
```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON fisher.* TO 'bf'@'localhost';
Query OK, 0 rows affected (0.007 sec)
```

Okay, my user has privileges. Let's `FLUSH` to be sure our privileges are updated.



```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)
```

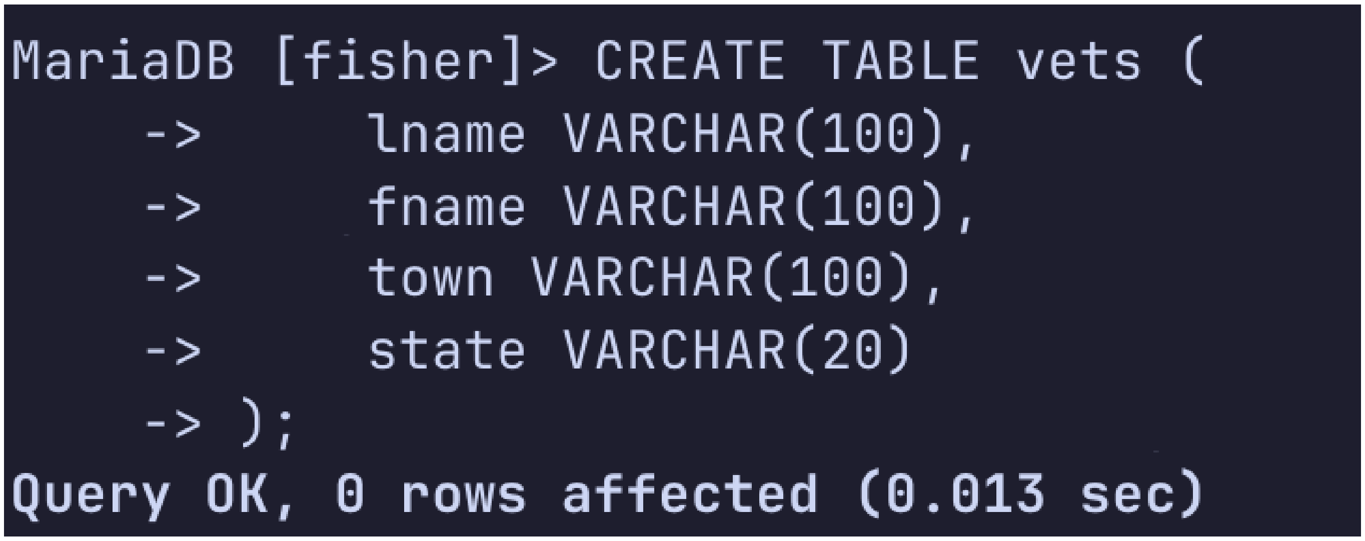
Now, let's exit the root account and sign into the `bf` account. I'm using the `--local-infile` flag to allow loading files from my computer to the database.



```
MariaDB [(none)]> EXIT;
Bye
📦[b@mariadb-ppc ~]$ mariadb -u bf -p --local-infile=1 fisher
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
```

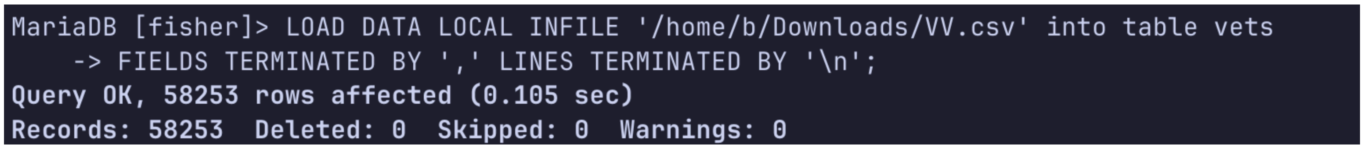
Continuing on, I'm tasked to create a table called `vets`, with the following fields:

```
lname varchar(100), fname varchar(100), town varchar(100), state varchar(20)
```



```
MariaDB [fisher]> CREATE TABLE vets (
    ->      lname VARCHAR(100),
    ->      fname VARCHAR(100),
    ->      town VARCHAR(100),
    ->      state VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.013 sec)
```

This container shares the home directory of my host machine, so I don't have to use podman to copy the CSV in. Now, we're tasked to read in the file, which contains the last name, first name, town, and state of Vietnam Veterans who were KIA.

