

# Spatial Databases: PostGIS

Option GIS-Python

hes.  
**so**  
**business.**

Jean-Paul Calbimonte



School of Management

Bachelor of Science HES-SO (BSc) in Business  
Information Technology

## > Before we start ...



install QGIS: useful for visualizing and connecting with PostGIS

<https://www.qgis.org>

# > PostGIS

## What is PostGIS?



- Extension to the PostgreSQL object-relational database
- Allows GIS (Geographic Information Systems) objects to be stored in the database.
- Support for R-Tree spatial indexes.
- Functions for analysis and processing of GIS objects.

# > PostGIS geometry types



## Geometries in WKT (well known text):

- `POINT (0 0)`
- `LINESTRING (0 0,1 1,1 2)`
- `POLYGON ((0 0,4 0,4 4,0 4,0 0),  
          (1 1, 2 1, 2 2, 1 2,1 1))`
- `MULTIPOINT ((0 0), (1 2))`
- `MULTILINESTRING ((0 0,1 1,1 2), (2 3,3 2,5 4))`
- `MULTIPOLYGON (((0 0,4 0,4 4,0 4,0 0), (1 1,2 1,2 2,1 2,1 1)),  
                  ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))`
- `GEOMETRYCOLLECTION (POINT(2 3),  
                      LINESTRING(2 3,3 4))`

Essentially same types as in Python/shapely

## > Install Postgis: options

- Option 1: Install locally
- Option2: Install on docker

# > Option1: Install PostgreSQL locally

Windows:

<https://www.postgresql.org/download/windows/>

Mac:

<http://postgresapp.com/>

<https://www.postgresql.org/download/macosx/>



## Postgres.app

The easiest way to get started with PostgreSQL on the Mac

Introduction Downloads Documentation GitHub ← 4665 Stars!

## Latest Release

If you're new to Postgres, this is the file you should download. It includes everything you need to get started with PostgreSQL and PostGIS.

### Postgres.app with PostgreSQL 11

Postgres.app v2.2.2 · Requires macOS 10.12 · Download Size 70MB

PostgreSQL 11.2 / PostGIS 2.5.1 / plv8 2.3.9

↓ Download

## > Option 1: Install PostGIS locally

<https://postgis.net/install/>

PostGIS is an optional extension in PostgreSQL

To enable it enter into a `psql` console:

```
-- Enable PostGIS (includes raster)
CREATE EXTENSION postgis;

-- Enable Topology
CREATE EXTENSION postgis_topology;
```

Other PostGIS extensions can be added if needed (e.g. 3D etc.)

# > Option 1: Install pgAdmin locally

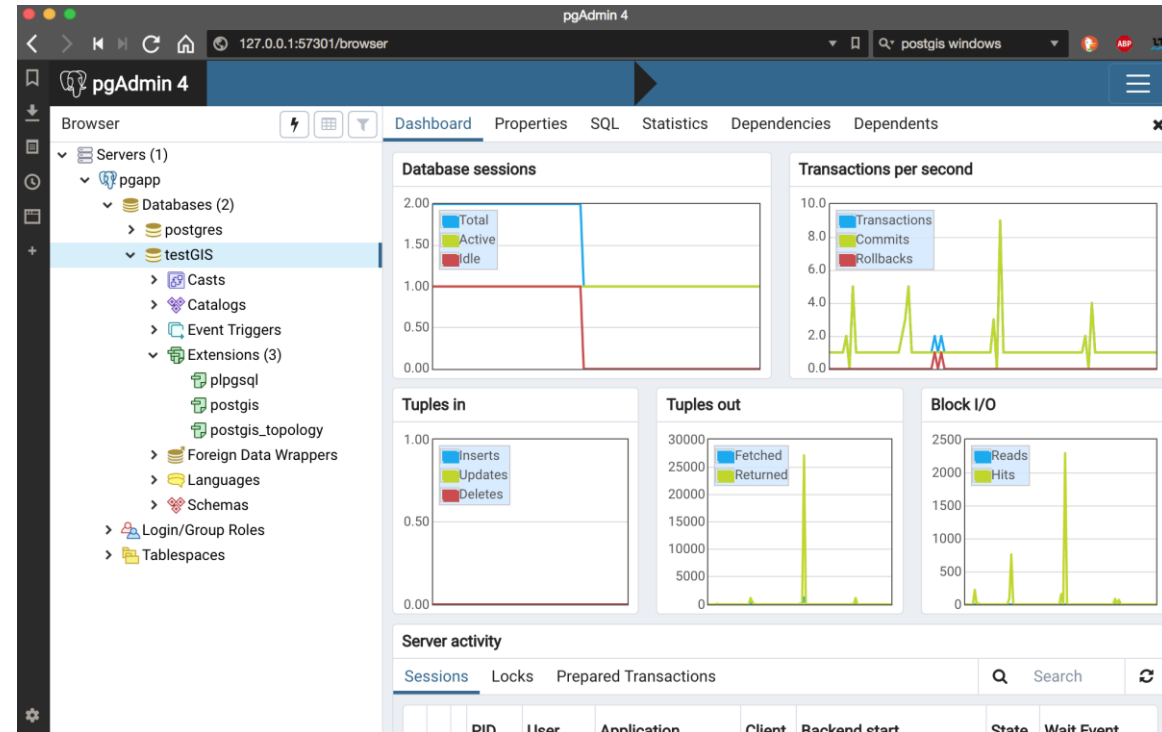
<https://www.pgadmin.org>

pgAdmin is a tool for managing PostgreSQL databases

The screenshot shows the 'Create - Server' dialog box in pgAdmin, with the 'Connection' tab selected. The fields are filled with the following values:

- Host name/address: localhost
- Port: 5432
- Maintenance database: postgres
- Username: postgres
- Password: (empty)
- Save password?:
- Role: (empty)
- Service: (empty)

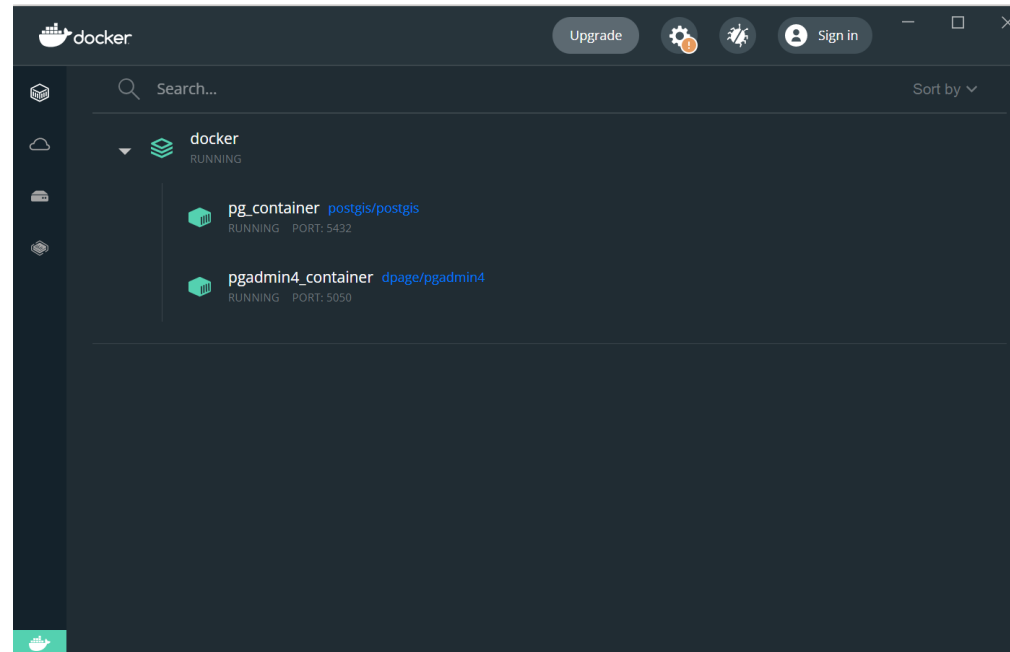
A red error message at the bottom states: "Name must be specified." The 'Save' button is disabled.



connect to your local PostgreSQL installation

## > Option 2: Install on docker

Or don't install anything and use docker






<https://www.docker.com/>

# Develop faster. Run anywhere.

The most-loved Tool in Stack Overflow's 2022 Developer Survey.

