Getting Started Guide and Tutorial

Once you have followed the installation procedure and have viewmydb installed correctly so that you can run it, you are now ready to start using the program and connect to one of your databases. This guide will briefly show you how to quickly set up databases for both MySQL and PostgreSQL so that you have a database to connect to and can start using the viewmydb interface. If you already have databases set up and are able to connect to them, then you can skip to the section explaining how the program works.

Getting set up

Again, this section of the guide assumes that you have no databases setup for MySQL or PostgreSQL. If you already have MySQL and PostgreSQL setup, then you can skip this section.

When setting up PostgreSQL, the biggest thing to keep in mind is how PostgreSQL treats user roles in association to databases. When you installed PostgreSQL, the installation procedure created a user role for you called “postgres” that works with the default Postgres role. The first thing to do is to log into that user account by typing the following command.
$ sudo -u postgres -i

You should now be logged in as the ‘postgres’ user and can now access the PostgreSQL command line utility psql. For this guide, we will not go over how to create a new role or associate databases with your main user account and instead we will just be using the deafault ‘postgres’ user account. However, here is a good guide that goes in more depth on what you can do to set up PostgreSQL
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-14-04

Now that you are logged in as the postgres user, you can create a PostgreSQL database with the following command.
postgres~$ createdb [new database name];

You can now use this new database as the database you connect to when you run the viewmydb program.

IMPORTANT! Note that unless you go in and make changes to your PostgreSQL user roles, you will have to be logged in as the default ‘postgres’ user when you run the viewmydb program in order to connect to your database. So, make sure you run the program from the account of a postgres user.

Now, we can set up MySQL. If you are still logged in as the postgres user, then switch back to your main user account. You can just type exit at the command prompt and it will go back to your main user account.

In order to run the MySQL command line interface that was installed when you installed the MySQL Server, you can run the following command
$ mysql -uroot -p

This will open up the mysql command prompt and you can then enter in a command to create a database.
mysql> CREATE DATABASE [new database name];

Now, you have a database created that you can use to connect to when running the viewmydb program.

IMPORTANT! When running viewmydb and connecting to a MySQL database, unless you made changes to your MySQL setup, then you should use the username ‘root’ and the password that you created when you installed MySQL.

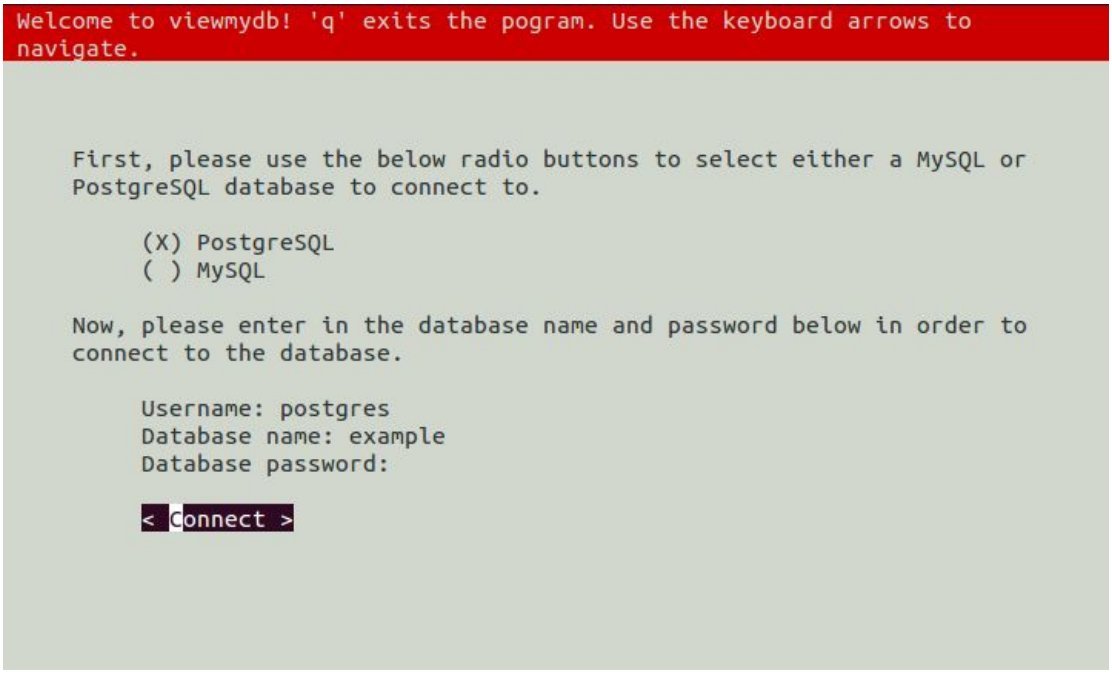
You now have databases created in both MySQL and PostgreSQL that you should be able to connect to from the viewmydb program and use within the program.

Using the viewmydb Program/Tutorial Walkthrough
Now that everything is set up, you can run the viewmydb program and start using it. Remember that if you are using the default postgres account or a database that was created with that account, you need to run the program from that user account. If you set up your own user account and database, run the program from that user account. Also, with MySQL, if you are using the database created in the previous section, then connect as user “root”.

