

Author: Michael O'Donnell

Date: 10/23/19

Goal: Create pseudo-data in python, load into mySQL database as 3 tables, run practice join queries in mySQL

1. **Created pseudo-data in python with the following script:**

https://github.com/odonnell31/python_data_science_basics/blob/master/SQL/generate_puesdo_data.py

In the above script, 3 dataframes are created and exported as CSV files. Small sample snip below:

```
40  ### Second, building puesdo-dataset for table: buildings ###
41
42  # build empty dataframe to match buildings table headers:
43  buildings_columns = ['building_id', 'building_address', 'city',
44                      'floors', 'units', 'built_year']
45  buildings = pd.DataFrame(columns = buildings_columns)
46
47  # streets and cities to create addresses
48  street = ["Pearl", "Peach", "Gordon", "2nd", "Yellowstone",
49           "Winners", "El Camino Real", "Bluestone",
50           "Green", "Astor", "6th"]
51
52  street_suffix = ["St", "Ave", "Ln", "Ct", "Rd",
53                 "Way", "Dr", "Pl", "Drive", "Road",
54                 "Street"]
55
56  city = ["Orlando", "Los Angeles", "Seattle", "San Francisco", "Chicago",
57         "Portland", "Wilmington", "Miami", "Denver", "Boulder", "New York"]
58
59  # fill buildings table with 10 buildings (lots of data generated randomly)
60  for b in range(1,11):
61      new_building = {'building_id': b,
62                    'building_address': str(str(random.randint(1,1000))+ " "+
63                                           street[b]+" "+street_suffix[random.randint(1,10)]),
64                    'city': city[random.randint(0,10)],
65                    'floors': random.randint(3,12),
66                    'units': apartments[apartments.building_id == b].apartment_id.count(),
67                    'built_year': random.randint(1904, 1995)}
68      new_building_df = pd.DataFrame(columns = buildings_columns,
69                                   data = [new_building])
70      buildings = buildings.append(new_building_df)
```

2. **Created 3 tables in mySQL and loaded them with the CSV's created in step (1):**

https://github.com/odonnell31/python_data_science_basics/blob/master/SQL/apartment_create_tables.sql



The screenshot shows a MySQL command-line interface with a toolbar at the top. The toolbar includes icons for file operations (save, open, print), editing (undo, redo, cut, copy, paste), and navigation (home, search, limit to 1000 rows, star, magnifying glass, and a refresh icon). The SQL queries are as follows:

```
7 • CREATE TABLE apartments (apartment_id int, building_id int,
8     vacant_status int, rent int, pet_friendly int,
9     PRIMARY KEY (apartment_id));
10
11 • LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\apartments.csv'
12 INTO TABLE apartments
13 FIELDS TERMINATED BY ','
14 ENCLOSED BY '"'
15 LINES TERMINATED BY '\\n'
16 IGNORE 1 ROWS;
17
18
19
20 • CREATE TABLE buildings (building_id int, building_address VARCHAR(75),
21     city VARCHAR(25), floors int, units int, built_year int,
22     PRIMARY KEY (building_id));
23
24 • LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\buildings.csv'
25 INTO TABLE buildings
26 FIELDS TERMINATED BY ','
27 ENCLOSED BY '"'
28 LINES TERMINATED BY '\\n'
29 IGNORE 1 ROWS;
30
31
32
33 • CREATE TABLE tenants (tenant_id int, tenant_name VARCHAR(50),
34     apartment_id int, renter_income int,
35     lease_start DATE, lease_end DATE,
36     PRIMARY KEY (tenant_id));
37
38 • LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\tenants.csv'
39 INTO TABLE tenants
40 FIELDS TERMINATED BY ','
41 ENCLOSED BY '"'
42 LINES TERMINATED BY '\\n'
43 IGNORE 1 ROWS;
```

3. Ran practice queries on the newly created mySQL database:

https://github.com/odonnell31/python_data_science_basics/blob/master/SQL/query_apartment_db.sql

Limit to 1000 rows

```

1 # find the average rent of an apartment in each city:
2
3 • SELECT buildings.city, apartments.rent
4 FROM apartments
5 JOIN buildings ON apartments.building_id = buildings.building_id
6 GROUP BY buildings.city
7 ORDER BY apartments.rent DESC;

```

Result Grid

	city	rent
▶	Boulder	5282
	Wilmington	4125
	Denver	3349
	Portland	2861
	Orlando	2669
	Seattle	1718
	Chicago	1472
	Los Angeles	1265

Limit to 1000 rows

```

10 # Find the top 10 tenants with the largest income to rent ratio:
11
12 • SELECT tenants.tenant_name, (tenants.renter_income / apartments.rent) AS "Rent_Ratio",
13     apartments.rent, tenants.renter_income AS "income"
14 FROM tenants
15 JOIN apartments on tenants.apartment_id = apartments.apartment_id
16 ORDER BY "Rent_Ratio" DESC
17 LIMIT 10;

```

Result Grid

	tenant_name	"Rent_Ratio"	rent	income
▶	ADAM MOORE	268.2865	768	206044
	CASSY MARTIN	266.9012	769	205247
	CHUCK BROWN	263.2099	791	208199
	JIM HARRIS	257.2632	855	219960
	SAMANTHA TAYLOR	239.8380	753	180598
	JOEL ROBINSON	212.3269	823	174745
	EMILY MARTINEZ	211.8353	935	198066
	MICHAEL HERNANDEZ	201.2781	1086	218588
	JULIA JONES	198.9381	921	183222
	MICHAEL THOMAS	198.1950	1036	205330

```
19 # Which city has the greatest number of vacant apartments?
20
21 • SELECT buildings.city, COUNT(apartments.vacant_status) AS "Vacants"
22 FROM buildings
23 JOIN apartments
24 ON buildings.building_id = apartments.building_id
25 WHERE apartments.vacant_status = 1
26 GROUP BY buildings.city
27 ORDER BY "Vacants" DESC;
```

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	city	"Vacants"
▶	Los Angeles	102
	Boulder	80
	Denver	55
	Portland	53
	Chicago	52
	Wilmington	52
	Seattle	46
	Orlando	44

Limit to 1000 rows

```
29
30 # Which city has the most pet friendly apartments?
31
32 • SELECT buildings.city, COUNT(apartments.apartment_id) AS "Dog friendly apts"
33 FROM buildings
34 JOIN apartments ON buildings.building_id = apartments.building_id
35 WHERE apartments.pet_friendly = 0
36 GROUP BY buildings.city
37 ORDER BY COUNT(apartments.apartment_id) DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [A](#)

	city	Dog friendly apts
▶	Los Angeles	106
	Boulder	92
	Seattle	66
	Denver	58
	Chicago	53
	Wilmington	51
	Portland	48
	Orlando	41

Limit to 1000 rows

```
38
39 # How many apartments in each city have an address that contains "Rd" or "Road"?
40
41 • SELECT buildings.city, COUNT(apartments.apartment_ID) AS "Apts on a 'Road'"
42 FROM buildings
43 JOIN apartments
44 ON buildings.building_id = apartments.building_id
45 WHERE buildings.building_address LIKE "%Rd"
46 OR buildings.building_address LIKE "%Road"
47 GROUP BY buildings.city
48 ORDER BY COUNT(apartments.apartment_ID) DESC;
49
50
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [A](#)

	city	Apts on a 'Road'
▶	Portland	99