Download Docker Desktop

 Windows

 Apple Chip

 Linux

 Intel Chip

Install docker desktop

# > PostGIS with docker

Download the [docker-compose.yml](#) file from Cyberlearn in a folder in your computer (for example a folder named postgis)

## INTRO TO POSTGIS



docker-compose



Fichier texte

# > PostGIS: Use docker compose

```
1  version: '3.8'
2  services:
3    db:
4      container_name: pg_container
5      image: postgis/postgis
6      restart: always
7      environment:
8        POSTGRES_USER: root
9        POSTGRES_PASSWORD: root
10       POSTGRES_DB: test_db
11     ports:
12       - "5432:5432"
13   pgadmin:
14     container_name: pgadmin4_container
15     image: dpage/pgadmin4
16     restart: always
17     environment:
18       PGADMIN_DEFAULT_EMAIL: admin@admin.com
19       PGADMIN_DEFAULT_PASSWORD: root
20     ports:
21       - "5050:80"
```

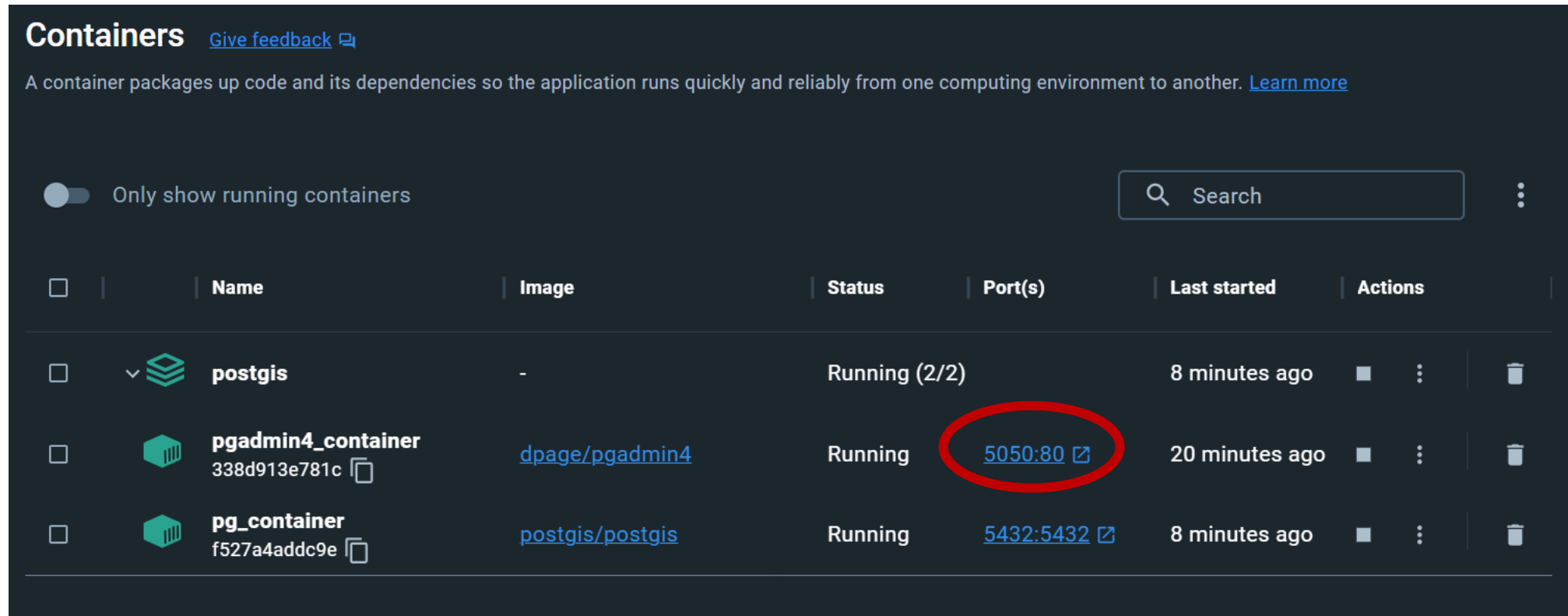
Now open a terminal in that folder and type:

```
>docker compose up
```

```
C:\Users\jpcik\git\postgis>docker compose up
[+] Running 0/18
- db Pulling                                     3.0s
- 3f9582a2cbe7 Pulling fs layer                 0.6s
- 0d9d08fcl1a1a Pulling fs layer               0.6s
[+] Running 0/32b Pulling fs layer             0.6s
- db Pulling                                     3.1s
- 3f9582a2cbe7 Downloading [>]                ] 327.7kB/... 0.7s
- 0d9d08fcl1a1a Downloading [>]                ] 44.3kB/... 0.7s
```

# > Running docker

Now you should have both postgres/postgis (the database) and PgAdmin (the admin tool) running



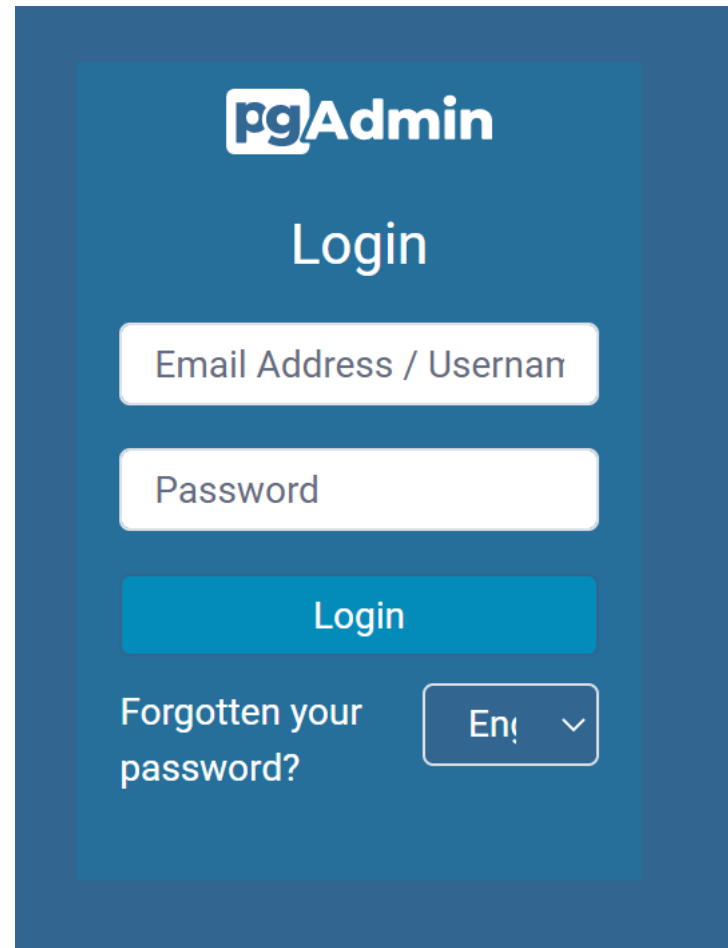
The screenshot shows the Docker Desktop interface with a list of running containers. The 'pgadmin4\_container' is highlighted with a red circle around its port mapping '5050:80'. The interface includes a search bar, a toggle for 'Only show running containers', and a table with columns for Name, Image, Status, Port(s), Last started, and Actions.

	Name	Image	Status	Port(s)	Last started	Actions
<input type="checkbox"/>	postgis	-	Running (2/2)		8 minutes ago	■ ⋮ 🗑️
<input type="checkbox"/>	pgadmin4_container 338d913e781c 📄	<a href="#">dpage/pgadmin4</a>	Running	<b>5050:80</b> 🗑️	20 minutes ago	■ ⋮ 🗑️
<input type="checkbox"/>	pg_container f527a4addc9e 📄	<a href="#">postgis/postgis</a>	Running	<a href="#">5432:5432</a> 🗑️	8 minutes ago	■ ⋮ 🗑️

Click here to open the Admin tool pgAdmin

# > pgAdmin

You can login with [admin@admin.com](mailto:admin@admin.com), and password 'root'



pgAdmin

Login

Email Address / Username

Password

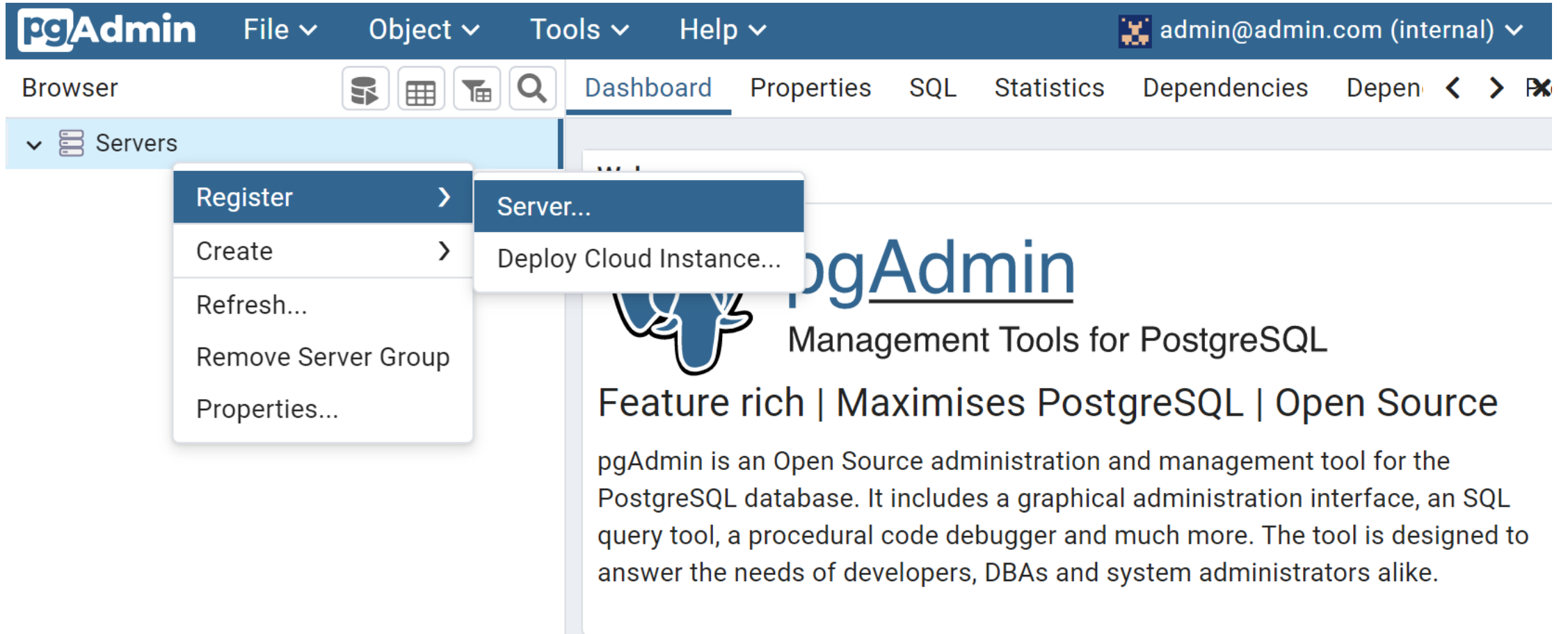
Login

Forgotten your password?