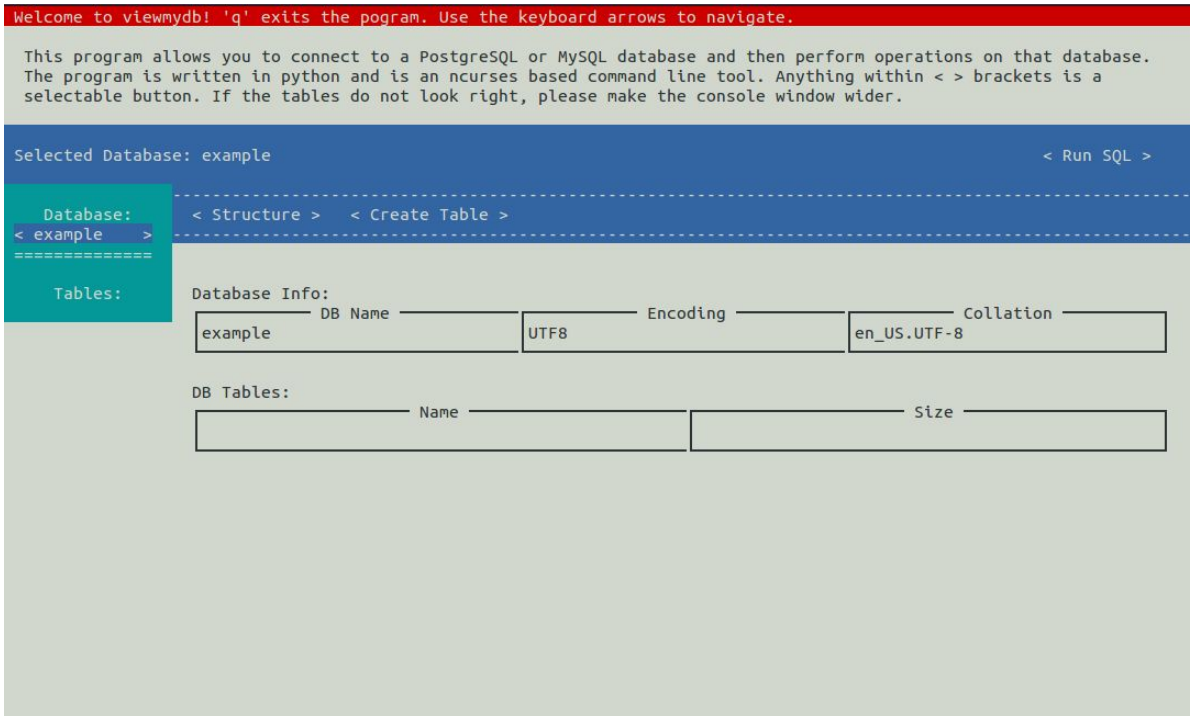
Run the program with the following command
$ viewmydb

You will see the login screen first and you can go ahead and enter your credentials and select which database type you want to connect to, either MySQL or PostgreSQL. Go ahead and connect to a database and you will be directed to the main program interface.

For this guide/tutorial we will go through some of the functionality of the program using a PostgreSQL database. The interface and functionality are identical for both PostgreSQL and MySQL, so anything covered in this walkthrough is also possible using MySQL. This guide will start with a new database like the one you might have created early in the getting set up section. The database created for this guide is called “example” and here is a view of what it looks like to log in to that database. In this case, we have not set up a postgres password, so it is not necessary to include one, however it is recommended to set up a password for your database.



Once you login, you will be shown the main user interface, which should look like the below image. It is recommended that you resize your terminal window to make it wider so that some of the UI elements will render correctly, such as larger tables.



As you can see, we do not have any tables created for this database. However, you can see that you are currently selecting the database “example”. In the UI, whatever database or table you have currently selected will show up in top view next to either “Selected Database” or “Selected Table”. Since we have selected the “example” database, we can view its structure (which is the default view shown above) or we can create a table. Let’s go ahead and create a table by navigating to the Create Table button and clicking on it. Then, fill in the table name and number of fields for the new table like below.

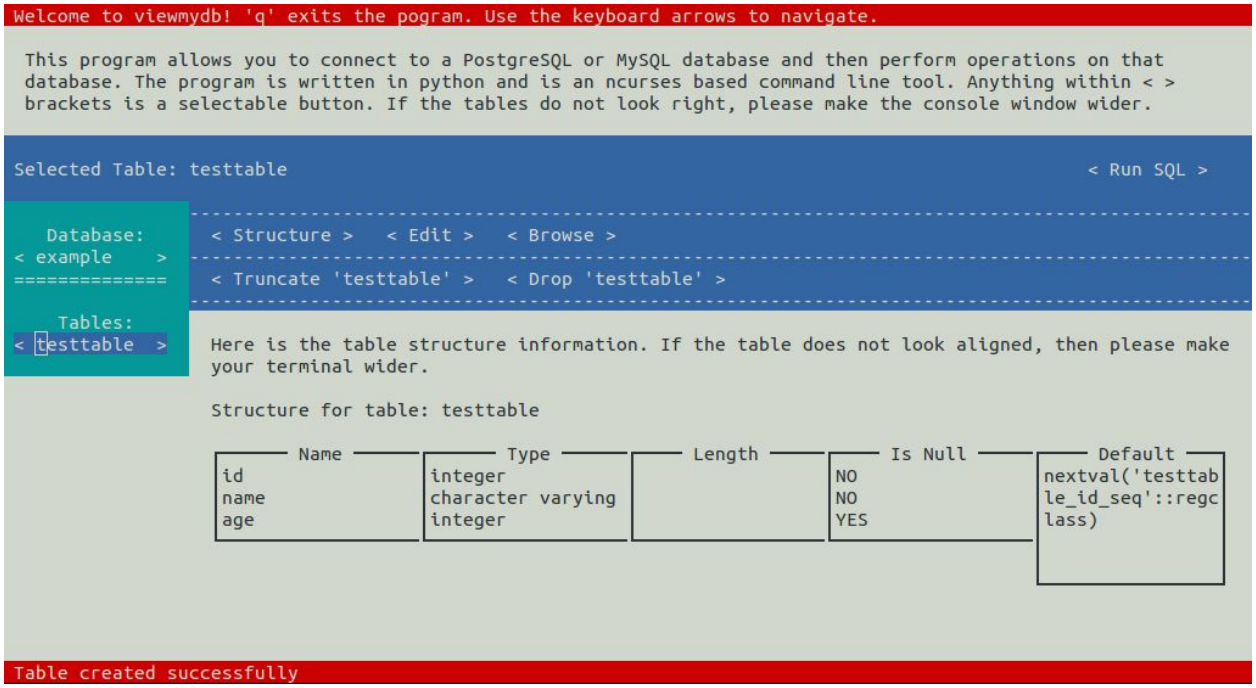
Click next step and you will be prompted to enter in information for each of the fields that you designated to create. Here is the information we are going to enter:
Field 1
Name: id
Type: serial
check Not Null and Primary Key

