Lecture6-PostgreSQL

January 29, 2018

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
# let's look at how to do more than just read data from PostgreSQL (!)
# first let's populate the sailing club dataframes by querying using SQLAlchemy
import sqlalchemy
```

In [3]:
```
from sqlalchemy import create_engine
engine = create_engine('postgresql+psycopg2://mikejcarey:postquel@localhost/mikejcarey')
```

In [4]:
```
boatquery = "SELECT * FROM boats ORDER BY bid"
boats = pd.read_sql_query(boatquery, con=engine)
boats
```

Out[4]:

	bid	bname	color
0	101	Interlake	blue
1	102	Interlake	red
2	103	Clipper	green
3	104	Marine	red

In [5]:
```
sailorquery = "SELECT * FROM sailors"
sailors = pd.read_sql_query(sailorquery, con=engine)
sailors
```

Out[5]:

	sid	sname	rating	age	major
0	22	Dustin	7.0	45.0	CS
1	29	Brutus	1.0	33.0	EE
2	31	Lubber	8.0	55.5	Econ
3	32	Andy	8.0	25.5	Math
4	58	Rusty	10.0	35.0	CS
5	64	Horatio	7.0	35.0	CS
6	71	Zorba	10.0	16.0	CS
7	74	Horatio	9.0	35.0	Math
8	85	Art	4.0	25.5	Music
9	95	Bob	3.0	63.5	Econ
10	101	Joan	3.0	NaN	Math
11	107	Johannes	NaN	35.0	None

```
In [6]: reservequery = "SELECT * FROM reserves"
        reserves = pd.read_sql_query(reservequery, con=engine)
        reserves
```

Out[6]:

	sid	bid	date
0	22.0	101.0	1998-10-10
1	22.0	102.0	1998-10-10
2	22.0	103.0	1998-10-08
3	22.0	104.0	1998-10-07
4	31.0	102.0	1998-10-11
5	31.0	103.0	1998-11-06
6	31.0	104.0	1998-11-12
7	64.0	101.0	1998-09-05
8	64.0	102.0	1998-09-02
9	74.0	103.0	1993-09-08
10	NaN	103.0	1998-09-09
11	1.0	NaN	2001-01-11
12	1.0	NaN	2002-02-02

```
In [7]: # AND NOW:  let's explore how to add more data to a PostgreSQL table from a dataframe
        #           (so first we'll create some new reservations to add to the reserves table)
        moreres = reserves
        moreres['date'] = "2018-01-27"

In [8]: moreres
```

Out[8]:

	sid	bid	date
0	22.0	101.0	2018-01-27
1	22.0	102.0	2018-01-27
2	22.0	103.0	2018-01-27
3	22.0	104.0	2018-01-27
4	31.0	102.0	2018-01-27
5	31.0	103.0	2018-01-27
6	31.0	104.0	2018-01-27
7	64.0	101.0	2018-01-27
8	64.0	102.0	2018-01-27
9	74.0	103.0	2018-01-27
10	NaN	103.0	2018-01-27
11	1.0	NaN	2018-01-27
12	1.0	NaN	2018-01-27

```
In [9]: # here comes the insertion of new data from the moreres dataframe
        moreres.to_sql("reserves", con=engine, if_exists='append', index=False)

In [10]: reserves = pd.read_sql_query(reservequery, con=engine)
         reserves
```

Out[10]:

	sid	bid	date
0	22.0	101.0	1998-10-10

