

Lab 3: Structured Query Language (SQL)

We know how to use command line, and GUI console to manipulate the database, load the data, and back-up and restore the data (Lab 1 and Lab 2). Now it is time to do some serious businesses with our data in a database. This lab is designed and developed based on the data and examples from chapter 3 in Prof. Shekhar's book.

Objective

The goal for you to take away from this lab is simple and forward:

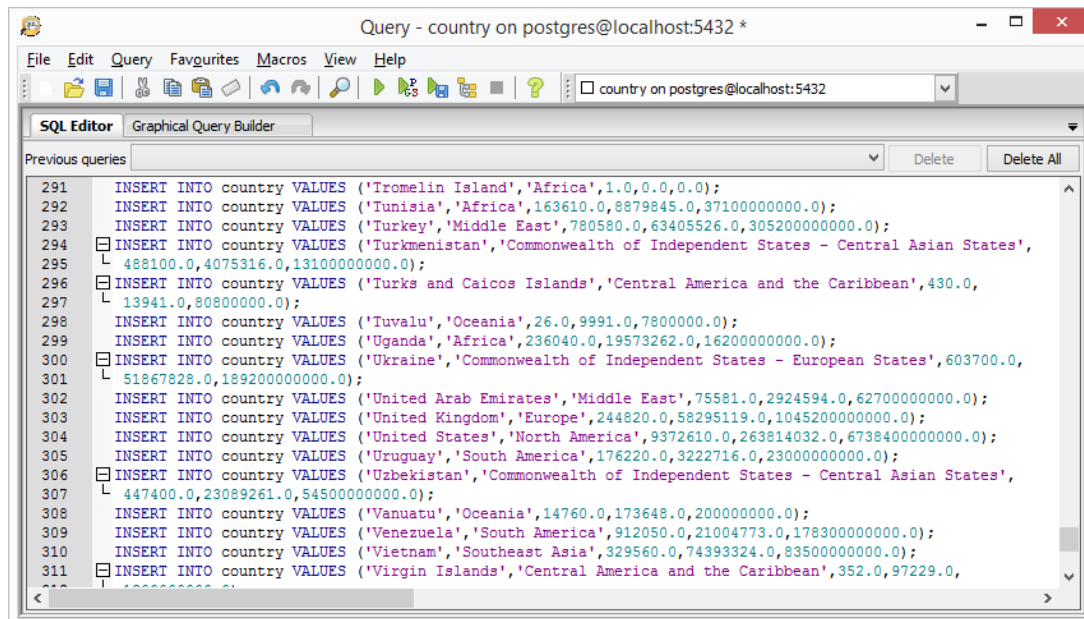
- Conquer and master SQL

Lab Data and Scripts

Please download the data and scripts (Lab4.zip) in the Lab section at Learn@UW system. If you are using Windows, put it under “C:\lab4” (if you do not have a “C:\lab4” folder, please create one). Unzip the data, and review them.

