This assignment is a bit shorter than the past few. Let's get started:

1. Read in CSV:

If you haven't already done so, make sure you have both the vets table in your database.
Replace your States table, if you already have it from class, with the data in the States_Territories.csv
file (given). This file includes Territories like GUAM, Northern Marinara Islands etc.

I do have an existing States table. First things first, we need to replace the data in this table with the data in States_Territories.csv.

```
MariaDB [fisher]> TRUNCATE TABLE states;
Query OK, 0 rows affected (0.014 sec)
```

I truncate the table to wipe the existing data in the table.

Next, let's read our new CSV into the table.

```
MariaDB [fisher]> LOAD DATA LOCAL INFILE '/home/b/Downloads/States_Territories.csv' INTO TABLE states FIELDS TERMINATED BY ','
 ENCLOSED BY '"' LINES TERMINATED BY '\n';
Query OK, 58 rows affected (0.010 sec)
Records: 58  Deleted: 0  Skipped: 0  Warnings: 0
```

Now double checking the CSV read in properly.

```
MariaDB [fisher]> SELECT * FROM states LIMIT 5;
+-------------------+--------+
| StateName         | Abbrev |
+-------------------+--------+
| STATE(TERRITORY)  |        |
| Alabama           | AL     |
| Alaska            | AK     |
| Arizona           | AZ     |
| Arkansas          | AR     |
+-------------------+--------+
5 rows in set (0.001 sec)
```

Annoyingly, the header read in. Let's clean that up:

```
MariaDB [fisher]> DELETE FROM states WHERE StateName='STATE(TERRITORY)';
Query OK, 1 row affected (0.008 sec)

MariaDB [fisher]> SELECT * FROM states LIMIT 5;
+----------------+--------+
| StateName      | Abbrev |
+----------------+--------+
| Alabama        | AL     |
| Alaska         | AK     |
| Arizona        | AZ     |
| Arkansas       | AR     |
| American Samoa | AS     |
+----------------+--------+
5 rows in set (0.001 sec)
```

Great! Our CSV is properly read in, and we can continue.

2. Obtain Count of Vets From Each State, Descending:

For this task, we need to combine a few of the things I've used individually into one query: `GROUP BY, ORDER BY` and `COUNT(*)`, as well as using `IN` for the first time.

```
MariaDB [fisher]> SELECT state,
    -> COUNT(*) AS count
    -> FROM vets
    -> WHERE state IN (
    -> SELECT Abbrev FROM states
    -> )
    -> GROUP BY state
    -> ORDER BY count DESC;
+-------+-------+
| state | count |
+-------+-------+
| CA    |  5575 |
| NY    |  4120 |
| TX    |  3414 |
| PA    |  3141 |
| OH    |  3092 |
```

This query grabs the state and count for any state in vets that matches an Abbreviated state in states, then groups by state and sorts the output by the count in descending order.

3. Obtain List of States in vets But Not States:

Pretty simple, I'll just slightly alter the previous command to contain `NOT IN` instead of `IN`.

```
MariaDB [fisher]> SELECT state, COUNT(*) AS count  FROM vets WHERE state NOT IN ( SELECT Abbrev FROM states ) GROUP BY state ORD
ER BY count DESC;
+-------+-------+
| state | count |
+-------+-------+
| XC    |    58 |
| XP    |    32 |
| VQ    |    15 |
| XG    |     7 |
| XM    |     5 |
| XJ    |     4 |
| SO    |     4 |
| ZZ    |     3 |
| XE    |     3 |
| XB    |     3 |
| XN    |     3 |
| XF    |     2 |
| XI    |     2 |
| CZ    |     2 |
| XA    |     1 |
| XS    |     1 |
+-------+-------+
16 rows in set (0.044 sec)
```

This query shows all states/territories not contained in the states table, as well as their counts.

4. Source .sql files:

Something different now: sourcing .sql files and observing their results. First, we do *StudentExample.sql*.

```
MariaDB [fisher]> SOURCE StudentExample.sql
Query OK, 1 row affected (0.001 sec)

Database changed
Query OK, 0 rows affected (0.024 sec)
```

We can see the database changed, and we are no longer in `fisher`. Let's list the tables of our new database:

```
MariaDB [mynewpaltz]> SHOW TABLES;
+----------------------+
| Tables_in_mynewpaltz |
+----------------------+
| CourseTable          |
| EnrollmentTable      |
| StudentTable         |
+----------------------+
3 rows in set (0.001 sec)
```

This new database looks like an example of data stored by a university regarding a class.

Now, let's source *EmployeesDatabase.sql*:

```
MariaDB [mynewpaltz]> SHOW TABLES;
+----------------------+
| Tables_in_mynewpaltz |
+----------------------+
| CourseTable          |
| EnrollmentTable      |
| StudentTable         |
| departments          |
| employees            |
+----------------------+
5 rows in set (0.000 sec)
```

We can see that we now have tables `departments` and `employees` in our mynewpaltz database.

That concludes the assignment.