# NRW-Förderprojekt: OpenHygrisC Data Processing for Education (OpenHyPE)

- Gefördert durch das Ministerium für Landwirtschaft und Verbraucherschutz NRW (MLV)
- Laufzeit: 15.12.2021 - 31.06.2023

## Table of Contents

## 1. Zusammenfassung

Das Landesamt für Natur-, Umwelt- und Verbraucherschutz (LANUV) stellt umfangreiche Grundwassermessdaten bezüglich Menge und Qualität als offene Daten bereit. Die landeseigenen Internet-Datenplattformen ELWAS und HygrisC sind zwar optimiert, die Daten zu finden und herunterzuladen, aber sie sind nicht leicht zu bedienen und bieten selbst nur sehr einge-schränkte Analysemöglichkeiten der raum- und zeitbezogenen Daten. Deshalb sollen im Projekt OpenHyPE ein erster Grundstock praxisorientierter Lehrmaterialien entwickelt werden, die beschreiben, wie man selbst eine Geodatenbank – OpenHyPE DB genannt – aufbauen kann, die dann mit den unter dem

Namen OpenHgrisC bereitgestellten offenen Grundwasserdaten gefüllt wird.

Die OpenHyPE DB basiert auf ProstgreSQL/PostGIS und bildet das Zentrum eines Systems zur Analyse und Darstellung von Umweltdaten. Es wird beschrieben, wie das geographische Informationssystem QGIS sowie die Programmiersprache Python genutzt werden können, um die Daten zu selektieren, zu analysieren und in Form von Karten oder Zeitreihen darzustellen. Alle verwendeten Software-Produkte sind „Free and Open Source Software“ (FOSS). Das offene Lehrmaterial wird als „Open Educational Resource“ (OER) veröffentlicht und richtet sich aufgrund des abgestuften Schwierigkeits-grads sowohl an Studierende als auch Schülerinnen und Schüler des Landes Nordrhein-Westfalen und darüber hinaus. Die initiale Anschubfinanzierung des OpenHyPE-Projekts soll dazu genutzt werden, den wertvollen Umweltdatenbestand des Landes bei jungen Menschen bekannter zu machen und durch die Verbindung von Umweltwissenschaften mit Informatik einen Beitrag zur interdisziplinären MINT-Förderung allgemein sowie zur Bildung für nachhaltige Entwicklung (BNE) im Besonderen zu leisten.

## 1.1 Einführung / Problembeschreibung

Das Land Nordrhein-Westfalen (NRW) betreibt über das LANUV umfassende und professionelle Messnetze zur Erfassung von Umweltdaten. Im Rahmen von Open.NRW und angetrieben durch die INSPIRE-Direktive der Europäischen Union sowie weitere Direktiven wie zum Beispiel die EU-Wasserrahmenrichtlinie (WRRL) werden vom Land NRW umfangreiche Daten-Produkte offen zugänglich und frei nutzbar auf verschiedenen Plattformen zur Verfügung gestellt (Free and Open Data).

Das Land NRW ist in Deutschland ein Vorreiter bei der Bereitstellung von offenen und (kosten)freien Geodaten. Diese Daten sind ein wahrer Schatz und bilden die Grundlage für potentiell massiven Erkenntnisgewinn im Bereich Umwelt- und Naturschutz. Trotzdem scheint es so zu sein, dass nur ein vergleichsweise kleiner Personenkreis dieses Potential wirklich nutzt. Deshalb hat sich das Projekt OpenHyPE zur Aufgabe gemacht, diesen Datenbestand in die Hochschullehre einzubauen und entsprechendes frei zugängliches Lehrmaterial zu entwickeln, dass nicht nur von Studierenden sondern auch zum Teil von Schülerinnen und Schülern genutzt werden kann, um die Grundzüge der Umweltdatenverarbeitung zu lernen. Die Anschubfinanzierung soll genutzt werden, um erste Schritte der Entwicklung solchen Training-Materials umzusetzen.

Wir verfolgen dabei das Paradigma des „Problem based learning“: Die notwendigen Kenntnisse und Fähigkeiten werden anhand einer konkreten gesellschaftlich relevanten Problemstellung identifiziert und vermittelt. Die Lösung der als bedeutsam erkannten Fragestellung ist die Motivation für das Lernen.

Am Anfang wollen wir das Material anhand des Problemfelds „Grundwasserschutz“ entwickeln. Das Ministerium für Umwelt, Landwirtschaft, Natur- und Verbraucherschutz NRW (MULNV) betreibt über den „Landesbetrieb Information und Technik Nordrhein-Westfalen“ (IT.NRW) ein eigenes wasserbezogenes Datenportal namens ELWAS-WEB. Darin werden auch Daten der landesweiten Grundwasserdatenbank HygrisC vorgehalten. ELWAS und HygrisC bieten Außenstehenden nur eingeschränkte Möglichkeiten der explorativen Datenanalyse. Aus Sicht des Usability Engineering, das sich mit der Anwenderfreundlichkeit technischer Systeme befasst, sind hinsichtlich der Benutzbarkeit sowie der Datenanalyse-Möglichkeiten Verbesserungen wünschenswert, denn gerade die explorative Datenanalyse und das Data Mining helfen, Strukturen und Zusammenhänge zwischen den Daten zu erkennen. ELWAS und HygrisC sind deshalb nur bedingt geeignet, Grundlagen der Umweltdatenanalyse zu vermitteln, können aber im Unterricht als Begleitmaterial einfließen.

Auf dem Portal OpenGeodata.NRW werden umfangreiche Daten mit Raumbezug – auch Geodaten genannt – zur Verfügung gestellt, die oft einen Zeitbezug haben, wie z.B. Landnutzungsänderungen oder Messdatenreihen zur Wasserqualität. Dort liegen auch Auszüge der HygrisC-Grundwasser-Datenbank des Landes NRW, die unter dem Namen OpenHygrisC veröffentlicht werden. Diese Grundwasserdaten können in idealer Weise als Grundlage zum Aufbau einer eigenen Umweltdatenbank dienen, anhand derer die Lernenden Konzepte des Datenmanagements und der Datenanalyse kennenlernen.

## 1.2 Projektziele

Folgende Komponenten sollen realisiert werden:

- Entwicklung der OpenHyPE Geodatenbank auf Basis von PostgreSQL/PostGIS zur Verwaltung raum- und zeitbezogener Daten zu Grundwasserqualität und -menge
- Problembezogenes freies Online-Kursmaterial (OER), Tutorials, Video-Tutorials, Anleitungen, Programm-Code, unter Verwendung von Free and Open Source Software (FOSS):
    - Vorstellung des Landesamts für Natur, Umwelt- und Verbraucherschutz (LANUV)
    - Einführung in den Grundwasserschutz
    - Einführung in das Geographische Informationssystem QGIS
    - Einführung in die relationale Datenbank PostgreSQL und die Abfragesprache SQL
    - Einführung in die Geodatenbank-Erweiterung PostGIS
    - Einführung in die Verarbeitung von Geodaten mit der Programmiersprache Python
    - Installation des OpenHyPE Datenbank-Managementsystems
    - Diskussion des Datenmodells und Hochladen der OpenHygrisC-Daten des LANUV
    - Automatisches Erstellen von Diagrammen zu Zeitreihen der Wasserqualität
    - Automatisches Erstellen von Karten zur Grundwasserchemie
    - Erstellen einfacher Dashboards mit interaktiven Online-Grafiken und -Karten
    - Einführung in Data Mining (Deskriptive Statistik, Suchen nach Zusammenhängen)

# 2.Implementation

## 2.1 Data flow

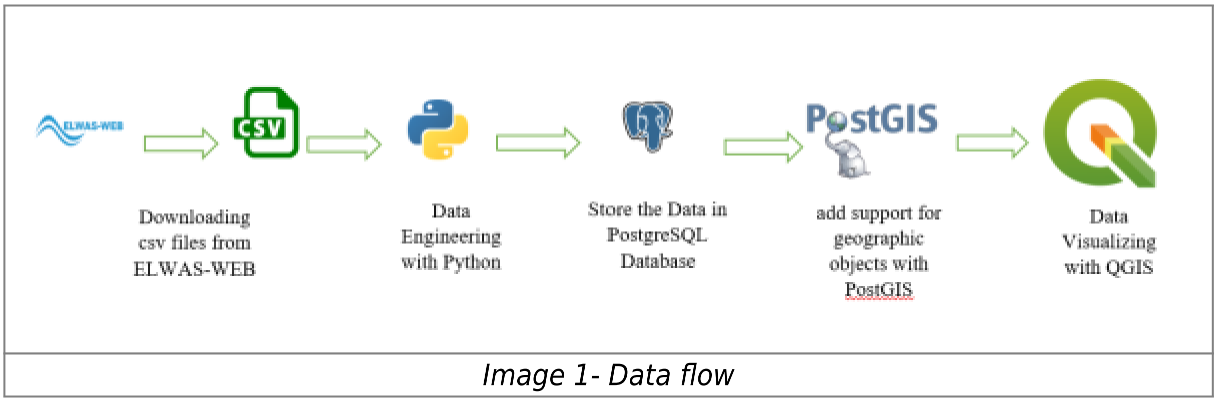

ELWAS-WEB
CSV
PostGIS
Downloading csv files from ELWAS-WEB
Data Engineering with Python
Store the Data in PostgreSQL Database
add support for geographic objects with PostGIS
Data Visualizing with QGIS

