Lab 1: *PostgreSQL* Tutorial I: GUI (*pgAdmin III*)

This is a basic introduction into pgAdmin III, the comprehensive database design and management console for Postgres databases. You'll also be introduced to the fundamental features in pgAdmin and be ready to start creating databases, entering data, and building multi-table queries using the Graphical Query tool all without writing any SQL.

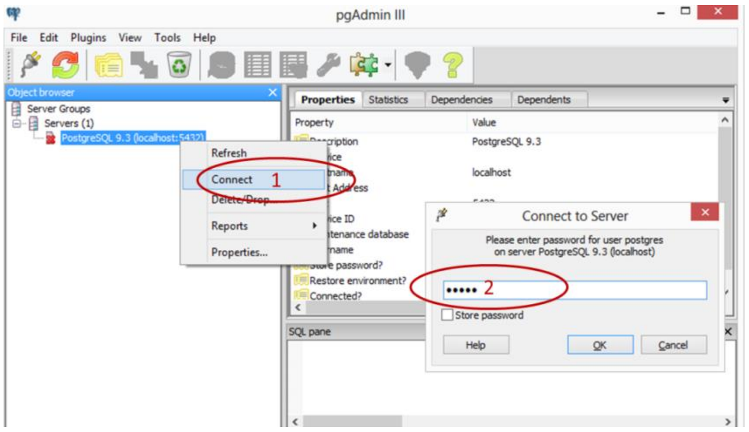
Please first go through Part I: Exercise Tutorial, then finish tasks listed in Part II: Lab Assignment.

Part I: Exercise Tutorial

1. Getting started with *pgAdmin III*

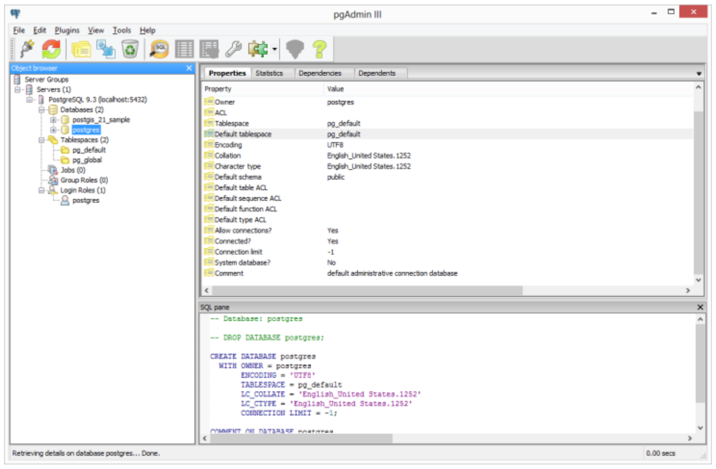
Once pgAdmin III has been installed, a server will be needed to be connected. In our case, we will use "localhost" as the server.

➔ Right click "**PostgresSQL 9.3**" at the left "Object browser" panel to connect local host server, and select "**Connect**", and enter "*postgres*" as the password.



Please enter password for user postgres
on server PostgreSQL 9.3 (localhost)

The server should now be shown in the left hand box and you can navigate your way round your database tables. For the first time using the PostgreSQL, only two database available, including a spatial database (postgis_21_sample) created while installing the PostGIS, and a default database named postgres on each PostgreSQL server installation.



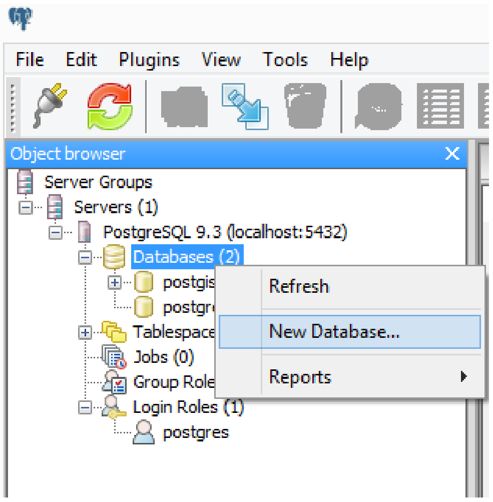
Property	Value
Owner	postgres
ACL	
Tablespace	pg_default
Default tablespace	pg_default
Encoding	UTF8
Collation	English_United States.1252
Character type	English_United States.1252
Default schema	public
Default table ACL	
Default sequence ACL	
Default function ACL	
Default type ACL	
Allow connections?	Yes
Connected?	Yes
Connection limit	-1
System database?	No
Comment	default administrative connection database

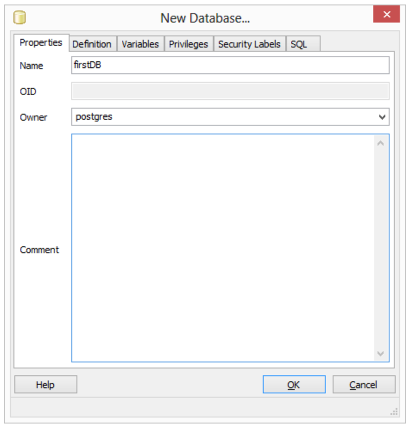
```
-- Database: postgres

-- DROP DATABASE postgres;

CREATE DATABASE postgres
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'English_United States.1252'
       LC_CTYPE = 'English_United States.1252'
       CONNECTION LIMIT = -1;

COMMENT ON DATABASE postgres
```

Retrieving details on database postgres... Done.

2. Create your first DB

➔ Right click **Servers(1)->PostgreSQL 9.3 (localhost: 5432) ->Databases(2)**

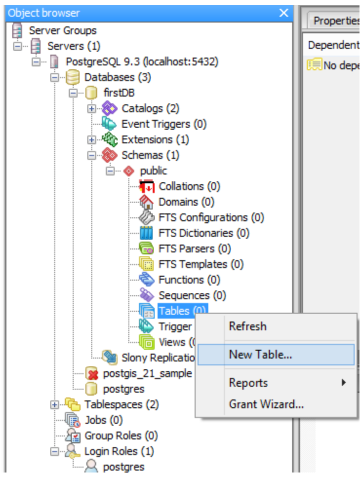




Name: firstDB
OID:
Owner: postgres

➔ Enter "*firstDB*" as the DB name, choose "postgres" as owner, and click "OK" as the right figure.

