Exercise # 6

To add shapefile and raster to your database, you can go through two means, again. One is to use QGIS and connect it to your spatial database, another is to use, command line utilities mentioned above. For that change your directory to where you sample data is placed as

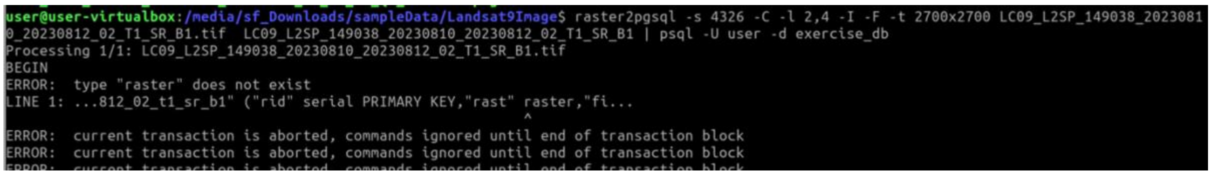
cd /data_directory/

and type

raster2pgsql -s 4326 -C -I -F LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1.tif \
LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1 | psql -U user -d exercise_db

You will need to type this whole thing in a single line, and remove the blackslashes '/'

However, you will be greeted with the following screen. Can you guess why this happens?

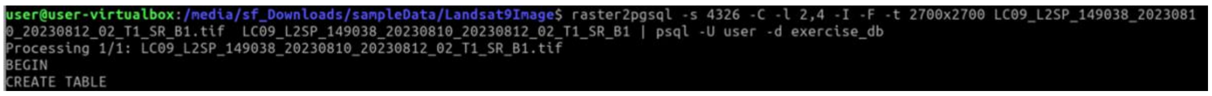


```
user@user-virtualbox:/media/sf_Downloads/sampleData/Landsat9Image$ raster2pgsql -s 4326 -C -l 2,4 -I -F -t 2700x2700 LC09_L2SP_149038_2023081
0_20230812_02_T1_SR_B1.tif  LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1 | psql -U user -d exercise_db
Processing 1/1: LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1.tif
BEGIN
ERROR:  type "raster" does not exist
LINE 1: ...812_02_t1_sr_b1" ("rid" serial PRIMARY KEY,"rast" raster,"fi...
                                                             ^
ERROR:  current transaction is aborted, commands ignored until end of transaction block
ERROR:  current transaction is aborted, commands ignored until end of transaction block
ERROR:  current transaction is aborted, commands ignored until end of transaction block
```

You will need to create an extension for the database that can handle, rasters. For that before you do that, you need to,

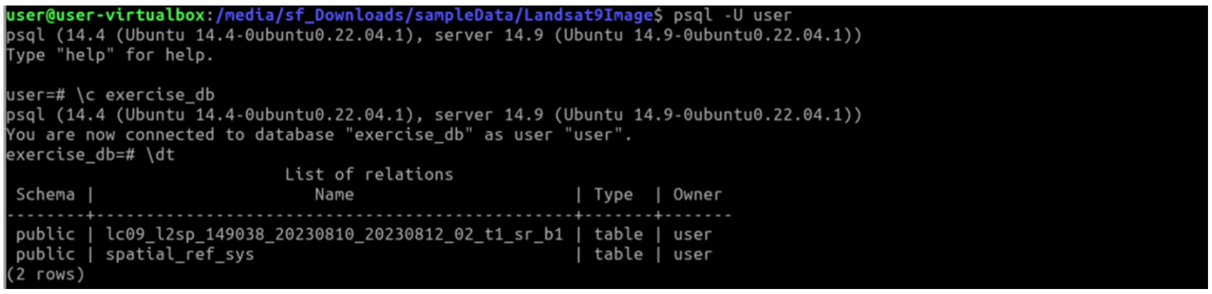
CREATE EXTENSION postgis_raster;

This will raster enable your database, and if you run the above command again, you will get the following output.



```
user@user-virtualbox:/media/sf_Downloads/sampleData/Landsat9Image$ raster2pgsql -s 4326 -C -l 2,4 -I -F -t 2700x2700 LC09_L2SP_149038_2023081
0_20230812_02_T1_SR_B1.tif  LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1 | psql -U user -d exercise_db
Processing 1/1: LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1.tif
BEGIN
CREATE TABLE
```