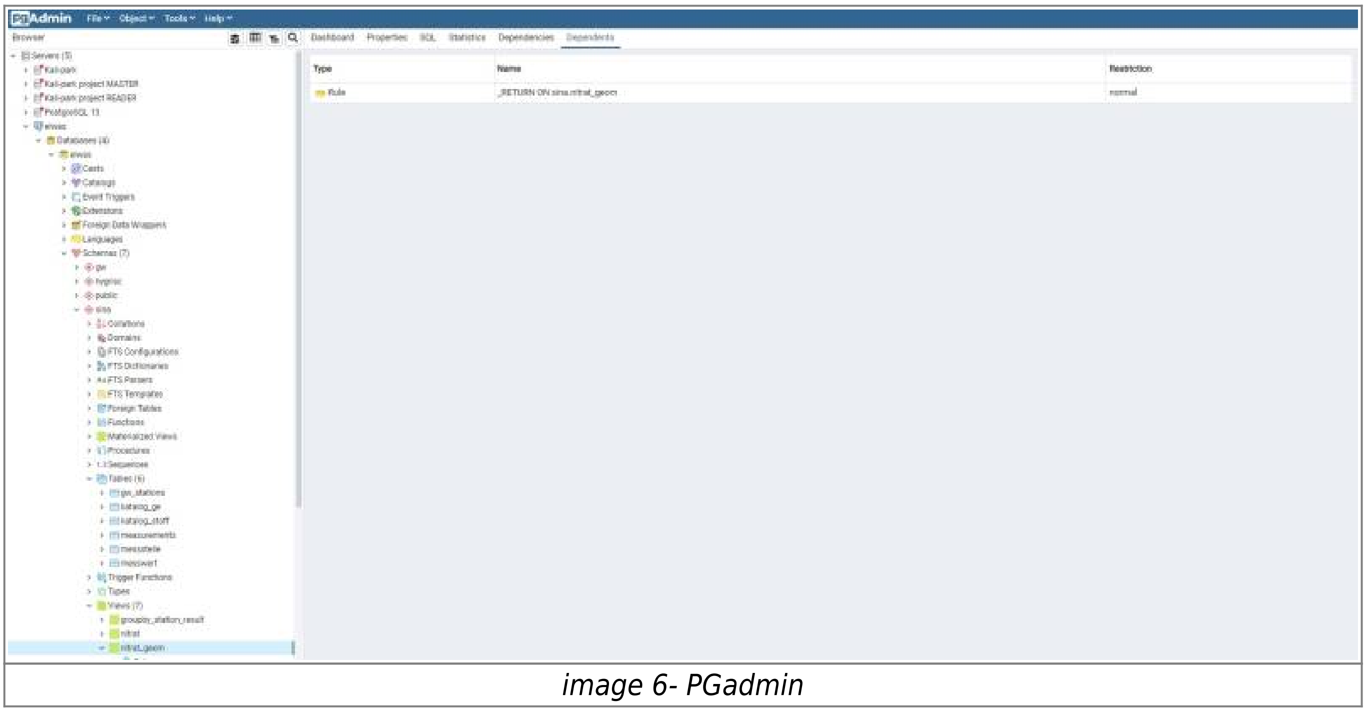
*Image 1- Data flow*

# 2.2 PostgreSQL/PostGIS

PostgreSQL is also known as Postgres is a free and open-source relation database management system and according to the www.pgadmin.org, “Postgres Database is the most advanced open-source database in the world”. We can store the time series data as well as geometry data in Postgres. In this project, we have used PGadmin which is the most popular and powerful open-source administration platform for Postgres Database.

**PostGIS:** PostGIS is technically an extension of the PostgreSQL database which helps to add support for geographical objects. PostGIS is open-source and free to use.

The below image shows the PGadmin tool.



| Type | Name | Restriction |
|---|---|---|
| Rule | _RETURN ON sina.nitrat_geom | normal |

*image 6- PGadmin*

Watching the below videos to understand how we can create schemas and tables in the PostgreSQL database.

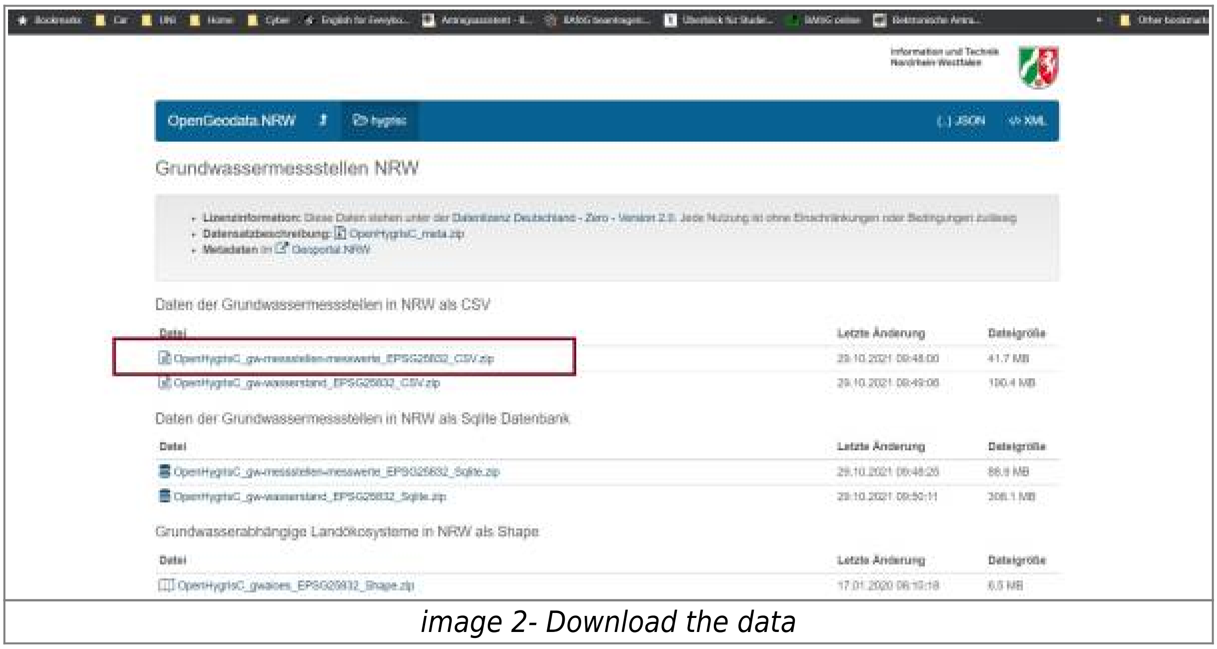
| | |
|---|---|
| | |
| *video1- Create Schema in PGadmin* | *video2- Create tables in PGadmin* |

Create a database and schema based on the above video.

# 2.3 Data Engineering

## 2.3.1 Downloading the data

In the first step, The data must be downloaded from here. The first zip file should be downloaded and extracted which consists of four CSV files and one instruction. image 2 shows which zip file should be downloaded.



Grundwassermessstellen NRW

- Lizenzinformation: Diese Daten stehen unter der Datenlizenz Deutschland - Zero - Version 2.0. Jede Nutzung ist ohne Einschränkungen oder Bedingungen zulässig
- Datensatzbeschreibung: OpenHygrisC_meta.zip
- Metadaten im Geoportal.NRW

Daten der Grundwassermessstellen in NRW als CSV

| Datei | Letzte Änderung | Dateigröße |
|---|---|---|
| OpenHygrisC_gw-messstellen-messwerte_EPSG25832_CSV.zip | 29.10.2021 09:48:00 | 41.7 MB |
| OpenHygrisC_gw-wasserstand_EPSG25832_CSV.zip | 29.10.2021 09:49:06 | 190.4 MB |

Daten der Grundwassermessstellen in NRW als Sqlite Datenbank

| Datei | Letzte Änderung | Dateigröße |
|---|---|---|
| OpenHygrisC_gw-messstellen-messwerte_EPSG25832_Sqlite.zip | 29.10.2021 09:48:26 | 88.8 MB |
| OpenHygrisC_gw-wasserstand_EPSG25832_Sqlite.zip | 29.10.2021 09:50:11 | 308.1 MB |

Grundwasserabhängige Landökosysteme in NRW als Shape

| Datei | Letzte Änderung | Dateigröße |
|---|---|---|
| OpenHygrisC_gwaoes_EPSG25832_Shape.zip | 17.01.2020 06:10:18 | 6.5 MB |

*image 2- Download the data*

After extracting the above zip file then we can consider the four CSV files and instruction file which is highly recommended to read first.