En ↓

# > pgAdmin

You can register a new server



The screenshot displays the pgAdmin web interface. At the top, there is a dark blue navigation bar with the 'pgAdmin' logo and menu items: 'File', 'Object', 'Tools', and 'Help'. On the right side of this bar, the user is logged in as 'admin@admin.com (internal)'. Below the navigation bar is a 'Browser' section with icons for database, table, and search, and a search input field. The main content area shows a 'Servers' menu expanded, with options: 'Register', 'Create', 'Refresh...', 'Remove Server Group', and 'Properties...'. The 'Register' option is highlighted, and a sub-menu is visible with 'Server...' and 'Deploy Cloud Instance...'. In the background, a promotional banner for pgAdmin is visible, featuring the text: 'pgAdmin Management Tools for PostgreSQL', 'Feature rich | Maximises PostgreSQL | Open Source', and a paragraph describing the tool as an open-source administration and management tool for PostgreSQL.

# > pgAdmin

You may use these parameters and save:

Register - Server ↗ ✕

General **Connection** Parameters SSH Tunnel Advanced

Host name/address	<input type="text" value="db"/>
Port	<input type="text" value="5432"/>
Maintenance database	<input type="text" value="postgres"/>
Username	<input type="text" value="root"/>
Kerberos authentication?	<input type="checkbox"/>
Password	<input type="password" value="..."/> <span>👁</span>
Save password?	<input type="checkbox"/>
Role	<input type="text"/>
Service	<input type="text"/>

? ? ✕ Close ↺ Reset 💾 Save

# > pgAdmin

And you are ready to start.

The screenshot displays the pgAdmin web interface. The top navigation bar includes 'pgAdmin', 'File', 'Object', 'Tools', and 'Help' menus, along with a user dropdown for 'admin@admin.com (internal)'. Below the navigation bar is a 'Browser' pane on the left showing a tree view of the database structure: Servers (1) > postgres > Databases (2) > postgres > test\_db. The 'test\_db' database is selected and expanded, showing its components: Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, and Subscriptions. Below the tree view are 'Login/Group Roles' and 'Tablespaces'. The main dashboard area contains six charts: 'Database sessions' (Total, Active, Idle), 'Transactions per second' (Transactions, Commits), 'Tuples in' (Inserts, Updates), 'Tuples out' (Fetched, Returns), and 'Block I/O' (Reads, Hits). All charts show zero activity. The bottom right corner of the interface includes the text 'School of Management | 18' and a small icon set.

# > PostGIS: first steps

In the query editor  check if installed, and version:

```
SELECT postgis_full_version();
```

```
POSTGIS="2.4.6 r17068" PGSQL="100" GEOS="3.6.3-CAPI-1.10.3 80c13047" PROJ...
```

- ▼ Servers (1)
  - ▼ pgapp
    - ▼ Databases (2)
      - > postgres
      - ▼ testGIS
        - > Casts
        - > Catalogs
        - > Event Triggers
        - ▼ Extensions (3)
          - plpgsql
          - postgis
          - postgis\_topology
        - > Foreign Data Wrappers
        - > Languages
        - > Schemas
      - > Login/Group Roles
      - > Tablespaces

extensions also visible in the sidebar

# > PostGIS geometries

Create basic geometries:

```
CREATE TABLE geometries (name varchar, geom geometry);
```

```
INSERT INTO geometries VALUES  
('Point', 'POINT(0 0)'),  
('Linestring', 'LINESTRING(0 0, 1 1, 2 1, 2 2)'),  
('Polygon', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),  
('PolygonWithHole', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),  
                               (1 1, 1 2, 2 2, 2 1, 1 1))'),  
('Collection', 'GEOMETRYCOLLECTION(POINT(2 0),  
                                     POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))');
```

```
SELECT name, ST_GeometryType(geom), ST_NDims(geom),  
       ST_SRID(geom), ST_NumGeometries(geom)  
FROM geometries;
```

"Point"	"ST_Point"	2	0	1
"Linestring"	"ST_LineString"	2	0	1
"Polygon"	"ST_Polygon"	2	0	1
"PolygonWithHole"	"ST_Polygon"	2	0	1
"Collection"	"ST_GeometryCollection"	2	0	2

# > Point & LineStrings

## Point

```
SELECT ST_X(geom), ST_Y(geom), ST_asText(geom)
FROM geometries
WHERE name = 'Point';
```

```
"0" "0" "POINT(0 0)"
```

## LineString

```
SELECT ST_Length(geom), ST_Npoints(geom)
FROM geometries
WHERE name = 'LineString';
```

```
"3.41421356237309" 4
```

# > Polygons

```
SELECT name, ST_Area (geom) ,  
          ST_NRings (geom) ,  
          ST_AsText (ST_InteriorRingN (geom, 1) ) ,  
          ST_AsText (ST_ExteriorRing (geom) )  
FROM geometries  
WHERE name LIKE 'Polygon%';
```

"Polygon"	"1"	1	"LINESTRING(0 0,1 0,1 1,0 1,0 0)"	
"PolygonWithHole"	"99"	2	"LINESTRING(1 1,1 2,2 2,2 1,1 1)"	"LINESTRING(0 0,10 0,10 10,0 10,0 0)"

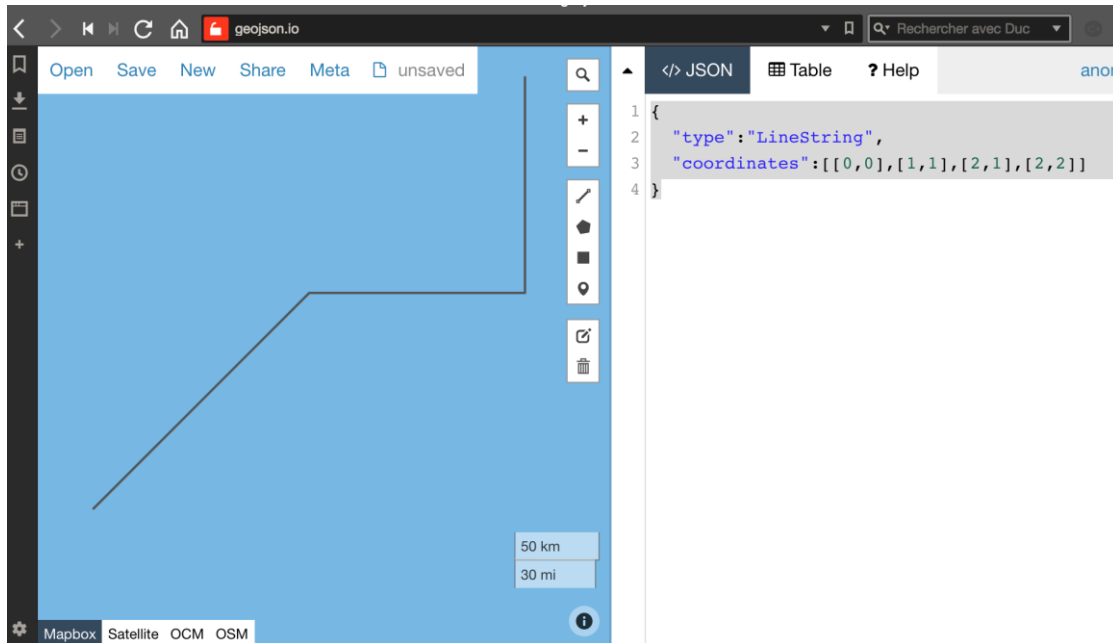
## > Conversion functions

- **ST\_AsText**: Returns the Well-Known Text (WKT) representation of the geometry/geography without SRID metadata.
- **ST\_AsBinary**: Returns the Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.
- **ST\_EndPoint**: Returns the last point of a LINESTRING geometry as a POINT.
- **ST\_AsEWKB**: Returns the Well-Known Binary (WKB) representation of the geometry with SRID meta data.
- **ST\_AsEWKT**: Returns the Well-Known Text (WKT) representation of the geometry with SRID meta data.
- **ST\_AsGeoJSON**: Returns the geometry as a GeoJSON element.
- **ST\_AsGML**: Returns the geometry as a GML version 2 or 3 element.
- **ST\_AsKML**: Returns the geometry as a KML element. Several variants. Default version=2, default precision=15.
- **ST\_AsSVG**: Returns a Geometry in SVG path data given a geometry or geography object.

