Overview

For this project, we created our own database using pgAdmin 4. With pgAdmin4 we are able to import our CSV file with the data using SQL queries. Once we confirm our database is complete and in BCNF form, we can use our Python to create a CLI interface using scripts so we can allow users to modify the database however they want.

pgAdmin 4

With pgAdmin 4 we can import any CSV data to your database. To create tables, you can choose to use Queries or use the interface provided by pgAdmin. Using the interface, you can create a table by clicking on the left side, if you click “Databases” it will open a table containing your database, click on your database and click on “schema” the schema will have a drop down menu one of them will be “table”. If you right click “table” it will give you an option to create your table with the column and the type of each column.

To create a table using SQL queries, you can create a table using SQL commands.

An example is: “`CREATE TABLE tutorials (id int, tutorial_name text);`”

To Import you CSV file to your table, you can use the command:

```
COPY table_name(Columns_in_table)
FROM 'Path to File'
DELIMITER ','
CSV HEADER;
```

If the data you imported and the data type you set up are incompatable, you can go to the properties of the table to change the type or clear out the contents of the data with this command:

```
TRUNCATE TABLE Table_name
RESTART IDENTITY;
```

Python Script:

The main script prompts the user to enter a value corresponding to the SQL command they want to do and then calls the imported functions to execute the scripts within those functions.

insert.py:

This file contains many functions that ask for the users to enter the data to the corresponding tables since when you insert into a table you would need to enter all information about that data. It prompts the user to enter what table they would like to access to insert data into with choice() and then depending on the input the corresponding table would ask the user to input the data to insert into the table with the SQL query INSERT.

delete.py:

Similar to Insert.py we prompt the user to enter what table they would like to access to with choice() and then depending on the input the corresponding table would ask the user to input the data of the column name and the value to match so we can use it to execute the SQL query DELETE to delete data.

update.py:

We ask the user to enter what table they would like to access to with choice() and then depending on the input the corresponding table would ask the user to input the data of the column name and the value to match so we can use the information to execute the SQL query UPDATE the data for the table.

select_1.py:

We ask the user to enter what table they would like to access to with choice() and then depending on the input the corresponding table would ask the user to input the data of the column name and the value to match so we can use the information to execute the SQL query SELECT the data for the table. This allows the user to search for data have the CLI display the information.

aggregate.py:

This script is a little different, in addition to the choice function, we also have the aggregate function aggregate(). The aggregate function lets the user choose what they want to do with data: add, find average, count, find the minimum and find the maximum. We first tell the user to make a choice on what table they want to modify and then ask the user how they modify the data. Depending on how the user wants to aggregate data, we would perform the corresponding SQL queries for sum(), avg(), count(), min() and max().

sort.py

This script has a special function as well, the user has a choice function to choose what function along with a choice if they want to sort sqlsort() the data by ascending or descending. Depending on what the user chooses to sort by we run SQL query command corresponding to what the user wanted with the SELECT * with desc/asc command.

join.py

This script doesn't have a choice function, instead we get it immediately from the user's input. We select the table name using SQL queries SELECT and join it with a second table we get from the user input with INNER JOIN with the matching columns.

group.py

This script uses the choice function to allow the users to choose a table. Then allow the users to type in the two columns they want to group the columns within the table. We can group it using the SQL query GROUP BY.

sub.py

This script is used for subqueries, we give the users a choice to input what table they wanted to subquery first. Once the table is chosen, we ask the user to enter the column name from the table and another column table from the second table. We use the SQL query command SELECT *from IN (SELECT * from)

transaction.py

This script doesn't have a choice function, instead we get it immediately from the user's input. We allow the user to select what they want to do for a transaction() they have the option to update or rollback. If the user selects update we select the table name with user input using SQL queries UPDATE the data with a new value for each transaction. If we want to rollback we give them the option to rollback with connection.rollback()