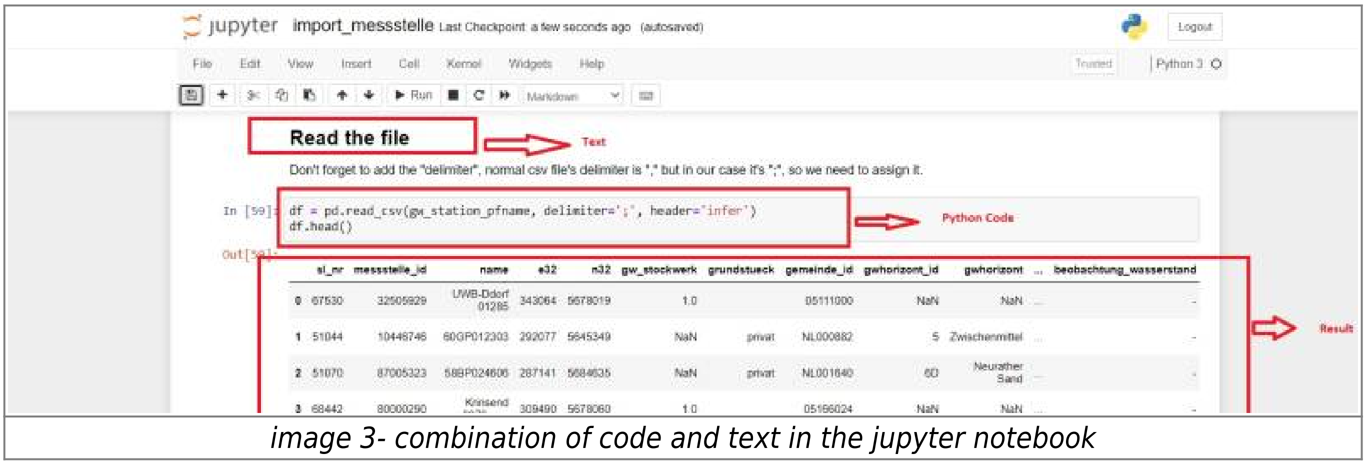
## 2.3.2 Python

Python is an object-oriented programming language that helps the programmer to write logic and clear code for small as well as large projects.

Python is used in several ways such as:

- AI and machine learning
- Data analytics
- Data visualisation
- Programming applications

In this project, we have used Python for data engineering, data pre-processing and data analysis. Jupyter Notebook is used in this project to write python codes. The Jupyter Notebook is an open-source web application that data scientists simply can write the code and make it easier to document it. Simply we can combine python codes, text, images, comments and the result of the codes on the same page. The below image shows how code, text and the result of the code can be seen on a single page.



**Read the file**

Don't forget to add the "delimiter", normal csv file's delimiter is "," but in our case it's ";", so we need to assign it.

```
In [59]: df = pd.read_csv(gw_station_pfname, delimiter=';', header='infer')
         df.head()
```

```
Out[59]:
   sl_nr  messstelle_id        name     e32      n32  gw_stockwerk  grundstueck  gemeinde_id  gwhorizont_id    gwhorizont  ...  beobachtung_wasserstand
0  67530       32505929    UWB-Ddorf  343064  5678019           1.0                  05111000            NaN           NaN  ...                        -
                               01285
1  51044       10448746  60GP012303  292077  5645349           NaN       privat     NL000882              5  Zwischenmittel  ...                        -
2  51070       87005323  58BP024606  287141  5684635           NaN       privat     NL001640             6D      Neurather  ...                        -
                                                                                                                   Sand
3  68442       80000290    Krinsend  309490  5678060           1.0                  05166024            NaN           NaN  ...                        -
```

image 3- combination of code and text in the jupyter notebook

## 2.3.3 Anaconda

Anaconda is an open-source distribution for python and R. It is used for data science, machine learning, deep learning, etc. With the availability of more than 300 libraries for data science, it becomes fairly optimal for any programmer to work on anaconda for data science Anaconda is used in this project. An environment on Anaconda has been created to install all the packages needed for this project.

VideoXXX- How to install Anaconda on Windows

VideoXXX- How to install Anaconda on Mac OS X

**Environment in Anaconda:** A conda environment is a directory that contains a specific collection of conda packages that are used in the project.

**How to create Conda environment:** The below video shows how to create a Conda environment, how to activate it, how to install different packages on the environment and how to deactivate the environment.

*VideoXXX- How to create Conda environment*

**Python packages:** a package is a collection of modules that have the same aim together. These modules are like functions and can help us to write code easily. The packages which are needed should be installed and imported before use.

To get more details about conda environment, I highly recommend visiting the below webpage.

https://towardsdatascience.com/manage-your-python-virtual-environment-with-conda-a0d2934d5195

**openhype environment:** In this project, "openhype" environment has been created in order to install all the necessary packages which are needed. openhype environment has been created based on the above video.

Several packages are related to data science but in this project, we have used the below packages. There are two ways to install the below packages:

**install a package manually:** In this way you need to install each package manually into openhype environment by the command prompt.

- **pandas:** pandas is one of the most important and popular packages of python among Data scientists for data manipulation and analysis. We can do Data cleaning, Data pre-processing, fill the data, visualize the data, Data inspection, Loading and saving data and much more.

In this project, we have used pandas, to read the CSV files, clean the data and do data engineering. The below code should be written into the openhype environment on Anaconda prompt

```
conda install pandas
```

- **sqlalchemy:** SQLAlchemy package is like a bridge between python programming language and database. we have used this package to connect to our database.

```
conda install sqlalchemy
```

- **psycopg2:** With the help of this package, our python program can communicate with the PostgreSQL database.

```
conda install psycopg2
```

- **geopandas:** With the help of this package we are able to work with geospatial data in Python. according to the geopandas website " It

is extends the data types used by pandas to allow spatial operations on geometric types".

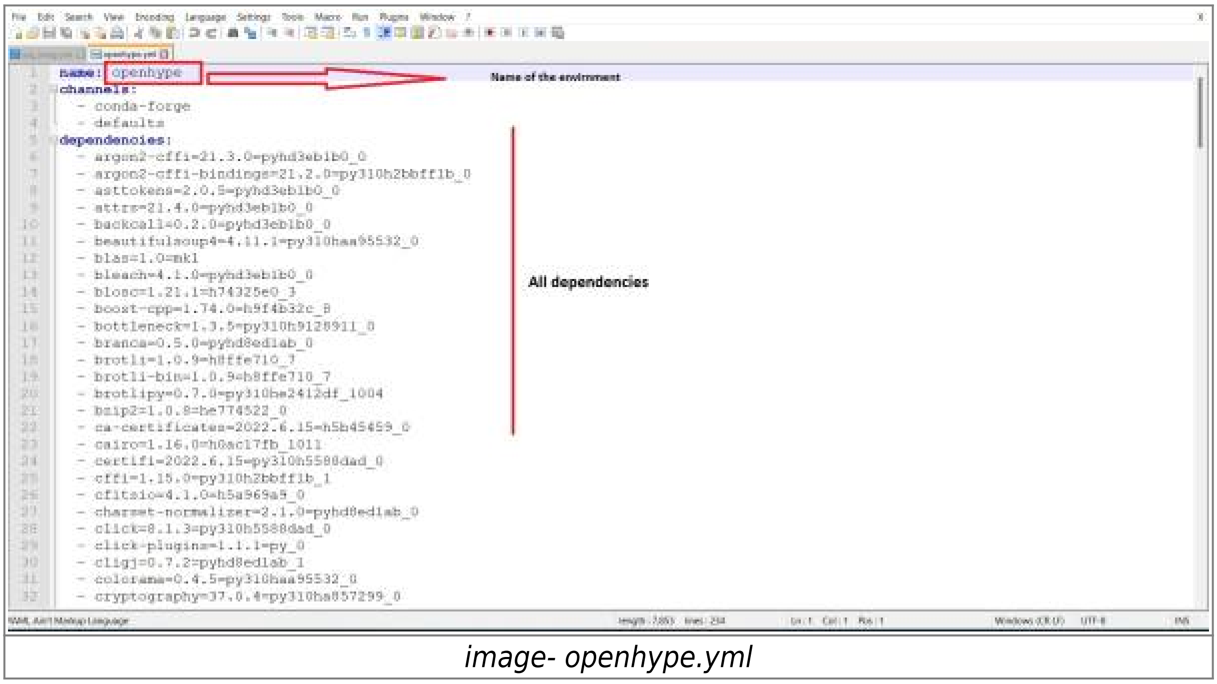
```
conda install --channel conda-forge geopandas
```

Some packages need to specify the channel to install and that's why in the above code we have specified the channel.

```
conda install jupyter notebook
```

All the above packages must be installed into the openhype environment with Anaconda prompt.