➔ The "firstDB" will show up.

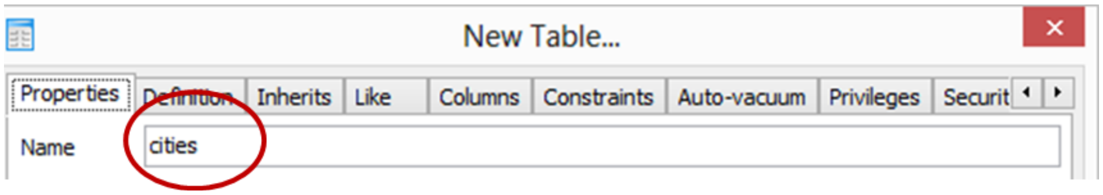


3. Add tables to the DB through the GUI

In this practice, we will add a table to firstDB to record weather data in various cities across the country. To do this, we need:

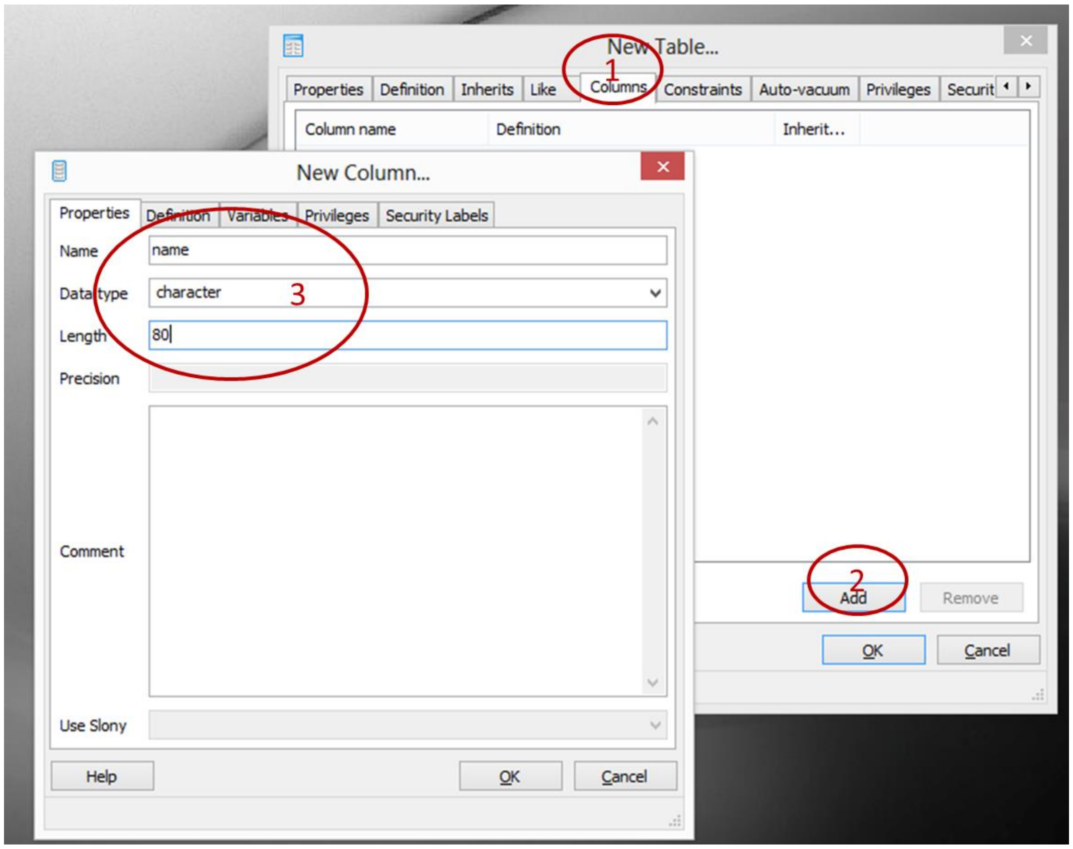
➔ Right click **firstDB->Schemas->public->Tables**, to bring up create "**New Table**" dialogue

➔ Set the table name as "*cities*"



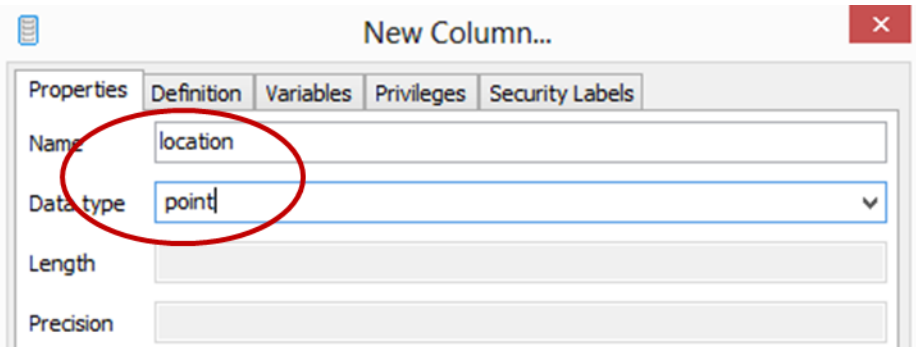
Name: cities

➔ Add fields to the table: Click "**Columns**" tab, click "**Add**" button, and enter "*name*" in the Name text box, "Data type" as character and "Length" as 80



Name: name
Data type: character
Length: 80
Precision:
Comment:
Use Slony:

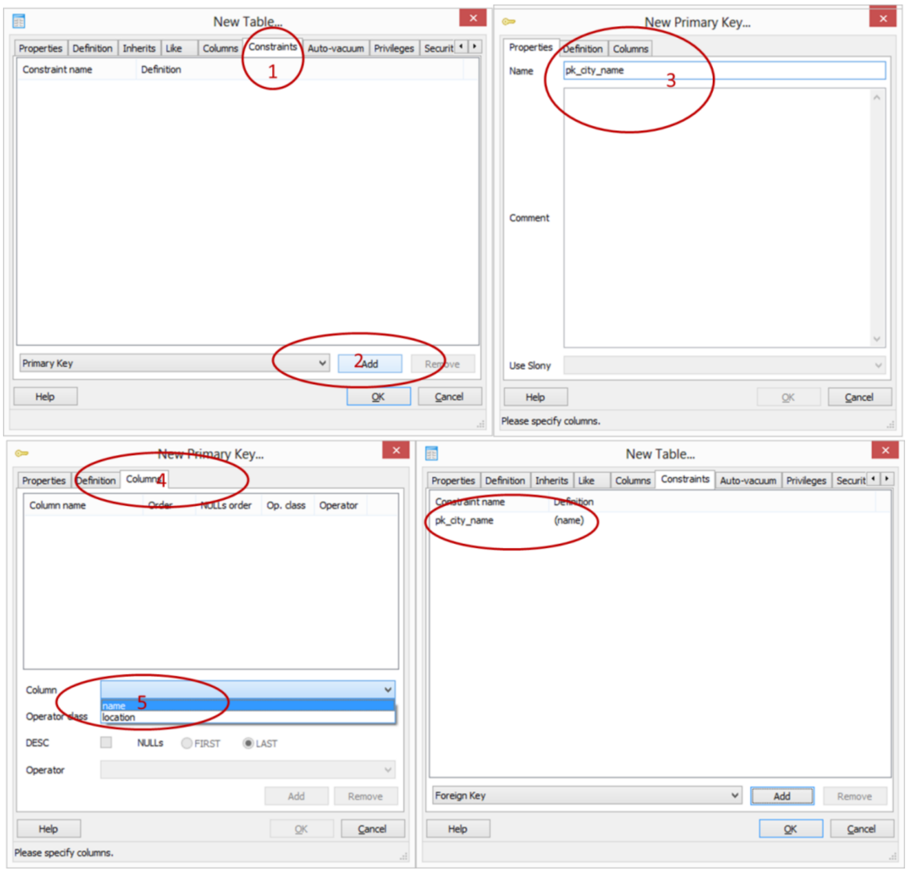
→ Add one more field to the table: click "**Add**" button again, and enter "*location*" in the Name text box, and "Data type" as "Point".



Name: location
Data type: point
Length:
Precision:

→ Finally, let's add one simple constraint for the table. This would be our primary key, which would prevent duplicated cities from being add to the table. This can be done by:

→ Click "**Constraints**" tab -> "**Add**" button. A new dialogue for adding a new primary key will show up. Enter the "*pk_city_name*" as the key name, click "**Columns**", and connect the key to a city name field of the table we created by selecting "*name*". Then we will see that our primary key is created and added in the panel.



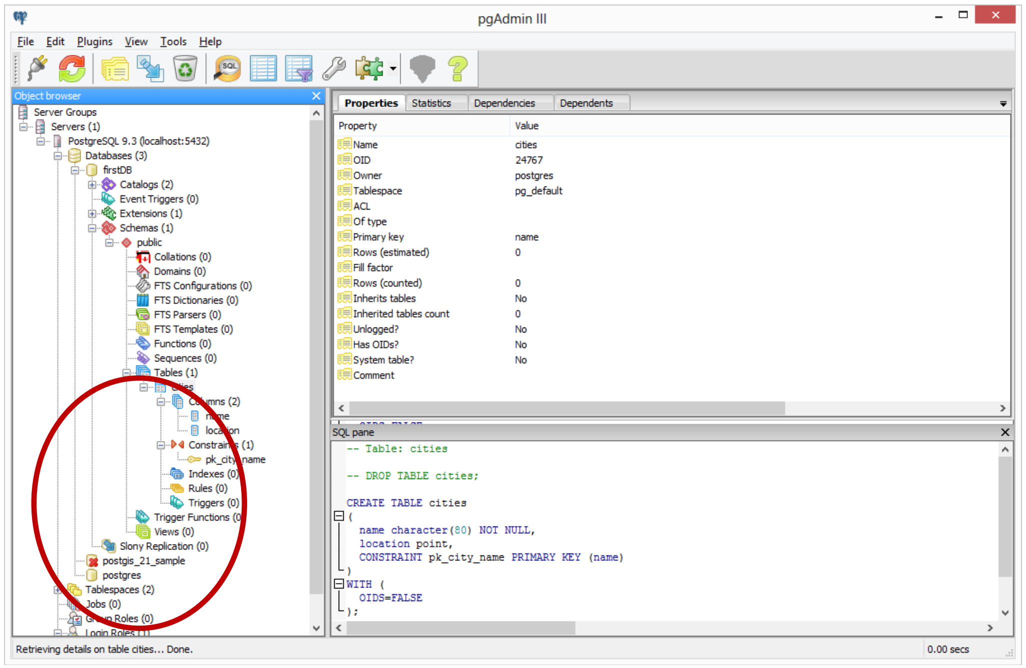
Constraint name	Definition
pk_city_name	(name)

Name: pk_city_name

Column name	Order	NULLs order	Op. class	Operator

Column: name, location

Now we have specified the table name, column names, their types, and a simple constraint (primary key) to prevent from duplicated records with the same city being added to the table. To the result of our work, you can go to the object browser and expand the table "cities".



Property	Value
Name	cities
OID	24767
Owner	postgres
Tablespace	pg_default
ACL	
Of type	
Primary key	name
Rows (estimated)	0
Fill factor	
Rows (counted)	0
Inherits tables	No
Inherited tables count	0
Unlogged?	No
Has OIDs?	No
System table?	No
Comment	

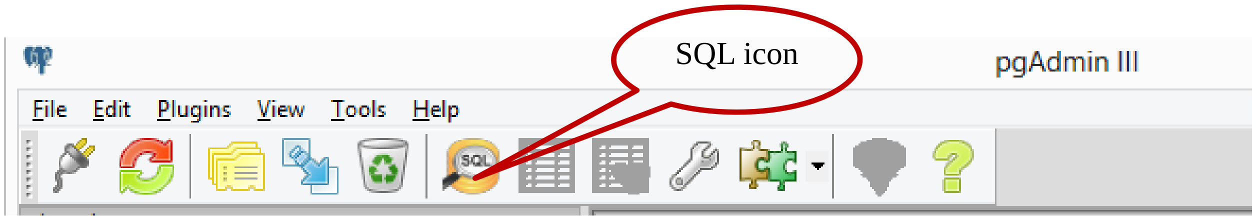
```
-- Table: cities

-- DROP TABLE cities;

CREATE TABLE cities
(
  name character(80) NOT NULL,
  location point,
  CONSTRAINT pk_city_name PRIMARY KEY (name)
)
WITH (
  OIDS=FALSE
);
```

