Connecting to and Operations on a Database

lesson_3_1_2

Installing PostgreSQL on the Playground Server

At the command line, the first one will require your password:

- `sudo apt update`
- `sudo apt install postgresql postgresql-contrib`

Gain Access to PSQL Command Line

- psql is the interactive terminal for working with PostgreSQL

At the command line:

- `sudo -u postgres psql`

You are not logged in as the "postgres" superuser.

Create User, Database and Grant Access

Create Database

- `CREATE DATABASE cloud_user;`

Create User

- `CREATE USER cloud_user WITH ENCRYPTED PASSWORD 'cloud_user';`

Grant Access to Database by User

- `GRANT ALL PRIVILEGES ON DATABASE cloud_user TO cloud_user;`

You now have a database you can access named cloud_user as the user cloud_user.

Leave PSQL

- `\q`

Configure PostgreSQL For Remote Access

- PostgreSQL installs with all access to remote users turned off; this is a good thing, think security

To allow remote access:

- `sudo nano /etc/postgresql/10/main/pg_hba.conf`

  Add a last entry to the file:

  - `host all all 0.0.0.0/0 md5`

- `CTRL-x` to exit and type `Y` and `Enter\Return` to save

- `sudo nano /etc/postgresql/10/main/postgresql.conf`

  Find the section labeled `CONNECTIONS AND AUTHENTICATION` and above this line

  - `#listen_addresses = 'localhost'` add
  - `listen_addresses = '*'`

- `CTRL-x` to exit and type `Y` and `Enter\Return` to save

- `sudo systemctl restart postgresql.service`

Install Postgresql Driver to Your Virtual Environment

- `conda activate python_data_course`
- `conda install psycopg2`

Start and connect to the Jupyter Notebook server as usual.

Server Operations Using Python's Psycopg2

```
import pandas as pd
import psycopg2

CONNECT_DB = "host=localhost port=5432 dbname=cloud_user user=cloud_user
password=cloud_user"
```

A Word About Database Connections

In the cell below you will see I used a `try, except, finally` block. There are a couple of main reasons and I thought it important to call them out.

- Exceptions:
  Exceptions can occur when trying to connect to a database like PostgreSQL. Maybe the server is down. Or perhaps it has already exceeded the maximum number of collections. It is important to catch those errors and report out to the user (even if it is only you).

- Connections:
  There are a limited number of connections a database server can accept. While this is a ratehr large number, it

is possible to reach that number and be refused a connection. Be a good neighbor and always only open a connection for an operation and then close it. Don't open a connection and leave it open while you review the data you have gotten.

Create Table

```
create_table_query = '''CREATE TABLE tips (
    ID SERIAL PRIMARY KEY,
    weekday varchar (10),
    meal_type varchar (10),
    wait_staff varchar (10),
    party_size smallint,
    meal_total float4,
    tip float4
); '''

try:
    # Make connection to db
    cxn = psycopg2.connect(CONNECT_DB)

    # Create a cursor to db
    cur = cxn.cursor()

    # Send sql query to request
    cur.execute(create_table_query)
    records = cxn.commit()

except (Exception, psycopg2.Error) as error :
    print ("Error while connecting to PostgreSQL", error)

finally:
    #closing database connection.
    if(cxn):
        cur.close()
        cxn.close()
        print("PostgreSQL connection is closed")

print(f'Records:\n {records}')
```

Add the Data to Table

```
try:
    # Make connection to db
    cxn = psycopg2.connect(CONNECT_DB)
```

```
        # Create a cursor to db
        cur = cxn.cursor()

        with open('./tips.csv', 'r') as f:
            # skip first row, header row
            next(f)
            cur.copy_from(f, 'tips', sep=",")
            cxn.commit()

    except (Exception, psycopg2.Error) as error :
        print ("Error while connecting to PostgreSQL", error)

    finally:
        #closing database connection.
        if(cxn):
            cur.close()
            cxn.close()
            print("PostgreSQL connection is closed")
            print("tips table populated")
```

Selecting Data From a Server

Use `.fetchall()` with LIMIT or TOP (#)

- LIMIT works for most databases, but does not work with SQL Server
- TOP (#) is used in place of LIMIT on SQL Server

```
def db_server_fetch(sql_query):
    try:
        # Make connection to db
        cxn = psycopg2.connect(CONNECT_DB)

        # Create a cursor to db
        cur = cxn.cursor()

        # Send sql query to request
        cur.execute(sql_query)
        records = cur.fetchall()

    except (Exception, psycopg2.Error) as error :
        print ("Error while connecting to PostgreSQL", error)

    finally:
        #closing database connection.
        if(cxn):
            cur.close()
            cxn.close()
            print("PostgreSQL connection is closed")
        return records
```

Test table is populated by `Select`ing the first five rows.

```
select_query = '''SELECT * FROM tips LIMIT 5;'''

records = db_server_fetch(select_query)
print(records)
```

```
def db_server_change(sql_query):
    try:
        # Make connection to db
        cxn = psycopg2.connect(CONNECT_DB)

        # Create a cursor to db
        cur = cxn.cursor()

        # Send sql query to request
        cur.execute(sql_query)
        records = conn.commit()

    except (Exception, psycopg2.Error) as error :
        print ("Error while connecting to PostgreSQL", error)

    finally:
        #closing database connection.
        if(cxn):
            cur.close()
            cxn.close()
            print("PostgreSQL connection is closed")
        return records
```

Add a new record with the following data:
On Saturday, new wait staff Alfred had one person at Breakfast for 10.76 and received a 0.50 tip.

```
add_data = '''INSERT INTO tips
    (id, weekday, meal_type, wait_staff, party_size, meal_total, tip)
    VALUES
    (504, 'Saturday', 'Breakfast', 'Alfred', 1, 10.76, 0.50);'''
```

```
db_server_change(add_data)
```

Make a SELECT Request to Get New Records

```
select_query = '''SELECT * FROM tips WHERE wait_staff='Alfred';'''

records = db_server_fetch(select_query)
print(records)
```

Accessing a SQL Database With Pandas

pandas.read_sql( ) - loads data from database
pandas.to_sql( ) - write data to database

CAUTION: Please don't write to a database unless you know what you are doing and are authorized. If you are not, your permission should allow read only.

```
def pandas_db_server_fetch(sql_query):
    try:
        # Make connection to db
        cxn = psycopg2.connect(CONNECT_DB)

        # Send sql query to request and create dataframe
        df = pd.read_sql(sql_query, cxn)

    except (Exception, psycopg2.Error) as error :
        print ("Error while connecting to PostgreSQL", error)

    finally:
        #closing database connection.
        if(cxn):
            cxn.close()
            print("PostgreSQL connection is closed")
        return df
```

```
select_query = '''SELECT * FROM tips WHERE wait_staff='Alfred';'''

alfred_df = pandas_db_server_fetch(select_query)
alfred_df.head()
```

```
tips_df = pandas_db_server_fetch('''SELECT * FROM tips;''')
```

```
tips_df.head()
```

```

```