**Load all the packages into the environment by a yml file:** In this way, you need to create an environment (which in our project is Openhype) and load a yml file which consists of the all packages that are needed for this project. The below image shows the content of our openhype.yml file.



```
name: openhype
channels:
  - conda-forge
  - defaults
dependencies:
  - argon2-cffi=21.3.0=pyhd3eb1b0_0
  - argon2-cffi-bindings=21.2.0=py310h2bbff1b_0
  - asttokens=2.0.5=pyhd3eb1b0_0
  - attrs=21.4.0=pyhd3eb1b0_0
  - backcall=0.2.0=pyhd3eb1b0_0
  - beautifulsoup4=4.11.1=py310haa95532_0
  - blas=1.0=mkl
  - bleach=4.1.0=pyhd3eb1b0_0
  - blosc=1.21.1=h74325e0_3
  - boost-cpp=1.74.0=h9f4b32c_8
  - bottleneck=1.3.5=py310h9128911_0
  - branca=0.5.0=pyhd8ed1ab_0
  - brotli=1.0.9=h8ffe710_7
  - brotli-bin=1.0.9=h8ffe710_7
  - brotlipy=0.7.0=py310he2412df_1004
  - bzip2=1.0.8=he774522_0
  - ca-certificates=2022.6.15=h5b45459_0
  - cairo=1.16.0=h0ac17fb_1011
  - certifi=2022.6.15=py310h5588dad_0
  - cffi=1.15.0=py310h2bbff1b_1
  - cfitsio=4.1.0=h5a969a9_0
  - charset-normalizer=2.1.0=pyhd8ed1ab_0
  - click=8.1.3=py310h5588dad_0
  - click-plugins=1.1.1=py_0
  - cligj=0.7.2=pyhd8ed1ab_1
  - colorama=0.4.5=py310haa95532_0
  - cryptography=37.0.4=py310ha857299_0
```

*image- openhype.yml*

With the below code, we are able to create an environment based on the yml file. Please be aware that this yml file should be in the same directory as our anaconda directory, otherwise you need to write the full path of the file. openhype.yml can be found from here

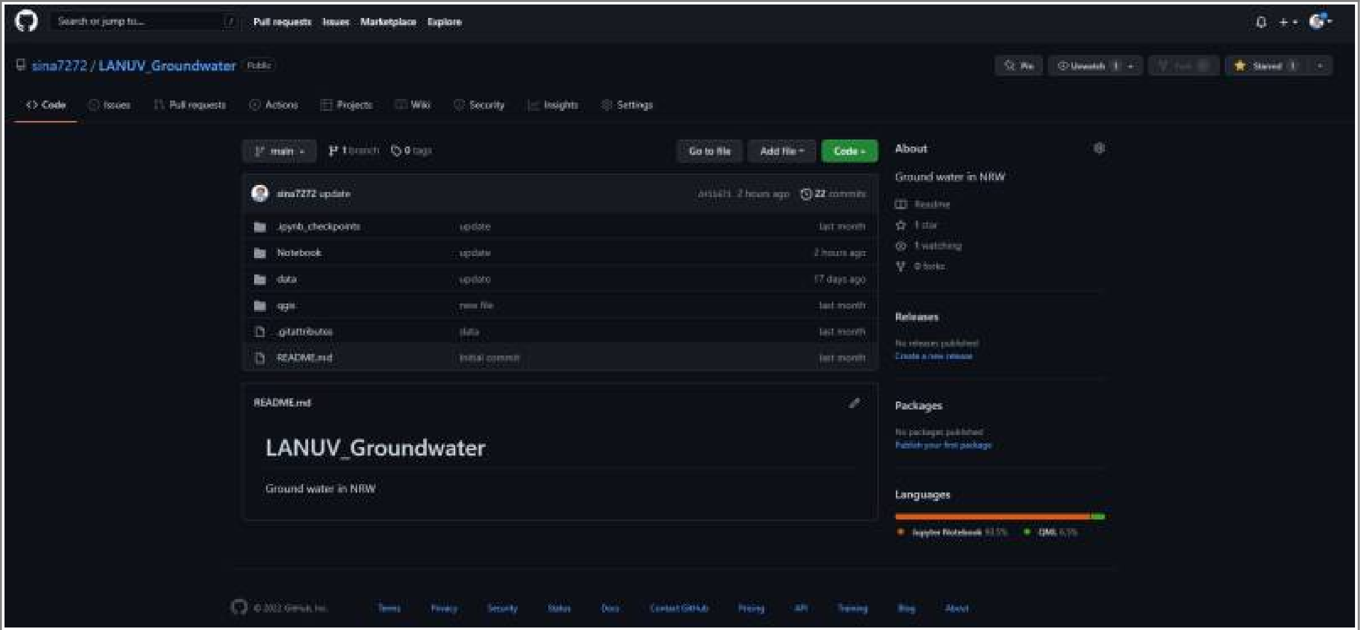
```
conda env create -f openhype.yml
```

How to create a yml file for the environment: With the below code you can export the yml file from the existing environment.

```
conda env export > openhype.yml
```

Since we have downloaded the four CSV files in the previous chapter, now is the time to read our CSV files and start to clean them in order to make them ready to import into our database.

please refer to our Notebook section of the code on Github (the below image).



sina7272 / LANUV_Groundwater Public

sina7272 update 2 hours ago 22 commits

.ipynb_checkpoints update last month
Notebook update 2 hours ago
data update 17 days ago
qgis new file last month
.gitattributes data last month
README.md Initial commit last month

README.md

LANUV_Groundwater

Ground water in NRW

Ground water in NRW

No releases published
Create a new release

No packages published
Publish your first package

© 2022 GitHub, Inc.

*image 5- Github*

There below four notebooks should be run separately, in order to import data into the database.

- import_gemeinde.ipynb:

In this notebook, we will import the data of all geminde into the database.

- import_katalog_stoff.ipynb:

In this notebook, we will import the data of all the catalogue substances into the database.

- import_messstelle.ipynb:

In this notebook, we will import the data of all stations into the database.

- import_messwert.ipynb:

In this notebook, we will import the data of all values into the database.

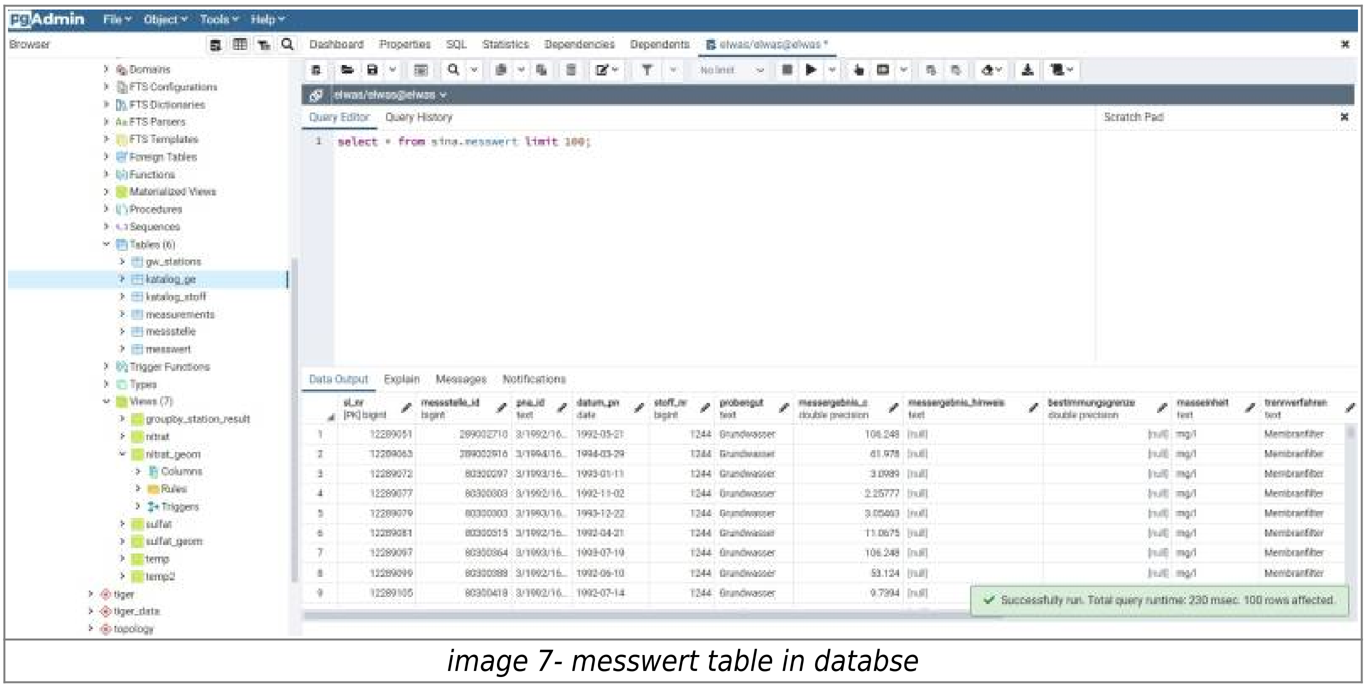
## 2.4 Observation Data in the Database

In the previous section, we downloaded the data, cleaned and imported them to the database successfully and now it's time to see the data in the database. as we know, the SQL command is valid in the PostgresSQL database, as a result of that, we will run some basic SQL commands to see the data.