Field 2
Name: name
Type: varchar
check Not Null and Unique

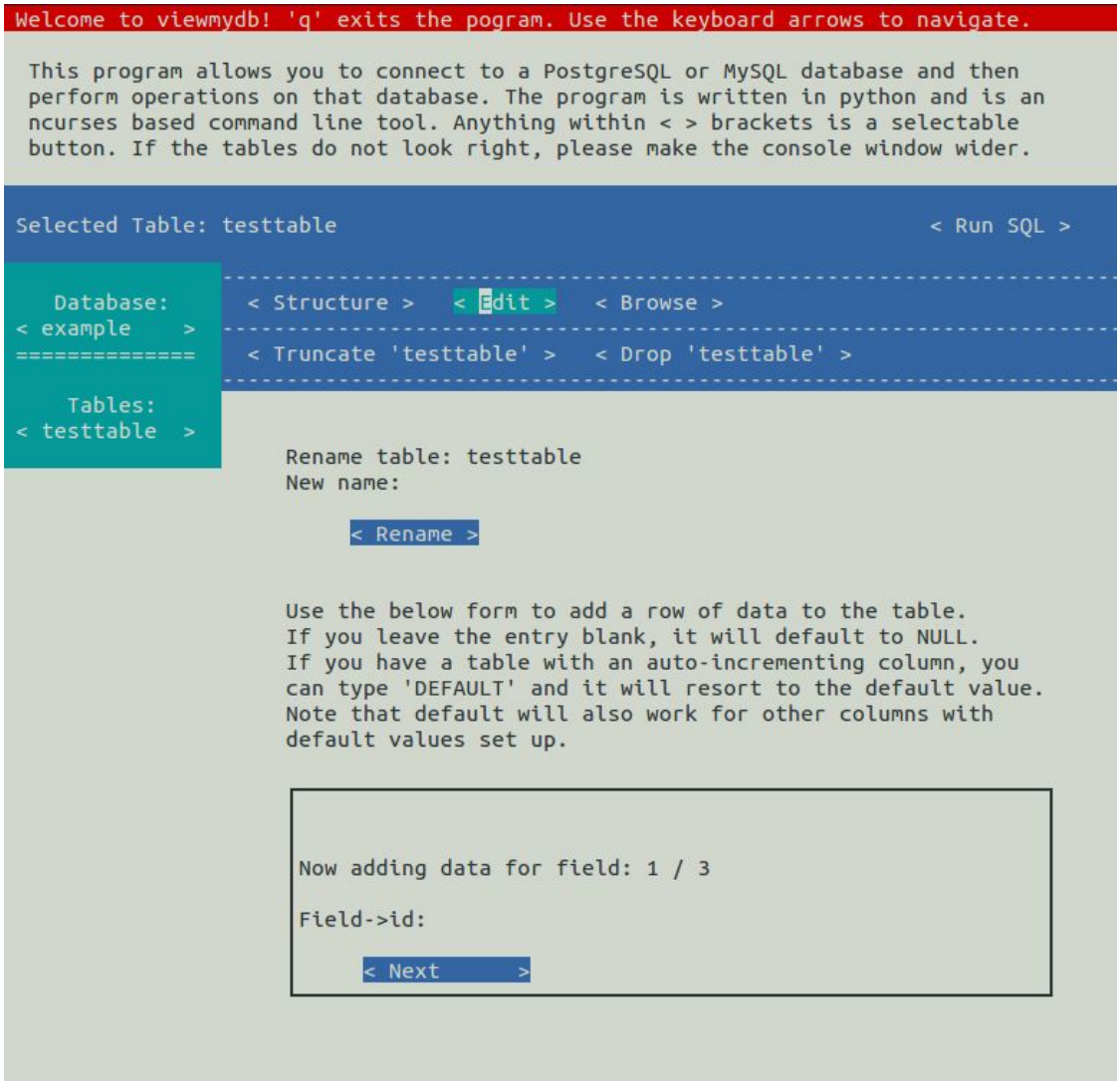
Field 3
Name: age

Type: integer
do not check Not Null and have None checked

Enter all of this in and click create to create your table. You should notice a success message at the bottom of the screen and you should now be able to see your table name show up in the left most column. Go ahead and navigate to the table name and click on it to see the below view.

