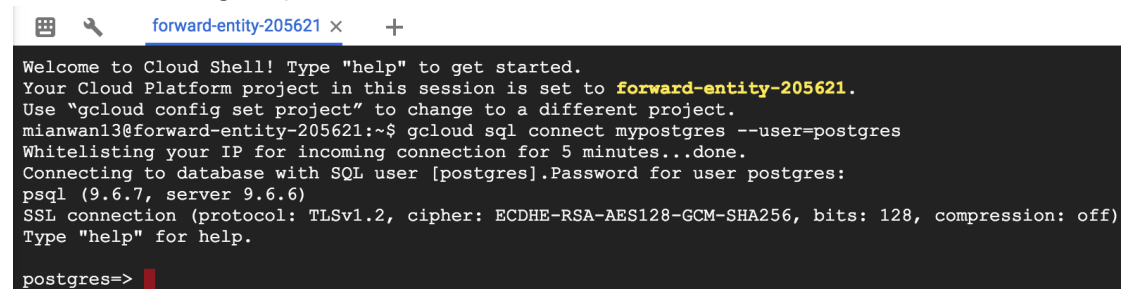


Homework 3 Spatial Database (7pts)

In this homework, you are going to work with spatial data - you will create some data, visualize it, do queries on it, and visualize the query results. The exercise will give you a taste of working with spatial data, use of a spatial file format and spatial query functions, all of which are quite useful from a real-world (or job interview) perspective. **Note that you need to log in to GCP using the credit you got in last homework.**

Preparation

1. Follow the guide of the link <https://cloud.google.com/sql/docs/postgres/quickstart> to create a PostgreSQL instance, and connect to it. You should see the shell as follows:



```
forward-entity-205621 x +
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to forward-entity-205621.
Use "gcloud config set project" to change to a different project.
mianwan13@forward-entity-205621:~$ gcloud sql connect mypostgres --user=postgres
Whitelisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [postgres]. Password for user postgres:
psql (9.6.7, server 9.6.6)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES128-GCM-SHA256, bits: 128, compression: off)
Type "help" for help.

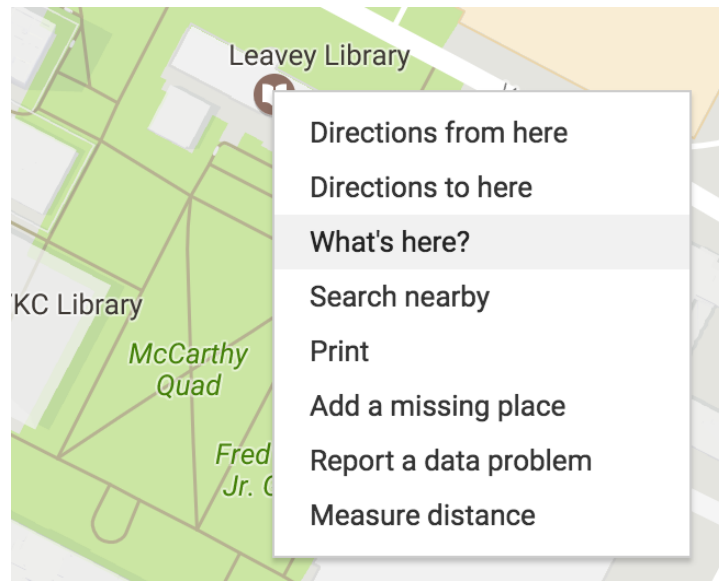
postgres=>
```

2. [Download Google Earth](#) on your laptop, install it.

Assignment Tasks (6pts)

1. You need to create (generate) [latitude,longitude] spatial coordinates for 11 locations.
 - a. One of those needs to be where your home/apartment/dorm room is.
 - b. The other 10 would have to be spread out - spatially distinct, at least 100 feet between adjacent locations (and at most 'several hundred feet' - we don't want to cover a huge region overall!). You can do either way of the following:
 - If you are on campus, you can obtain the coords of its four corners (Exposition/Vermont, Vermont/Jefferson, Jefferson/Figueroa, Figueroa/Exposition), and get coordinates for 6 spots inside the campus (classrooms, labs, offices, restaurants, landmarks..).
 - If you are a DEN student, get your coordinates from your place of work or neighborhood (again, make sure they are not too close to each other or too far apart).
 - c. How would you obtain spatial coordinates at a location?

You can get the coordinate by using Google Maps easily. Select any point and right click, then choose "What's here?". You will be able to see the coordinate.



2. Now that you have **11** coordinates and their label strings (ie. text descriptions such as "Tommy Trojan", "SAL", "Chipotle".), you are going to create a KML file (.kml format) out of them using a text editor.

KML is a map-oriented file format, with XML tags. Specifically, each location you surveyed will be a 'placemark' in your .kml file (specified using coords and labels). Here is more detail. The .kml file with the 11 placemarks is going to be your starter file, for doing visualizations and queries. We provide a .kml skeleton to get you started (just edit it to put in your coords and labels).