First, we want to see our tables, with the below code, we are selecting all the columns (* means all the columns) from our schema which is consist of our table (in this case is **sina**) and with the name of the table. and then because the size of the table is huge and we want only to see the first 100 rows then we just limit it to 100.

```
select * from sina.messwert limit 100;
```

The below image (image 7) shows the result of the above command in PGadmin.



```
select * from sina.messwert limit 100;
```

| | sl_nr [PK] bigint | messstelle_id bigint | pna_id text | datum_pn date | stoff_nr bigint | probengut text | messergebnis_c double precision | messergebnis_hinweis text | bestimmungsgrenze double precision | masseinheit text | trennverfahren text |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 12289051 | 289002710 | 3/1992/16... | 1992-05-21 | 1244 | Grundwasser | 106.248 | [null] | [null] | mg/l | Membranfilter |
| 2 | 12289063 | 289002916 | 3/1994/16... | 1994-03-29 | 1244 | Grundwasser | 61.978 | [null] | [null] | mg/l | Membranfilter |
| 3 | 12289072 | 80300297 | 3/1993/16... | 1993-01-11 | 1244 | Grundwasser | 3.0989 | [null] | [null] | mg/l | Membranfilter |
| 4 | 12289077 | 80300303 | 3/1992/16... | 1992-11-02 | 1244 | Grundwasser | 2.25777 | [null] | [null] | mg/l | Membranfilter |
| 5 | 12289079 | 80300303 | 3/1993/16... | 1993-12-22 | 1244 | Grundwasser | 3.05463 | [null] | [null] | mg/l | Membranfilter |
| 6 | 12289081 | 80300515 | 3/1992/16... | 1992-04-21 | 1244 | Grundwasser | 11.0675 | [null] | [null] | mg/l | Membranfilter |
| 7 | 12289097 | 80300364 | 3/1993/16... | 1993-07-19 | 1244 | Grundwasser | 106.248 | [null] | [null] | mg/l | Membranfilter |
| 8 | 12289099 | 80300388 | 3/1992/16... | 1992-06-10 | 1244 | Grundwasser | 53.124 | [null] | [null] | mg/l | Membranfilter |
| 9 | 12289105 | 80300418 | 3/1992/16... | 1992-07-14 | 1244 | Grundwasser | 9.7394 | [null] | | | |

✓ Successfully run. Total query runtime: 230 msec. 100 rows affected.

*image 7- messwert table in databse*

Now with the above SQL command, we are able to see the others three tables that we have (The below codes).

```
select * from sina.messstelle limit 100;
```

```
select * from sina.katalog_stoff limit 100;
```

```
select * from sina.katalog_ge limit 100;
```

Now we want to see more details for our tables and we will run the below codes.

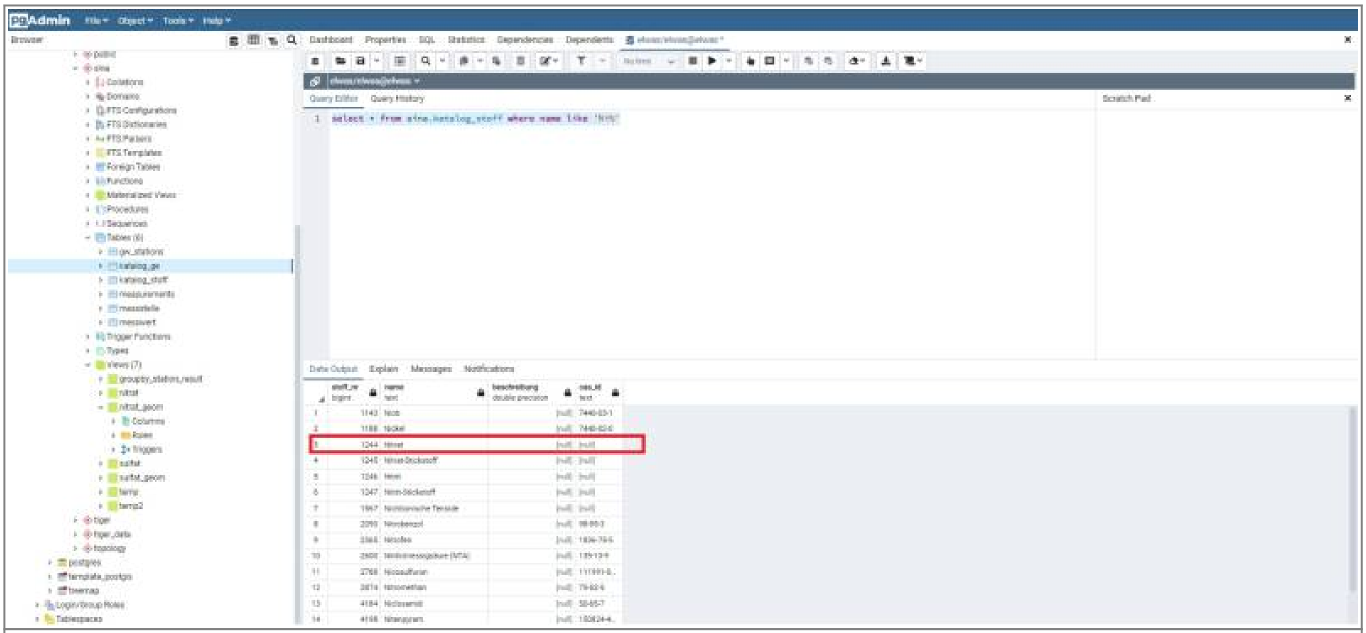
**Count the rows of each table:** with the below code, we can see how many rows we have in each table.

```
select count (*) from sina.messwert;
```

```
select count (*) from sina.messstelle;
```

**Filter the data based on Nitrate only:** In this case first we need to find out what the substance number of the Nitrate is. With below code, we can find

```
select * from sina.katalog_stoff where name like 'Ni%';
```



```
select * from sina.katalog_stoff where name like 'Ni%'
```

| | stoff_nr<br>bigint | name<br>text | beschreibung<br>double precision | cas_id<br>text |
|---|---|---|---|---|
| 1 | 1143 | Niob | [null] | 7440-03-1 |
| 2 | 1188 | Nickel | [null] | 7440-02-0 |
| 3 | 1244 | Nitrat | [null] | [null] |
| 4 | 1245 | Nitrat-Stickstoff | [null] | [null] |
| 5 | 1246 | Nitrit | [null] | [null] |
| 6 | 1247 | Nitrit-Stickstoff | [null] | [null] |
| 7 | 1967 | Nichtionische Tenside | [null] | [null] |
| 8 | 2090 | Nitrobenzol | [null] | 98-95-3 |
| 9 | 2368 | Nitrofen | [null] | 1836-75-5 |
| 10 | 2606 | Nitrilotriessigsäure (NTA) | [null] | 139-13-9 |
| 11 | 2768 | Nicosulfuron | [null] | 111991-0... |
| 12 | 2874 | Nitromethan | [null] | 75-52-5 |
| 13 | 4184 | Niclosamid | [null] | 50-65-7 |
| 14 | 4198 | Nitenpyram | [null] | 150824-4... |

*Image 8- Find out the substance number (stoff_nr) of Nitrate*

As we can see in image 8, the substance number (stoff_nr) of Nitrate is **1244**

**Filter the data based on Nitrate only:**

```
select * from sina.messwert where stoff_nr = '1244';
```

Now we can filter the messwert table based on Nitrate. In this step we need to somehow save the above table. We can save this new table as a new “view”.