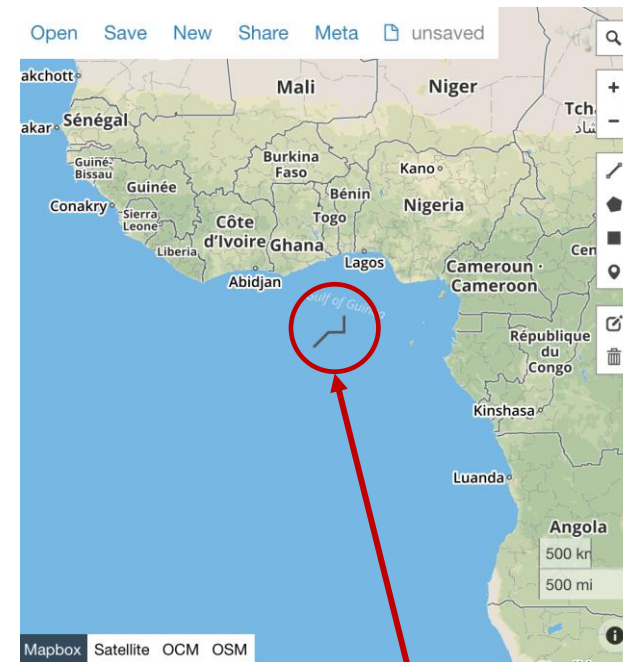
# > Formats: GeoJSON

```
SELECT ST_AsGeoJSON(geom) FROM geometries  
WHERE name = 'Linestring';
```

```
{  
  "type": "LineString",  
  "coordinates": [[0,0],[1,1],[2,1],[2,2]]  
}
```



visualize in geojson.io

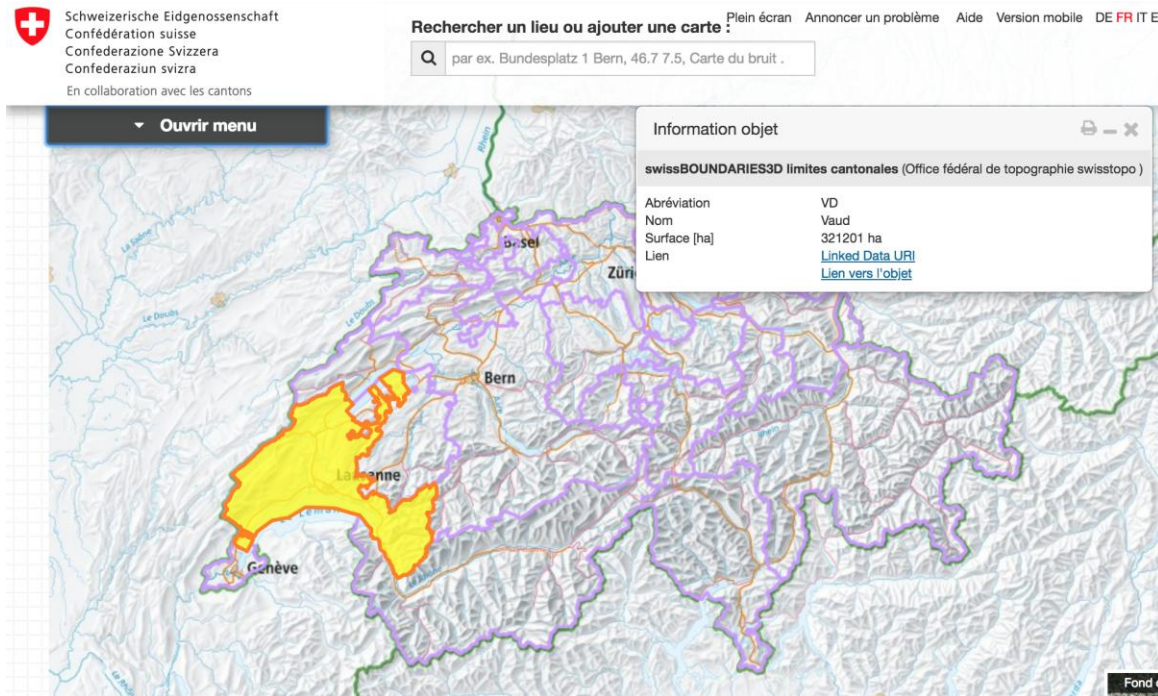


where in the world is this  
linestring?

# > Loading Shapefiles

Load shapefile as a Table with spatial objects in it

<https://map.geo.admin.ch/> Lots of official Swiss maps

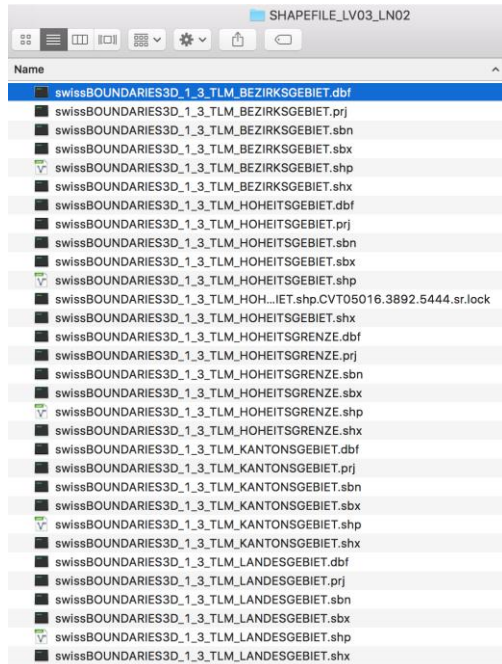


swissBOUNDARIES3D:  
Swiss limits

<http://data.geo.admin.ch/ch.swisstopo.swissboundaries3d-kanton-flaeche.fill/data.zip>

# > Loading Shapefiles

Lots of files inside the shapefile zip:



5 different layers:

borders (multi lines)

	Geométrie	Description
TLM_HOHEITSGRENZE	Polyligne	Limites administratives (frontières nationale, cantonale, de district, communale)
TLM_HOHEITSGEBIET	Polygone	Unités administratives de base (communes)
TLM_BEZIRKSGBIET	Polygone	Territoires des districts
TLM_KANTONSGBIET	Polygone	Territoires des cantons
TLM_LANDESGBIET	Polygone	Territoires des pays

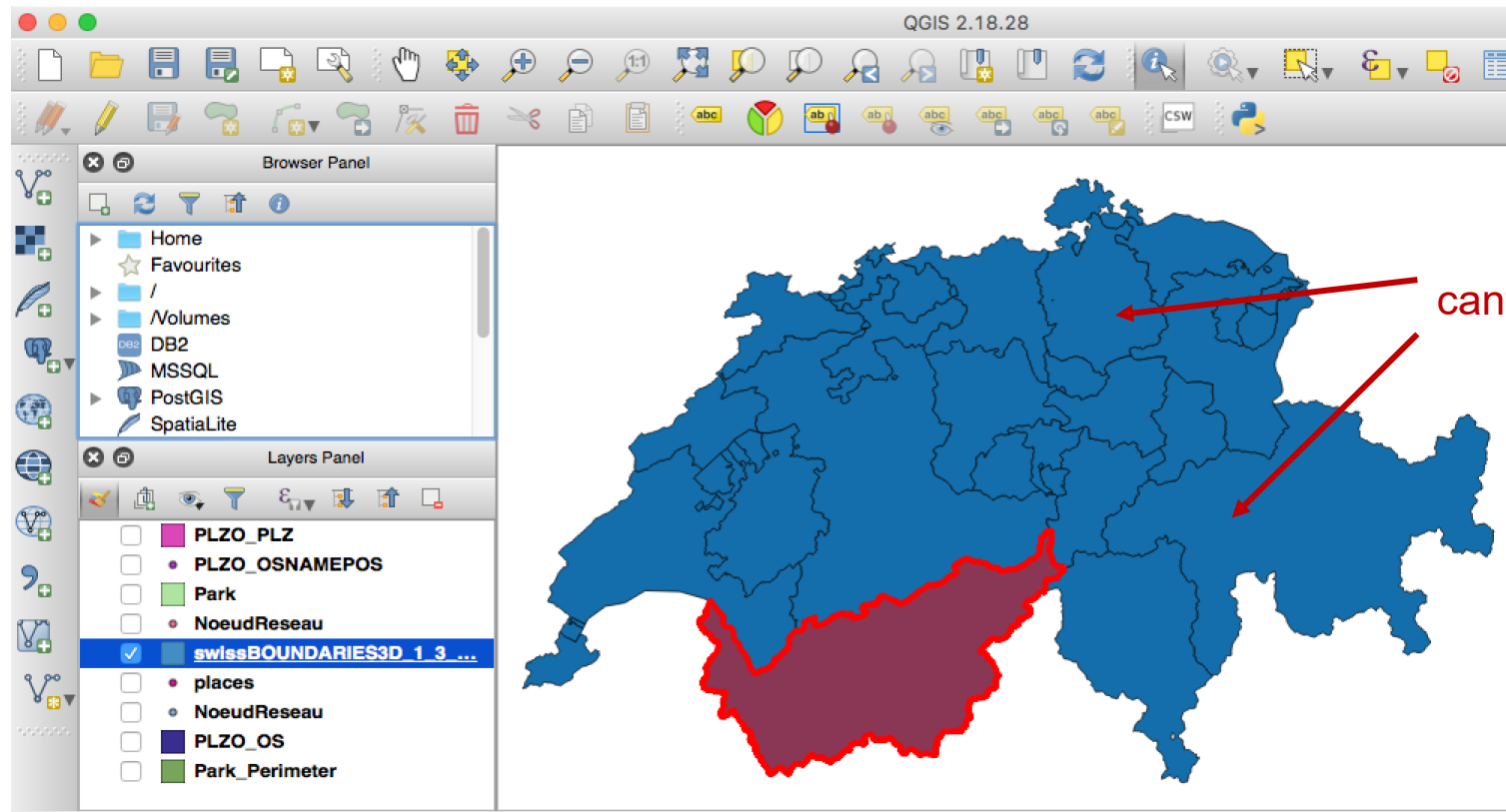
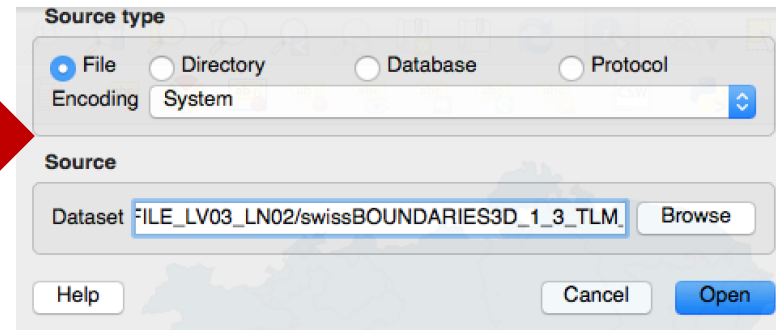
municipality polygons  
districts polygons  
canton polygons