Retrieving details on table cities... Done.   0.00 secs

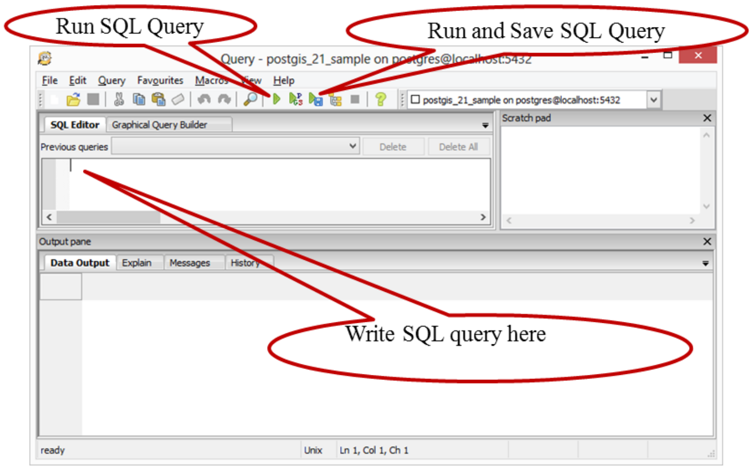
4. Add tables to the DB through the SQL Query Tool

Besides to add the table step by step through the graphic use interface (GUI), we can also create tables for the database through the script. To query and operate the data within *PostgreSQL*, an SQL query tool is necessary and provided in *pgAdmin III*. Let's see a shortcut to create a table with SQL query tool.

➔ Clicking the **SQL icon** will produce another window in which the SQL query should be written.



Following figure is a screenshot of the SQL window where the queries are written.



Following are **SQL command** that we can use to create a weather table:

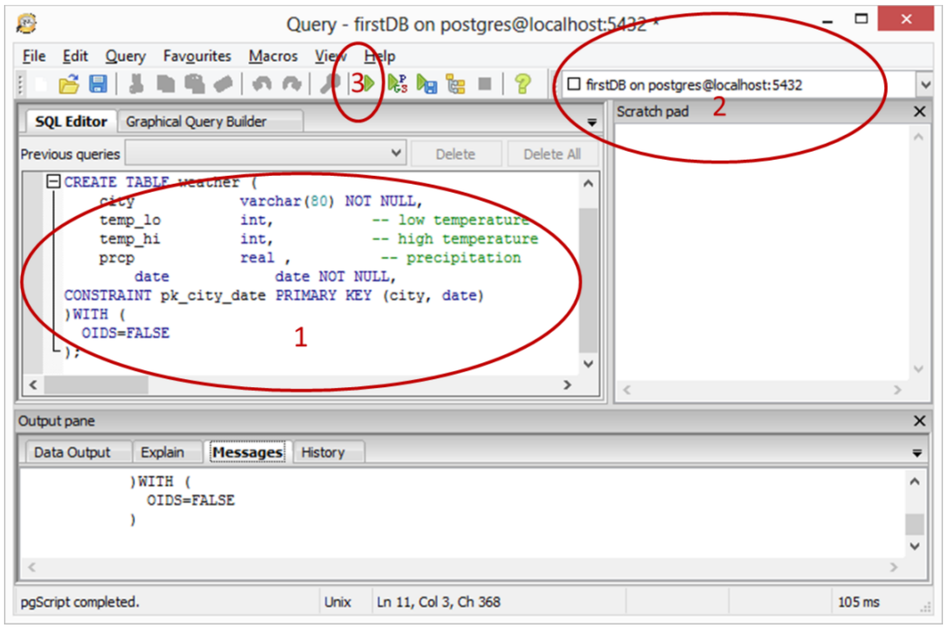
```
DROP TABLE IF EXISTS weather;
CREATE TABLE weather (
        city         varchar(80) NOT NULL,
        temp_lo      int,         -- low temperature
        temp_hi      int,         -- high temperature
        prcp         real ,       -- precipitation
        date         date         NOT NULL,
       CONSTRAINT pk_city_date PRIMARY KEY (city, date)
)WITH (
  OIDS=FALSE
);
```

White space (i.e., spaces, tabs, and newlines) can be used freely in SQL commands. That means you can type the command aligned differently than above, or even all on one line. Two dashes ("--") introduce comments. Whatever follows them is ignored up to the end of the line. SQL is case insensitive about key words and identifiers, except when identifiers are double-quoted to preserve the case (not done above).

"varchar(80)" specifies a data type that can store arbitrary character strings up to 80 characters in length. "int" is the normal integer type. "real" is a type for storing single precision floating-point numbers. "date" should be selfexplanatory. (Yes, the column of type date is also named date. This might be convenient or confusing — you choose.)

PostgreSQL supports the standard SQL types int, small int, real, double precision, char(N), varchar(N), date, time, timestamp, and interval, as well as other types of general utility and a rich set of geometric types. PostgreSQL can be customized with an arbitrary number of user-defined data types. Consequently, type names are not syntactical key words, except where required to support special cases in the SQL standard.

➔ Now let's copy the SQL to the *"SQL editor"* window. Before you execute the command by click "***Execute pgScript***" button, you need to **make sure that you are connecting to the correct database (DB) server and DB as well.** In our case, our DB name is "firstDB".