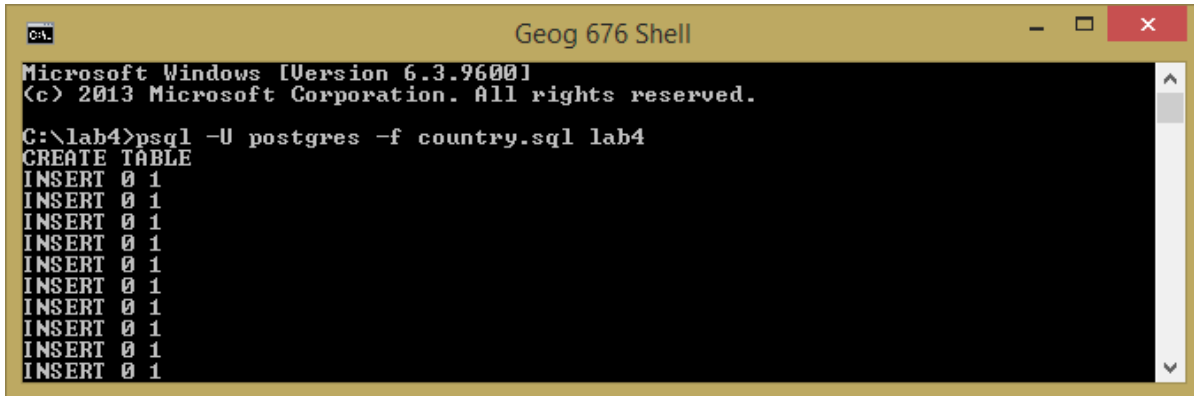
1. Database Preparation

Create a database with any name you preferred (e.g., lab4). To create a table named “country” in your database and load data (country records) into this table, run SQL statements stored in file “C:/lab4/country.sql”. To run this sql file, you can **either** directly open “country.sql”, then copy, paste, and execute the SQL statements in *pgAdmin III* SQL console (Lab 1) as below:



```
Query - country on postgres@localhost:5432 *
File Edit Query Favgrites Macros View Help
country on postgres@localhost:5432
SQL Editor Graphical Query Builder
Previous queries Delete Delete All
291 INSERT INTO country VALUES ('Tromelin Island','Africa',1.0,0.0,0.0);
292 INSERT INTO country VALUES ('Tunisia','Africa',163610.0,8879845.0,37100000000.0);
293 INSERT INTO country VALUES ('Turkey','Middle East',780580.0,63405526.0,305200000000.0);
294 INSERT INTO country VALUES ('Turkmenistan','Commonwealth of Independent States - Central Asian States',
295 488100.0,4075316.0,13100000000.0);
296 INSERT INTO country VALUES ('Turks and Caicos Islands','Central America and the Caribbean',430.0,
297 13941.0,80800000.0);
298 INSERT INTO country VALUES ('Tuvalu','Oceania',26.0,9991.0,7800000.0);
299 INSERT INTO country VALUES ('Uganda','Africa',236040.0,19573262.0,16200000000.0);
300 INSERT INTO country VALUES ('Ukraine','Commonwealth of Independent States - European States',603700.0,
301 51867828.0,189200000000.0);
302 INSERT INTO country VALUES ('United Arab Emirates','Middle East',75581.0,2924594.0,62700000000.0);
303 INSERT INTO country VALUES ('United Kingdom','Europe',244820.0,58295119.0,1045200000000.0);
304 INSERT INTO country VALUES ('United States','North America',9372610.0,263814032.0,6738400000000.0);
305 INSERT INTO country VALUES ('Uruguay','South America',176220.0,3222716.0,23000000000.0);
306 INSERT INTO country VALUES ('Uzbekistan','Commonwealth of Independent States - Central Asian States',
307 447400.0,23089261.0,54500000000.0);
308 INSERT INTO country VALUES ('Vanuatu','Oceania',14760.0,173648.0,200000000.0);
309 INSERT INTO country VALUES ('Venezuela','South America',912050.0,21004773.0,178300000000.0);
310 INSERT INTO country VALUES ('Vietnam','Southeast Asia',329560.0,74393324.0,83500000000.0);
311 INSERT INTO country VALUES ('Virgin Islands','Central America and the Caribbean',352.0,97229.0,
```

Or you can run “*psql -U postgres -f country.sql lab4*” command in command line console (Lab 2):



```
C:\lab4>psql -U postgres -f country.sql lab4
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

2. SQL Queries

Now write SQL queries for the queries listed in following subsections. *Please run and test your queries in the pgAdmin III console or psql shell command to make sure there is no syntax error.* Please submit all the written SQL statements in a *single .txt* file to Dropbox at Learn@UW.

2.1: Simple Selection (11 pts)

- 1). List countries (including all columns) from rows where name contains 'Ocean'. (1 pts)
- 2). List countries (including all columns) from rows where area is zero. (1 pts)
- 3). List names of all regions (with no duplicates). (3 pts)

Hint: keyword DISTINCT.

- 4). List names and population of populous (population over 10 million) countries. (3 pts)
- 5). List name, population, region of populous (population over 10 million) European and Asian countries. (3 pts)

Hint: you can use either logical operator “Or”, or keyword “IN”, to make sure your result set include both European and Asian.

2.2: Summarization and Aggregation (15 pts)

- 1). List name, area for countries with area between 0.5 million sq. km. and 1 million sq. km. (the unit for “area” column is sq. km.) How many countries are there? (2 pts)

Hint: use either logical operator AND or keyword BETWEEN...AND.

- 2). List number of countries, total area, and average area per country for all regions of the world. (3 pts)
- 3). Select regions which have at least 5 countries with population over 10 million. (3 pts)
- 4). List population, GDP, population density for countries of the world in order of GDP. (3 pts)

Hint: population density is equal to population/area (area not equal to 0).

- 5). Get the average population, average GDP, and average population density for regions with average population over 10 million of the world in descending order of average GDP. (4 pts)

Hint: use all 6 clauses to create this query statement.

2.3: Nested Queries (12 pts)

- 1). Find names of countries with GDP more than the total GDP of Africa. (4 pts)
- 2). Find the region with largest GDP. Do not use MAX. You will need SUM to compute region totals. (4 pts)

Hint: keyword ALL

Refer: [http://technet.microsoft.com/en-us/library/ms187074\(v=sql.105\).aspx](http://technet.microsoft.com/en-us/library/ms187074(v=sql.105).aspx)

- 3). Find the countries in Asia with population larger than the largest population of North American countries. (4 pts)

2.4: Insert, Update and Delete (12 pts)

- 1). Examine the rows with non-zero area but zero gdp and population. Delete these rows assuming these do not constitute countries. (1 pts)
- 2). Determine the region for Mauritius. Update region for Mauritius to be Africa. (1 pts)
- 3). Update GDP of United States to reflect 5 percent increase per year since 1995 (The data in the table was collected for countries in 1995). (3 pts)

*Hint: you can use *power* mathematics function, it is can be used as $POWER(a, b)$, which means a is raised to the power of b . For example: $POWER(3, 2) = 9$. For more mathematic function that are supported in SQL, please refer to:*

<http://www.postgresql.org/docs/9.3/static/functions-math.html>.

- 4). Hong Kong merged with China in 1997. Add the population, area, gdp of Hong Kong to China. Delete Hong Kong. (3 pts)

Hint: To update multiple columns with value from other tables, you can use do something like this:

```
UPDATE table1
SET col1 = othertable.col1,
    col2 = othertable.col2
FROM othertable
[WHERE table1.col3 = XXX];
```

- 5). Insert a new country, Korea in Asia region. Its area, population and gdp are sum of each in two countries, 'Korea, North' and 'Korea, South'. Delete 'Korea, North' and 'Korea, South' countries. (4 pts)

Hint: to insert this new row to the country table, you may write and run two SQL statements:

- 1) First insert only name and region first.
- 2) Then update the gdp, population and area of this new row with the values that you calculate the sum of the gdp, population, and area of Korean North and South.