```
1   22.0  102.0  1998-10-10
2   22.0  103.0  1998-10-08
3   22.0  104.0  1998-10-07
4   31.0  102.0  1998-10-11
5   31.0  103.0  1998-11-06
6   31.0  104.0  1998-11-12
7   64.0  101.0  1998-09-05
8   64.0  102.0  1998-09-02
9   74.0  103.0  1993-09-08
10   NaN  103.0  1998-09-09
11   1.0    NaN  2001-01-11
12   1.0    NaN  2002-02-02
13  22.0  101.0  2018-01-27
14  22.0  102.0  2018-01-27
15  22.0  103.0  2018-01-27
16  22.0  104.0  2018-01-27
17  31.0  102.0  2018-01-27
18  31.0  103.0  2018-01-27
19  31.0  104.0  2018-01-27
20  64.0  101.0  2018-01-27
21  64.0  102.0  2018-01-27
22  74.0  103.0  2018-01-27
23   NaN  103.0  2018-01-27
24   1.0    NaN  2018-01-27
25   1.0    NaN  2018-01-27
```

In [11]:
```
# now let's see how to insert new data into a PostgreSQL table much more directly,
# carefully separating the statement and its parameters to avoid SQL injection risk
# (https://security.openstack.org/guidelines/dg_parameterize-database-queries.html)
```

In [12]:
```
values = (77, 777, "2018-01-07")
engine.execute("INSERT INTO reserves VALUES (%s, %s, %s)", values)
```

Out[12]:
```
<sqlalchemy.engine.result.ResultProxy at 0x10b114278>
```

In [13]:
```
reserves = pd.read_sql_query(reservequery, con=engine)
reserves[reserves.sid > 70]
```

Out[13]:
```
     sid    bid        date
9   74.0  103.0  1993-09-08
22  74.0  103.0  2018-01-27
26  77.0  777.0  2018-01-07
```

In [14]:
```
# values here can be either a Python tuple or a Python array
values = [88, 888, "2018-01-08"]
engine.execute("INSERT INTO reserves VALUES (%s, %s, %s)", values)
```

Out[14]:
```
<sqlalchemy.engine.result.ResultProxy at 0x10e472ef0>
```

```
In [15]: reserves = pd.read_sql_query(reservequery, con=engine)
         reserves[reserves.sid > 70]
```

Out[15]:

	sid	bid	date
9	74.0	103.0	1993-09-08
22	74.0	103.0	2018-01-27
26	77.0	777.0	2018-01-07
27	88.0	888.0	2018-01-08

```
In [16]: # let's look at how we can "UPSERT" new data in a similarly direct manner in PostgreSQL
         # where "UPSERT" means "INSERT if it's not there, and UPDATE (replace) what's there oth
         # (Note: this technique may prove useful later on, when working with the user data in t
         boats
```

Out[16]:

	bid	bname	color
0	101	Interlake	blue
1	102	Interlake	red
2	103	Clipper	green
3	104	Marine	red

```
In [17]: instmt = "INSERT INTO boats (bid, bname, color) VALUES (%s, %s, %s)"
         instmt
```

```
Out[17]: 'INSERT INTO boats (bid, bname, color) VALUES (%s, %s, %s)'
```

```
In [18]: values = (101, 'Sunfish', 'yellow')
         # engine.execute(instmt, values)
```

```
In [19]: upstmt = "INSERT INTO boats (bid, bname, color) VALUES (%s, %s, %s) ON CONFLICT (bid) D
         upstmt
```

```
Out[19]: 'INSERT INTO boats (bid, bname, color) VALUES (%s, %s, %s) ON CONFLICT (bid) DO UPDATE
```

```
In [20]: values = (101, 'Sunfish', 'yellow')
         engine.execute(upstmt, values)
```

```
Out[20]: <sqlalchemy.engine.result.ResultProxy at 0x10e4973c8>
```

```
In [21]: boats = pd.read_sql_query(boatquery, con=engine)
         boats
```

Out[21]:

	bid	bname	color
0	101	Sunfish	yellow
1	102	Interlake	red
2	103	Clipper	green
3	104	Marine	red

```
In [22]: values = (101, 'Interlake', 'blue')
         engine.execute(upstmt, values)
```

```
Out[22]: <sqlalchemy.engine.result.ResultProxy at 0x10e497978>
```

```
In [23]: boats = pd.read_sql_query(boatquery, con=engine)
         boats
```

```
Out[23]:    bid      bname  color
         0  101  Interlake   blue
         1  102  Interlake    red
         2  103    Clipper  green
         3  104     Marine    red
```

