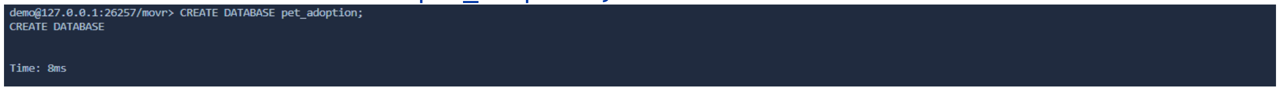
Set Up Dog Shelter Database: CREATE DATABASE

Animals 4 Homes is getting ready to open and help dogs find loving homes.

To get started, we need to set up a SQL database to store and manage each dog as they come in and get adopted.

First, we need to create a database for our data. We'll name it pet_adoption using the following SQL statement:

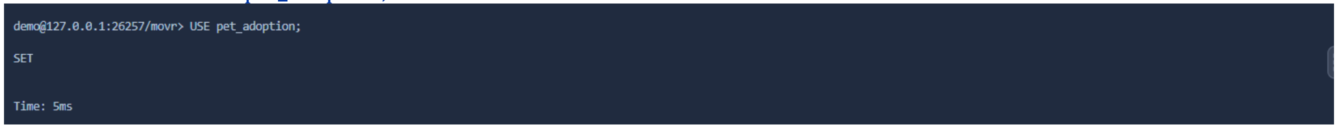
```
CREATE DATABASE pet_adoption;
```



```
demo@127.0.0.1:26257/movr> CREATE DATABASE pet_adoption;
CREATE DATABASE


Time: 8ms
```

Creating a database doesn't automatically set it as the active database, so now let's select the pet_adoption database with the `USE` command:

```
USE pet_adoption;
```



```
demo@127.0.0.1:26257/movr> USE pet_adoption;

SET


Time: 5ms
```

Next, we need to set up the tables that will store our data.

For this project, let's create just two tables: animals and adoptions.

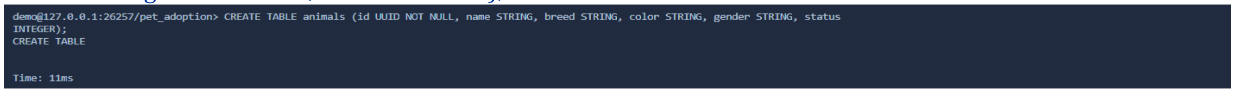
Table #1: A Table for Animals: CREATE TABLE & UUID

The first table, animals, will maintain a list of the dogs that come

through our shelter. Create the animal table with the following

command:

```
CREATE TABLE animals (id UUID NOT NULL, name STRING, breed STRING, color STRING,
gender STRING, status INTEGER);
```



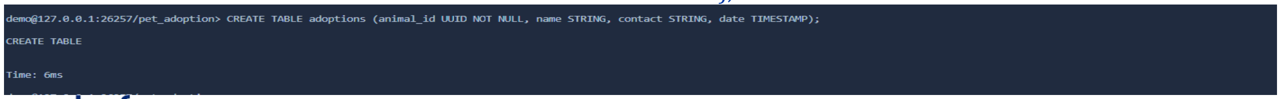
```
demo@127.0.0.1:26257/pet_adoption> CREATE TABLE animals (id UUID NOT NULL, name STRING, breed STRING, color STRING, gender STRING, status
INTEGER);
CREATE TABLE


Time: 11ms
```

Table #2: The List of Adoptions: TIMESTAMP

The second table will be named adoptions. It will be used to track all adoption transactions.

Create this adoptions table using the following command:

```
CREATE TABLE adoptions (animal_id UUID NOT NULL, name STRING, contact STRING,
date TIMESTAMP);
```

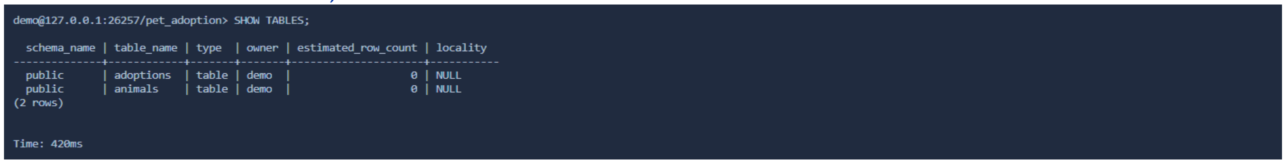


```
demo@127.0.0.1:26257/pet_adoption> CREATE TABLE adoptions (animal_id UUID NOT NULL, name STRING, contact STRING, date TIMESTAMP);

CREATE TABLE


Time: 6ms
```

Verify Database Setup: SHOW TABLES & COLUMNS

Run this command to get the list of tables in the current database and check that we have both the animals table and the adoptions table:

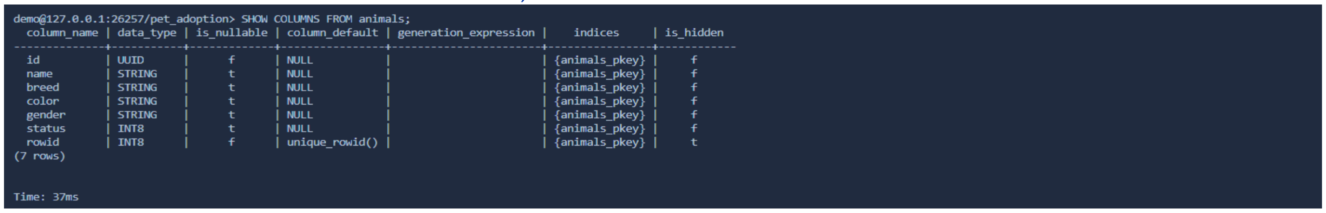
```
SHOW TABLES;
```



```
demo@127.0.0.1:26257/pet_adoption> SHOW TABLES;

  schema_name | table_name | type  | owner | estimated_row_count | locality
--------------+------------+-------+-------+---------------------+-----------
  public      | adoptions  | table | demo  |                   0 | NULL
  public      | animals    | table | demo  |                   0 | NULL
(2 rows)


Time: 420ms
```

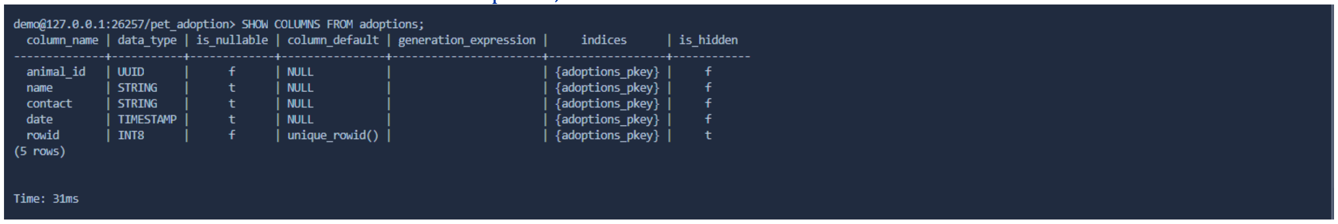
If you see both tables, then you can run these two statements to make sure that the columns of each are correct:

```
SHOW COLUMNS FROM animals;
```



```
demo@127.0.0.1:26257/pet_adoption> SHOW COLUMNS FROM animals;
  column_name | data_type | is_nullable | column_default | generation_expression |    indices     | is_hidden
--------------+-----------+-------------+----------------+-----------------------+----------------+------------
  id          | UUID      |      f      | NULL           |                       | {animals_pkey} |     f
  name        | STRING    |      t      | NULL           |                       | {animals_pkey} |     f
  breed       | STRING    |      t      | NULL           |                       | {animals_pkey} |     f
  color       | STRING    |      t      | NULL           |                       | {animals_pkey} |     f
  gender      | STRING    |      t      | NULL           |                       | {animals_pkey} |     f
  status      | INT8      |      t      | NULL           |                       | {animals_pkey} |     f
  rowid       | INT8      |      f      | unique_rowid() |                       | {animals_pkey} |     t
(7 rows)


Time: 37ms
```

```
SHOW COLUMNS FROM adoptions;
```



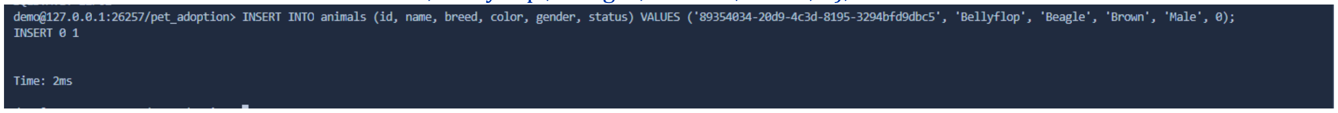
```
demo@127.0.0.1:26257/pet_adoption> SHOW COLUMNS FROM adoptions;
  column_name | data_type | is_nullable | column_default | generation_expression |     indices      | is_hidden
--------------+-----------+-------------+----------------+-----------------------+------------------+------------
  animal_id   | UUID      |      f      | NULL           |                       | {adoptions_pkey} |     f
  name        | STRING    |      t      | NULL           |                       | {adoptions_pkey} |     f
  contact     | STRING    |      t      | NULL           |                       | {adoptions_pkey} |     f
  date        | TIMESTAMP |      t      | NULL           |                       | {adoptions_pkey} |     f
  rowid       | INT8      |      f      | unique_rowid() |                       | {adoptions_pkey} |     t
(5 rows)


Time: 31ms
```

Now that our database schema looks good, we're ready to start accepting dogs at our shelter.