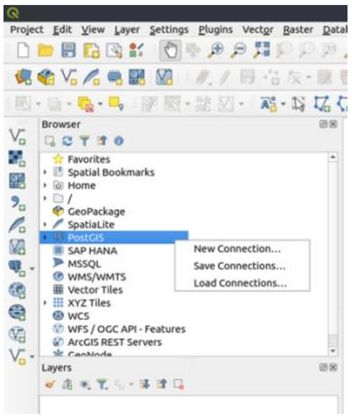
The imported file will be in the table showing as follows.



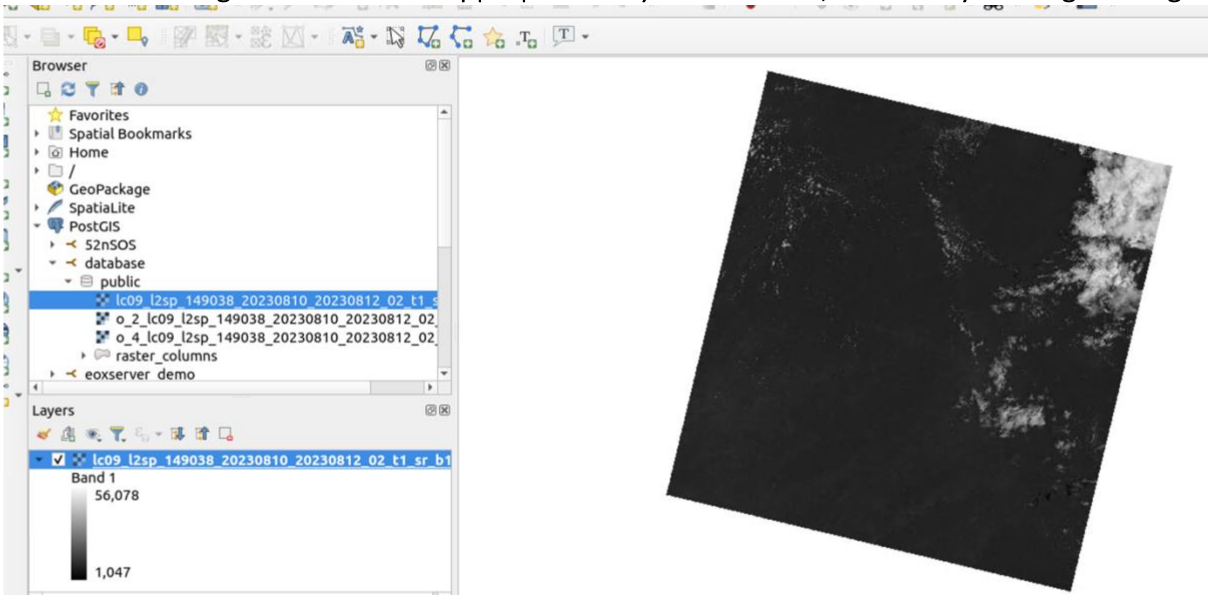
```
user@user-virtualbox:/media/sf_Downloads/sampleData/Landsat9Image$ psql -U user
psql (14.4 (Ubuntu 14.4-0ubuntu0.22.04.1), server 14.9 (Ubuntu 14.9-0ubuntu0.22.04.1))
Type "help" for help.

user=# \c exercise_db
psql (14.4 (Ubuntu 14.4-0ubuntu0.22.04.1), server 14.9 (Ubuntu 14.9-0ubuntu0.22.04.1))
You are now connected to database "exercise_db" as user "user".
exercise_db=# \dt
                             List of relations
 Schema |                      Name                      | Type  | Owner
--------+------------------------------------------------+-------+-------
 public | lc09_l2sp_149038_20230810_20230812_02_t1_sr_b1 | table | user
 public | spatial_ref_sys                                | table | user
(2 rows)
```

Let's have a look at our file in QGIS, for that you need to create a connection to your db.



Add the following details or those appropriate to your database, and voila you are good to go.