```
In [24]: engine.execute("DELETE FROM Reserves WHERE date >= '2018-01-01'", con=engine)
         reserves = pd.read_sql_query(reservequery, con=engine)
         reserves
```

```
Out[24]:      sid    bid        date
         0   22.0  101.0  1998-10-10
         1   22.0  102.0  1998-10-10
         2   22.0  103.0  1998-10-08
         3   22.0  104.0  1998-10-07
         4   31.0  102.0  1998-10-11
         5   31.0  103.0  1998-11-06
         6   31.0  104.0  1998-11-12
         7   64.0  101.0  1998-09-05
         8   64.0  102.0  1998-09-02
         9   74.0  103.0  1993-09-08
         10   NaN  103.0  1998-09-09
         11   1.0    NaN  2001-01-11
         12   1.0    NaN  2002-02-02
```

```
In [25]: # but wait, what if my incoming data doesn't have a natural primary key?
         # or what if I want to timestamp my data based on when I have inserted it?
         # or both...?
         !cat /Users/mikejcarey/desktop/teaching/STATS170ab/ContactExample.sql
```

```
CREATE TABLE Contacts (
  cid SERIAL PRIMARY KEY,
  sid int,
  contact varchar,
  ts timestamp DEFAULT now()
);
```

```
In [26]: contactquery = "SELECT * FROM Contacts"
         contacts = pd.read_sql_query(contactquery, con=engine)
         contacts
```

```
Out[26]: Empty DataFrame
         Columns: [cid, sid, contact, ts]
         Index: []
```

```
In [27]: values = (22, "dustin22@cs.wisc.edu")
         engine.execute("INSERT INTO Contacts (sid, contact) VALUES (%s, %s)", values)

Out[27]: <sqlalchemy.engine.result.ResultProxy at 0x10e4728d0>

In [28]: contacts = pd.read_sql_query(contactquery, con=engine)
         contacts

Out[28]:    cid  sid              contact                          ts
         0    4   22  dustin22@cs.wisc.edu 2018-01-29 20:28:57.169259

In [29]: values = (85, "art123@arts.wisc.edu")
         engine.execute("INSERT INTO Contacts (sid, contact) VALUES (%s, %s)", values)

Out[29]: <sqlalchemy.engine.result.ResultProxy at 0x10e472e80>

In [30]: contacts = pd.read_sql_query(contactquery, con=engine)
         contacts

Out[30]:    cid  sid              contact                          ts
         0    4   22  dustin22@cs.wisc.edu 2018-01-29 20:28:57.169259
         1    5   85  art123@arts.wisc.edu 2018-01-29 20:28:57.210865

In [31]: values = (85, "art123@arts.wisc.edu")
         engine.execute("INSERT INTO Contacts (sid, contact) VALUES (%s, %s)", values)
         contacts = pd.read_sql_query(contactquery, con=engine)
         contacts

Out[31]:    cid  sid              contact                          ts
         0    4   22  dustin22@cs.wisc.edu 2018-01-29 20:28:57.169259
         1    5   85  art123@arts.wisc.edu 2018-01-29 20:28:57.210865
         2    6   85  art123@arts.wisc.edu 2018-01-29 20:28:57.261582

In [32]: # reset the Contacts world for the next time through this notebook...
         # engine.execute("TRUNCATE Contacts RESTART IDENTITY CASCADE", con=engine)
         engine.execute("DELETE FROM Contacts", con=engine)

Out[32]: <sqlalchemy.engine.result.ResultProxy at 0x10e4a1b70>

In [33]: contacts = pd.read_sql_query(contactquery, con=engine)
         contacts

Out[33]: Empty DataFrame
         Columns: [cid, sid, contact, ts]
         Index: []

In [34]: # enough about PostgreSQL again for awhile - on to Twitter...!
```