# **Spatial Databases**

Lecture notes for Geoinformatics/Cartography Master Students Dr. UNGVÁRI Zsuzsanna assistant professor EJTEInstitute of Cartography and Geoinformatics

## Contents

Foreword	4
Chapter 1: Installing pgAdmin4, PostgreSQL and POSTGIS, a short introduction to pgAdmin4	5
Importing a spatial database	12
Chapter 2: DBeaver, the free universal database tool	18
Installing DBeaver on Windows is straightforward:	18
DBeaver User Interface	19
Database Stucture in DBeaver	20
Querying in the SQL Console	25
Chapter 3. Basics SQL	27
Chapter 4. Key to basics SQL	39
Chapter 5: Comparision sheet to PostgreSQL and MySQL	43
Chapter 6. The PostGIS functions	47
Getting familiar with PostGIS	47
Representing Points, Lines and Polygons in WKT	47
Geometry type, Dimension, Coordinate dimension and Number of Nodes	50
Advanced in PostGIS	52
Projections, Spatial Reference Identifiers	52
Measurements	53
Bounding boxes and bounding geometries	57
Accessing other geometric information	58
Geometry validation	59
Creating geometries	59
Understanding spatial relations – Geoprocessing	60
Combining geometries	61
Others	62
Conversion function in PostGIS	62
Chapter 7: Practices in POSTGIS	63
Chapter 8: Key	70
Chapter 9. Data import	80
Creating a new database, and importing data from SQL files	80
Adding extensions (PostGIS, PostGIS Raster, PostGIS Topology)	80
Uploading Data into a PostgreSQL Database from CSV – Using an Existing table structure	81
Uploading Data to a PostgreSQL database from a CSV – Missing table structure	86
Creating a backup	88
Upload a table from an another one.	89
Data conversion	89

Chapter 10: Key	101
Troubleshooting FAQ	. 94
Shapefile import with PostGIS Bundle	. 92
The Geography data type	. 91

# Foreword

These lecture notes were created for the Spatial Databases course in the Geoinformatics and Cartography Master's programme at Eötvös Loránd University, Budapest, Hungary. It also includes an introductory section on database management and queries. Based on the author's experience, the best way to master SQL is through extensive practice with queries, clauses, functions, etc. Therefore, this note is consists of the following parts:

- A description of **basic SQL** with several practical examples.
- A chapter on **PostGIS**, including theoretical concepts, function descriptions, and numerous examples.
- Practical guidance on **installing PostgreSQL and PostGIS**, as well as uploading databases and performing other related tasks.

The author strongly recommends readers to have some prior experience with QGIS or ArcGIS, as it simplifies the understanding of spatial databases.

The tasks in these note are based on **PostgreSQL 13** and **PostGIS 3.1**. Newer versions may have some differences, so please refer to the **official documentation** for updates.

I wish you an enjoyable learning experience!

the author

Reviewed by Dr. Gede Mátyás

# Chapter 1: Installing pgAdmin4, PostgreSQL and POSTGIS,

# a short introduction to pgAdmin4

"pgAdmin is the most popular and feature rich open source administration and development platform for PostgreSQL, the most advanced open source database in the world".<sup>1</sup> However, pgAdmin itself is only a database management tool. To use it with a PostgreSQL database, PostgreSQL must be installed alongside pgAdmin. I recommend using the following website to install these programs. Install the available newest version. (*Note: when I wrote these lecture notes, I used the PostgreSQL version 13 and pgAdmin4. If you will found some differences in syntax or in pictures, that caused by the different software versions. In this case, please check the documentation!)* 

#### https://www.postgresql.org/download/windows/

Click on **Download the installer**, then select your computer's operating system and the software version.

$\leftrightarrow \rightarrow C$	O A https://www.postgresql.org/download/windows/			<b>I</b> 1	1
Home About Do	wnload Documentation Community Developers Support Donate Your account	Search for	(	2 6	
	August 8th, 2024: PostgreSQL 16.4, 15.8, 14.13, 13.16, 12.20, and 17 Beta 3 Released!				
Quick Links	Windows installers 🖶				
Downloads     Packages     Source	Interactive installer by EDB				
<ul> <li>Software Catalogue</li> </ul>	Download the installer certified by EDB for all supported PostgreSQL versions.				
<ul> <li>File Browser</li> </ul>	Note! This installer is hosted by EDB and not on the PostgreSQL community servers. If you have issues with the website webmaster@enterprisedb.com.	it's hosted on, please contac	:t		
	This installer includes:				
	<ul> <li>The PostgreSQL server</li> <li>pgAdmin, a graphical tool for managing and developing your databases</li> <li>StackBuilder, a package manager for downloading and installing additional PostgreSQL tools and drivers. Stackbuilt migration, replication, geospatial, connectors and other tools.</li> </ul>	der includes management, ir	ntegration,	Decum	
	This installer can run in graphical or silent install modes.	Home About I	Download	Docum	ent
	The Windows installer is designed to be a straightforward, fast way to get up and running with PostgreSQL. In the unlike please report them on the installer Github page. Issues with the installed packages should be reported to the appropria				
	Advanced users can also download a zip archive of the binaries, without the installer. This download is intended for use another application installer.	Quick Links		Wi	n
	Dist former automatic	- Dowolaada			
	Platform support	🗣 Képmetsző			0
	The installers are tested by EDB on the following platforms. They can generally be expected to run on other comparable Windows:	Képernyőkép a vágólapra máso Válassza ki a megjelöléshez és	olva és ment megosztásh	ve oz.	

<sup>&</sup>lt;sup>1</sup> <u>https://www.pgadmin.org/</u>

$\leftarrow \ \rightarrow \ C$	○ A https://www.	enterprisedb.com/downloads/p	oostgres-postgresql-downloads			E \$	V 🖲 É
& E	DB EDB Postgres AI Serv	vices Resources Com	pany		Sigr	n In V Talk to an Expert	٩
C	Download F	POSTGRESC	ĴΓ				
	PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32	
	16.4	postgresql.org	postgresql.org	8	<b>B</b>	Not supported	
	15.8	postgresql.org 🗗	postgresql.org	0	<b>B</b>	Not supported	
	14.13	postgresql.org 🗗	postgresql.org	0	ė.	Not supported	
	13.16	postgresql.org	postgresql.org	<u>e</u>	Ċ.	Not supported	
	12.20	postgresql.org	postgresql.org	ů.	ė.	Not supported	
	9.6.24*	8	8	0	ė.	ė.	
	9.5.25*	0		8	<b>B</b>	8	

The installion process is straightforward:

- 1. Click **Next** through the setup and accept the default options.
- 2. Set the port number. The default is **5432**. If it is not already in use, keep it.
- 3. Choose a username and a password (*for example: postgres/appletree*). Do not forget these credentials!