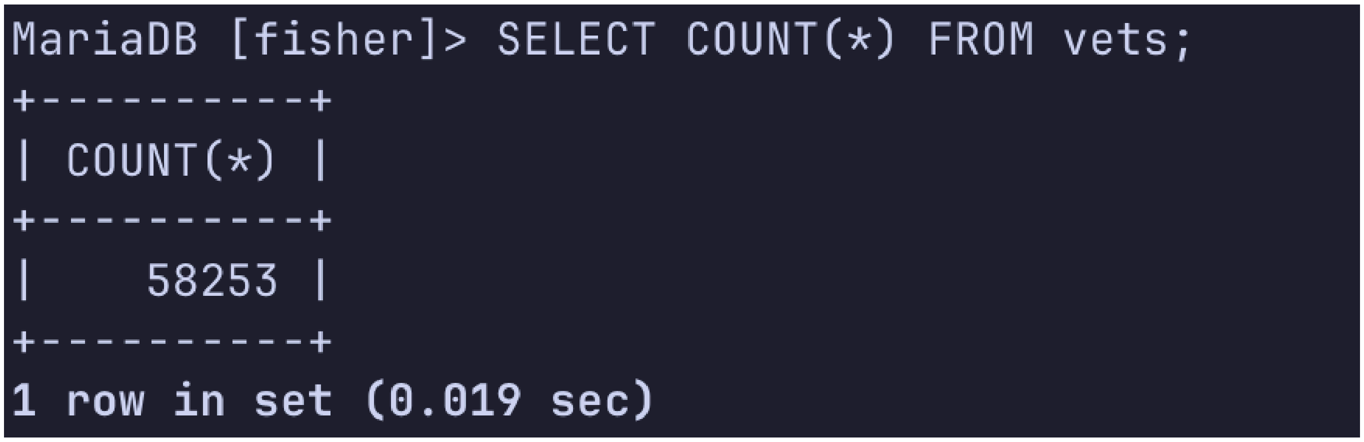


```
MariaDB [fisher]> LOAD DATA LOCAL INFILE '/home/b/Downloads/VV.csv' into table vets
    -> FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';
Query OK, 58253 rows affected (0.105 sec)
Records: 58253  Deleted: 0  Skipped: 0  Warnings: 0
```

Great! Looks like our CSV file properly read into our `vets` table.

Now, we have a few SQL queries to run:

```
(a) How many veterans are listed in the file?
```