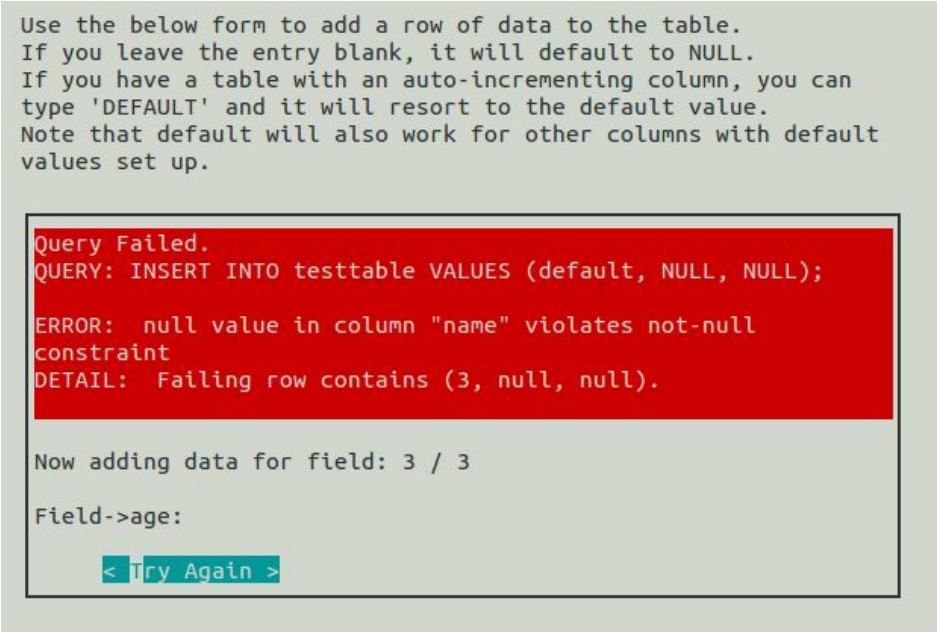


As you can see, you have now selected the table “testtable” as shown up top. Since you have now selected a table instead of a database, you have more options for what you can do. The default view for a table, shown above, is the table structure, but we can also edit the table, browse the data in the table, truncate the table, or drop the table. Since we just created the table, the table is empty and we should now add some data into it. This can be done by going to the “Edit” button and clicking it, which shows the below view.

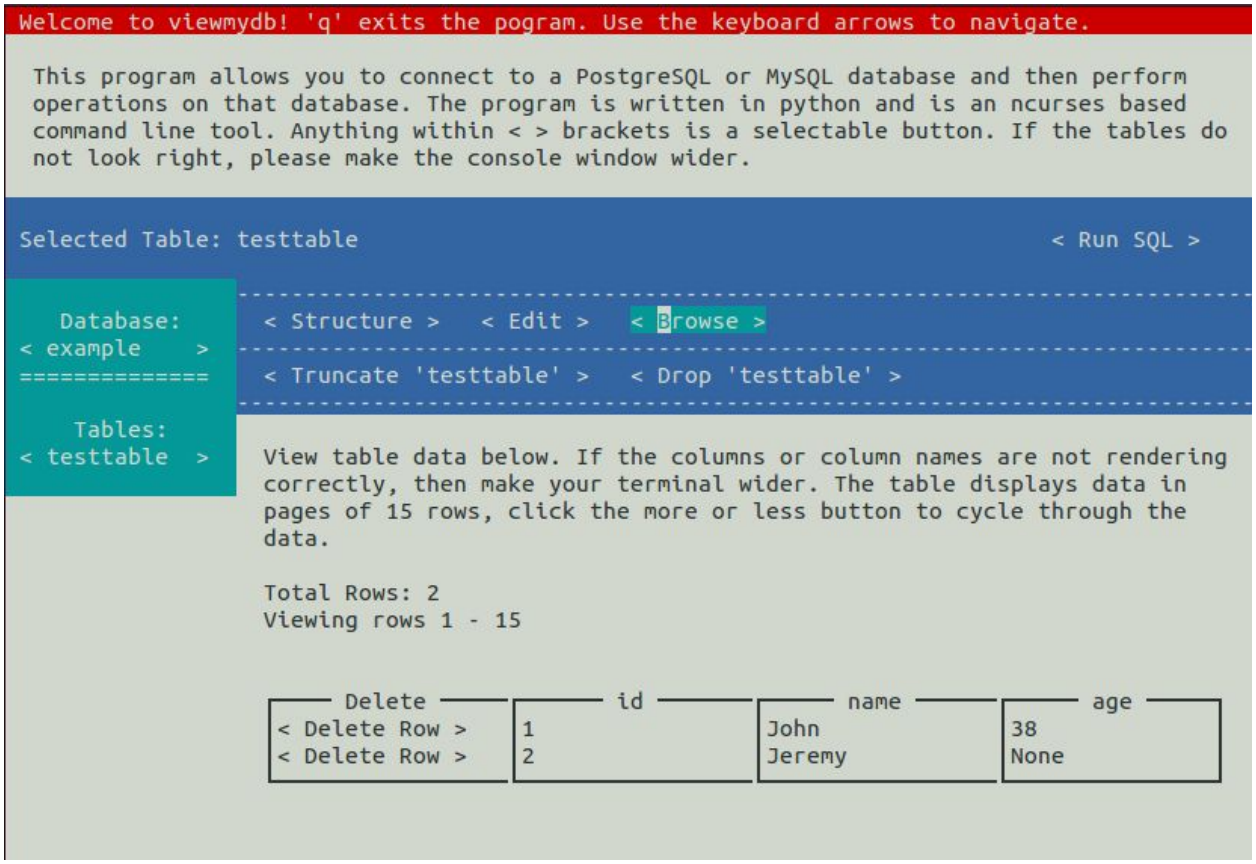


As you can see, you can also rename the table name from within this view as well. To do this, all you would need to do is enter a new table name into the “New name:” field and then click “Rename”. Now, let’s create some data for this table. Within the box, it will step you through each column field one by one and allow you to enter data for the field. For the first field “id”, since we set it to type SERIAL, which is auto-incrementing, we can enter its value as “default” and it will auto-increment the id field. Enter in the following data to create a row in the table
id: default
name: John
age: 38
Once you finish this and add the data, you should see a success message at the bottom showing you that the data was added successfully. Let’s do this again with the following data
id: default
name: Jeremy
age: (leave blank)