country polygons

# > Loading Shapefiles

Open one shapefile layer in QGIS

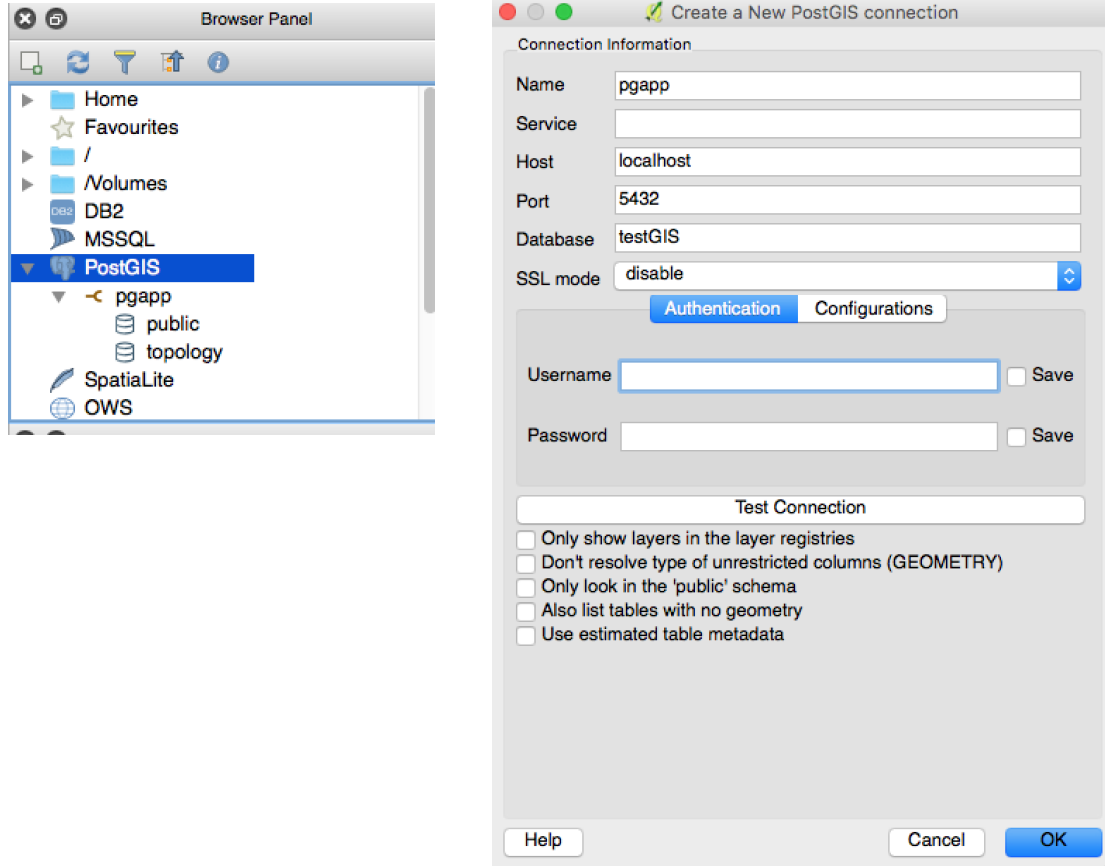
e.g. the canton polygons  
(TLM\_KANTONSgebiet)



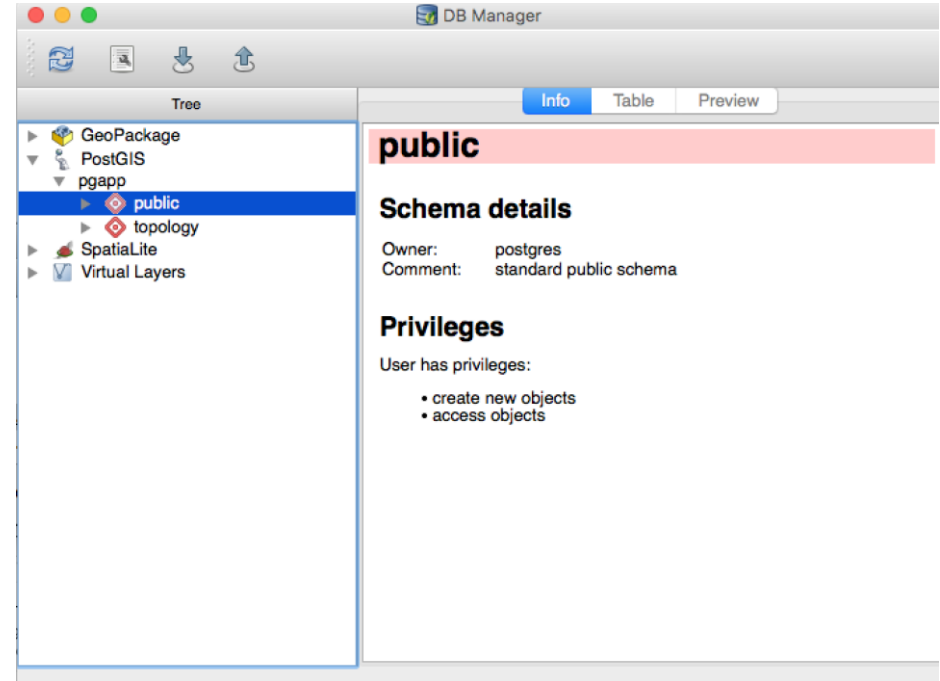
canton polygons

# > Loading Shapefiles

Connect to PostGIS from QGIS:  
PostGIS->new connection

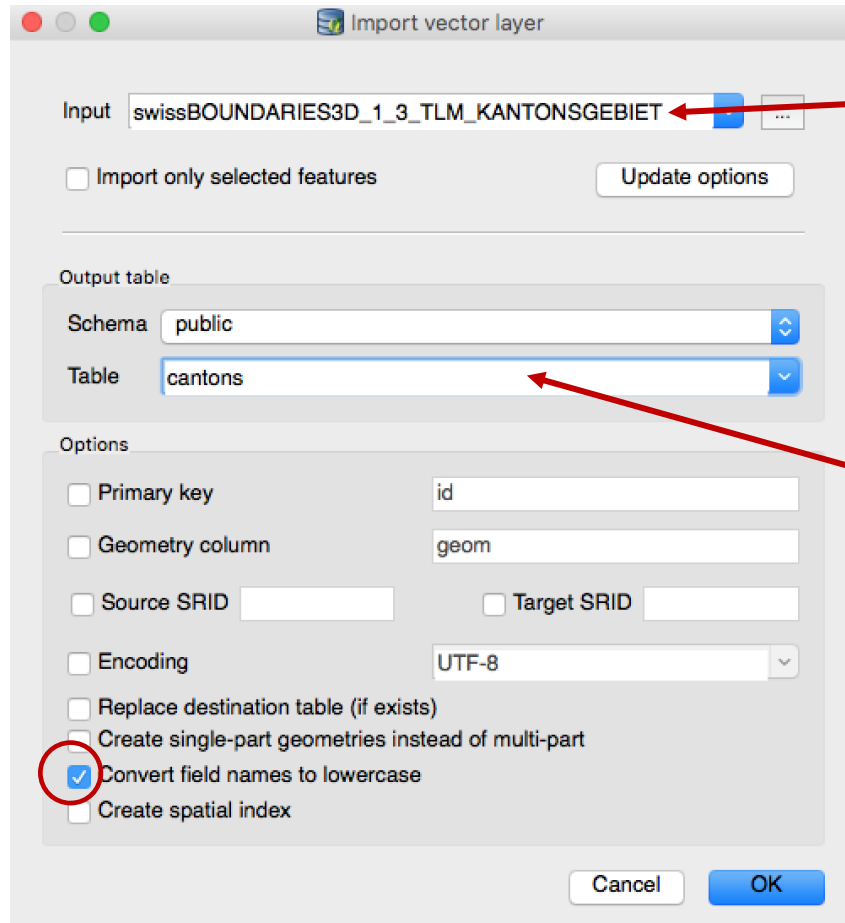


Then from the QGIS menu  
Database->DBManager



# > Loading Shapefiles

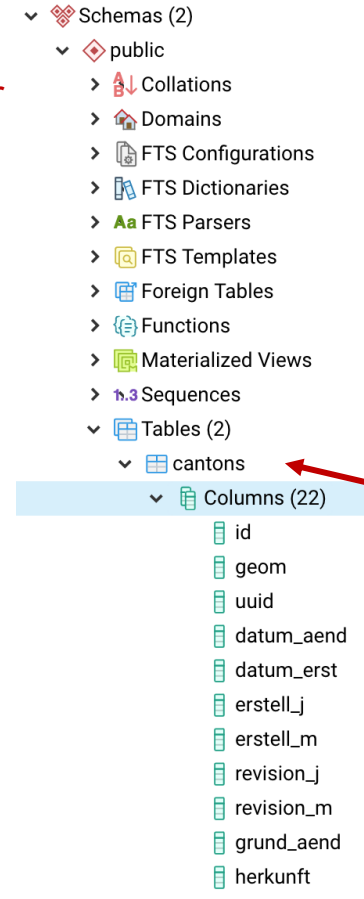
Then import the vector layer as a new table 'cantons'



shapefile layer

new table

Go back to pgAdmin, and refresh, the new table is there:

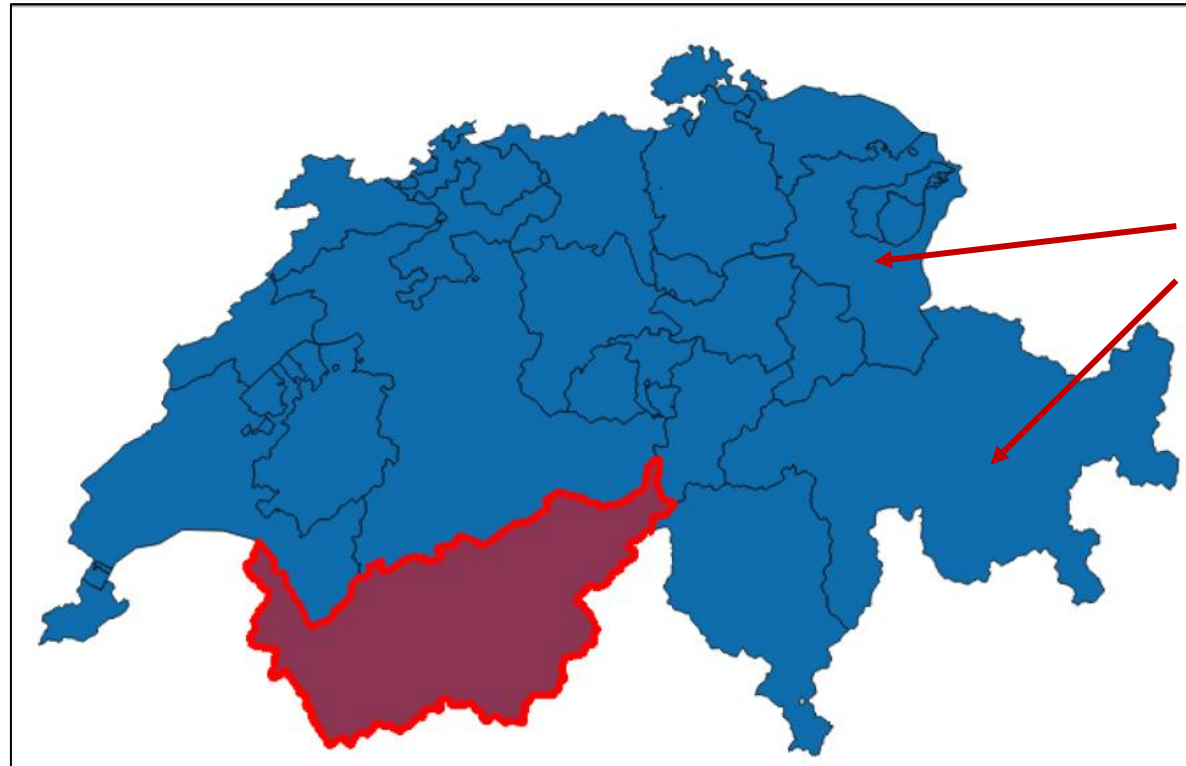


new table and imported columns

# > Loading Shapefiles

Load with command line tool shp2pgsql:

```
> shp2pgsql.exe -I -s 21781  
swissBOUNDARIES3D_1_3_TLM_KANTONSgebiet.shp cantons |  
psql.exe -h localhost -p 5433 -d postgres -U postgres
```



canton polygons

# > Loading Shapefiles

Go back to pgAdmin, and refresh, the new table is there:

The screenshot shows the pgAdmin database tree structure. Under 'Schemas (2)', the 'public' schema is expanded. Under 'Tables (2)', the 'cantons' table is expanded, showing a list of 22 columns: id, geom, uuid, datum\_aend, datum\_erst, erstell\_j, erstell\_m, revision\_j, revision\_m, grund\_aend, and herkunft. A red arrow points from the text 'new table and imported columns' to the 'cantons' table entry.

- ▼ Schemas (2)
  - ▼ public
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > Aa FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Sequences
    - ▼ Tables (2)
      - ▼ cantons
        - ▼ Columns (22)
          - id
          - geom
          - uuid
          - datum\_aend
          - datum\_erst
          - erstell\_j
          - erstell\_m
          - revision\_j
          - revision\_m
          - grund\_aend
          - herkunft

# > Querying

Get all from the cantons

```
SELECT * FROM cantons;
```

Data Output Explain Messages Notifications Geometry Viewer

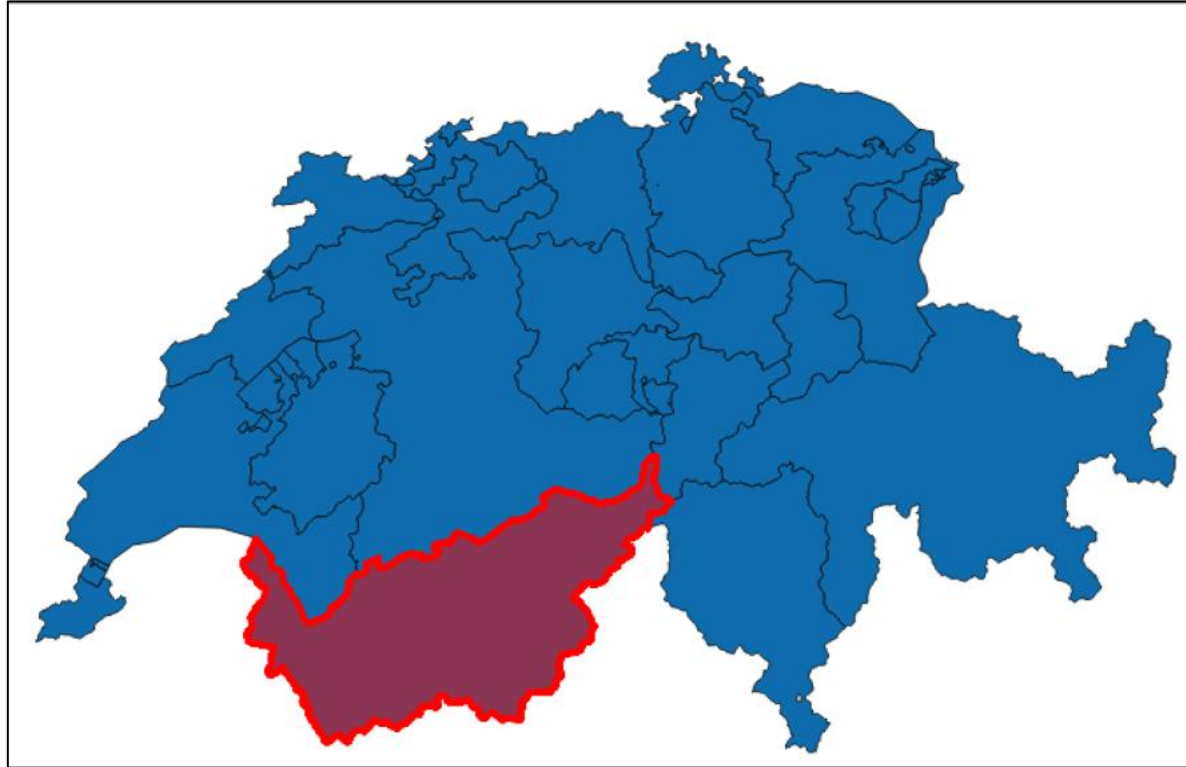
	id integer	geom geometry	uuid character varying (38)	datum_aend date	datum_erst date	erstell_j integer	erstell_m character varying (20)
1	1	01060000A0155...	{0B2364ED-49E0-4D53-A3...	2018-11-22	2012-10-26	2012	10
2	2	01060000A0155...	{DDD56CEF-0E61-4EED-8...	2018-11-22	2012-10-26	2012	10
3	3	01060000A0155...	{54B25E50-30A7-4995-AD...	2018-11-22	2012-10-26	2012	10
4	4	01060000A0155...	{921DFEF2-6D91-4CB8-9C...	2018-11-22	2012-10-26	2012	10
5	5	01060000A0155...	{95F10F52-8B2F-4D6A-AF...	2018-11-22	2012-10-26	2012	10
6	6	01060000A0155...	{05D55405-466B-4ECC-83...	2017-12-04	2012-10-26	2012	10
7	7	01060000A0155...	{FB7105B8-6D7C-4787-84...	2018-11-22	2012-10-26	2012	10
8	8	01060000A0155...	{B01E1FB4-9A9B-48AC-B...	2015-12-09	2012-10-26	2012	10
9	9	01060000A0155...	{A7C284E4-45C4-44E2-AB...	2016-12-09	2012-10-26	2012	10

```
SELECT name, id FROM cantons;
```

	name character varying (254)	id integer
1	Graubnden	1
2	Bern	2
3	Valais	3
4	Vaud	4
5	Ticino	5
6	St. Gallen	6
7	Zrich	7
8	Fribourg	8
9	Luzern	9

# > Querying

How many cantons do we have in this representation of Switzerland?