**What is views:** A view is a database object that is of a stored query. A view can be accessed as a virtual table in PostgreSQL. It means that we can do whatever we are able to do with views same as tables. The below code create views for us:

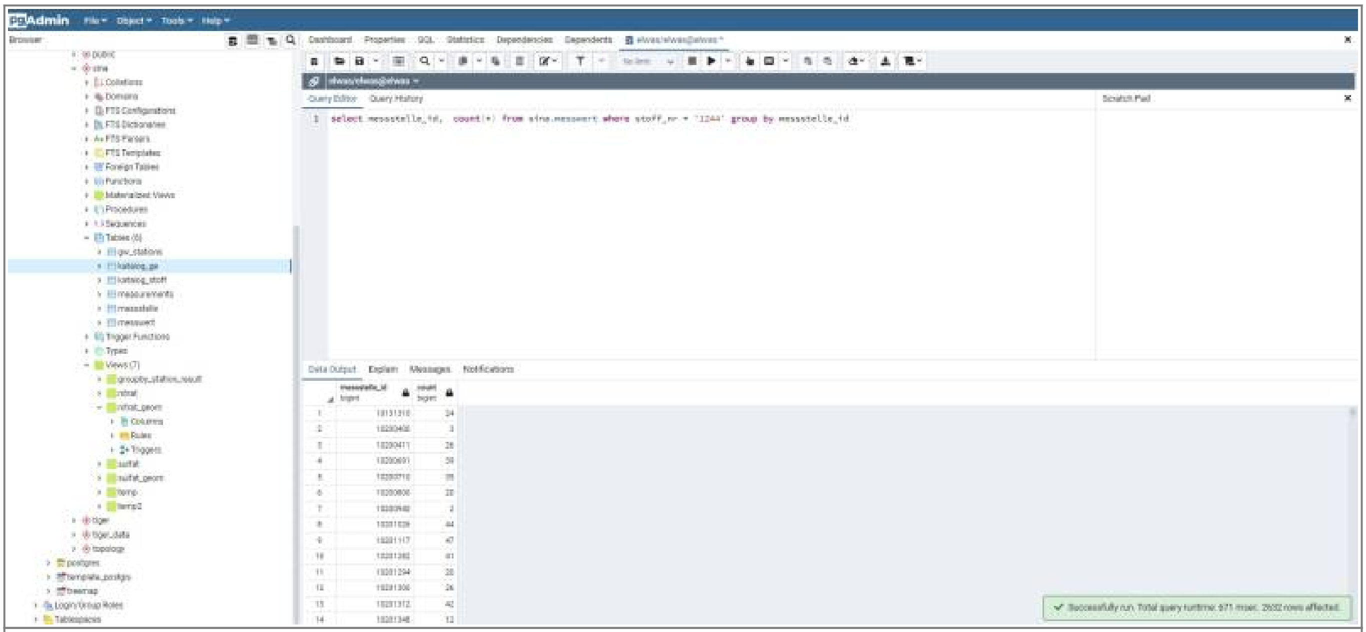
```
create view sina.nitrat as (select * from sina.messwert where stoff_nr =
'1244');
```

Now we have “nitrat” view which simply can call same as tables with the below code. This view is filter of our messwert table based on “1244” which is “Nitrate”

```
select * from sina.nitrat ;
```

**Group by the two tables:** In this section we want to group by our two tables (messwert and messstelle) only in Nitrate. These two tables have a column **messstelle_id** which simply means station id.

```
select messstelle_id,  count(*) from sina.messwert where stoff_nr = '1244'
group by messstelle_id;
```



```
select messstelle_id,  count(*) from sina.messwert where stoff_nr = '1244' group by messstelle_id
```

| | messstelle_id<br>bigint | count<br>bigint |
|---|---|---|
| 1 | 10131310 | 24 |
| 2 | 10200408 | 3 |
| 3 | 10200411 | 28 |
| 4 | 10200691 | 59 |
| 5 | 10200710 | 35 |
| 6 | 10200806 | 20 |
| 7 | 10200940 | 2 |
| 8 | 10201029 | 44 |
| 9 | 10201117 | 47 |
| 10 | 10201282 | 41 |
| 11 | 10201294 | 20 |
| 12 | 10201306 | 26 |
| 13 | 10201312 | 42 |
| 14 | 10201348 | 12 |

✓ Successfully run. Total query runtime: 971 msec. 2932 rows affected.

*Image 9- Group by the two tables (messwert and messstelle)*

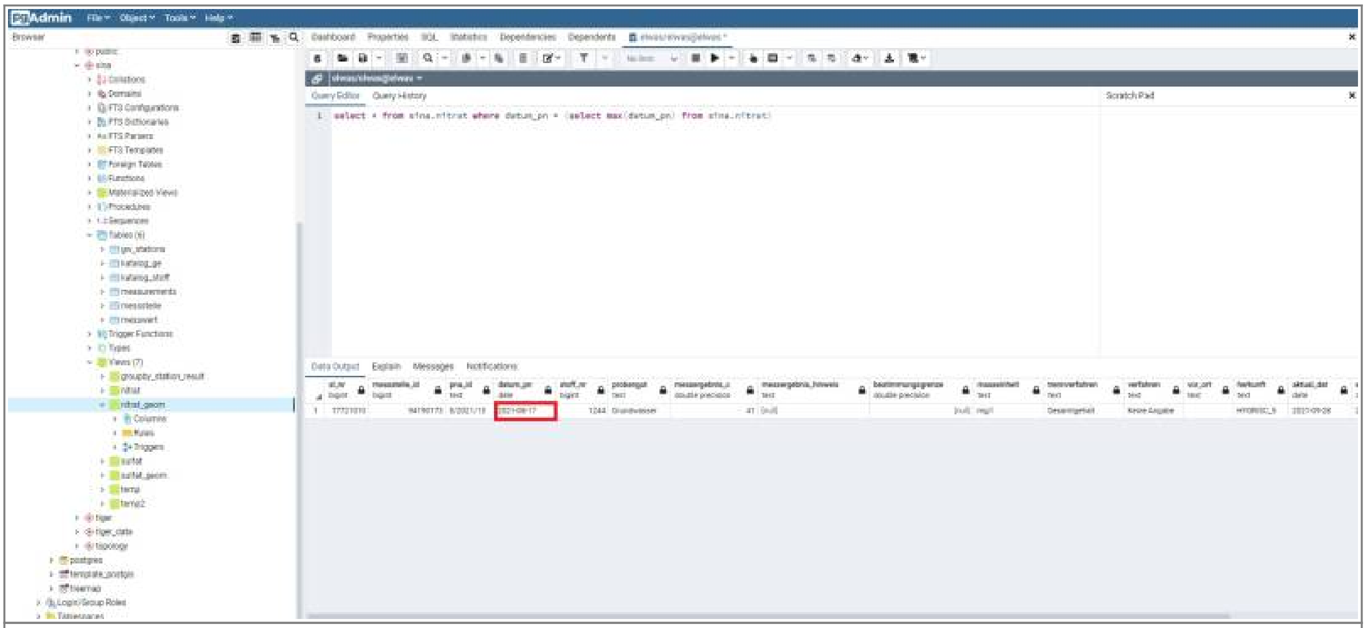
Image 9 shows that each station id has how many single measurements for the Nitrate only. I highly recommend opening the below website to get more deep into how “group by” works and how we can use it.

https://www.w3schools.com/sql/sql_orderby.asp

*Video 3- How group by works*

**Maximum date in nitrat table:**

```
select * from sina.nitrat where datum_pn = (select max(datum_pn) from
sina.nitrat);
```



```
select * from sina.nitrat where datum_pn = (select max(datum_pn) from sina.nitrat)
```

