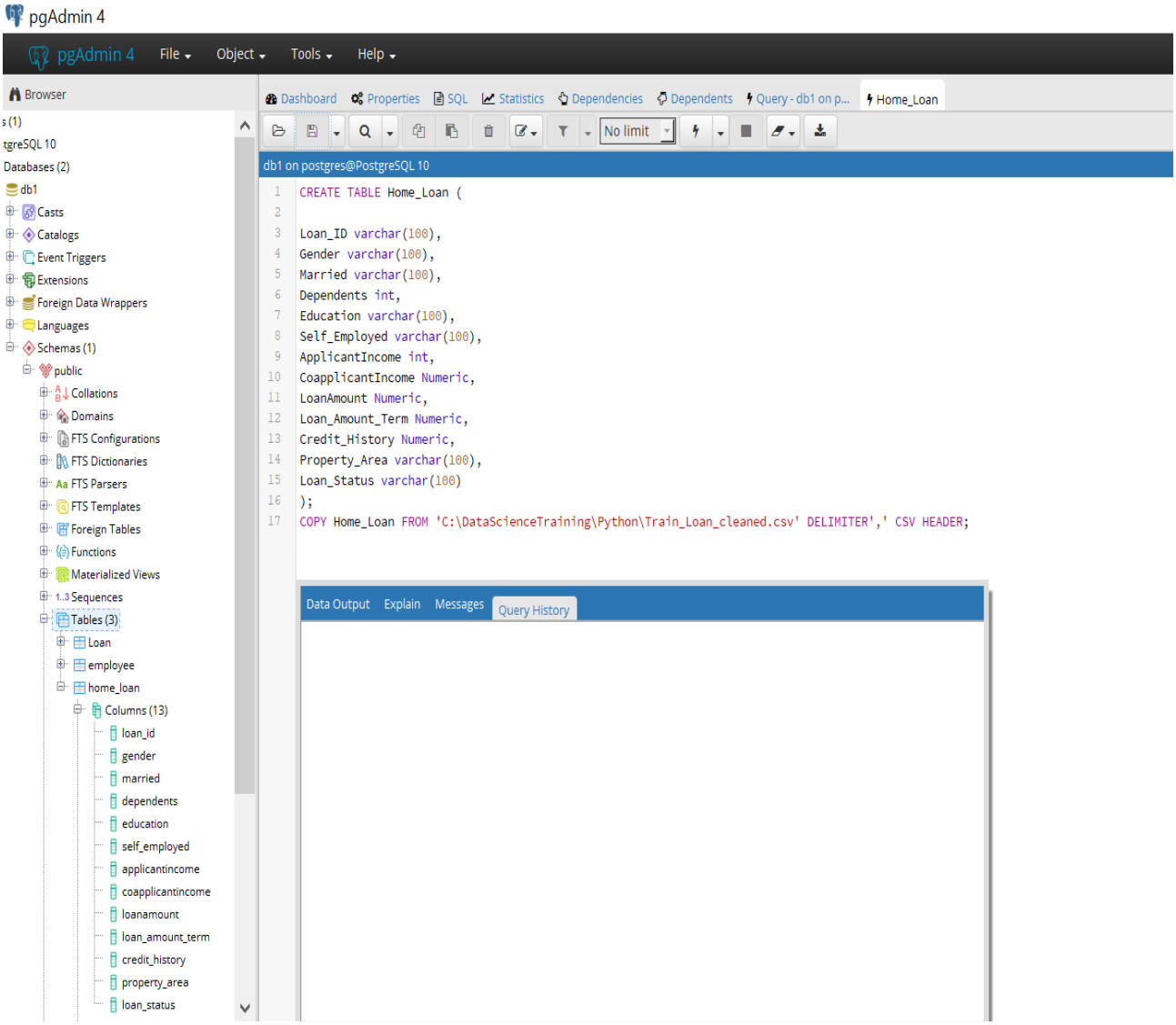
CREATING POSTGRES TABLE:

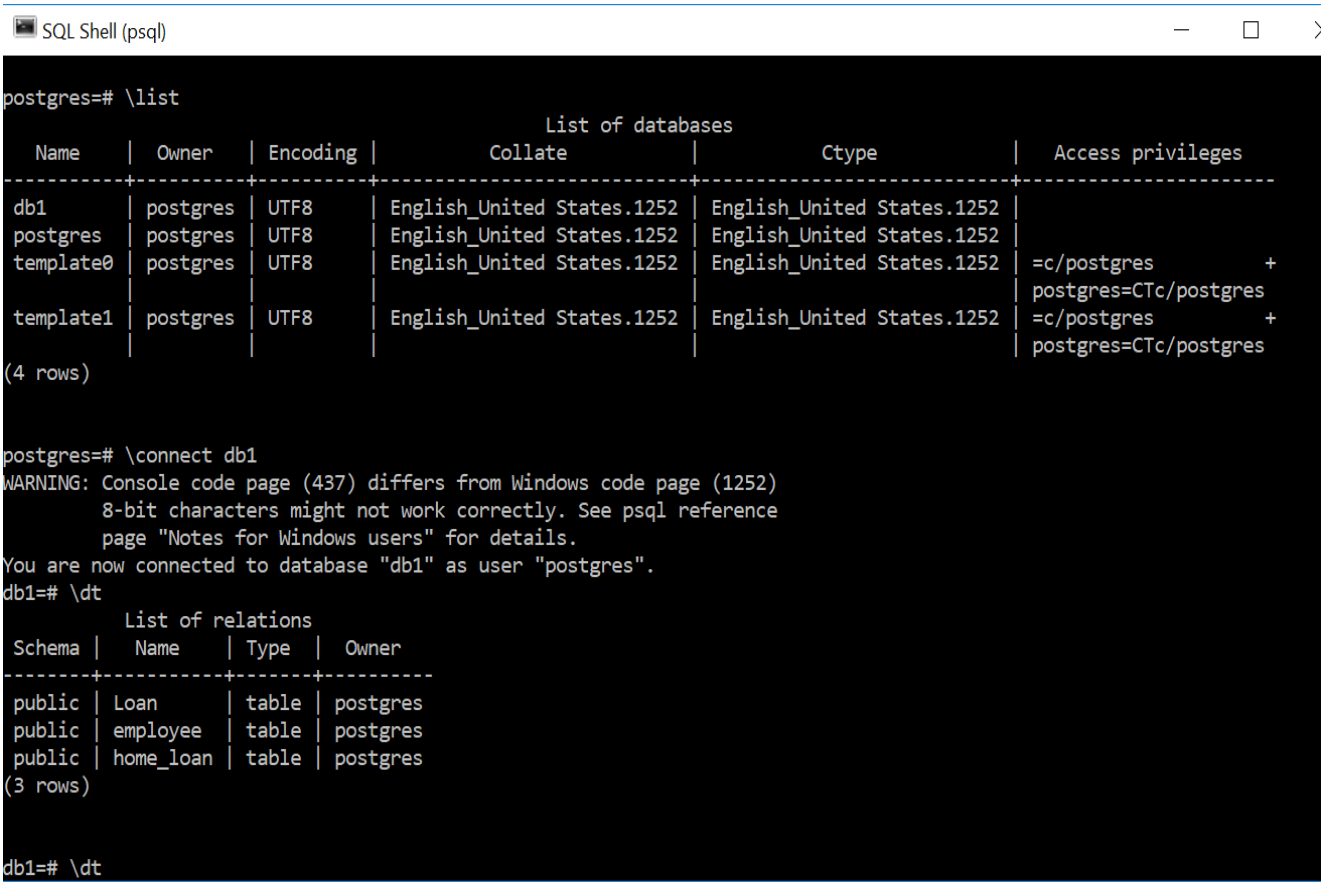
I have used the Loan Prediction dataset to create the Postgres table. You can download the dataset from the link below.

https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/

Step 1: Write the query tool as follows



Step 2: To view the list of tables in your database including the table that you created above use the following commands



Reading Data in Pandas Data frame from Postgres Table:

Step 1: Install the Python package ‘pscycopg2’ in your anaconda prompt using the following command.

pip3 install psycopg2

Also, you can use ‘pygresql’ package to connect PostgreSQL databases using Python. Use the following command to install it

pip install pygresql

To set up a Python virtual environment (using SSH) and install the above package follow the steps given in the below link

https://www.a2hosting.com/kb/developer-corner/postgresql/connecting-to-postgresql-using-python

The ‘psycopg2’ module content:

psycopg2.**connect**(*dsn=None*, *connection_factory=None*, *cursor_factory=None*, *async=False*, ***kwargs*)

Create a new database session and return a new **connection** object.

```
conn = psycopg2.connect("dbname=test user=postgres password=secret")
```

or using a set of keyword arguments:

```
conn=psycopg2.connect(dbname="test", user="postgres", password="secret")
```

The basic connection parameters are:

- **dbname** – the database name (**database** is a deprecated alias).
- **user** – user name used to authenticate.
- **password** – password used to authenticate.
- **host** – database host address (defaults to UNIX socket if not provided).
- **port** – connection port number (defaults to 5432 if not provided).

class **connection**

Handles the connection to a PostgreSQL database instance. It encapsulates a database session.

Connections are created using the factory function **connect()**.

cursor (*name=None*, *cursor_factory=None*, *scrollable=None*, *withhold=False*)

Return a new **cursor** object using the connection.

commit ()

Commit any pending transaction to the database.

By default, Psycopg opens a transaction before executing the first command, if **commit()** is not called, the effect of any data manipulation will be lost.

The connection can be also set in “autocommit” mode: no transaction is automatically open, commands have immediate effect.

rollback ()

Roll back to the start of any pending transaction. Closing a connection without committing the changes first will cause an implicit rollback to be performed.

close ()

Close the connection now (rather than whenever **del** is executed). The connection will be unusable from this point forward; an **InterfaceError** will be raised if any operation is attempted with the connection. The same applies to all cursor objects trying to use the connection. Note that closing a connection without committing the changes first will cause any pending change to be discarded as if a ROLLBACK was performed.

Refer the following documentation for more information about connecting Postgres Database using Python psycopg2 package.

http://initd.org/psycopg/docs/

http://initd.org/psycopg/docs/usage.html

To know more about connecting Postgres with Pandas and Jupyter refer the following link

http://jgardiner.co.uk/blog/read_sql_pandas

How to import Large data into Postgres SQL? Refer following Links

https://www.youtube.com/watch?v=DGtJAJUIxXw

https://data36.com/sql-where-clause-tutorial-beginners-ep2/