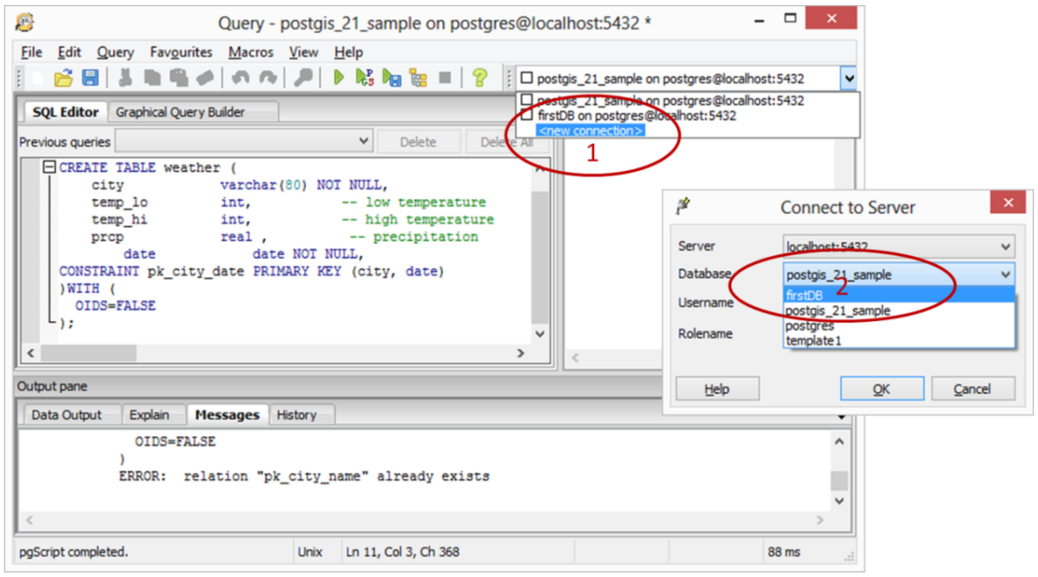


```
CREATE TABLE weather (
    city            varchar(80) NOT NULL,
    temp_lo         int,            -- low temperature
    temp_hi         int,            -- high temperature
    prcp            real ,           -- precipitation
        date            date NOT NULL,
CONSTRAINT pk_city_date PRIMARY KEY (city, date)
)WITH (
  OIDS=FALSE
);
```

```
            )WITH (
              OIDS=FALSE
            )
```

pgScript completed. | Unix | Ln 11, Col 3, Ch 368 | 105 ms

If the connection is not to the "firstDB", you can scroll down the DB connection drop box as below and select "**new connection**". A DB connection dialogue will show up and you can select the appropriate one.

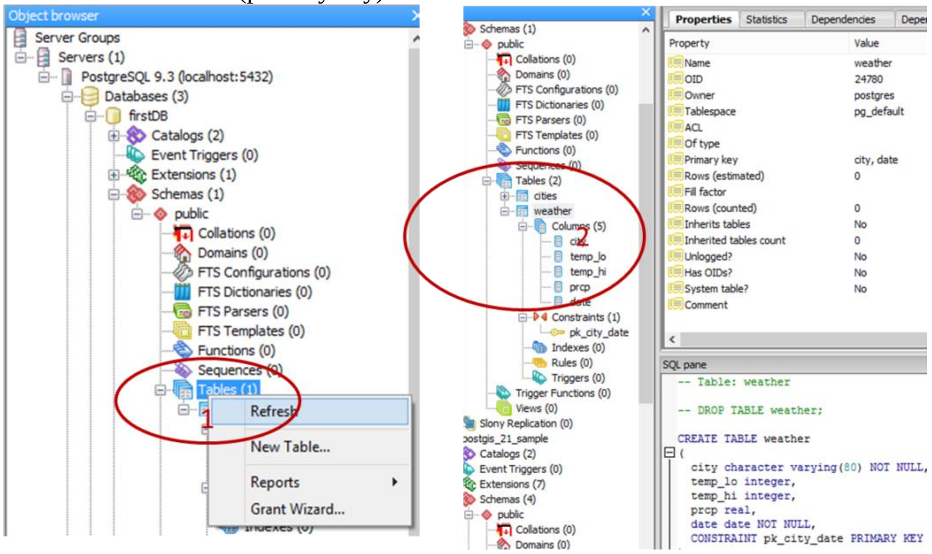


```
CREATE TABLE weather (
    city            varchar(80) NOT NULL,
    temp_lo         int,            -- low temperature
    temp_hi         int,            -- high temperature
    prcp            real ,           -- precipitation
        date            date NOT NULL,
CONSTRAINT pk_city_date PRIMARY KEY (city, date)
)WITH (
  OIDS=FALSE
);
```

```
            OIDS=FALSE
          )
          ERROR:  relation "pk_city_name" already exists
```

pgScript completed. | Unix | Ln 11, Col 3, Ch 368 | 88 ms

- ➔ Now let's execute the command by click "***Execute pgScript***" button.
- ➔ Now back to Object browser and **right click "Tables", select Refresh**, you will notice our new weather table with 5 columns and one constraint (primary key) created.



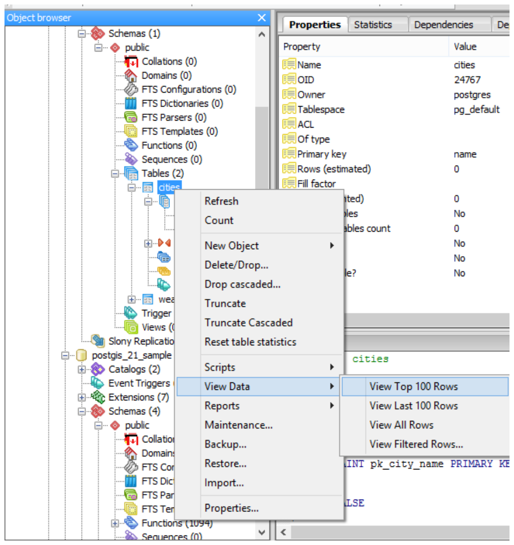
Property	Value
Name	weather
OID	24780
Owner	postgres
Tablespace	pg_default
ACL	
Of type	
Primary key	city, date
Rows (estimated)	0
Fill factor	
Rows (counted)	0
Inherits tables	No
Inherited tables count	0
Unlogged?	No
Has OIDs?	No
System table?	No
Comment	

```
-- Table: weather

-- DROP TABLE weather;

CREATE TABLE weather
(
  city character varying(80) NOT NULL,
  temp_lo integer,
  temp_hi integer,
  prcp real,
  date date NOT NULL,
  CONSTRAINT pk_city_date PRIMARY KEY
```