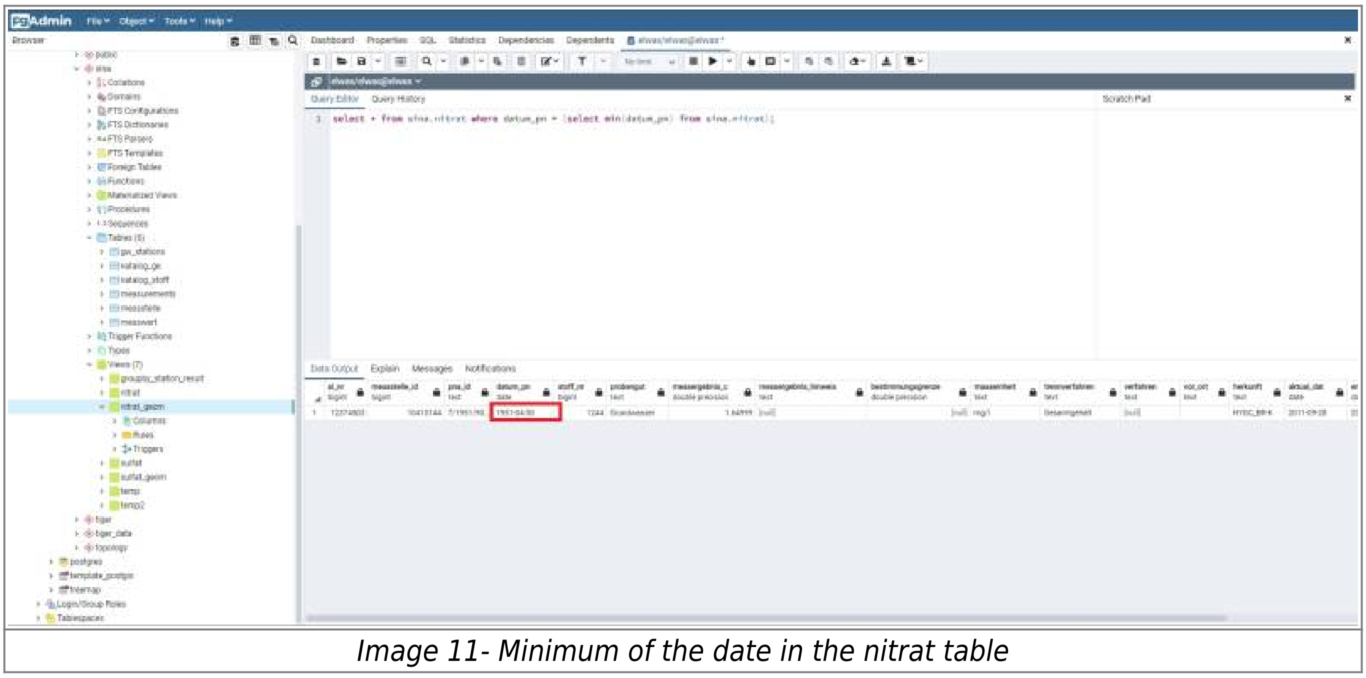
| | st_nr<br>bigint | messstelle_id<br>bigint | pns_id<br>text | datum_pn<br>date | stoff_nr<br>bigint | probengut<br>text | messergebnis_c<br>double precision | messergebnis_hinweis<br>text | bestimmungsgrenze<br>double precision | masseinheit<br>text | trennverfahren<br>text | verfahren<br>text | vor_ort<br>text | herkunft<br>text | aktual_dat<br>date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 17721010 | 94190173 | 8/2021/18 | 2021-08-17 | 1244 | Grundwasser | 41 | [null] | [null] | mg/l | Gesamtgehalt | Keine Angabe | | HYGRISC_5 | 2021-09-28 |

Image 10- Maximum of the date in the nitrat table

As we can see in Image 10, the maximum date is **2021-08-17**

**Minimum date in nitrat table:**

```
select * from sina.nitrat where datum_pn = (select min(datum_pn) from
sina.nitrat);
```



```
select * from sina.nitrat where datum_pn = (select min(datum_pn) from sina.nitrat);
```

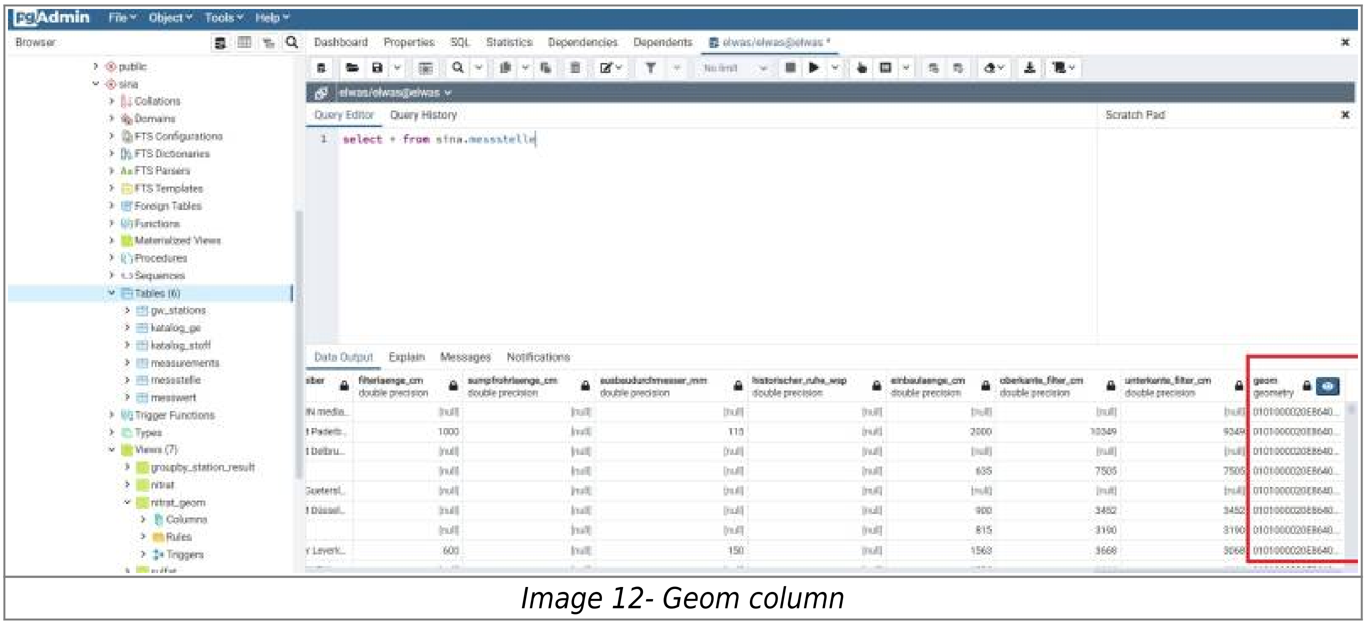
| | st_nr<br>bigint | messstelle_id<br>bigint | pns_id<br>text | datum_pn<br>date | stoff_nr<br>bigint | probengut<br>text | messergebnis_c<br>double precision | messergebnis_hinweis<br>text | bestimmungsgrenze<br>double precision | masseinheit<br>text | trennverfahren<br>text | verfahren<br>text | vor_ort<br>text | herkunft<br>text | aktual_dat<br>date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 12374803 | 10410144 | 7/1951/90 | 1951-04-30 | 1244 | Grundwasser | 1.64999 | [null] | [null] | mg/l | Gesamtgehalt | [null] | | HYEC_BR-K | 2011-09-28 |

Image 11- Minimum of the date in the nitrat table

As we can see in Image 11, the minimum date is **1951-04-30**

**Create geometry column in messstelle table:** In this section, we want to create a geometry column from **e32** and **n32** columns from the messstelle table. with the below code, we are able to create a new column and we set the name as **geom**

```
ALTER TABLE sina.messstelle ADD COLUMN geom geometry(Point, 25832);
UPDATE sina.messstelle SET geom = ST_SetSRID(ST_MakePoint(e32, n32), 25832);
```



```
select * from sina.messstelle
```

| …iber | filterlaenge_cm<br>double precision | sumpfrohrlaenge_cm<br>double precision | ausbaudurchmesser_mm<br>double precision | historischer_ruhe_wsp<br>double precision | einbaulaenge_cm<br>double precision | oberkante_filter_cm<br>double precision | unterkante_filter_cm<br>double precision | geom<br>geometry |
|---|---|---|---|---|---|---|---|---|
| N media… | [null] | [null] | [null] | [null] | [null] | [null] | [null] | 0101000020E8640… |
| 1 Paderb… | 1000 | [null] | 115 | [null] | 2000 | 10349 | 9349 | 0101000020E8640… |
| 1 Delbru… | [null] | [null] | [null] | [null] | [null] | [null] | [null] | 0101000020E8640… |
|  | [null] | [null] | [null] | [null] | 635 | 7505 | 7505 | 0101000020E8640… |
| Guetersl… | [null] | [null] | [null] | [null] | [null] | [null] | [null] | 0101000020E8640… |
| 1 Düssel… | [null] | [null] | [null] | [null] | 900 | 3452 | 3452 | 0101000020E8640… |
|  | [null] | [null] | [null] | [null] | 815 | 3190 | 3190 | 0101000020E8640… |
| r Leverk… | 600 | [null] | 150 | [null] | 1563 | 3668 | 3068 | 0101000020E8640… |

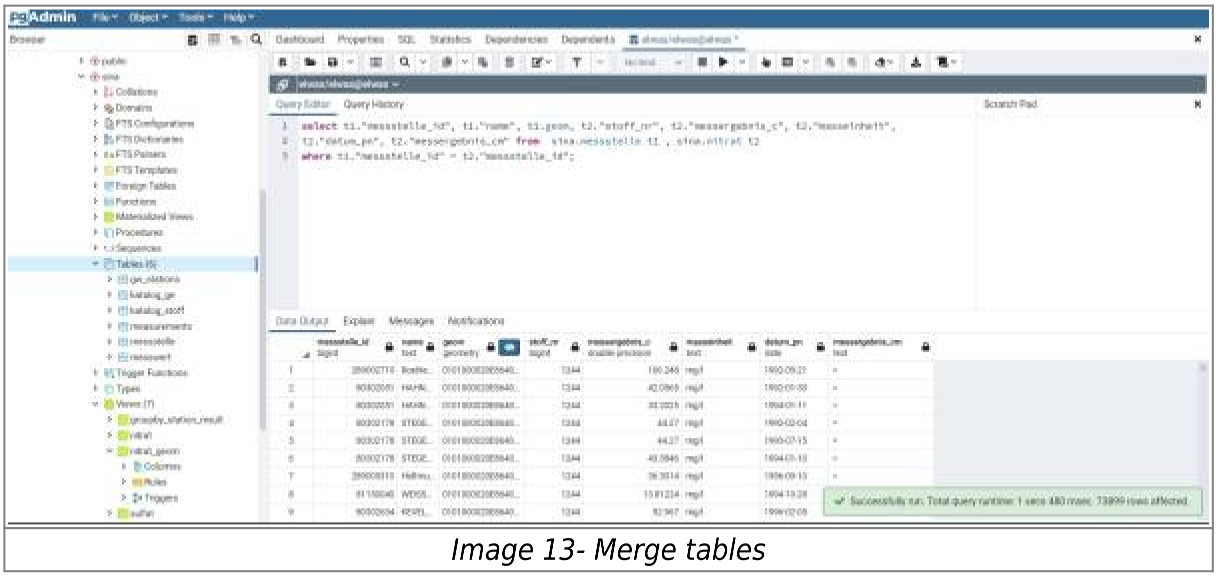
Image 12- Geom column

Now the messstelle table has one more column (geom) which consists of the geometry information of the location of each station.