Finally, you will set up the PostgreSQL DB with pgAdmin. While pgAdmin is a professional, robust and stable software, its user interface is not very user-friendly. Therefore, in the next chapter, the author will suggest an alternative solution (DBeaver).

However, if you prefer or are more familiar with pgAdmin, you can still use it. Keep in mind that some tasks in this guide are written specifically for DBeaver.

To handle spatial data, you need to install the PostGIS extension. Please visit the following website to download and install PostGIS on Windows using one of the two methods below:

#### Method 1: Download and Install the Setup File:

- 1. Open this website: https://download.osgeo.org/postgis/windows/
- 2. Find the appropriate version of PostGIS and download the setup file (EXE).

$\leftarrow \rightarrow C$	C A https://download.osgeo.org/postgis/windows/	53	⊠ 🕄 ් =

#### Index of /postgis/windows/

File Name 1	<u>File Size</u> ↓	<u>Date</u> ⊥
Parent directory/	-	-
<u>extras/</u>	-	2014-Jan-16 05:49
<u>pg10/</u>	-	2022-Aug-21 03:35
<u>pg11/</u>	-	2023-Jun-02 15:53
<u>pg12/</u>	-	2024-May-23 14:13
<u>pg13/</u>	-	2024-May-23 14:14
<u>pg14/</u>	-	2024-May-23 14:15
<u>pg15/</u>	-	2024-May-23 14:16
<u>pg16/</u>	-	2024-May-23 14:17
<u>pg80/</u>	-	2012-Oct-31 17:38
<u>pg81/</u>	-	2012-Oct-31 18:15
<u>pg82/</u>	-	2012-Oct-31 18:13
<u>pg83/</u>	-	2012-Oct-31 18:18
<u>pg84/</u>	-	2013-Mar-16 05:00
<u>pg90/</u>	-	2013-Sep-08 15:57
<u>pg91/</u>	-	2014-May-18 19:15
<u>pg92/</u>	-	2015-Aug-16 04:28
<u>pg93/</u>	-	2016-Oct-16 06:38
<u>pg94/</u>	-	2018-Dec-10 00:02
<u>pg95/</u>	-	2018-Dec-09 01:17

3. The installer and follow the instructions to complete the installation.

#### Index of /postgis/windows/pg16/

File Name 4	File Size 4	Date 1
Parent directory/	-	-
archive/	-	2024-May-23 14:15
postgis-bundle-pg16-3.4.2x64.zip	129.3 MiB	2024-Apr-19 16:05
postgis-bundle-pg16-3.4.2x64.zip.md5	103 B	2024-Apr-19 16:05
postgis-bundle-pg16x64-setup-3.4.2-1.exe	117.0 MiB	2024-May-19 04:44
postgis-bundle-pg16x64-setup-3.4.2-1.exe.md5	75 B	2024-May-19 05:45

#### Method 2: Install via Application Stack Builder

1. PostgreSQL includes the Application Stack Builder. Open it.



2. Select your PostgreSQL version, and click Next.



- 3. Expand the Spatial Extensions list and check PostGIS Bundle.
- 4. Choose the folder where you want to download the PostGIS Bundle.

😻 Stack Builder 4.2.1	×
	Please select the applications you would like to install.          Categories         Add-ons, tools and utilities         Database Drivers         Database Server         Registration-required and trial products         Spatial Extensions         PostGIS 3.4 Bundle for PostgreSQL 16 (64 bit) v3.4.2 (v3.4.1 inst         Web Development
	< <u>B</u> ack <u>N</u> ext > <u>C</u> ancel

5. Accept the license agreement, then review and confirm the components to be installed.

PostGIS Bundle 3.4.2 for P	ostgreSQL x64 16 Setup	-		×
Ch Ch Ch Ch Ch Ch	hoose Components hoose which features of PostGIS 64 16 you want to install.	Bundle 3.4.2 f	for Postgr	eSQL
Check the components you wa install. Click Next to continue.	nt to install and uncheck the com	ponents you d	lon't want	to
Select components to install: Space required: 290.5 MB	PostGIS Bundle     Create spatial database     Register PROJ_LIB Env Va     Register GDAL_DATA Env     Enable Key GDAL Drivers     Enable All GDAL Drivers     Allow Out-db Rasters     Register SSL Bundle	Position you over a com see its desc	ur mouse ponent to ription.	
Nullsoft Install System v3.09 ——	< Back	Next >	Car	ncel

- 6. Set the destination folder to your PostgreSQL directory and confirm.
- 7. The installation process will begin. Please wait until it is completed.

PostGIS Bundle 3.4.2 for PostGIS	stgreSQL x64	16 Setup	—	
Cho Chi Pos	ose Install I pose the folde stgreSQL x64 1	Location r in which to ins 16.	stall PostGIS Bund	dle 3.4.2 for
Setup will install PostGIS Bundle install in a different folder, click	3.4.2 for Post Browse and se	greSQL x64 16 elect another fo	in the following f lder. Click Next to	older. To o continue.
Destination Folder C:\Program Files\PostgreSQ	<u>L\16</u>		Brov	vse
Space required: 290.5 MB Space available: 373.5 GB				
Nullsoft Install System V3.09		< <u>B</u> ack	<u>N</u> ext >	Cancel

Now, you are ready to start working with pgAdmin.

## pgAdmin quick start guide

The following sections present some basics functions of pgAdmin.

#### Starting pgAdmin

First, launch pgAdmin. The initial step is logging in. You must remember the username and password you set during the installation process. After entering your password, you will gain access to your

#### databases. This is the welcome screen:

👎 pgAdmin 4									- 0 3	$\times$
pg Admin	File 🗸	Object 🗸	Tools 🗸	Help 🗸						
Browser	0%	<b>III T</b>	Q >_	Dashboard Properties SQL	Statistics	Dependencies	Dependents			×
> EServers				Welcome						
				PgAdmin is an Open Source addr procedural code debugger and	min nt Tools fo ses Poste ninistration an nuch more. T	or PostgreSQ greSQL   Op nd management 1 'he tool is designe	L en Source tool for the Postgre ed to answer the ne	eSQL database. It includes a graphical administratio eeds of developers, DBAs and system administrato	on interface, an SQL query tool, a rs alike.	
				Quick Links						
					Add New Se	erver		Configure pgAd	imin	
				Getting Started						
				PostgreSQL Documentation	on	pgAdmin	Website	Planet PostgreSQL	Community Support	
								Aktiválja a W Aktiválja a Windo	'indowst ws rendszert a Gépházban.	
🗄 🔎 Írjon	n ide a ker	eséshez	10	🛍 🛤 🗖 🍪 🚱	Q	<b>VI (R</b> )		🐣 8°C Időnként napos 🖍	0 🖻 📾 🌈 📥 14:44	6

#### Navigating the Interface

In the left column (**Object Explorer**), open the **Servers** menu. Under **Servers**, you can find the **Databases**. By default, there is always an empty, initial DB called *postgres*.