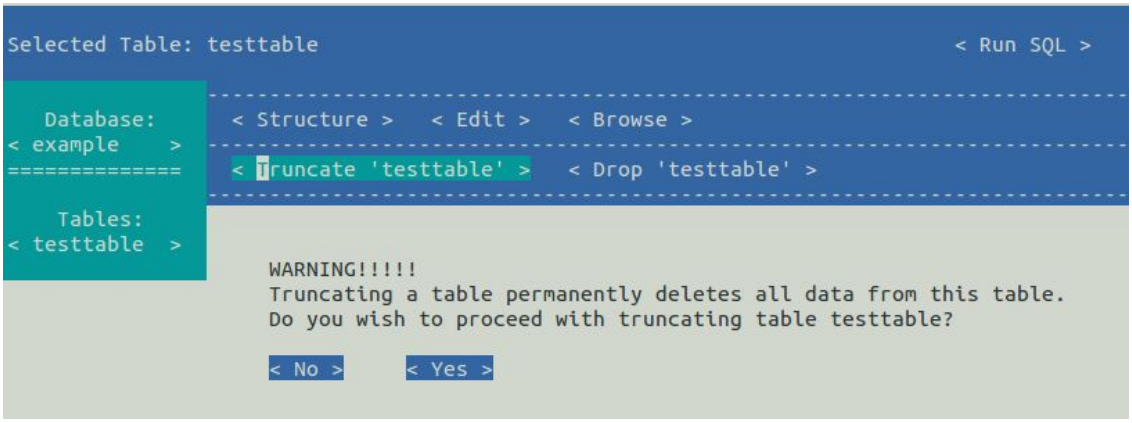
And again
id: default
name: (leave blank)
age: (leave blank)
Submitting this last entry with a blank name should show an error message and the data will not be added because of this. This is because when we created the table, we made it so that name could not be Null, so it gives us this error below.

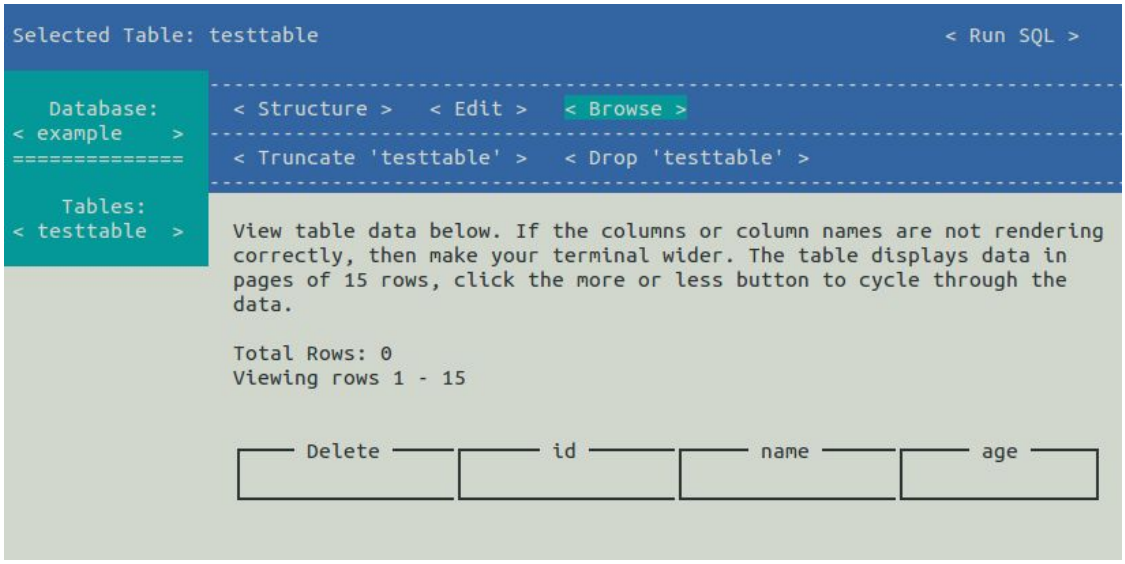


You can click Try Again and it will reset the view and let you attempt to add data again if you want to do that. However, now that we have added some data, let's click on the "Browse" button so that we can see the data that we added. Clicking on "Browse" will show the below view where you can see the two rows of data that we just added.