Browser
Favorites
Spatial Bookmarks
Home
/
GeoPackage
SpatiaLite
PostGIS
52nSOS
database
public
lc09_l2sp_149038_20230810_20230812_02_t1_s
o_2_lc09_l2sp_149038_20230810_20230812_02
o_4_lc09_l2sp_149038_20230810_20230812_02
raster_columns
eoxserver_demo
Layers
lc09_l2sp_149038_20230810_20230812_02_t1_sr_b1
Band 1
56,078
1,047

Note in the above, example you can see same 3x files, kindly ignore that since these were some overviews created. In your case you will only get a single file.

To understand more about the flags used in raster2pgsql above you can see the following.

# -s use srid 4326
# -I create spatial index
# -C use standard raster constraints
# *.tif load all these files
# -F include a filename column in the raster table
# -t tile the output <grid x grid>

Kindly make sure that you name your db connection the same, as your database you want to connect to. This will make it easier for you to identify the files etc. Otherwise, you will have an issue.

To get you further acquainted with the raster processing capabilities of postgres, let’s work with a digital elevation model since, the same can be processed for hydrological analysis.

We shall be using the following functions in postgis_raster.

1. ST_Slope
2. ST_Hillshade
3. ST_Aspect
4. ST_TPI
5. ST_TRI
6. ST_Roughness

These functions are very useful and are similarly implemented in other spatial analysis utilities, as well. The general syntax for these is,

```
ST_Slope (raster rast, integer nband=1, text pixeltype=32BF, text units=DEGREES, double precision scale=1.0, boolean interpolate_nodata=FALSE);
```

Let’s first download a DEM tile, this can be done using the following means.

You can follow the following link and download a tile of your choice; I will strongly recommend picking a region always known to you

https://dwtkns.com/srtm30m/

Alternatively, you can use Google Earth Engine (GEE) to download a tile of your choice, using the following code. You might need to alter it for your choice of study area. The following code is for entire region of Pakistan.

```
// Load SRTM elevation V4 dataset
var dataset = ee.Image('CGIAR/SRTM90_V4');
var elevation = dataset.select('elevation');
Map.setCenter(69.3, 30.3, 4);
// Load a country border as a region of interest (roi).
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');
var roi = countries.filterMetadata('country_na', 'equals', 'Pakistan');
// Clip the image to the region of interest.
Map.addLayer(elevation.clip(roi));
elevation = elevation.clip(roi);
// Export the image, specifying scale and region.
Export.image.toDrive({
  image: elevation,
  region: roi,
  description: 'imageToDriveExample',
  scale: 30,
  maxPixels: 1e12
});
```

Once, you have your area of interest files, downloaded, you can proceed to import the file to your database. It is recommended that you create a new database and import your files accordingly. You can name the database elevationData.

But the files you get from the first, website, will be in hgt format. You can directly consume that in postgis but if you want, you can convert it to GeoTIFF using the following line of code.