#### **Importing the Library Database**

Now, let's import the *library* DB from *library.sql* file.

#### Create a New Database

- Right-click on Servers, then select Create  $\rightarrow$  New Database.
- A dialog window will appear.

🥽 Create - Database	9	2 ×
General Definition	Security Parameters Advanced SQL	
Database	library	
Owner	🐣 postgres	*
Comment		
[ [ i ][ ? ]	× Cancel 🖧 Reset	Save

- Enter a name for the database and click the **Save** button.
- The database is now created.

#### **Executing SQL Queries**

- Right-click on the *library* database and select **Query Tool**.
- Open the *library.sql* file in a text editor (e.g., Notepad++).
- Copy all the text from the file.



- Paste the copied text into the **Query Tool** window.
- Click the **Play** button to run the queries.

This set of queries will create the necessary tables in the *library* database, and populate them with data. Once the process is complete, you should receive the message: '*The query was successful*'.



For a detailed description of the *library* database structure and its tables, refer to the Appendix.

## Importing a spatial database

In the following chapters, we will be working in DBeaver, a universal database management tool. However, DBeaver has a weakness: importing process of large size SQL files (30–50MB +) often results int the error message *"importing process has failed*". To avoid this issue, I recommend using pgAdmin to import *hungary.sql*, our sample spatial database for this semester. pgAdmin is a more stable and robust tool for handling large imports.

#### **Creating the Hungary Database**

- 1. Open pgAdmin.
- 2. Right-click on **Databases**  $\rightarrow$  **Create**  $\rightarrow$  **Database**.
- 3. Enter *hungary* as the database name and click **Save**.

💜 paAdmin 4									
File V Obj	ject 🗸	Tools 🗸	Help 🗸						
Browser 🗊 🗊		Q >_	Dashboard	Pro	perties	SQL	Statistics	Dependencies	Deper
<ul> <li>Servers (1)</li> <li>         PostgreSQL 13     </li> <li>         Databases (5)     </li> </ul>			Server sess	ions					
> Skonyvtar		Creat	e	>	Data	abase		I	
> > postgres		Refre	sh						
> 🚍 zsuzsi2			4						
> 📑 zsuzsi3			2						
> 📤 Login/Group Roles			0						
> 🔁 Tablespaces									
			Tuples in					Inserts	Updat
			1						

#### Adding the PostGIS Extension

Since the *hungary* DB contains spatial data, you must first enable the PostGIS extension before import ing the SQL file.

- 1. In pgAdmin, locate the **Extensions** section under your database.
- 2. Right-click on **Extensions**  $\rightarrow$  **Create**  $\rightarrow$  **Extension**
- 3. In the dialog window, search for **PostGIS** and select it.
- 4. Click **Save** to add the extension.

🕆 Create	- Extension	2 ×
General	Definition SQL	
Name	postgis	~
Cascade	?	
Commen	t	h

#### Importing the hungary.sql File

- **1.** Open the **Query Tool**:
  - a. Right-click on the *hungary* database and select **Query Tool**.
  - b. Alternatively, you can click on the **barrel icon** in the toolbar or access it via **Tools** → **Query Tool**.
- 2. Open *hungary.sql* in a text editor (e.g., Notepad++).

- 3. Copy all the text from the file.
- 4. Paste the copied text into the **Query Tool** window.
- 5. Click the **Play** button to execute the queries. Wait for 20–30 seconds while the database is created and populated with data.



#### Viewing Databases in pgAdmin

Once the database is imported, you will see the following structure in the database tree. The **Schemas** contains the tables of the DB. To view a table, select it and click on the table icon above the **Object** 

A select \* query will run automatically, displaying all columns (fields) and rows (records). This is the **Data Output** section.

🏷 Subscriptions

g <mark>Admin</mark>	File ~ Object ~ Tools ~	Help	• ~								
rowser	\$ III Ta Q >_	Dasi	hboard Prope	rties SQL Statistics	Dependencies Dependencies	dents 🔠 public.autopal	ya/mo/postgres@Postgre	SQL 13			
v	♦ public	5	🖕 🔒 🗡	ŝ Q ⊻ ∰ ¥ f	i 🗊 🗹 🝸	🖌 No limit 🖌 🔳	Y	5 5 💇 🛓 '	<b>.</b> ~		
	> A B↓ Collations	~	mo/postares@	PostareSOI 13							
	> 🏠 Domains	Ouerv Editor Ouerv History									
	FTS Configurations	Quei	Query Editor Query History								
	FTS Dictionaries	1	SELECT * FR	ROM public.autopalya							
	Aa FIS Parsers     ETS Tomplaton	2	окрек ву ар	D_1d ASC							
	> Foreign Tables										
	> (iii) Functions	Data	Output Explr	in Managan Natifia	ationa						
	> 📴 Materialized Views	Data		in messages nounc							
	> (() Procedures		[PK] integer	character varying (254)	ap_hu_ed_dire character varying (254)	ap_alt_name character varying (254)	character varying (254)	ap_highway character varying (254)	ap_int_ref character varying		
	> 13 Sequences	1	1	way/4293741	backward	[null]	[null]	motorway	E 653		
	<ul> <li>Iables (29)</li> <li>Boutopolyo</li> </ul>	2	2	way/4380568	forward	[null]	[null]	motorway	[null]		
	> mautopaiya	3	3	way/4380570	forward	[null]	yes	motorway	E 71		
	> 📑 baranya_ffi	4	4	way/4380571	forward	[null]	[null]	motorway	E 71		
	> 🛅 baranya_nok	5	5	way/4380572	forward	[null]	[null]	motorway	E 71		
	> 🛅 baranya_ossz	6	6	way/4380573	forward	[null]	[null]	motorway	E 71		
	> 🛅 belterulet	7	7	way/4380792	backward	[null]	[null]	motorway	E 71		
	> 📑 elsorendu	8	8	way/4381283	backward	[null]	[null]	motorway	E 75		
	> epuletek	9	9	way/4383240	forward	[null]	[null]	motorway	E 71		
	> == folyo_reluter	10	10	way/4383748	forward	[null]	[null]	motorway	E 71		
	> == folyok	11	11	way/4681971	[nuli]	[nuli]	[null]	motorway	[null]		
	>  geolada_eov	12	12	way/4948018	backward	[nuli]	[null]	motorway	E 65;E 75		
	> 🚍 jarasok	13	13	way/4948020	backward	[nuli]	yes Aktiváli	angtowayhdowst	E 65;E 75		
	> 🛅 masodrendu	14	14	way/4965393	backward	[null]	[null] Aktiválja	a motatway is rendszert a G	66665E075n.		