5. Add data to the table

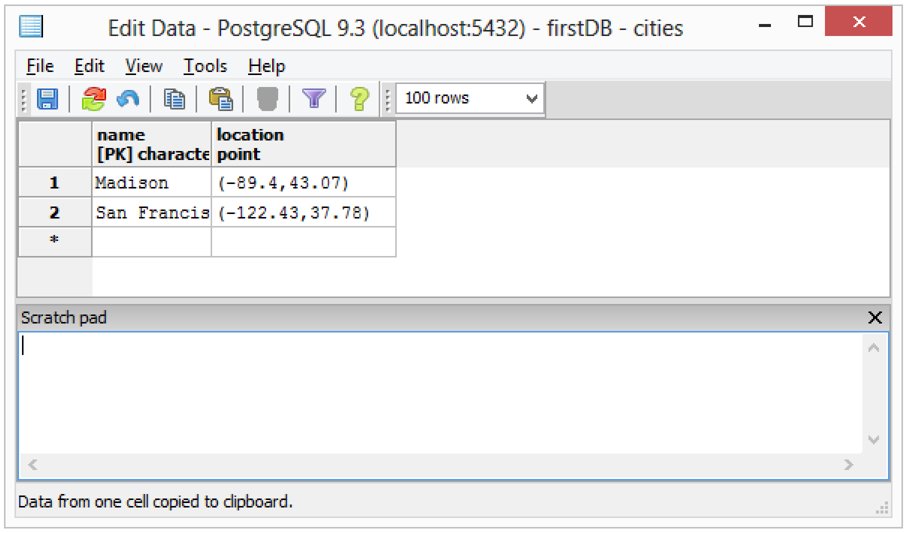
Now we have successfully created two tables, it is time to add some data.

- ➔ Manually add data entries in a tabular form. For example, we can enter data entries for cities table, by **right click "cities" -> "view data" -> "View Top 100 rows"**.



Property	Value
Name	cities
OID	24767
Owner	postgres
Tablespace	pg_default
ACL	
Of type	
Primary key	name
Rows (estimated)	0
Fill factor	

- ➔ Enter the "Madison, (89.40, 43.07)" as name and location for one row, and add "San Francisco, (-122.43,37.78)" as another row as below (you might need to **click the Refresh button** **after enter each row**).


Edit Data - PostgreSQL 9.3 (localhost:5432) - firstDB - cities
File Edit View Tools Help
100 rows
Scratch pad

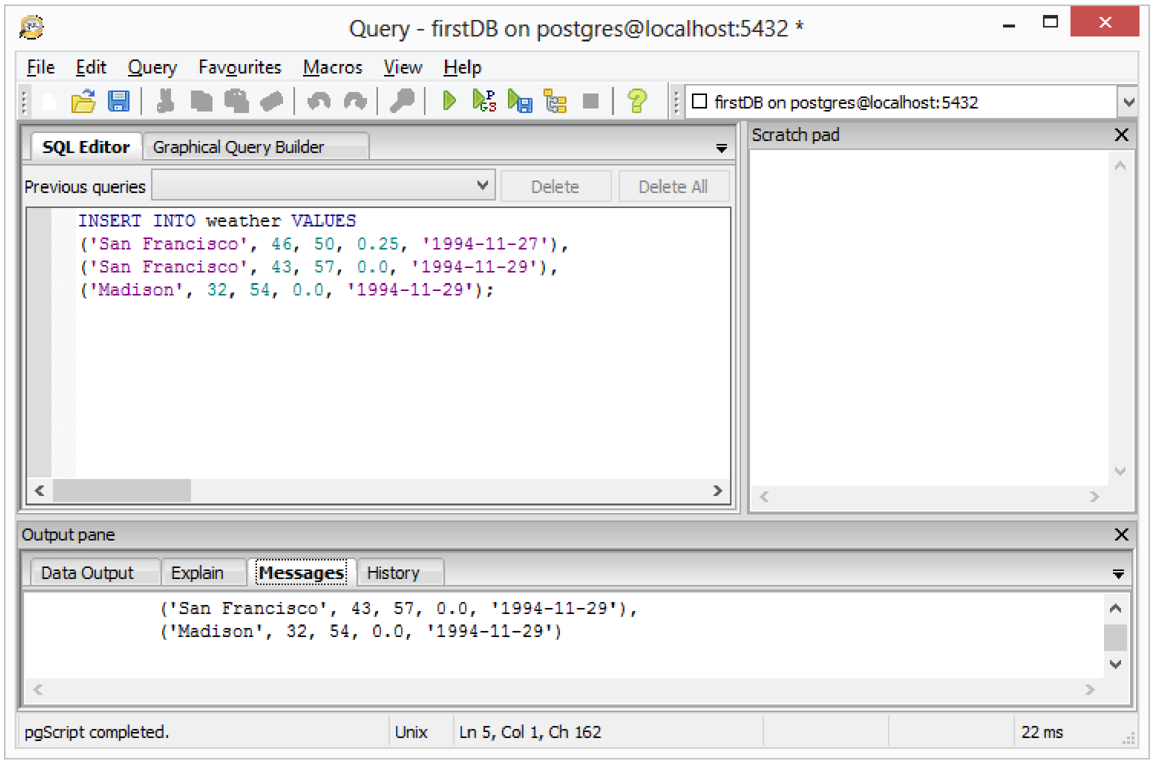
	name [PK] characte	location point
1	Madison	(-89.4,43.07)
2	San Francis	(-122.43,37.78)
*		

Data from one cell copied to clipboard.