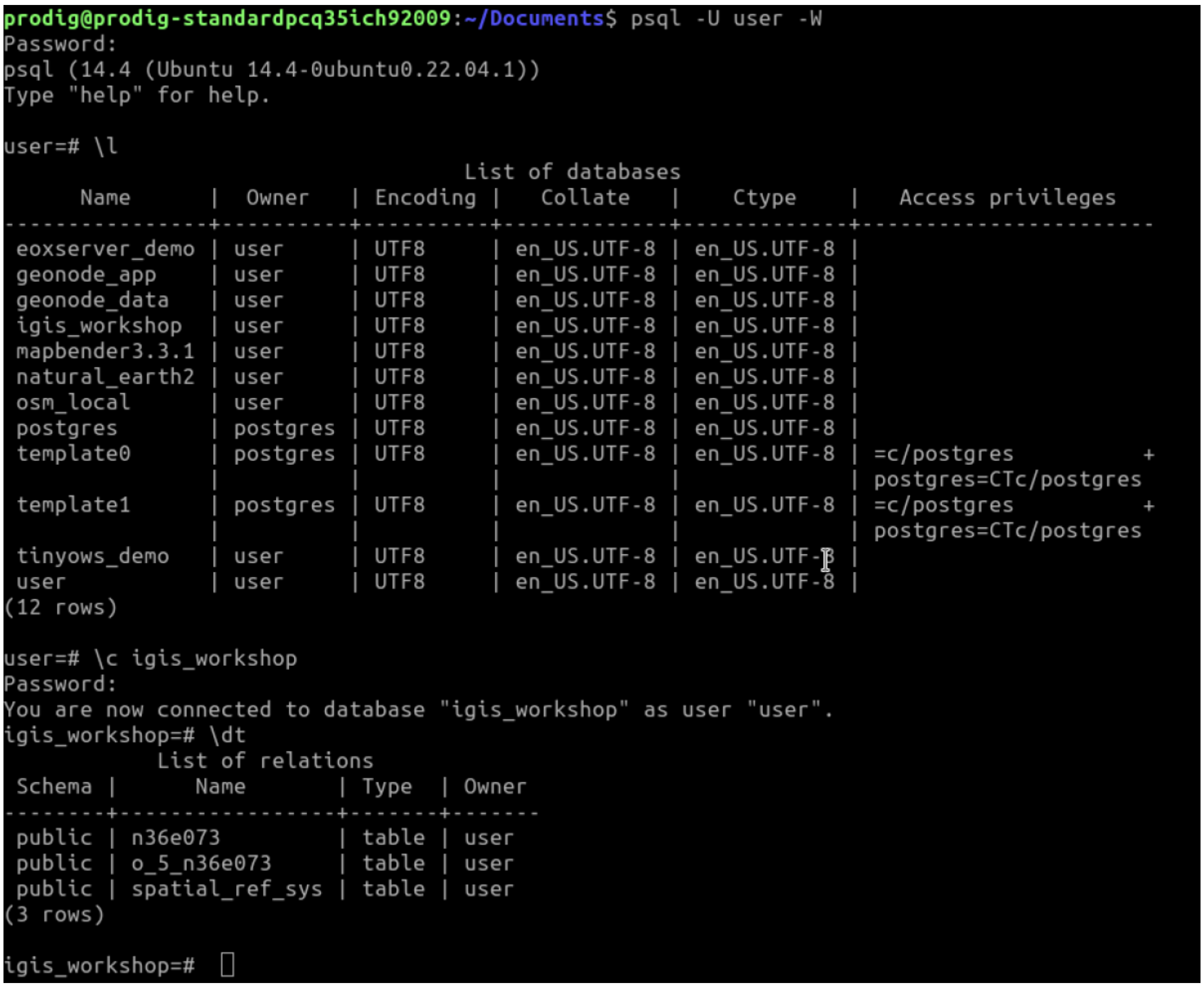
raster2pgsql -G # lists all formats that postgis supports.
gdalwarp -r cubic <input>.hgt <output>.tif

the -r cubic options allow for appropriate resampling of the elevation raster. If you want to skip this flag, you may do so.

To import your elevation dataset, you can use the following command, but ensure that you are already in the working directory where the downloaded elevation datasets are present.

raster2pgsql -C -I -M -F -Y -l 5 -s 4326 -t 100x100 N36E073.TIF 100x100 N36E073 | psql -U user -d exercise_db

Can check if the file has been imported properly into your database.