#### Viewing Databases in pgAdmin

If the table contains geometry data, click on the geometry column header to open the **Geometry Viewer**. If the geometry projection is EPSG: 4326 (Plate Carée projection on WGS84 ellipsoid), an OpenStreetMap (OSM) basemap will appear in the background. This projection works with geographic coordinates (degrees). If the projection uses meter units, such as in the *motorway* table, the OSM basemap will not be visible.



To exit the Geometry viewer, return the Data Output tab.

Data Output	Explain	Messages	Notifications	Geometry Viewer					
								•	0
Editing ]	Data i	n Tables	5						

To modify data, click the **small pencil icon** in the field header.



A small editor window will open. Make the necessary changes.



Click Save the Data Changes. in the Data Output toolbar.



#### **Running Queries**

Open the Query Tool as described earlier. Type your SQL query. Click the Play button to run it.

₿ <sup>2</sup>	hungary/postg	pres@PostgreSQL 16	~ \$	¢.
	8 - /-	▼		
Quer	y Query Histo	ry		2
1	Select * fr	<b>om</b> np_tk <b>order by</b> np_name		
Data	Output Mess	ages Notifications		2
=+	• ľ ·	1 5 ± ~		
	np_gid [PK] integer	np_name character varying (254)	np_tipus character varying (50)	np_geom geometry
1	1	Aggteleki Nemzeti Park	nemzeti park	0106000020945C00000200000010300000010000008C040000E622B20778AD274191
2	2	Balaton-felvidéki Nemzeti Park	nemzeti park	0106000020945C000016000000103000000300000048030000D3B42CAB6A3E1F411E
3	90	Bihar-sík Tájvédelmi Körzet	tájvédelmi körzet	0106000020945C000001000000103000000100000020000007535099EC241294146,
4	141	Bihar-sík Tájvédelmi Körzet	tájvédelmi körzet	0106000020945C0000010000001030000001000000E0000008642BC02807294197

#### **Creating a New Table**

Right click on **Tables**  $\rightarrow$  **Create**  $\rightarrow$  **Table** In the first tab, enter the table name. Navigate to the **Columns** tab and add field name, data type and other properties (e.g primary key, NULL constraints, etc.). The SQL tab will display the generated SQL query. Click **Save** to create the table.

ene	eral	Columns Ad	vanced Constr	aints	Partitions Para	meters Se	curity SQL	
he	rite	d from table(s)	Select to inher	it from	1			
olu	ımr	IS						
		Name 👻	Data type		Length/Precision	Scale	Not NULL?	Primary key
fi	Ē	pr_szam	integer	Ŧ			No	No
fi	Ô	pr_nev	"char"	Ŧ			No	No
fi	Î	pr_id	serial	Ŧ			No	Yes
fi	Ô	pr_geom	geometry	*			No	No
i		?				× Cancel	A Reset	B Save
i en	Crea	? ate - Table I Columns Ad	dvanced Const	raints	Partitions Para	× Cancel meters Se	curity SQL	Save
i en 1 2 3 4 5 6 7 8 9	Creater and the creater and th	? ate-Table I Columns Ace EATE TABLE IF pr_szam inte pr_nev "char pr_id serial pr_geom geom PRIMARY KEY	dvanced Const NOT EXISTS pr ger, ", hetry, (pr_id)	raints	Partitions Para	<b>≭</b> Cancel meters Se	ct Reset	Save
i en 1 2 3 4 5 6 7 8 9 0	Creater and the creater and th	? ate-Table I Columns Act EATE TABLE IF pr_szam inte pr_nev "char pr_id serial pr_geom geom PRIMARY KEY FER TABLE publ OWNER to pos	dvanced Const NOT EXISTS pu ger, ", , hetry, (pr_id) .ic.proba .tgres;	raints	Partitions Para	★ Cancel meters Se	Curity SQL	Save

### Adding New Column

Expand the **Columns** section in **Object Explorer**. Right click on  $\rightarrow$  **Create**  $\rightarrow$  **Column**. In the dialog window, enter the column name (**General** tab); select the data type (**Definition** tab). Configure additional properties if needed. The SQL tab will show the generated SQL query. Click **Save** to apply changes.

<ul> <li>✓ I Tables (29)</li> <li>✓ I autopaly</li> <li>✓ I Colur</li> </ul>	a nns (14)	_	
8	Create	>	Column
	Refresh Search Objects PSQL Tool (Beta) Query Tool		



#### **Deleting Columns or Tables**

To remove a column or table: right-click on the column/table name and select **Delete**.

#### **Modifying Column Properties**

To change column properties (e.g., name, type): Right-click on the column name and select **Properties**. Modify the desired attributes in the dialog window.

#### **Refreshing the Database View**

If changes are not visible, then right-click on **Tables** or **Database** and select **Refresh**.

# Chapter 2: DBeaver, the free universal database tool

DBeaver Community is a free cross-platform database tool for developers, database administrators, analysts, and everyone working with data. It supports all popular SQL databases like MySQL, MariaDB, PostgreSQL, SQLite, Apache Family, and more.<sup>2</sup>

Advantages:

- Basic support for relational databases: MySQL, SQL Server, PostgreSQL and others
- Data Editor
- SQL Editor
- Database schema editor
- Basic ER Diagrams
- Basic charts
- Data export/import
- Task management
- Database maintenance tools

We will use this software to handle spatial databases (PostgreSQL+PostGIS).

You can download it from here: <u>https://dbeaver.io/download/</u>

#### Windows installation:

Installing DBeaver on Windows is straightforward:

- 1. Download the installer from the link above.
- 2. Run the setup file.
- 3. Click Next and accept the default settings when prompted.

## Creating a New Connection in DBeaver

When you start DBeaver for the first time, you will see the **New Connection** wizard. You can also access this menu anytime via **Database**  $\rightarrow$  **New Database Connection**.