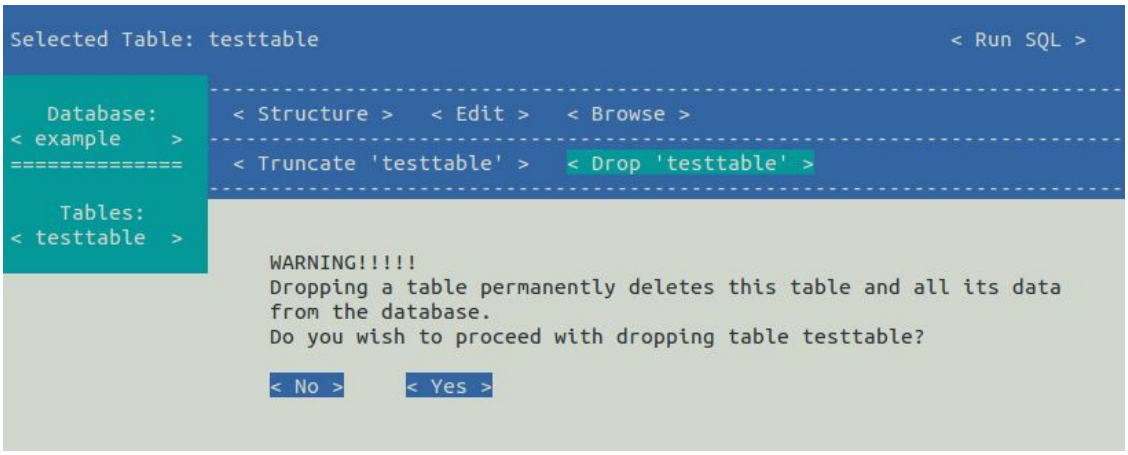


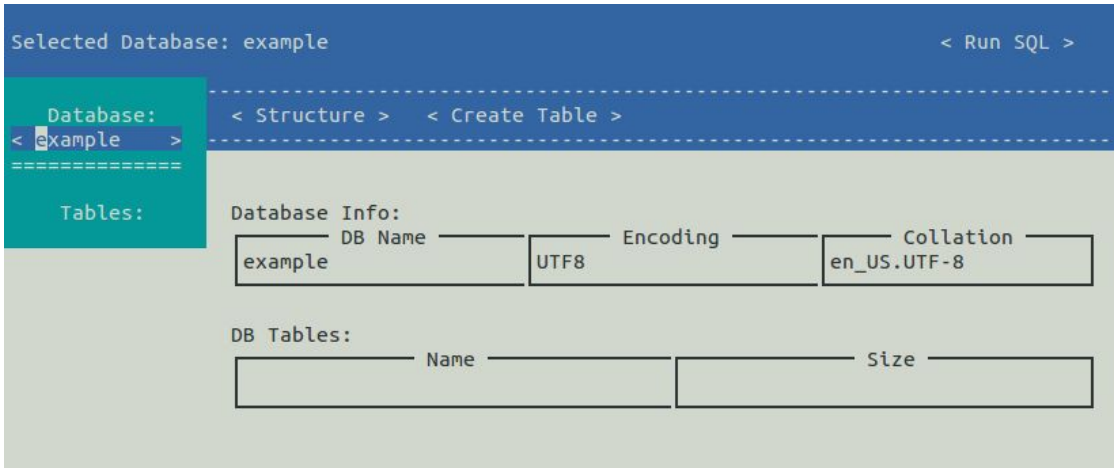
In the above view, if we wanted to, we could click on the “Delete Row” button and it would delete just that row of data from the table. Instead, let’s click on the “Truncate” button, which will delete all of the data from our table. As you can see, a warning message pops up, go ahead and click Yes and you should notice a success message at the bottom showing that the table has been truncated. Now go back to Browse and you should see that all of the data is gone. The below images illustrate this.



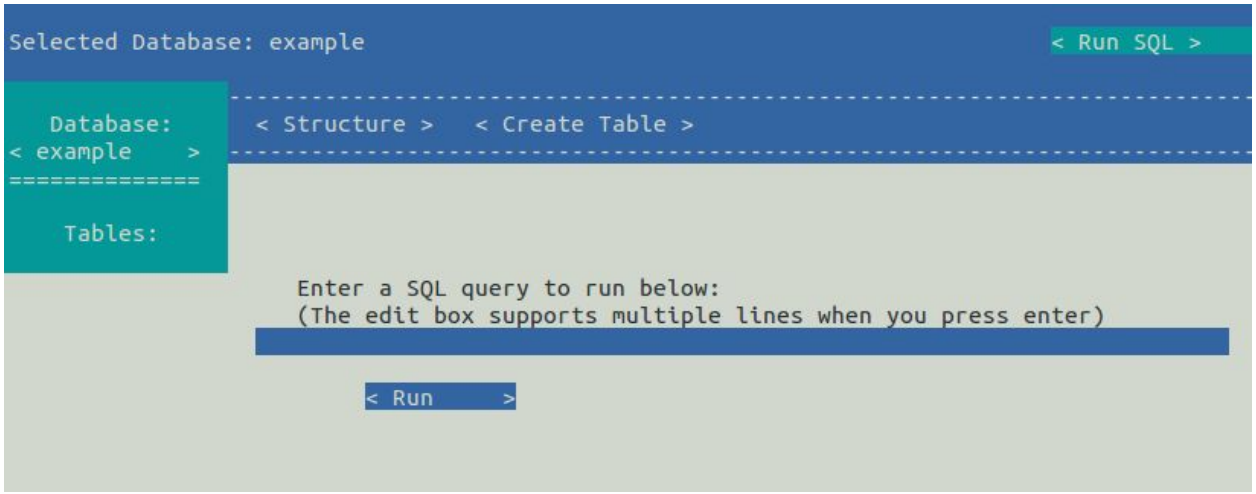


Now, let's drop our table, which deletes the table entirely. Go to "Drop" and click it and you should see a similar error message to truncate and go ahead and click it. You should see a message at the bottom showing you the table was dropped successfully and now the table is gone from the table selection column on the left. Also, you are no longer currently selecting the table you created and are now back at the default view of having the database "testtable" selected. The below images show this.



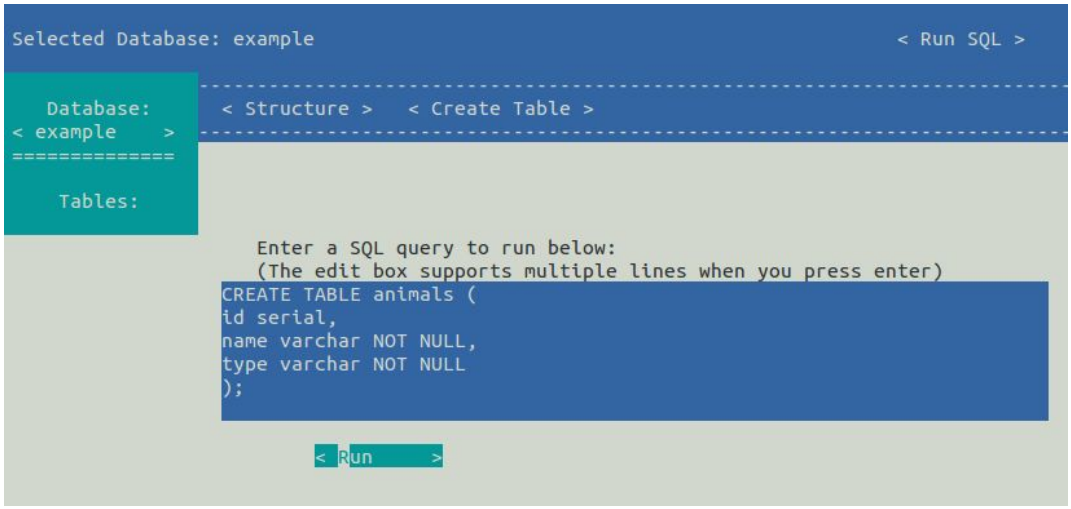


Now, we are back at square one, so let's explore one of the other features of the program. If you go up to the "Run SQL" button on click on that, it will give you this view