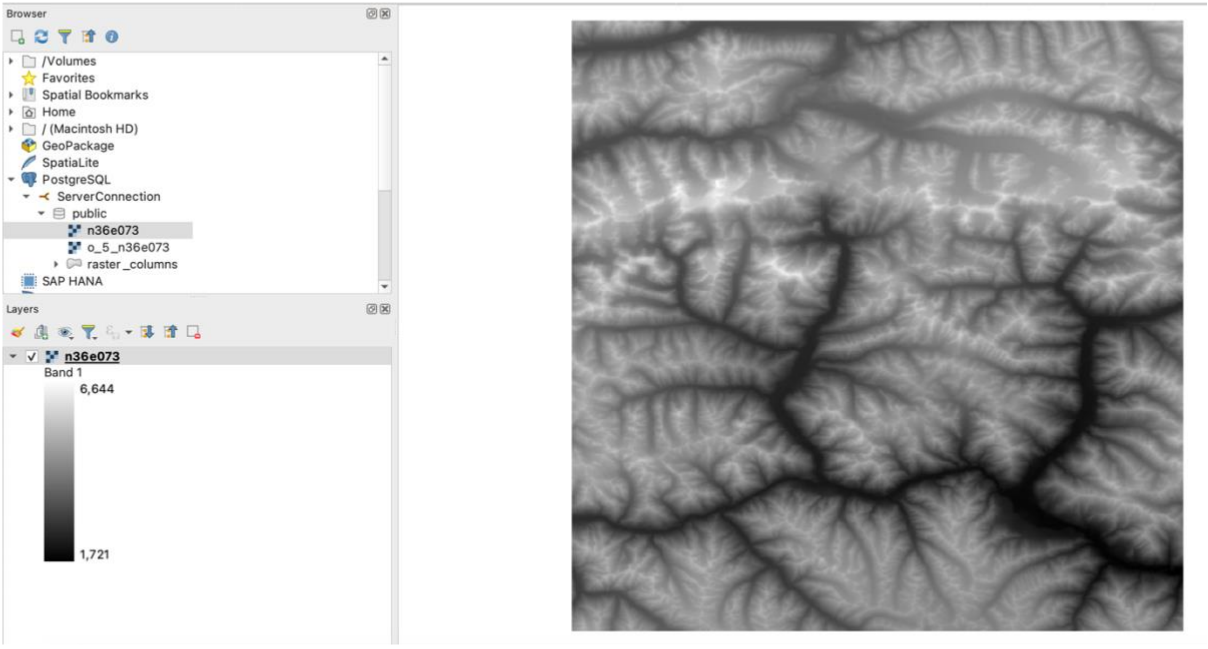
```
prodig@prodig-standardpcq35ich92009:~/Documents$ psql -U user -W
Password:
psql (14.4 (Ubuntu 14.4-0ubuntu0.22.04.1))
Type "help" for help.

user=# \l
                                    List of databases
      Name      |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
----------------+----------+----------+-------------+-------------+-----------------------
 eoxserver_demo | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 geonode_app    | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 geonode_data   | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 igis_workshop  | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 mapbender3.3.1 | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 natural_earth2 | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 osm_local      | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres       | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
                |          |          |             |             | postgres=CTc/postgres
 template1      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
                |          |          |             |             | postgres=CTc/postgres
 tinyows_demo   | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 user           | user     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(12 rows)

user=# \c igis_workshop
Password:
You are now connected to database "igis_workshop" as user "user".
igis_workshop=# \dt
            List of relations
 Schema |      Name       | Type  | Owner
--------+-----------------+-------+-------
 public | n36e073         | table | user
 public | o_5_n36e073     | table | user
 public | spatial_ref_sys | table | user
(3 rows)

igis_workshop=#
```

You can explore it in QGIS or have an overview in pgAdmin.



However, it is important here to enlist what the flags, included above do.

# -C This ensure raster data is not loaded with invalid information.
# -I Creates the GIST spatial index
# -M Run Vacuum analyze on the table
# -Y Use copy statement instead of insert, and allows for faster computation and less overheads, although it is less forgiving as compared to insert
# -l Overview factors, overviews are quick renders of the original data to say the least

ignore the database name here, since I have been testing the lines of code on two separate computers, hence the databases are differently name. You can remain specific to the names you desire.

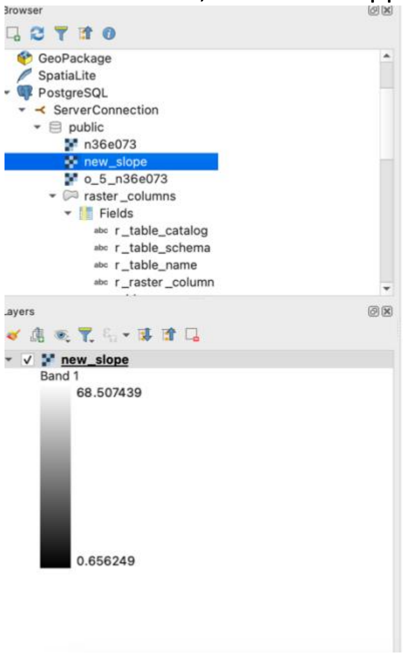
Also, you must have notices, -W flag. This is used when your user and the system users are different. You need to force a prompt for it to ask for your password, else you will continue to get authentication errors as it tries to log you in without a password.