#### Setting Up a PostgreSQL Connection

- 1. Select PostgreSQL and click Next.
- 2. Enter the following details: Server name localhost; port 5432 (default), username and password (same as in pgAdmin)
- 3. Check the **Show all databases** option.
- 4. Click **Finish** to complete the setup.

**Note:** The first time you connect to PostgreSQL, DBeaver will download the necessary drivers. Allow DBeaver to proceed with the download.

<sup>&</sup>lt;sup>2</sup> <u>https://dbeaver.io/</u>

	PostgreSQL			
🔞 Connect to	a database			×
Connection : PostgreSQL co	Settings onnection settings	(F	Postgre	SQ
Main PostgreS Server Connect by: URL: Host:	QL       Driver properties       SSH       SSL         Host       URL       jdbc:postgresql://localhost:5432/postgres         localhost       Gallost       Gallost	+ Network of Port: 5-	configuratio	ons
Database:	postgres	Show	all databa	ses
-Authenticatio Authenticatio Username: Password:	n: Database Native V postgres •••••• Save password			
Advanced Session role:	Local Client: PostgreSQL 16		~	
① <u>Connectio</u> Driver name:	n variables information ① Database documentation Connect PostgreSQL Dri	tion details (r	name, type, Driver lice	) ense
Test Connect	on < Back Next > Fini	ish	Cancel	

## DBeaver User Interface

DBeaver has a user-friendly interface, where frequently used menus are easily accessible. The figure below illustrates keyy components of the user interface. **Database Navigator** displays the DBs, you are connected to. **Editor** allows you to browse tables, view data, access the **ER Diagram**, and run queries.

A double-click in the **Database Navigator** opens the corresponding database, table, or column sheet.

1	File Edit Navigate Search SQL Editor Database Window Help	+							
- 1	🙀 🛡 🏀 💥 🗍 🖪 📮 🕞 Commit 🕞 Rollback 🍸 👻 Auto 🚱	DBeaver Sam	nple Database ( 👻	🗦 <no< th=""><th>ne&gt; 🚽 🔻</th><th>200</th><th><u>{5</u> + <b>∧</b> + P</th><th>🕞 Quick</th><th>c Access</th></no<>	ne> 🚽 🔻	200	<u>{5</u> + <b>∧</b> + P	🕞 Quick	c Access
- 1	🔲 Project - General 🕱 🛛 🕸 🖃 🕀 🛩 🗖	α EmpView δ	3						-
	Name DataSource Preview Size Modified	Properties	🖪 Data 📥 ER Diac	aram	📝 DBeav	er Sample	Database (SQLite)	💿 Views 🔻	- 🗔 EmpVie
	a 📴 Bookmarks Bookm 2017-11-03 22:27:58.4	06 Name		Valu				_	· ·
_	▷ □ 1 2017-11-03 22:27:01.0	31 Table Nam		Emp	View				
_	Customer DBeaver Sample Databa Table 783 2017-11-06 14:23:45.0	Table Type	c	VIEV	/				
_	11 Employeeld DBeaver Sample Databa Colum 469 2017-11-02 20:35:31.5	Catalog		VILV					
_	LastName DBeaver Sample Databa view E 502 2017-11-00 14:25:52.1	73 Schema							
_	Plavlist DBeaver Sample Databa Table P 783 2017-11-02 20:32:37.7	Table Desci	ription						
	Tables DBeaver Sample Databa Tables 1111 2017-11-06 14:27:20.1	57							
	ER Diagrams Diagra 2017-12-17 18:25:02.2	33							
	a 🗗 Scripts 2017-12-24 20:56:19.2	26							
	Gript-1.sql PostgreSQL - postgres select c 77 2017-12-21 20:34:29.0	85							
	ScriptA.sql DBeaver Sample Databa select*f 91 2017-12-24 20:28:39.1	15							
		٠		III					
-		Columns	Column Name	#	Data Type	Not Null	Auto Increment	Default Key	Descriptio
	·	Columns	ABC LastName	2	NVARCHAR(20)				
	😢 Database Navigator 🛛 🔪 👘 🤝 🧧	of Definition	ADC FirstName	3	NVARCHAR(20)				
1	Type part of object name to filter	-	ABC Title	4	NVARCHAR(30)				
	⊿ 📴 A		123 Reports To	5	INTEGER				
	Postgres		123 BirthDate	6	DATETIME				
	🔺 🔁 SQLite		123 HireDate	7	DATETIME				
	a 📑 DBeaver Sample Database (SQLite)		ABC Address	8	NVARCHAR(70)				
	Tables ()		ADC City	9	NVARCHAR(40)				
	⊿ 🖸 Views ()		ABC State	10	NVARCHAR(40)				
	⊳ roo EmpView		RBC Country	11	NVARCHAR(40)				
	p to test		PostalCode	12	NVARCHAR(10)				
	b El Sequences		HUC Phone	13	NVARCHAR(24)				
	Table Triggers		HDC Fax	14	INVARCHAR(24)				
	Data Types		mos email	15	INVARCHAR(00)				
	SQLite - New Connection								
		44.5				0 7	-	6 as 1 17b c	
		14 items				Q <b>Y</b>		<b>7 X   11</b> 294	∠e [ <sub>en</sub> Reve

😰 DBeaver 21.3.2 - <postgres> Script



## Database Stucture in DBeaver

When you open a database sheet, you can modify the database name and collation (character encoding, case sensivity, accent sensitivity, aphabetical order based on language). The default encoding is UTF-8.

postgres × Properties				
Name: DBA: Default Tablespace: Description:	postgres postgres pg_default default administr database	v ative connection	Default Encoding: Collate: Ctype: Connection Limit:	UTF8 Hungarian_Hungary.1250 Hungarian_Hungary.1250 Template Allow Connect
Schemas         Event Triggers         Extensions         Storage         System Info	Name          Image: Second state         Image: Secon	Comment system catalog schema standard public schema		

When you open a database, two important submenus appear in the navigation tree:

- 1. Schemas → public (default database schema). A Schema defines how data is organized within a relational database. It includes tables, fields, data types, and their relationships.
- 2. **Extensions** displays additional modules installed on your system. PostGIS is a key extension for handling spatial data (geometries).

To add a new extension: right-click on **Extensions**  $\rightarrow$  **Create New Extension**. If PostGIS is installed, find it in the list and enable it.

#### Viewing and Managing Tables in DBeaver



If you open **Schemas**, you can access **Tables**. To see the table details, make a double-click on table name or right click and **View Table**. This opens the table sheet, which consists of three tabs:

- 1. **Properties**: Table structure, settings, and metadata.
- 2. Data: View and edit table records.

