

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

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Browser' pane shows the database structure: PostgreSQL 10, Databases (2), db1, and Schemas (1) including 'public'. Under 'public', the 'Tables (3)' section is expanded, showing 'Loan', 'employee', and 'home_loan'. The 'home_loan' table is selected, and its columns are listed: loan_id, gender, married, dependents, education, self_employed, applicantincome, coapplicantincome, loanamount, loan_amount_term, credit_history, property_area, and loan_status.

The main pane shows the SQL editor with the following query:

```
1 CREATE TABLE Home_Loan (  
2  
3 Loan_ID varchar(100),  
4 Gender varchar(100),  
5 Married varchar(100),  
6 Dependents int,  
7 Education varchar(100),  
8 Self_Employed varchar(100),  
9 ApplicantIncome int,  
10 CoapplicantIncome Numeric,  
11 LoanAmount Numeric,  
12 Loan_Amount_Term Numeric,  
13 Credit_History Numeric,  
14 Property_Area varchar(100),  
15 Loan_Status varchar(100)  
16 );  
17 COPY Home_Loan FROM 'C:\DataScienceTraining\Python\Train_Loan_cleaned.csv' DELIMITER ',' CSV HEADER;
```

At the bottom, the 'Data Output' tab is active, showing an empty table structure for the query results.

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

```
SQL Shell (psql)
```

```
postgres=# \list
```

Name	Owner	Encoding	Collate	Ctype	Access privileges
db1	postgres	UTF8	English_United States.1252	English_United States.1252	
postgres	postgres	UTF8	English_United States.1252	English_United States.1252	
template0	postgres	UTF8	English_United States.1252	English_United States.1252	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	English_United States.1252	English_United States.1252	=c/postgres + postgres=CTc/postgres

```
(4 rows)
```

```
postgres=# \connect db1
```

WARNING: Console code page (437) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.

You are now connected to database "db1" as user "postgres".

```
db1=# \dt
```

Schema	Name	Type	Owner
public	Loan	table	postgres
public	employee	table	postgres
public	home_loan	table	postgres

```
(3 rows)
```

```
db1=# \dt
```

Reading Data in Pandas Data frame from Postgres Table:

Step 1: Install the Python package 'psycopg2' 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, Psycpg 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/>

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/>