NOTE - keep each label to be 15 characters or less (including spaces) - otherwise they might not be displayed properly. **KML file doesn't allow spaces between coordinate separator (ie. comma)!**

3. Load your kml file into Google Earth - that should show you your 11 sampled locations, on Google Earth's globe :) Take a snapshot (screenshot) of this, for submitting.
4. You need to use the above software to execute the following two spatial queries that you will need to write:
 - a. 1) compute the convex hull for your 11 points [a [convex hull](#) for a set of 2D points is the smallest convex polygon that contains the point set]. If you use Postgres, read [this](#). 2) Use the query's result polygon's coords, to create a polygon in your .kml file (edit the .kml file, add relevant XML to specify the KML polygon's coords). Load this into Google Earth, **visually verify that your 11 points are inside the convex hull, then take a screenshot.** (Note that even your data points happen to have a concave perimeter and/or happen to be self-intersecting, the convex hull, by definition, would be a tight, enclosing boundary (hull) that is a simple convex polygon. The convex hull is a very useful object - eg. see [this](#) discussion.)
 - b. assuming the points (your collected locations) are called #1,#2,#3,...#11, create a polygon using your points #1,#2,#3,#9,#10,#11 (in that order), and another polygon with the remaining points in order (#4,#5,#6,#7,#8). Then **write a query to find out if the two polygons disjoint** - the result would be (Boolean) true or false, depending on your coordinates. See [this](#). **Add these two polygons to your .kml file, and visually verify (in Google Earth) the overlap as being true or false. Take**

a screenshot. (feel free to REARRANGE the points #1,#2,#3,#9,#10,#11 to get a non self-crossing polygon (if you get a self-crossing polygon when you don't reorder and that bothers you!); likewise feel free to reorder #4 through #8. Doing so might give you a different result for the disjointing [compared to the result from the polygons where the points are all in ascending order], which is fine.

What to submit

- Your .kml file contents from step 4a & 4b above - with the placemarks, convex hull and two region polygons. Please also explain what these points map to, and why you choose them. (1 pt)
- Your two queries from step 4 - table creation commands (if you use Postgres and directly specify points in your queries, you won't have table creation commands, in which case you wouldn't need to worry about this part), and the queries themselves. (4 pts)
- Screenshots from step 4. (1 pt)
- List the issues you met in this homework, and your solutions. **(0 pt, but missing that will deduct 1 pt)**

Draw a Curve (1pt)

Using a selected point (such as OHE) as the center, compute (don't use GPS!) a set (sequence) of lat-long (ie. spatial) coordinates that lie along a pretty [Epicycloid curve](#) :) Create a new KML file with these points, visualize it on Google Earth, submit these three items: **your point generation code** (see below), the **resulting .kml file content (including first 10 coordinates)** and **a screenshot**.

Parametric Equations: ($a = 7$, $b = 3$)

$$x(t) = (a + b) \cos t - b \cos((a/b + 1)t)$$

$$y(t) = (a + b) \sin t - b \sin((a/b + 1)t)$$

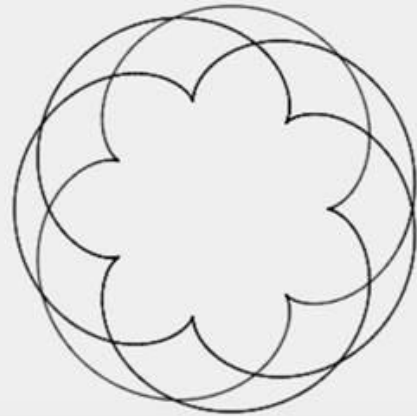
Using the above equations, loop through t from 0.00 to $n\pi$ (eg. 2π ; note that 'n' might need to be more than 2, for the curve to close on itself; and, t is in radians, not degrees), in steps of 0.01. That will give you the sequence of (x,y) points that make up the Epicycloid curve, which would/should look like the curve in the right side of the screengrab below, when $a = 7$, $b = 3$ (my JavaScript code for the point generation loop

is on the left):

```
var a=7, b=3;
var x0=a, y0=0;
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(150+10*x0, 150+10*y0);

var cos=Math.cos, sin=Math.sin, pi=Math.PI, nRev=
for(var t=0.0;t<(pi*nRev);t+=0.01) {
  var x=(a+b)*cos(t) - b*cos((a/b+1)*t);
  var y=(a+b)*sin(t) - b*sin((a/b+1)*t);
  ctx.lineTo(150+10*x,150+10*y);
}

ctx.stroke();
```



You need to ADD each (x,y) curve point to the (lat,long) of the center, ie. to that of the selected point- that will give you valid Epicycloid-based spatial coords for use in your .kml file. You can use any coding language you want, to generate (and visualize) the curve's coords: JavaScript, C/C++, Java, Python, MATLAB, Scala, Haskell, Ruby, R. You can also use Excel, SAS, SPSS, JMP etc., for computing and plotting the points.

Submission Guideline

- The submission MUST be a pdf file named [Student First Name]_[Student Last Name]_HW3.pdf
- The deadline is Friday, June 28, 2019 11:59 PM PDT. No submissions will be accepted past the deadline.