₨ * <postgres> Co</postgres>	onsole 🖽 county 🖽 libra	rian ×									- 0
💷 Properties 🖪 I	Data 🚠 ER Diagram				🖷 postgres	Databases	🔹 🗟 library	🛅 Schemas	▼ ■ public	🛅 Tables	👻 📰 librarian
Table Name: lib	orarian		Object ID:	19608							
Tablespace: pg	_default v		Owner:	postgres							
	Has Row-Level Security Parti	tions	Extra Options:								
Partition by:											
Comment:			ī.								
		_				_					
					-	· · ·					
Bt Columns	Column Name	#	Data type		Identity	Collation	Not Nul	l Default			Comment
E Constraints	123 li_code	1	int4				[V]				
Constraints	123 li_deptcode	2	int4				[]				
Foreign Keys	A-z li_name	3	varchar			<u>default</u>	[]				
Indexes	∧z li_job	4	varchar			<u>default</u>	[]				
Dependencies	<ul> <li>Ii_enter</li> </ul>	5	date				[]				
References	Ii_exit	6	date				[]				
Partitions	123 li_salary	7	int4				[]				
Triggers	123 li_premium	8	float4				[]				
Rules											
- Rules											
Policies											
i Statistics											
B Permissions											
ot DDL											
C Virtual											

3. **ER Diagram**: Visual representation of table relationships. The ER Diagram provides an overview of database structure: it displays tables, columns, data types, primary and foreign keys. If the database contains primary and foreign keys, they will be represented in the ER Diagram.



#### Working with Table Columns

The **Columns** tab displays field names, data types, encoding settings, and NULL constraints. There are some submenus in the right of Editor window.

1. **Tables**: To create a new column: Right-click  $\rightarrow$  **Create new column**. Here you can set the fields of the table with data type, character encoding and Null or Not Null values.

Always **Save** changes after modifying the table structure!

**Constraints:** Primary key or unique key ensures data integrity.

Foreign keys: Defines relationships between tables.

**Cascade delete and update**: If a primary key value is modified or deleted, related foreign key records will also be updated/deleted.

**Restricted delete and update**: Prevents modifications or deletions in the parent table if they would leave orphaned foreign keys.

Indexes: Improves search performance.

► * <postares< p=""></postares<>	> Console	arian ×									- 0
Properties	Data 🚠 ER Diagram			ę	postgres	Databases	🕶 🗟 library	Schemas	▼	🖿 Tables	🕶 📰 librarian
Table Name: Tablespace:	librarian pg_default  Has Row-Level Security Parl	itions	Object ID: Owner: Extra Options:	19608 postgres		۵					
Partition by: Comment:		•				~	_				
Et Columns	Column Name	#	Data type		Identity	Collation	Not Null	Default			Comment
D. Constantist	123 li_code	1	int4				[V]				
Constraints	123 li_deptcode	2	int4				[]				
Foreign Key	/s A-z li_name	3	varchar			default	[]				
Indexes	A-Z li_job	4	varchar			default	[]				
Dependence	ies 🕗 li_enter	5	date				[]				
References	<ul> <li>Ii_exit</li> </ul>	6	date				[]				
Dartitions	123 li_salary	7	int4				[]				
	123 li_premium	8	float4				[]				
Iriggers											
Rules											
Policies											
i Statistics											
B Permissions											
→T DDL											
😳 Virtual											

## Editing Table Data

Each cell in the **Data** tab is editable. **If you made any modifications, do not forget to save the changes!** After modifying data, always click **Save** at the bottom of the Editor.

⊵ * <postgres> Console</postgres>				librarian ×						
III F	🗄 Properties 🖪 Data 💩 ER Diagram 🦉 postgres 🏮 Databases 💌 🗟 library 🛗 Schemas 💌 🗟 public 🛅 Ta								🛅 Tables 🔻 📰 librarian	
ΞI	ibraria	an 🕼 Enter a SQL	expression to filter	results (use Ctrl+Space)					▶ <b>▼</b>	🕭 🕶 🏹 🕶 🔤 📼
Srid	۲	123 - li_code	123 Ii_deptcode	A-z li_name	^₂ li_job ▼	🕗 li_enter 🔹	🕗 li_exit 🔹	123 li_salary	123 li_premium	
Ĩ	1	3	1	🗹 Noah Coleman	Customer Service Representative	1989-07-06	1996-12-31	[NULL]	[NULL]	ane
¥	2	4	1	William Baker	Customer Service Representative	2004-03-15	[NULL]	52,100	0.3	S S
Te	3	5	1	James Butler	Storekeeper	1997-10-30	[NULL]	83,500	0.2	00
	4	6	2	Oliver Cross	Customer Service Representative	2004-02-21	[NULL]	52,300	0.3	
	5	7	2	Dylan Carter	Storekeeper	1989-01-23	[NULL]	84,100	0.2	
	6	8	2	Jacob Bell	Customer Service Representative	1993-02-22	[NULL]	56,700	0.3	
	7	9	3	Luke Bennett	Storekeeper	2004-06-15	[NULL]	83,400	0.2	C.
	8	10	3	Robert Cooper	Storekeeper	2002-01-15	[NULL]	89,500	0.2	
	9	11	3	David Cooper	Administrator	[NULL]	[NULL]	95,000	[NULL]	
	10	12	4	Daniel Jackson	Storekeeper	2005-01-03	[NULL]	83,000	0.2	
	11	13	4	Jack O'Neil	Team Leader	1991-09-06	[NULL]	116,000	0.4	
	12	14	4	Samantha Carter	Customer Service Representative	2001-01-25	2005-02-01	[NULL]	[NULL]	
	13	15	5	Ben Browder	Customer Service Representative	2000-03-15	[NULL]	52,200	0.3	
	14	16	5	Claudia Black	Administrator	1996-05-12	[NULL]	95,800	0.1	
	15	17	5	Cristopher Judge	Team Leader	1994-08-18	[NULL]	114,500	0.4	
	16	18	6	Teryl Rothery	Team Leader	1993-01-01	[NULL]	115,300	0.4	
	17	19	6	Amanda Tapping	Customer Service Representative	1985-08-13	[NULL]	50,500	0.3	
	18	20	6	Vanessa Angel	Customer Service Representative	1996-05-06	[NULL]	50,100	0.3	
	19	21	7	Katie Stuart	Administrator	1992-03-05	[NULL]	95,400	0.15	
	20	22	7	Morena Baccarin	Team Leader	1993-02-16	[NULL]	116,100	0.3	
	21	23	7	Joe Flanigan	Customer Service Representative	1991-03-21	[NULL]	50,000	[NULL]	
	22	24	1	Jennifer Keller	Storekeeper	2004-01-01	[NULL]	115,200	0.4	
	23	25	1	Kyra Thompson	Customer Service Representative	1991-07-22	[NULL]	50,800	0.3	
	24	26	2	Kate Hewlett	Team Leader	1992-07-06	[NULL]	115,000	0.4	
	25	27	2	Laura Cadman	Administrator	2000-05-18	[NULL]	95,000	[NULL]	
	26	28	3	Michelle Morgan	Team Leader	2001-06-14	[NULL]	115,400	0.45	
	27	29	4	Kate Heightmeyer	Administrator	1994-11-01	[NULL]	95,700	0.18	
	28	30	5	Alison Porter	Customer Service Representative	1994-11-01	[NULL]	50,300	0.31	
ord	29	31	6	Jeannie Miller	Storekeeper	2003-05-19	[NULL]	80,600	0.25	
Rec										
1										
	🧐 R	efresh ▼ 🛛 🛇 S	ave 🔻 🗵 Cancel	=> == == == =K <	> > 🖂 🗄 🗄 Export data 👻 🕸	₿ _200 2	9 i 29 ro	w(s) fetched - 0.00	00s, on 2025-05-11 a	t 19:39:25