However, this is not efficient way to enter the data. It is most useful for adding simple test data. On the other hand, we can use SQL query tool again, to enter data in batch. Following is a SQL command to insert three records to the weather table.

```
INSERT INTO weather VALUES
('San Francisco', 46, 50, 0.25, '1994-11-27'),
('San Francisco', 43, 57, 0.0, '1994-11-29'),
('Madison', 32, 54, 0.0, '1994-11-29');
```

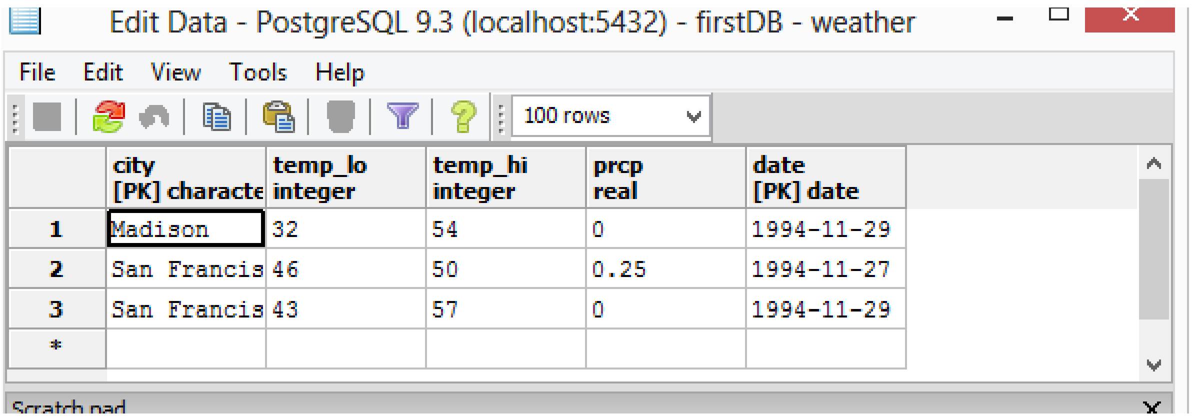
➔ Now let's erase SQL commands in the SQL Editor panel by clicking the 'clear edit window' tool bar.
➔ Copy the SQL to the *"SQL editor"* window, and click "Execute pgScript" button to run the command


Query - firstDB on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
firstDB on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History

```
INSERT INTO weather VALUES
('San Francisco', 46, 50, 0.25, '1994-11-27'),
('San Francisco', 43, 57, 0.0, '1994-11-29'),
('Madison', 32, 54, 0.0, '1994-11-29');
```

```
        ('San Francisco', 43, 57, 0.0, '1994-11-29'),
        ('Madison', 32, 54, 0.0, '1994-11-29')
```

pgScript completed. Unix Ln 5, Col 1, Ch 162 22 ms

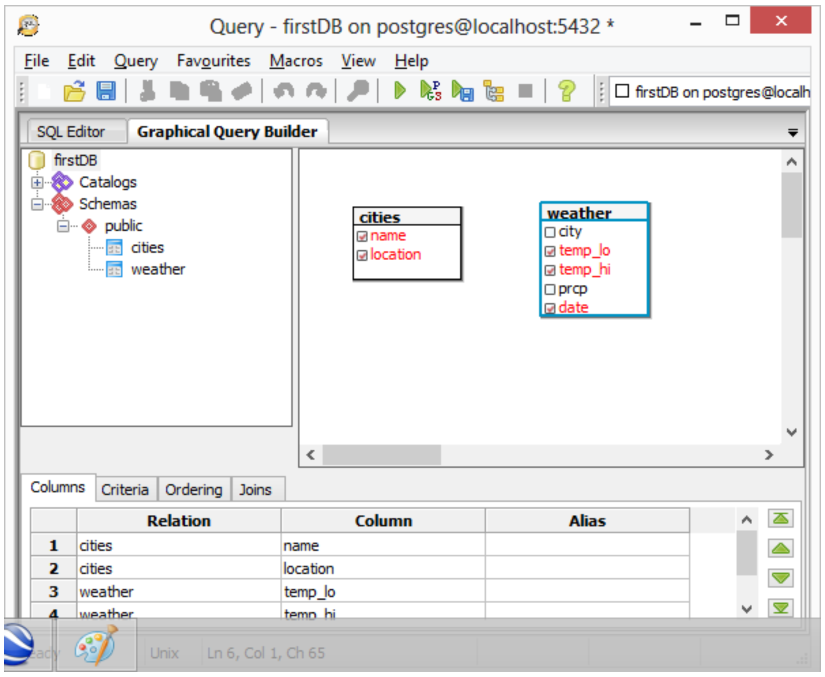

Edit Data - PostgreSQL 9.3 (localhost:5432) - firstDB - weather
File Edit View Tools Help
100 rows
Scratch pad

	city [PK] characte	temp_lo integer	temp_hi integer	prcp real	date [PK] date
1	Madison	32	54	0	1994-11-29
2	San Francis	46	50	0.25	1994-11-27
3	San Francis	43	57	0	1994-11-29
*					

6. Query the data through Graphical Query Builder

Now we have the data in multiple tables, we can use query tool to view all our whether data. In today's exercise, we will learn to use **Graphical Query Builder** rather than SQL Editor to build the SQL commands to view the data. The Graphical Query Builder (GQB) is part of the *Query Tool* which allows you to build simple SQL queries visually[1].