Add Dogs to Database: INSERT

They are eager to check in and look cute to visitors, so let’s add them to the system quickly. We can do this using an INSERT statement on the animals table that looks like the following:

INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('89354034-20d9-4c3d-8195- 3294bfd9dbc5', 'Bellyflop', 'Beagle', 'Brown', 'Male', 0);



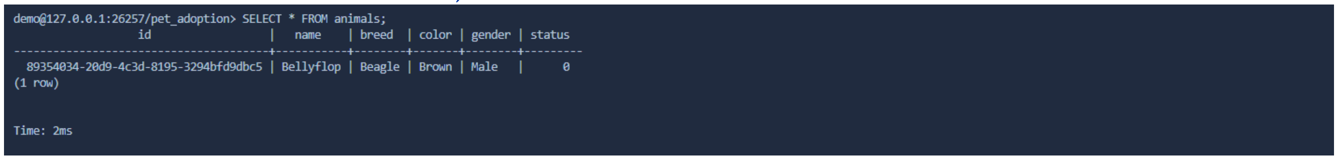
```
demo@127.0.0.1:26257/pet_adoption> INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('89354034-20d9-4c3d-8195-3294bfd9dbc5', 'Bellyflop', 'Beagle', 'Brown', 'Male', 0);
INSERT 0 1


Time: 2ms
```

Retrieve List of Dogs: SELECT * FROM

With the full list of dogs added to our database, we can try running some SELECT queries to look through them. The following are some small examples of possible SQL statements to run.

Get the full list of all properties of all dogs (defaults to a limit of 100 rows):

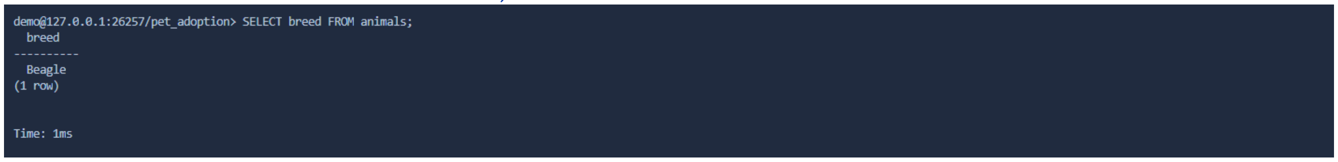
SELECT * FROM animals;



```
demo@127.0.0.1:26257/pet_adoption> SELECT * FROM animals;
                   id                  |   name    | breed  | color | gender | status
---------------------------------------+-----------+--------+-------+--------+---------
  89354034-20d9-4c3d-8195-3294bfd9dbc5 | Bellyflop | Beagle | Brown | Male   |      0
(1 row)


Time: 2ms
```

Get the breeds of all dogs:

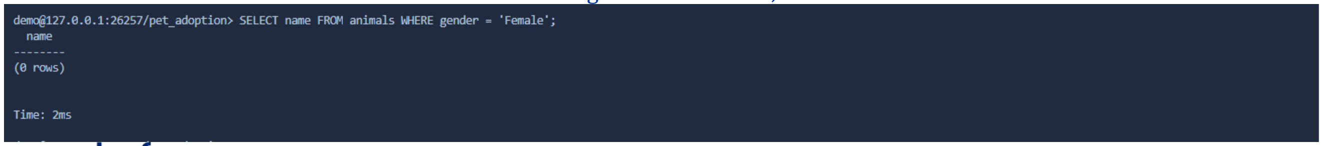
SELECT breed FROM animals;



```
demo@127.0.0.1:26257/pet_adoption> SELECT breed FROM animals;
  breed
----------
  Beagle
(1 row)


Time: 1ms
```

Get the names of only female dogs by including a WHERE clause:

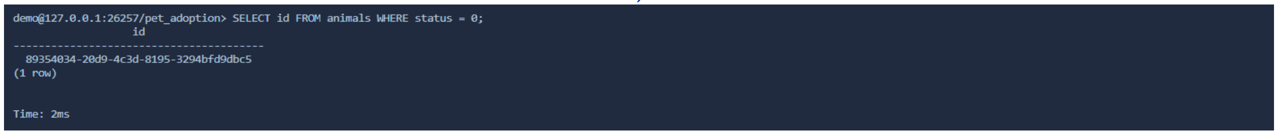
SELECT name FROM animals WHERE gender = 'Female';



```
demo@127.0.0.1:26257/pet_adoption> SELECT name FROM animals WHERE gender = 'Female';
  name
--------
(0 rows)


Time: 2ms
```

Get the IDs of dogs up for adoption:

SELECT id FROM animals WHERE status = 0;



```
demo@127.0.0.1:26257/pet_adoption> SELECT id FROM animals WHERE status = 0;
                   id
----------------------------------------
  89354034-20d9-4c3d-8195-3294bfd9dbc5
(1 row)


Time: 2ms
```