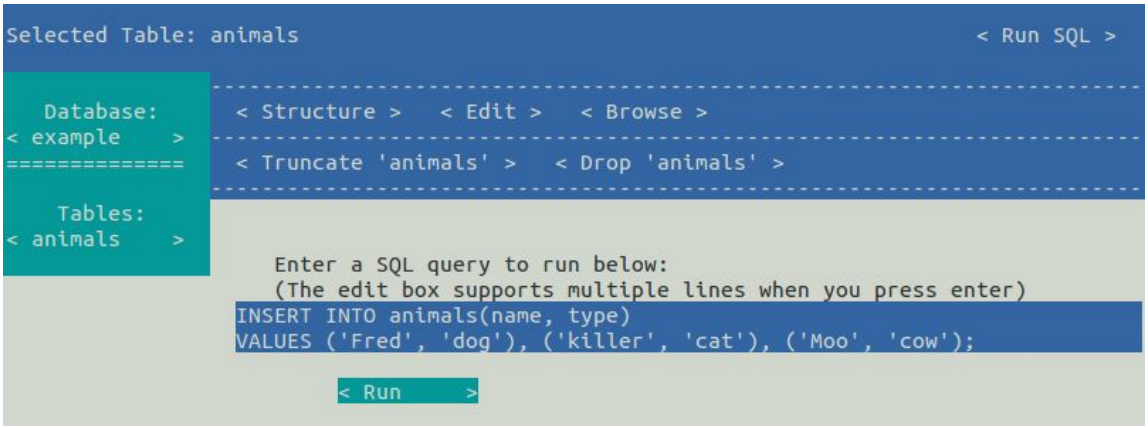


This feature is one of the more powerful features and what it does is allow you to enter in a direct SQL query in either PostgreSQL or MySQL, depending on whatever type of database you are connected to. Next, we will enter in some SQL code to show how this works and create another table since we deleted the last one. Enter in the below code to create a new table "animals" (as you can see, the input field allows for newlines, so you can format your query using the enter key)

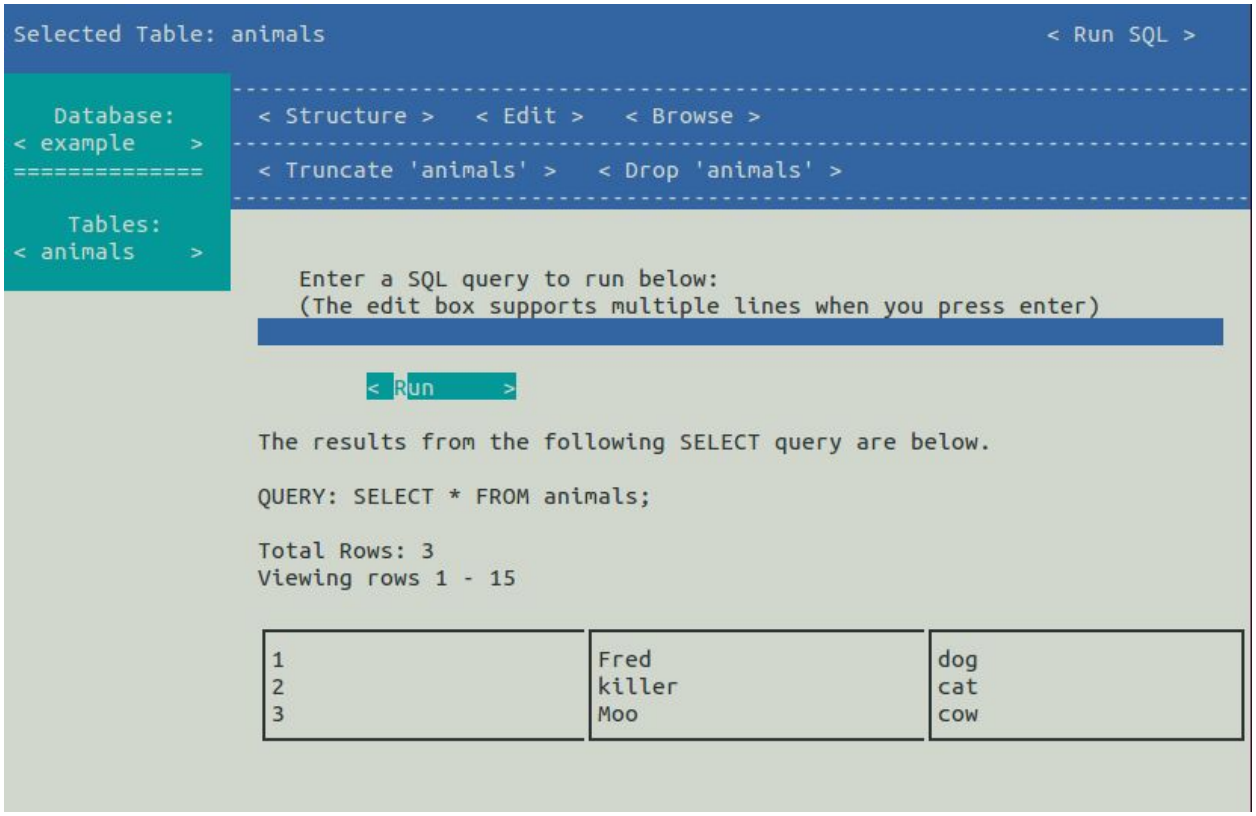
```
CREATE TABLE animals (
        id SERIAL,
        name VARCHAR NOT NULL,
        type VARCHAR NOT NULL
);
```