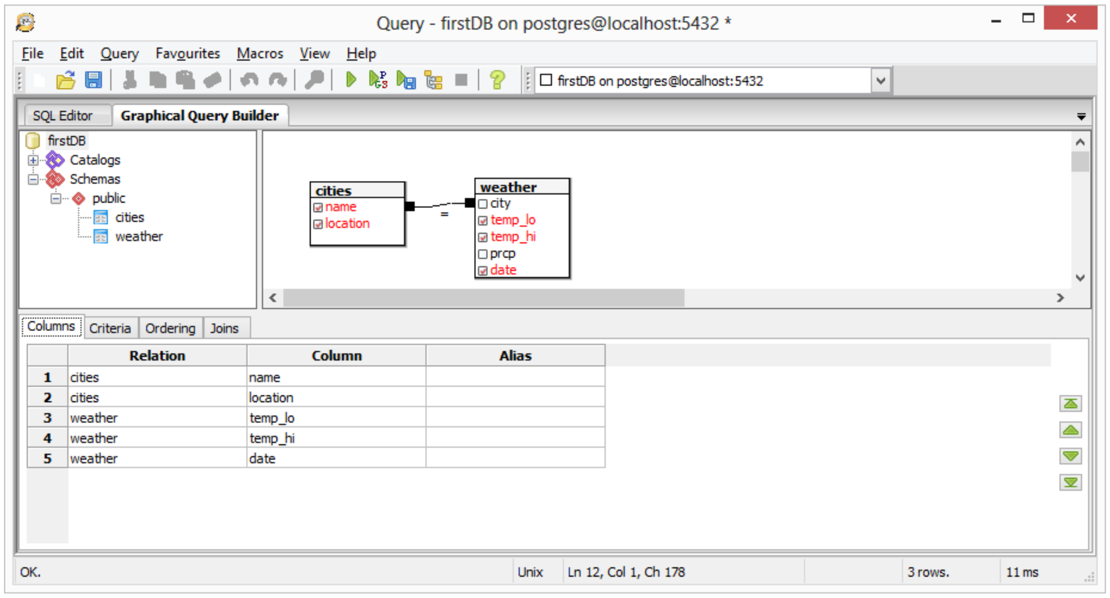
- ➔ Click "**SQL query tool**" to open the query window, and erase SQL commands in the SQL Editor panel by clicking the "clean edit window"tool bar so we can have a clean window. Then click "**Graphical Query Builder**"
- ➔ **Open "Schemas"-> "public".** Drag the cities *onto* the graphic query builder canvas. You will see a pane of "*cities"* showing up in the canvas. Drag the "*weather*" table onto the canvas as well. Now you can select the information you want to view from both tables. Under the cities pane, you can check "name", "location", and "check temp_lo", "temp_hi", and "date" from the weather table.



	Relation	Column	Alias
1	cities	name	
2	cities	location	
3	weather	temp_lo	
4	weather	temp_hi	

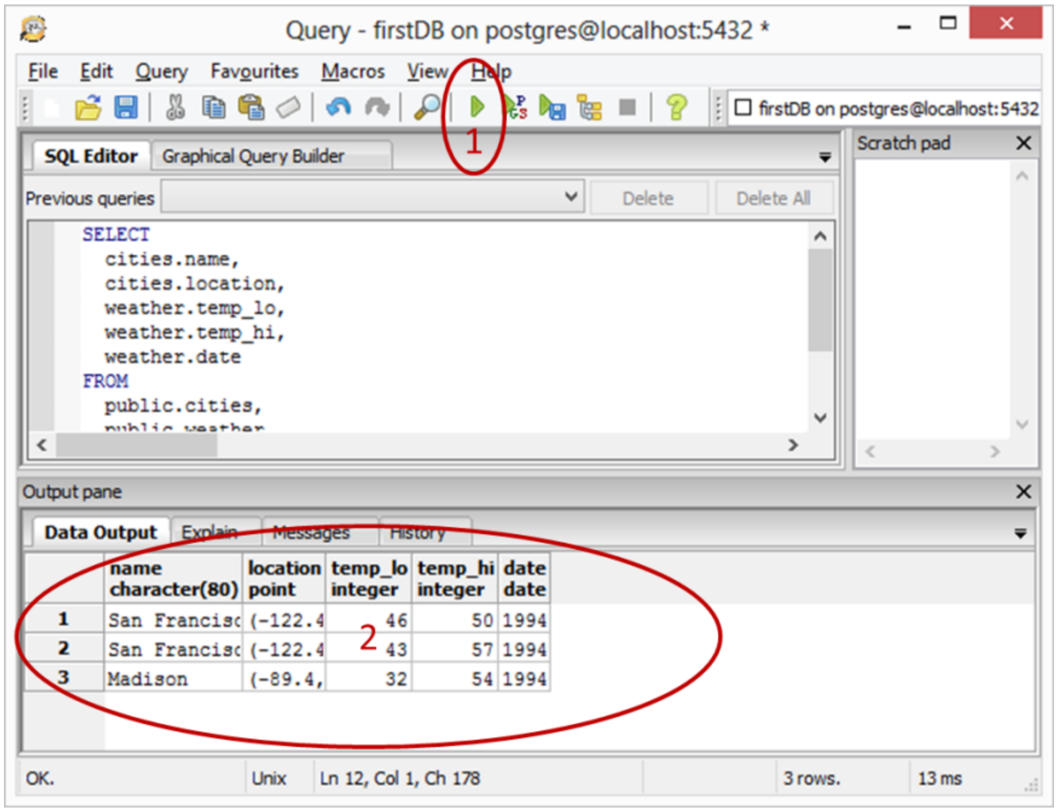
1 http://www.pgadmin.org/docs/dev/gqb.html

➔ Now you can do a join between the two tables so we can get the data from both tables by linking the name in the cities table to the city in the weather table. To create joins between relations, drag a column ("name" column in the cities table) from one relation onto another ("city" column in the weather table).



	Relation	Column	Alias
1	cities	name	
2	cities	location	
3	weather	temp_lo	
4	weather	temp_hi	
5	weather	date	

➔ Now run the query, and we can see down in the output pane, including San Francisco and Madison location and weather records joining from both table.



```
SELECT
  cities.name,
  cities.location,
  weather.temp_lo,
  weather.temp_hi,
  weather.date
FROM
  public.cities,
  public.weather
```

	name character(80)	location point	temp_lo integer	temp_hi integer	date date
1	San Francisc	(-122.4	46	50	1994
2	San Francisc	(-122.4	43	57	1994
3	Madison	(-89.4,	32	54	1994

Part II: Lab Assignment

Task 1*:* Insert at least two more records into "cities" and "weather" tables. You can make up any city and weather data. (25 pts)

Task 2*:* Use Graphical Query Builder to perform a query to list all the cities each with name, location, temp_lo, temp_hi, prcp, and date. (25 pts)

Directions: explain how you complete the tasks in texts, along with screenshots, and/or SQL code, to indicate your steps and results if necessary. Please turn in your lab report as a single *.pdf* file. Submit this file to the Dropbox at Learn@UW by the beginning of your lab period in one week.

Reference:

- Graphical Query Builder (GQB): http://www.pgadmin.org/docs/dev/gqb.html
- How To Create A Postgres Database Using pgAdmin : https://www.youtube.com/watch?v=1wvDVBjNDys