# > Querying

How many cantons in Switzerland?

```
SELECT COUNT(*) FROM cantons;
```

51

Something is wrong...

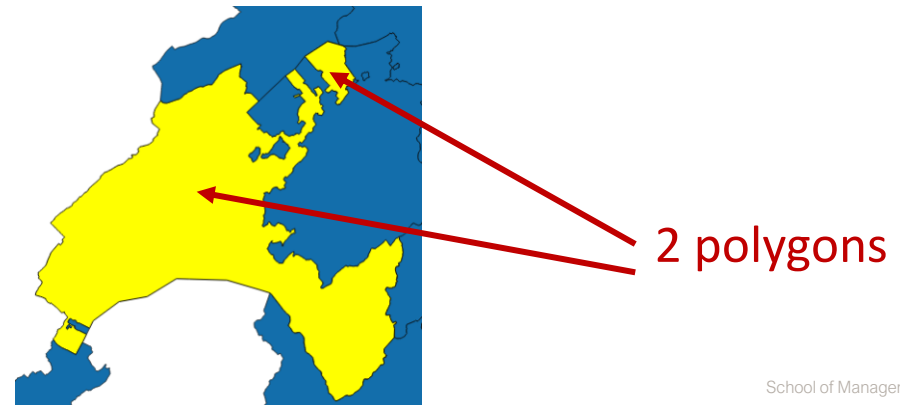
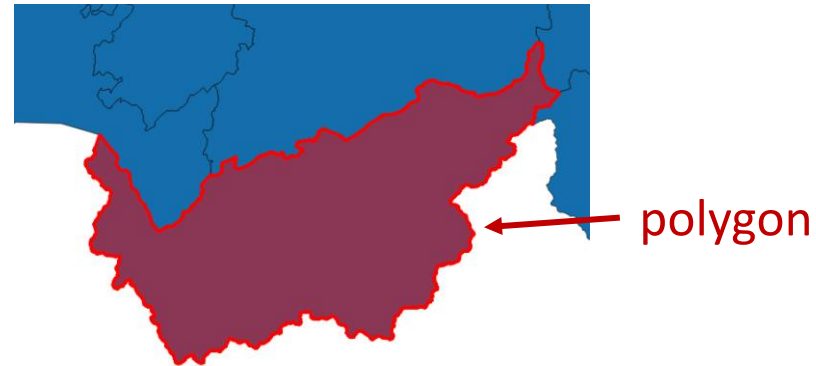
object id                      OFS canton id

```
SELECT name, id, kantonsnum FROM  
cantons WHERE name='Valais';
```

"Valais"	3	23
----------	---	----

```
SELECT name, id, kantonsnum FROM  
cantons WHERE name='Vaud';
```

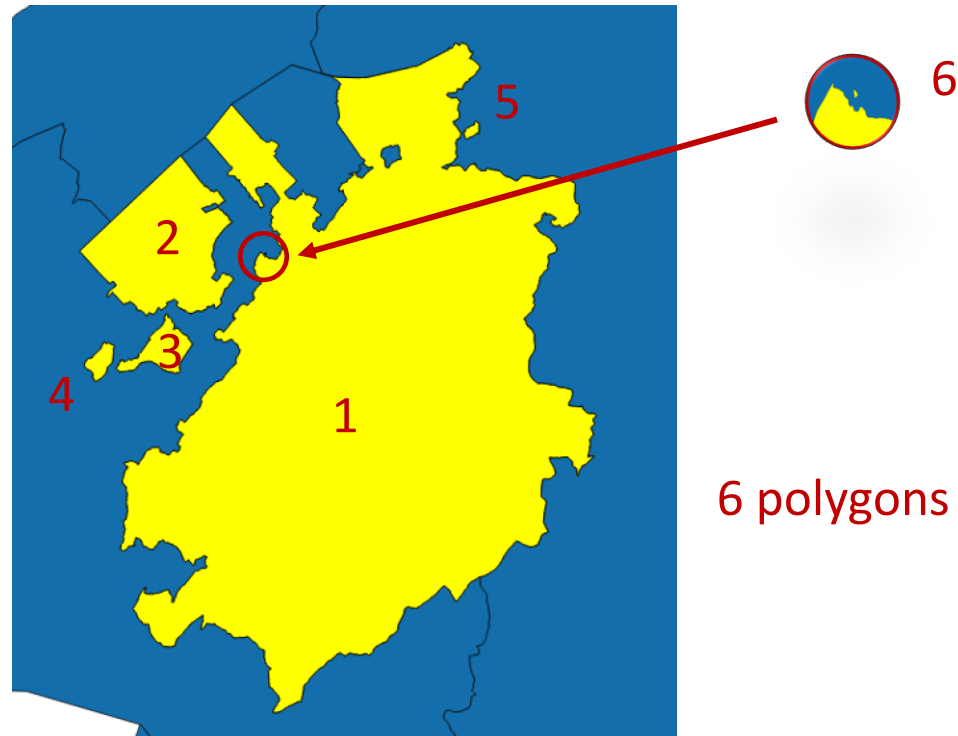
"Vaud"	4	22
"Vaud"	27	22



# > Querying

```
SELECT name, id, kantonsnum, kt_tteil  
FROM cantons WHERE name='Fribourg';
```

"Fribourg"	8	10	"1"
"Fribourg"	26	10	"2"
"Fribourg"	32	10	"3"
"Fribourg"	49	10	"6"
"Fribourg"	38	10	"4"
"Fribourg"	44	10	"5"



# > Querying

What is the area of Valais?

What is the area of Fribourg?

# > Querying

What is the area of Valais?

```
SELECT name, ST_area(geom)/1000000 AS areakm2
FROM cantons WHERE name='Valais';
```

```
"Valais"          5224.60307347788
```

```
SELECT name, ST_area(geom)/1000000 AS areakm2
FROM cantons WHERE name='Fribourg';
```

```
"Fribourg"      1520.68684827857
"Fribourg"      127.140151939312
"Fribourg"      16.613292752175
"Fribourg"      0.00728279967471749
"Fribourg"      5.59896387173263
"Fribourg"      1.36858498091714
```

Individual area of all 6 polygons of Fribourg

```
SELECT sum(ST_area(geom)/1000000) AS areakm2
FROM cantons WHERE name='Fribourg';
```

```
1671.41512462238
```

Aggregated area of Fribourg

```
SELECT kantonsfla AS areaHa
FROM cantons WHERE name='Fribourg' and kt_teil = '1';
```

# > Querying

What's the population and area of each canton?

Which cantons have the highest population density?

# > Querying

What's the population and area of each canton?

```
SELECT name, einwohnerz, kantonsfla  
FROM cantons WHERE kt_teil in ('1', '0') order by einwohnerz desc;
```

"Zrich"	1504346	"172894"
"Bern"	1031126	"595952"
"Vaud"	793129	"321201"
"Aargau"	670988	"140380"
"St. Gallen"	504686	"202820"
"Genève"	495249	"28249"
"Luzern"	406506	"149352"
"Ticino"	353709	"281216"

```
SELECT name, einwohnerz/kantonsfla as density  
FROM cantons WHERE kt_teil in ('1', '0') order by density desc;
```

"Basel-Stadt"	"52.4784844384303112"
"Genève"	"17.5315586392438670"
"Zrich"	"8.7009728504170185"
"Basel-Landschaft"	"5.5445167770973786"
"Zug"	"5.2536757005822477"
"Aargau"	"4.7797976919789144"
"Solothurn"	"3.4338920867860080"
"Thurgau"	"2.7536230426518359"
"Schaffhausen"	"2.7260572347697875"

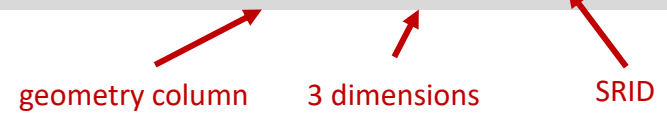
ordered by  
population density

# > PostGIS metadata

- public
  - Collations
  - Domains
  - FTS Configurations
  - FTS Dictionaries
  - FTS Parsers
  - FTS Templates
  - Foreign Tables
  - Functions
  - Materialized Views
  - Sequences
  - Tables (2)
    - cantons
    - spatial\_ref\_sys
  - Trigger Functions (2)
  - Types (19)
  - Views (4)
    - geography\_columns
    - geometry\_columns
    - raster\_columns
    - raster\_overviews

```
select * from geometry_columns;
```

```
"testGIS" "public" "cantons" "geom" 3 21781 "MULTIPOLYGON"
```



```
select * from spatial_ref_sys where srid=21781;
```

```
"21781" "EPSG" 21781 "PROJCS["CH1903 / LV03",GEOGCS["CH1903",DATUM["CH1903",SPHEROID["Bessel 1841",6377397.155,299.1528128,AUTHORITY["EPSG","7004"]],TOWGS84[674.374,15.056,405.346,0,0,0,0],AUTHORITY["EPSG","6149"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4149"]],PROJECTION["Hotine_Oblique_Mercator_Azimuth_Center"],PARAMETER["latitude_of_center",46.95240555555556],PARAMETER["longitude_of_center",7.439583333333333],PARAMETER["azimuth",90],PARAMETER["rectified_grid_angle",90],PARAMETER["scale_factor",1],PARAMETER["false_easting",600000],PARAMETER["false_northing",200000],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Y",EAST],AXIS["X",NORTH],AUTHORITY["EPSG","21781"]]"
```

```
"+proj=somerc +lat_0=46.95240555555556 +lon_0=7.439583333333333 +k_0=1 +x_0=600000 +y_0=200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no_defs "
```