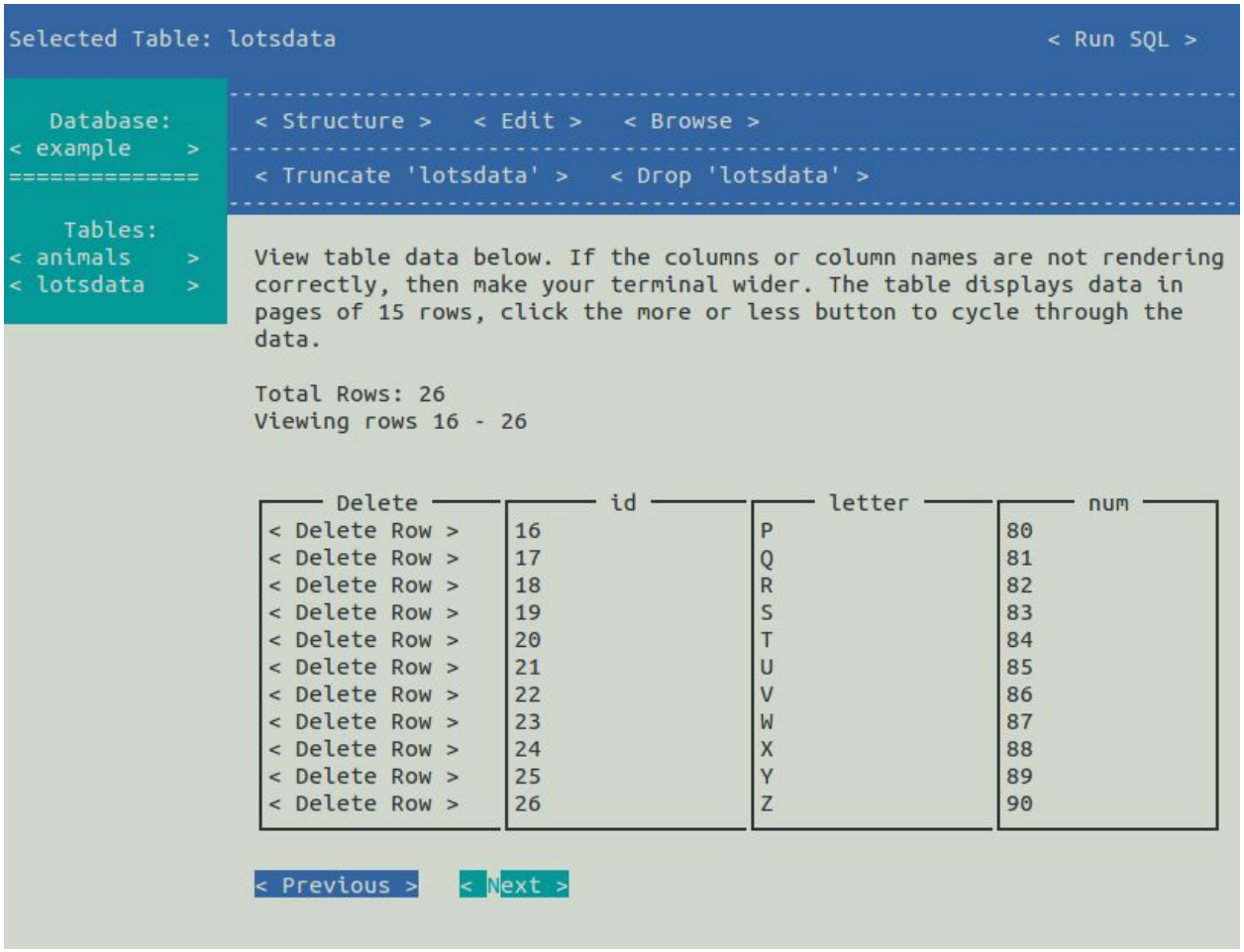
Once you click run, you should see a success message at the bottom showing that the query worked and you should also see the new table “animals” show up in the left side column. Now go back to “Run SQL” and we can add some data to this table. Note that you can click “Run SQL” at any time, whether you have currently selected a database or a table and it will work. This is because running a manual entered query is independent of the database or table currently selected. Enter in the following query
INSERT INTO animals(name, type)
VALUES (‘Fred’, ‘dog’),(‘killer’, ‘cat’),(‘Moo’, ‘cow’);



Now, if you select the “animals” table and then go to “Browse” you should see the data that was just added. However, let’s go back to the “Run SQL” view and enter one last query. When entering a manual query through Run SQL, if you enter a SELECT query that returns data, the data that was returned will be shown from within that view. If you enter anything other than a SELECT query, like the previous queries that we just entered, it will redirect you back to the main view. So, let’s try this SELECT query to get all of the data from the animals table.
SELECT * FROM animals;



As you can see above, the returned data is shown along with showing you the query that was run. Finally, in regards to the way that data is returned, if there is ever any more than 15 rows of data, the program will paginate the results so that you can cycle through it in groups of 15 rows at a time. Below is an example of how this works with a table that contains all of the uppercase letters and their decimal ASCII code. As you can see, Previous and Next buttons appear and allow you to cycle through the data.



That's all for the tutorial and the getting started guide. Hopefully this tutorial allowed you to become familiar with the functionality of the program so that now you can go in and use it for yourself. You can use it to manage databases that you already have set up or like we just did in the tutorial start from a clean database and build it within the program. Thanks!