#### **Filtering Data**

III P	operties 🖶 Data 🍰 ER Diagram	<b>E</b>	postgres 🔋 Data	abases 🔻 🗟	library 🛛 🖻 Schen	nas 🔻 🗈 public	🛅 Tables 🔻 📰 librarian
ΞI	orarian 🕃 Enter a SQL expression to filter results (use Ctrl+Space)					▶   ▼	<b>∂</b> ▼ <b>₹</b> ▼ ⇔ ▼ ⇒ ▼
Grid		^z li_job ▼ (	Øli_enter ▼ 🔍	li_exit 🔹	123 li_salary	<sup>23</sup> li_premium	

Below the Properties and Data tabs, you can filter data using SQL conditions.

#### select \* from katalogus where CONDITION

The first part (select \* from katalogus) is pre-written, you only need to fill in the WHERE condition. Additional filtering options:

- Sort data (ascending/descending)  $\rightarrow$  Right-click the column header.
- Filter by column value  $\rightarrow$  Click the **funnel icon** in the column header.

▦	Prope	rties 🖪 Data 品	ER Diagram			🧏 post	gres 📴 Databases 🔻
	np_tk	Enter a SQL e	xpression to filter results (use Ctrl+Spac	e)			
Grid	۲	123 np_gid 🔹	∧z np_name ↓ -	.   	Order by np name ASC		•
⊞	1	380	Zselici Tájvédelmi Körzet	Ξİ.	Order by np. name DESC	557108447 108038	
¥	2	378	Zempléni Tájvédelmi Körzet	Disab	Disable order by np_name		274360618 351822
- e	3	379	Zempléni Tájvédelmi Körzet		Disable order by np_name	C. 1. 544	59974855 362839.8
	4	72	Vértesi Tájvédelmi Körzet	<b>V</b> I	Filter by value	Ctrl+F11	097611697 235919
atial	5	70	Vértesi Tájvédelmi Körzet	<b>T</b>	Cell value	>	82854513 227385.4
Sp	6	71	Vértesi Tájvédelmi Körzet	TI	Custom		513534005 230084
Ø	7	373	Tokaj-Bodrogzug Tájvédelmi Körzet				162582136 306893
	8	377	Tokaj-Bodrogzug Tájvédelmi Körzet	<b>T</b>	np_name IS NULL		259381299 322932
	9	376	Tokaj-Bodrogzug Tájvédelmi Körzet	<b>T</b>	np_name IS NOT NULL		026664014 314306
	10	374	Tokaj-Bodrogzug Tájvédelmi Körzet	Υ.	Save filter settings		817436953 307384
	11	375	Tokaj-Bodrogzug Tájvédelmi Körzet		Bomovo all filtors/orderings		762996676 312552
	12	372	Tarnavidék Tájvédelmi Körzet	0	Contraction (There		521915162 314682
	13	52	Tápió-Hajta Vidéke Tájvédelmi Körz.	Ш¢	Customize fliters		278622201 225098
	1.4	51	Tápiá Upita Vidáko Tájvádolmi Körz	st +6	ivádalmi kärzat MULTIDOLVGO	סללחלווו ואר	0 21 24 05 4 4 77 22 4 72 5

### Querying in the SQL Console

To execute SQL queries, open the SQL Console: It is available in the Menu bar  $\rightarrow$  SQL Editor  $\rightarrow$  Open SQL Console, or right-click on Tables  $\rightarrow$  Open SQL Console or Icon bar  $\rightarrow$  Click on the SQL icon  $\rightarrow$  Open SQL Console

**Important note**: Always set the focus on Schema Public before running queries, otherwise you may get a "failed" error message.

<u>File Edit N</u> avigate Se <u>a</u> rch	<u>s</u> ql	Editor	<u>D</u> atabase	<u>W</u> indow	<u>H</u> elp			
🗱 🔻 🛡 🎨 💘 🛄 SQL 👻		Open	SQL script			F3	;	p
🗟 Database Navigator 🛛 🫅 Pro	17	Last eo	dited SQL sci	ript				)_tk
Enter a part of object name here	17	New S	QL script		Ctrl+]		on	
v 🗈 Databases	▶_	Open	SQL console					
🗸 🗟 hungary		Panels					>	o_ti
🗸 📠 Schemas		Layout	t		:		0	
🗸 📑 public		-						
🗸 🖬 Tables		Select	active schen	na				
> 🎞 autopaly	45	Set co	nnection fro	m navigato	r			
> 🎫 autout		A						
> 🎫 belterule		Auto-s	sync connect	ion with ha	vigator			
> == elsorende	L I	Open	separate con	nection			>	-
> 🎞 folvo fel	ulet				1	60K	~	5

To execute a query, click the orange **Play** button.



Now, let's try an SQL query in the next Chapter.

# **Chapter 3. Basics SQL**

#### **SQL Basics 1**

Use the Chapter 4. to check the correctness of your queries.