## > Spatial operations

- $ST\_Contains(A, B)$ : no points of B lie in the exterior of A, and at least one point of the interior of B lies in the interior of A.
- $ST\_Crosses(A, B)$ : the supplied geometries have some, but not all, interior points in common.
- $ST\_Disjoint(A, B)$ : the Geometries do not share any space together.
- $ST\_Distance(A, B)$ : 2-dimensional cartesian minimum distance
- $ST\_DWithin(A, B, radius)$ : the geometries are within the specified distance of one another.
- $ST\_Equals(A, B)$ : the given geometries represent the same geometry
- $ST\_Intersects(A, B)$ : the Geometries share any portion of space
- $ST\_Overlaps(A, B)$ : the Geometries share space, are of the same dimension, but are not completely contained by each other.
- $ST\_Touches(A, B)$ : the geometries have at least one point in common, but their interiors do not intersect.
- $ST\_Within(A, B)$ : the geometry A is completely inside geometry B

# > Spatial queries

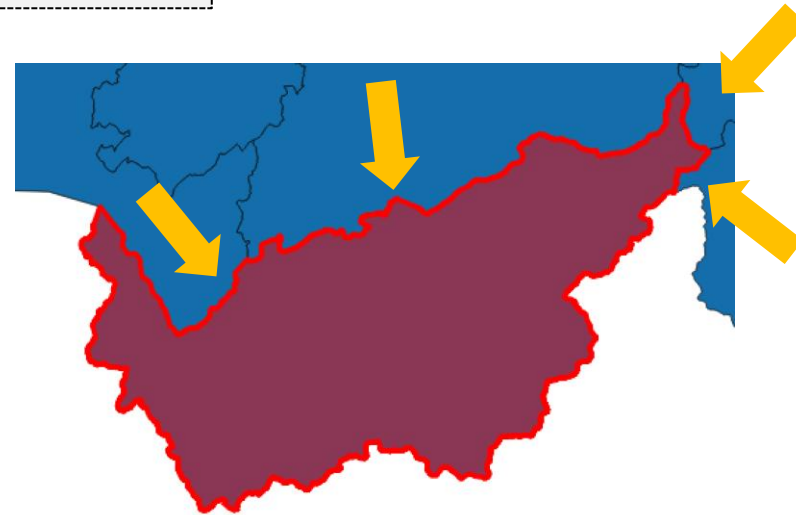
Which cantons border (touch) the canton of Valais?

# > Spatial queries

Which cantons border (touch) the canton of Valais

```
SELECT c2.name, c2.kt_teil
FROM cantons c1, cantons c2
WHERE ST_Touches(c1.geom, c2.geom)
and c1.name='Valais' and c1.id<>c2.id;
```

"Bern"	"1"
"Vaud"	"1"
"Ticino"	"0"
"Uri"	"0"

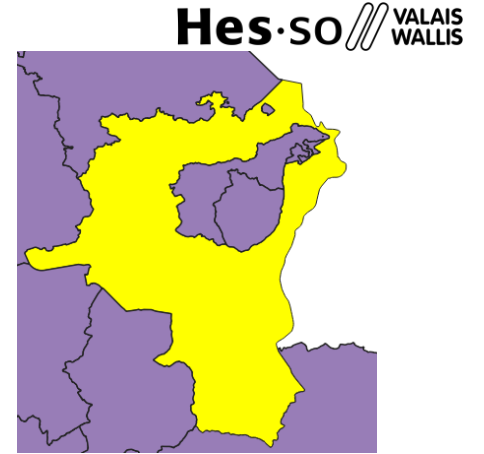


# > Spatial queries

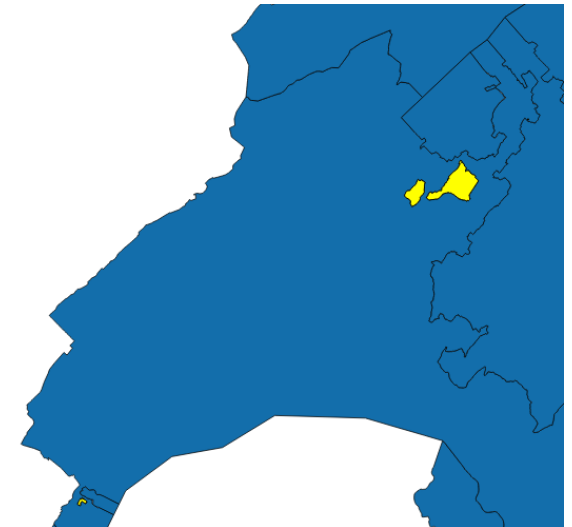
Which are the top 5 cantons with smaller borders?

# > Spatial queries

Which cantons are contained in St Gallen?



Which cantons are (partially) contained in Vaud?



# > Spatial queries

Which cantons are (partially) contained in Vaud?

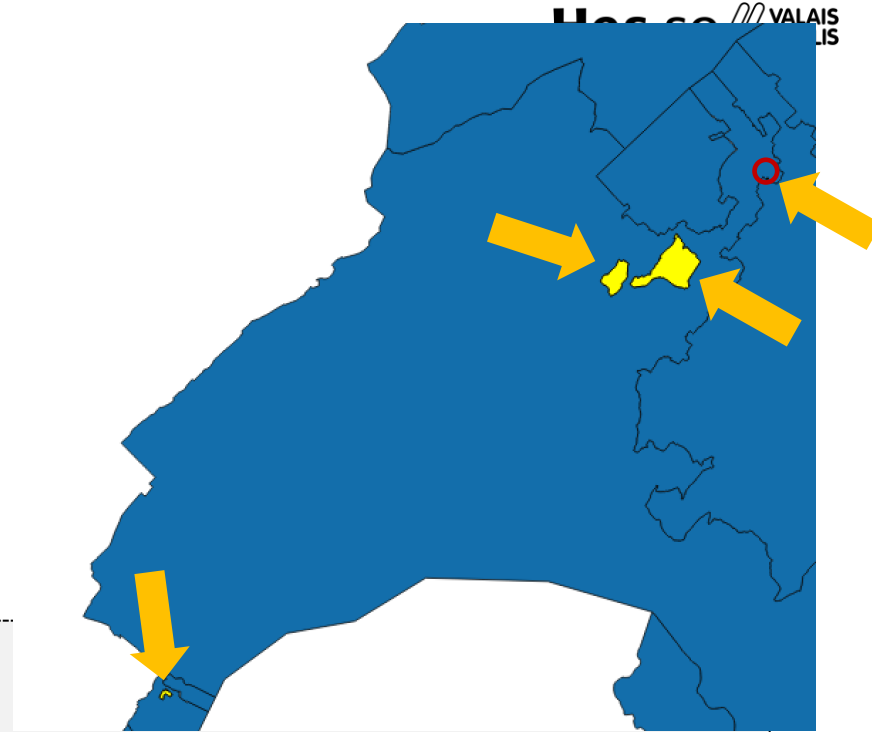
```
select c2.name, c2.kt_teil
from cantons c1, cantons c2
where ST_Contains(c1.geom, c2.geom) and
c1.name='Vaud' and c1.id<>c2.id;
```

No results!

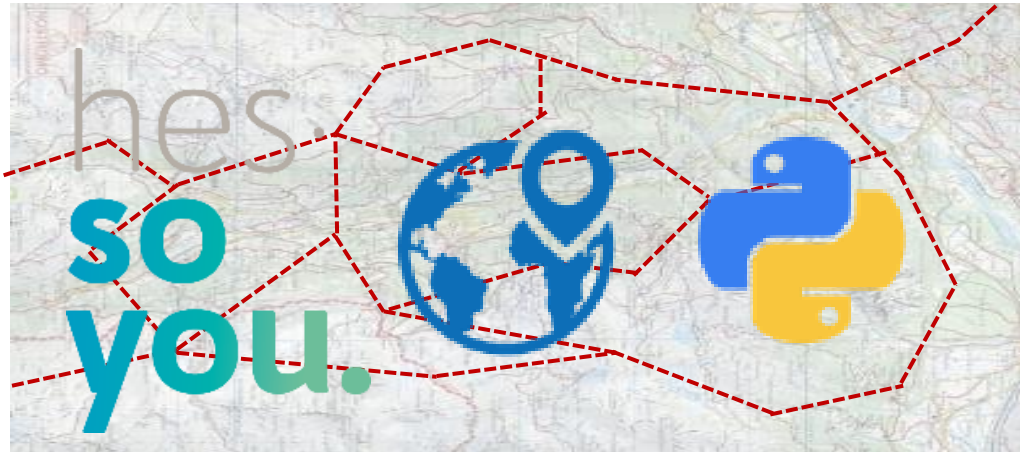
```
select c2.name, c2.kt_teil
from cantons c1, cantons c2
where ST_Contains(
  ST_Polygon(ST_ExteriorRing(ST_GeometryN(c1.geom, 1)), 21781),
  c2.geom)
and c1.name='Vaud' and c1.id<>c2.id;
```

Create polygon with the exterior of the canton border

SRID



```
"Fribourg" "3"
"Fribourg" "6"
"Fribourg" "4"
"Genève" "3"
```



School of Management  
Route de la Plaine 2  
3960 Sierre

[hevs.ch/heg](https://hevs.ch/heg)



Thank you for your attention.

swissuniversities