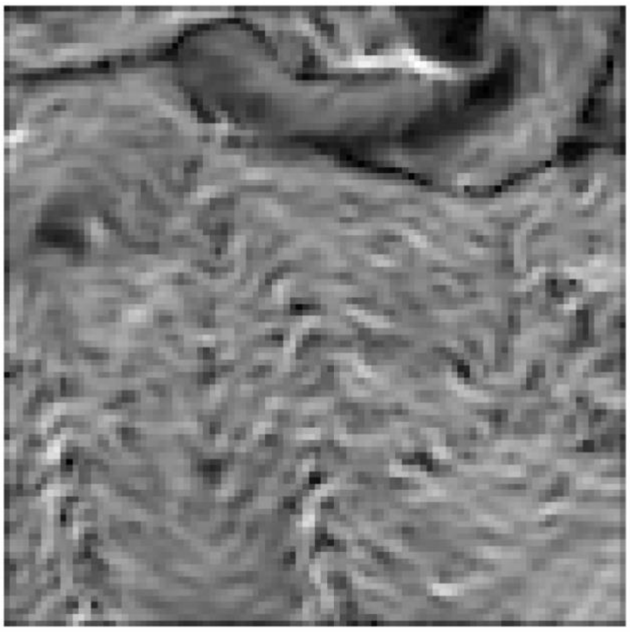
Let’s use the above-mentioned functions one by one on our data. For a start we shall start by calculating the slope of the given raster. This is done using the following command.

```
CREATE TABLE public.n36e073 slope AS
SELECT rid, ST_Slope(rast::raster, '1'::int,'32BF'::text,'DEGREES'::text,'111120'::double
precision) AS rast FROM n36e073 WHERE rid=’4’;
```

In the above, the pixel type is 32BF which refers to 32-bit floating point. While the units for distance is in degrees and elevation in meters. The scale value therefore is '111120'. If it was latitude longitude and feet, the scale value would have been 370400.

You will see the following file added to your schema. However, this takes a lot of time. Hence it is best to use, a smaller clipped section of the same.





Here I would want you to create similar rasters for, aspect, hillshade, Topographic Position Index (TPI) and Terrain Ruggedness Index (TRI).

But first let’s reclassify our slope raster into five broad classes. This can be done using the following command.

The values that we are picking are.

0-4.99
5-9.99
10-14.99
15-24.99
25-90 degrees
with 1-5 values.

The command therefore would be,

```
CREATE TABLE public.n36e073_reclass slope AS
SELECT rid,ST_Reclass(
  (ST_SLOPE(rast::raster,'1'::int,'32BF'::text,'DEGREES'::text,'111120'::doub
  le precision))::raster,
    '1'::int,   '0-5):1, 5-10:2, [10-15):3, 15-25):4,
  [25-90):5'::text,'32BF'::text)
  FROM n36e073;
```

For hillshade the command would be

ST_HillShade(raster n36e073, integer band=1, text pixeltype=32BF, double

```
  precision azimuth=315, double precision altitude=45, double precision
  max_bright=255, double precision scale=1.0, boolean
  interpolate_nodata=FALSE);
```

But this takes way too long so we need to learn to clip our rasters to meaningful area of interest. If that can be done, right the process will be faster and much more fun. Hence, let's work on that first.

For that you might want to open your raster in QGIS and quickly create a shapefile. It is imperative to know that the same can be created in postgres but for now we will do it on QGIS.

Once, you have created the shapefile, import this shapefile using shp2pgsql. And add it your database.

Once, you have done so it is now time to use this vector to clip your original raster. The syntax goes as follows.

```
CREATE TABLE schema_name.clip AS
SELECT ST_Clip(a.rast, b.geom, true)
FROM rasters.dem AS a, vectors.porto_parishes AS b
WHERE ST_Intersects(a.rast, b.geom);
```

Try the above and see how it works for you.