**Merge two tables:**

In this section, we want to merge the two tables (messwert and messstelle) based on the same column which is "messstelle_id". We need to select columns that we need from each tables and then merge them based on the "messstelle_id"

```
select t1."messstelle_id", t1."name", t1.geom, t2."stoff_nr",
t2."messergebnis_c", t2."masseinheit",
t2."datum_pn", t2."messergebnis_cm" from  sina.messstelle t1 , sina.nitrat
t2
where t1."messstelle_id" = t2."messstelle_id";
```



```
select t1."messstelle_id", t1."name", t1.geom, t2."stoff_nr", t2."messergebnis_c", t2."masseinheit",
t2."datum_pn", t2."messergebnis_cm" from  sina.messstelle t1 , sina.nitrat t2
where t1."messstelle_id" = t2."messstelle_id";
```

✔ Successfully run. Total query runtime: 1 secs 480 msec. 73899 rows affected.

Image 13- Merge tables

Now we need to create a view and save this SQL command as a new view. the name of this new view is "nitrat_geom".

```
create view sina.nitrat_geom as (select t1."messstelle_id", t1."name",
t1.geom, t2."stoff_nr", t2."messergebnis_c", t2."masseinheit",
 t2."datum_pn", t2."messergebnis_cm" from  sina.messstelle t1 , sina.nitrat
t2
where t1."messstelle_id" = t2."messstelle_id")
```

We need this new view for the next section in QGIS.

## 2.5 QGIS

QGIS is an open-source and free application that can support viewing, editing and analysis of geospatial data.

You can download QGIS for free from the below link.

https://qgis.org/en/site/

The below video shows how to download and install QGIS for windows which are highly recommended to watch before installing it.

*Video 4- How to download and install QGIS*

Now is the time to get to know about QGIS and the below video can help so much.

*Video 5- Get to know QGIS*

**Create a time series video:** In this section, we want to create a time series video to see how the nitrate concentration has been changed over time in North Rhine-Westphalia which is the most crowded state in Germany. First, we need to download the shapefile of the North Rhine-Westphalia state and load it into QGIS.

We will download three shapefiles,

- Whole state shapefile (dvg1bld_nw.shp)

- kreis shapefile (dvg1krs_nw.shp)
- Gemeinde shapefile (dvg1gem_nw.shp)

All the three shapefiles can be downloaded from here. After downloading, we need to load them to the QGIS to see them. The below video shows how to load the shapefile in QGIS.

*Video 6- How to import shapefile in QGIS*

Now we can see the map of NRW, kreis and Gemeinde. There are two options to create a video for time series.

**Locally with shapefile:** In here, we need to have a shapefile that consists of the nitrate concentration over time. download the notebook from here and run the python codes to create two shapefiles. then we should load these two shapefile to the QGIS. The first shapefile is consist of all stations in NRW and the second one is consist of the nitrate concentration.

The below video shows how we can load shapefiles to QGIS.

*Video 7- How to load shapefile in QGIS*

**Connect to Database:** The below video shows how we can connect our QGIS to Database and load the file.

*Video 8- How to connect database to QGIS and load the files*

# 3 Dashboard

This section will discuss how we can create an interactive dashboard for our data. An interactive dashboard is a tool that users can interrelate with data by analyzing, visualizing as well as monitoring the data.

Two approaches to creating a dashboard will be discussed in this section. The aim of this dashboard is a simple interactive dashboard in which users with no knowledge of programming can easily consider the data as well as a map. This kind of dashboard will help users to understand data better. Nowadays dashboards are widely used in several ways to help managers and decision-makers to make decision easier. one good example of such a kind of dashboard is the Covid-19 dashboard which each country also here in Germany people are widely used. The Covid-19 Dashboard aids people in noticing how many new cases and how many new deaths have been recorded in different periods of time.

In our case, we want to create a simple dashboard which shows the map of NRW as well as Nitrate and Sulfate concentration rates at different times.

### 3.1 Plotly Dash:

**Plotly:** Plotly is a computing company located in Montreal, Canada. They develop online data analytics and visualization tools. Plotly offers online graphing, analytics, and statistics tools for their users, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST. Plotly offers several open-source and enterprise products such as Dash which have been used for creating simple and interactive dashboards in this project.

**Dash:** Dash is a framework to build data apps rapidly not only in Python but also in R, Julia, and F#. According to Plotly official website, Dash is downloaded 800,000 times per month which shows that nowadays Dash getting more popular. Dash is a great framework for anyone who uses data with a customised user interface. Through a couple of simple patterns, Dash eliminated all of the technologies as well as protocols that are needed to make a full-stack web app with interactive data considerations. Another good feature is that Dash is running on web browsers so it means that no other application needs to run it.

If you would like to know more about Dashboard with Plotly Dash, click the link below. In the below link, you will find full tutorials about how to create a simple dashboard with Plotly Dash.

https://www.youtube.com/c/CharmingData

Dash is also offering some dashboards examples which could be really nice and helpful to get ideas. | Click here for Dash gallery

All the source codes of the dash gallery are available in | here

### 3.2 Panel:

# 4. Result

## Nitrate concentration 2000-2010

The below video has shown the concentration of nitrate in NRW from 2000 to 2010. The video is created with QGIS 3.16

## Nitrate concentration 2010-2020

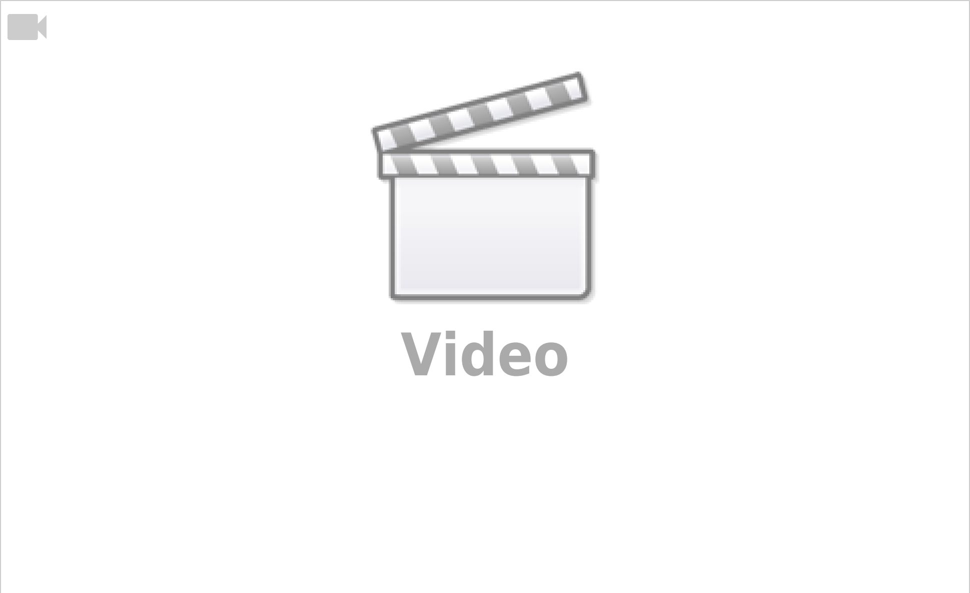
The below video has shown the concentration of nitrate in NRW from 2010 to 2020. The video is created with QGIS 3.16

## Sulfat concentration 2000-2010

The below video has shown the concentration of sulfate in NRW from 2000 to 2010. The video is created with QGIS 3.16

### Sulfat concentration 2010-2020

The below video has shown the concentration of sulfate in NRW from 2010 to 2020. The video is created with QGIS 3.16



# 5. Project codes

All the codes are available in below link.

Click here for project codes

# Weitere Infos

- EOLab-Wiki-Seiten zum Thema **Grundwasserdaten in NRW**