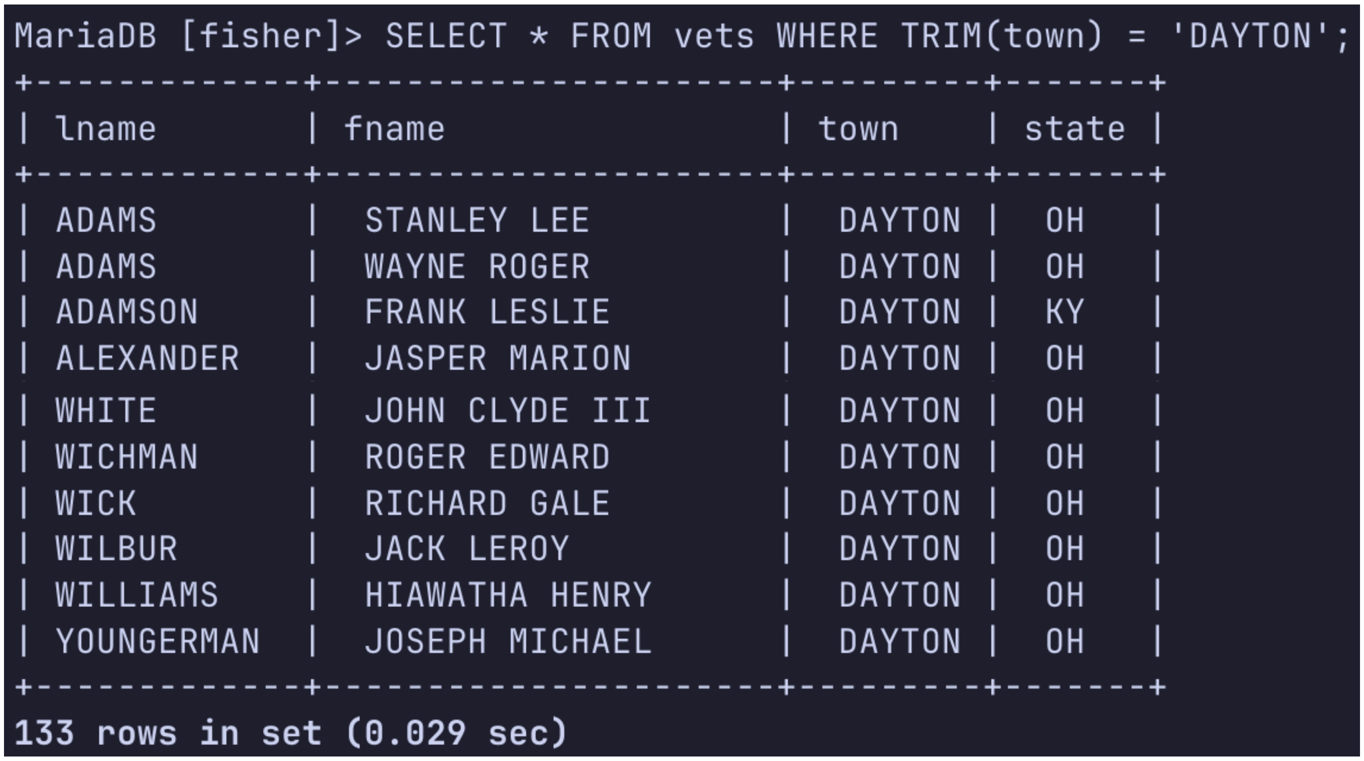


```
MariaDB [fisher]> SELECT COUNT(*) FROM vets;
+----------+
| COUNT(*) |
+----------+
|    58253 |
+----------+
1 row in set (0.019 sec)
```

By selecting all rows, we can see there are 58,253 rows, and since each row is a veteran, there are 58,253 veterans.

```
(b) How many of them are from DAYTON?
```

