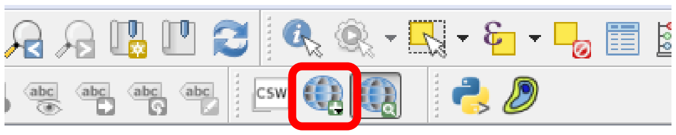
Getting started: Setup of your GIS environment

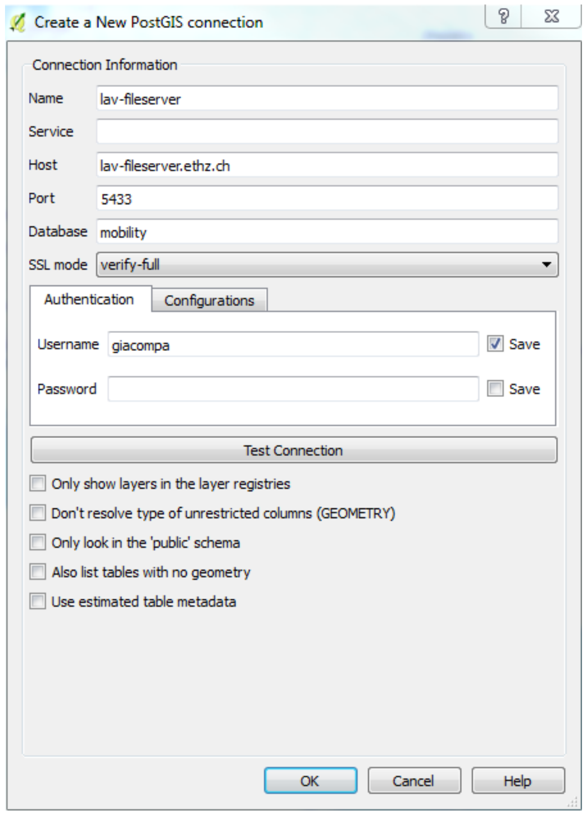
This document should be followed only by students who will work with GIS:

☐ QGIS:

- Download QGIS, which is a user-friendly and interactive software to manipulate GIS information. Go to https://www.qgis.org/it/site/forusers/download.html and download the 'QGIS Standalone Installer Version **2.18**' appropriate for your operative system. Install it.
- Once installed, open GQIG Desktop and install some Plugins. During the Thesis you may find useful to install different Plugins, for the moment you just need one. Go to Plugins -> Manage and Install Plugins.. -> Search 'QuickMapServices' -> Select it -> Install plugin -> Close
- To test, create a new project, click on the icon of QuickMapServices

  Go to OSM -> OSM Standard.
  And now you should see a map of the world. You can zoom in and out. What you just did is adding a **Raster** layer, which is a picture where you cannot select objects. These layers represent however very convenient backgrounds.
- Connect to the LAV server.
  In the Browser Panel (if you don't have it, add it from View -> Panels) right click on PostGIS -> New Connection..
  Fill the form with the same information as you did with for new server in pgAdmin (provide the password every time it is asked)



Create a New PostGIS connection

Connection Information

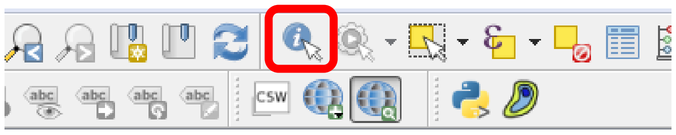
Name	lav-fileserver
Service	
Host	lav-fileserver.ethz.ch
Port	5433
Database	mobility
SSL mode	verify-full

Authentication | Configurations

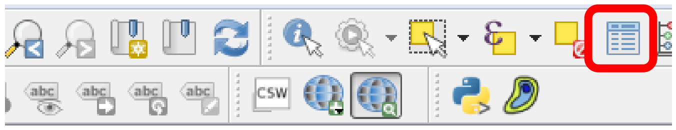
Username	giacompa	[✓ Save]
Password		Save

Test Connection
Only show layers in the layer registries
Don't resolve type of unrestricted columns (GEOMETRY)
Only look in the 'public' schema
Also list tables with no geometry
Use estimated table metadata
OK
Cancel
Help

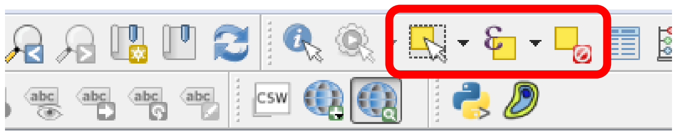
- Add a layer from the server.
  Once connected to the server you will see the list of all schemas (as in pgAdmin).
  Go to swissboundaries -> swissboundaries3d_1_3_tlm_2017_kantonsgebiet
  Drag it in the Layers Panel. The new layer should appear on the map, with the shapes of the different Swiss cantons. This is a Vector layer, made of objects (called **shapes** or **features**, in this example specifically they are polygons) with information. You can investigate a layer by selecting it in the Layers Panel (checking the box sets the visibility, clicking on the name activates the layer) and by using the tool:

For instance you can now click on the canton of Zürich (a **shape**) and the Identify Results panel on the right will appear, will all the **attributes** of the selected shape (such as population and area).
You can see all the attributes of all shapes of a layer by selecting the layer and either right click -> Open Attribute Table or click on the icon



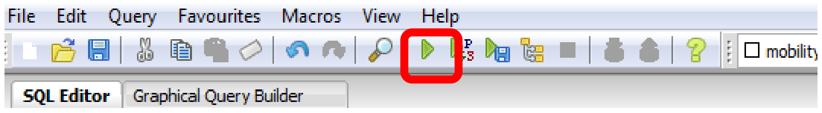
What you see is a table where each row is a shape and that can also be opened and processed in pgSQL, Python, Pandas…
QGIS contains many other useful functions. Check out especially the selection tools, that allow you to select features (=shapes) manually or depending on their properties and attributes.



Observe that the selection of rows in the attribute table is equivalent to the just mentioned selection of shapes.

☐ PostGIS:

- It is not a software that you need to install, but we list it here for your understanding. It is an extension of PostgreSQL, the dialect of SQL that we work with and we use on pgAdmin. Specifically, it is an extensions that enables the manipulation of GIS data. It is already installed on the server and you can already use it. As a good rule of thumb, any SQL function that starts with "ST_" comes from PostGIS and involves geometrical entities.
- For a quick test, open pgAdmin, connect to the server and go to (Databases) mobility -> (Schemas) swissboundaries -> (Tables) swissboundaries3d_1_3_tlm_2017_kantonsgebiet -> right click -> Scripts -> CREATE Script. This will open the SQL Editor with an already written query. This query has no filters and will return all rows and columns of the selected table. Don't run it for very big tables, but in this specific example you can: press F5 or click on



In the Output pane you will see the result of the query, which is the entire table. It is the very same as the attribute table in QGIS, but with an additional column **geom**, of type geometry. This is the column that contains that actual shape you saw in QGIS, while all other non-geometry columns are attributes of that shape. Being a vectorial description of a shape, often the geometry will be in non-human-readable format.
- You will see that you have 51 rows, although the Swiss cantons are 26: this is because every shape/row (in **this** table) represents a *continuous* part of a canton. E.g. the canton of Bern appears in 4 rows, because it is made of 4 disconnected territories. You can verify this also in QGIS, where you can singularly select only the individual continuous parts of canton Bern, but not the entire canton as a whole.
- We can write a query that merges parts belonging to the same canton:

```
SELECT kantonsnum, count(*) AS n_parts, ST_Union(geom) AS geom
FROM swissboundaries.swissboundaries3d_1_3_tlm_2017_kantonsgebiet
GROUP BY kantonsnum
```

The last GROUP BY statement implies we are grouping rows according to the column “kantonsnum” (since different parts of the same canton have the same “kantonsnum”; we could have chosen also the “name” field). When grouping rows, every output column not appearing in the GROUP BY statement must be somehow **aggregated** (PostgreSQL does not know which of the grouped rows to return). This is the case for the geometry: we want all the shapes of a single canton to be aggregated in a single geometry. This is done through the PostGIS function “ST_Union()”, which returns a geometry that is the “geographical” sum of the input geometries (find online additional details). The aggregate function “count()” does not need to refer to a column and it counts the number of input rows that are grouped in a single one. In this example it returns the number of discontinuous parts belonging to a single canton.
For more details on the SQL syntax, check SQL tutorials.

- If you run the new query you get a table with the 3 output columns and 26 rows, one for each canton. We cannot appreciate the merged geometries yet. For that we need the next and last step.

☐ Python and Jupyter Notebook:

- This is the environment where you can merge the elaboration capabilities of PostgreSQL/PostGIS, the graphical appeal of QGIS and the flexibility of Python/Pandas.
- Also Python requires a couple of extensions (packages) to work with GIS data. To install them, open a terminal and type:
  ```
  conda install geopandas
  ```
  you will be asked to install also other packages: accept and install them all.
- The setup is done; the tutorial continues (and ends) in a Jupyter Notebook. Go online to https://github.com/LAV-ESG/getting-started and download “gettingStartedNotebook_GIS.ipynb”. Save it locally, open it with Anaconda/Jupyter Notebook shortcut and go through it.