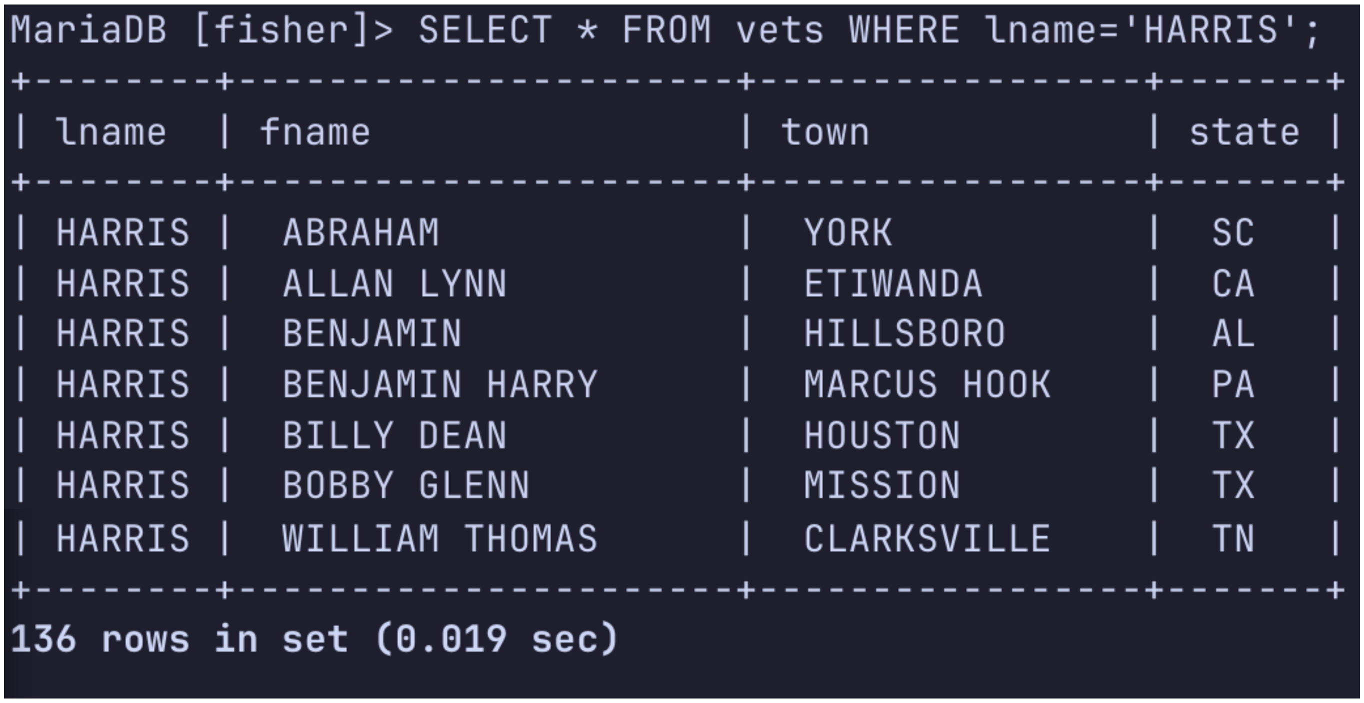
Originally, my query was: `SELECT * FROM vets WHERE town = 'DAYTON';`. This returned 0, which is incorrect. I believe this error occured because the data in the CSV parsed whitespace along with some of the data. To run the query as intended, I instead run: `SELECT * FROM vets WHERE TRIM(town) = 'DAYTON';`, which matches the trimmed entries of the town column with `'DAYTON'`. This outputs correctly:



```
MariaDB [fisher]> SELECT * FROM vets WHERE TRIM(town) = 'DAYTON';
+------------+----------------------+---------+-------+
| lname      | fname                | town    | state |
+------------+----------------------+---------+-------+
| ADAMS      |  STANLEY LEE         |  DAYTON |  OH   |
| ADAMS      |  WAYNE ROGER         |  DAYTON |  OH   |
| ADAMSON    |  FRANK LESLIE        |  DAYTON |  KY   |
| ALEXANDER  |  JASPER MARION       |  DAYTON |  OH   |
| WHITE      |  JOHN CLYDE III      |  DAYTON |  OH   |
| WICHMAN    |  ROGER EDWARD        |  DAYTON |  OH   |
| WICK       |  RICHARD GALE        |  DAYTON |  OH   |
| WILBUR     |  JACK LEROY          |  DAYTON |  OH   |
| WILLIAMS   |  HIAWATHA HENRY      |  DAYTON |  OH   |
| YOUNGERMAN |  JOSEPH MICHAEL      |  DAYTON |  OH   |
+------------+----------------------+---------+-------+
133 rows in set (0.029 sec)
```

We can see 133 veterans are from a town called Dayton, but it is worth noting some are from Dayton, OH, while others are in KY, VA, or others.

```
(c) List all veterans with last name 'HARRIS'
```



```
MariaDB [fisher]> SELECT * FROM vets WHERE lname='HARRIS';
+--------+----------------------+------------------+-------+
| lname  | fname                | town             | state |
+--------+----------------------+------------------+-------+
| HARRIS |  ABRAHAM             |  YORK            |  SC   |
| HARRIS |  ALLAN LYNN          |  ETIWANDA        |  CA   |
| HARRIS |  BENJAMIN            |  HILLSBORO       |  AL   |
| HARRIS |  BENJAMIN HARRY      |  MARCUS HOOK     |  PA   |
| HARRIS |  BILLY DEAN          |  HOUSTON         |  TX   |
| HARRIS |  BOBBY GLENN         |  MISSION         |  TX   |
| HARRIS |  WILLIAM THOMAS      |  CLARKSVILLE     |  TN   |
+--------+----------------------+------------------+-------+
136 rows in set (0.019 sec)
```

I've trimmed out the middle of the output, but we can see that there are 136 veterans with the last name HARRIS in our table.

Conclusion

This assignment involved installing MariaDB, creating a database and table to store Vietnam War veteran data, and importing records from a CSV file. After successfully loading the data, SQL queries were used to count total veterans, filter by location (DAYTON), and retrieve records by last name (HARRIS), demonstrating fundamental database management and query skills.