1. Get familiar with the library database! Open each data tables and review its content! At the end of these lecture notes, you will find a reference and explanation for all columns.

	public	: 📰 borrower >	<										
⊞	Properties 💀 Data 🚠 ER Diagram 🦉 postgres 🗊 Databases 💌 🗟 library 🛗 Schemas 💌 🗟 public 🖿 Tables 💌 🖽 borrower												
	borro	wer 🕼 Enter a SQL	expression to filter resul	ts (use Ctrl+Space)					) v v 👌	• • • •	⇒ ▼		
Grid	۰	123 • re_code	123 © re_deptcode	A-Z re_identifier	A-Z re_name	123 re_postal_code	A-z re_city	A-Z re_address	A-z re_status	A-z re_type			
I	1	30	1	45836/1997	John Nolan	3,000	Miskolc	BAROSS SQUARE 15.	1	Н	ane		
¥	2	31	2	22366/1992	Tim Bradford	1,078	Budapest	PIROSKA STREET 20.	1	Н	5		
Te	3	32	2	41501/1995	Nyla Harper	1,181	Budapest	HAVANNA UTCA 11	1	Н			
5	4	33	3	1657/1991	Angela Lopez	3,000	Eger	BEM RAKPART 2.	1	Н			
	5	34	4	31173/1992	Jackson West	1,182	Budapest	NAGYSZEBEN UTCA 33	1	Н			
	6	35	5	31214/1992	Lucy Chen	1,092	Budapest	CSERESZNYE UTCA 20	1	0			
	7	36	5	26613/1992	Wade Grey	1,213	Budapest	SZENT LAJOS UTCA 107	1	Н			
	8	37	7	39493/1994	Wesley Evers	1,149	Budapest	EGRESSY UTCA 13	1	Н			
	9	38	7	41678/1995	Talia Bishop	2,120	Dunakeszi	APOR UTCA 53.	I.	Н			
	10	20	7	62/11/1000	Pailov Nuno	2 210	Parec	TOMDA LITCA 2	1	LL			

SQL is a standard language for querying, modifying and manipulating databases.

An SQL statement looks like this:

```
select COLUMN_NAME(S) from TABLE_NAME
```

After select, write the field name(s), or use \* to retrieve all column in the result table.

2. List the identifier, purchase date, and the price for all publications from the catalogue table!

#### Ordering in the result table

In ascending order:

select COLUMN\_NAME from TABLE\_NAME order by COLUMN\_NAME asc
or simply:

select COLUMN\_NAME from TABLE\_NAME order by COLUMN\_NAME
In descending order:

select COLUMN NAME from TABLE NAME order by COLUMN NAME desc

3. List the ISBN number, title and the author from the publications table, in descending order by title!

#### The WHERE condition

If you want to filter the data, use a where condition in your query.

select COLUMN\_NAME from TABLE\_NAME where CONDITION order by COLUMN\_NAME

A condition consists of a column name, a relational operator and a value. Example:

#### ca\_price>800

Relational operators include: <, >, <=,>=, =, and <> (not equal).

Values can be numbers (integer e.g. 12 or float/real e.g. 12.345), or text (e.g. 'this is a text type data') and dates (e.g. '2024-09-15' or '2024.09.15').

- 4. List the ISBN number and the year of publication from the publications table!
- 5. Query those catalogue items that cost more than 3,000 Forints. List the inventory number and price!
- 6. Query those catalogue items, that are cheaper than 3,000 Forints. Print the inventory number, the price and the publication group code. Order the results by price in descending order.
- 7. Query the primary key and the title of publications that do not belong to publication group number 3!

#### AND / OR operators

Sometimes you need to apply multiple conditions. Use and and or operators combine them.

and: All conditions must be true.

or: At least one condition must be true.

```
select COLUMN_NAME from TABLE_NAME where CONDITION1 and CONDITION2
select COLUMN_NAME from TABLE_NAME where CONDITION1 or CONDITION2
```

#### Exercises

8. Print the data of catalogue items that are more expensive than 9,000 Forints and are marked as non-borrowable!

9. Print all data of reserved items from "borrowing" table!

10. Print all data of catalogue items that were purchased before 2004-03-01!

11. Print data of loans (borrowings) that were made after 2004-03-01!

12. List the inventory number, publication code, and status code of catalogue items, belonging to publication group 4!

13. List the name, job and date of entering of the librarians, ordered by the date in ascending order!

14. Print all data from the librarian table. Order the result by job in ascending order, and by the librarian name in descending order.

#### DISTINCT

The distinct keyword in SQL is used to eliminate duplicate values in the result set. It ensures that only unique (non-duplicated) records are returned for the columns specified in the query.

select distinct COLUMN\_NAME from TABLE\_NAME

15. Query the distinct job titles from the librarian table! (Every job should appear only once.)

#### Practice

16. Query librarians with the job title "Storekeeper" and "Customer Service Representative". Order the results by date of entering in descending order!

17. List the librarians whose salary is more than 50,000 Forints whose job is "Storekeeper". Order the results by date of entering in ascending order.

18. List librarians whose salary is less than 60,000 or more than 100,000.

19. List librarians who work in the department 2 or 5.

20. List the departments that are not located in Budapest!

21. List the department's locations from the department table! Every locations should appear only once!

#### **BETWEEN / NOT BETWEEN operators**

The between operator selects values within a given range. The values can be numbers or dates.

The between operator is inclusive: beginning and end values are included.

not between returns the values that are outside of the given range.

Examples:

where column\_name between value A and value B
where column\_name not between value A and value B

- 22. List the librarians whose salary is between 50,000 and 100,000!
- 23. List the librarians whose salary is not between 50,000 and 100,000!

#### **IN / NOT IN operators**

The in operator allows you to specify multiple values in a where clause. not in returns the values that are not in the specified list.

where column\_name in (value A, value B)
where column\_name not in (value A, value B)

24. List the librarians whose job is Storekeeper and Team Leader!

25. List the librarians whose job is neither Storekeeper nor Team Leader!

#### **Testing for NULL value (no data value)**

The is null operator is used to check if a column contains a null value. It is commonly used in where clauses to filter records where a column has no value (i.e., it is null).

is not null returns those records where the column contains a value.

where column\_name is null
where column name is not null

#### SQL ALIASES

SQL aliases are used to temporarily rename a column or table. They are often used to make column names more readable and exist only for the duration of the query.

column\_name AS new\_column\_name

26. Which publications were returned? (Check where there is a date in bo\_indate column in the borrowing table?)

27. Display the name, the job and the income (salary+premium calculates as a percentage of salary) of the librarians! Rename the column to salary.

#### LIKE / NOT LIKE Operators

The like operator is used in a where clause to search for a specified pattern in a column.

- % represents zero, one, or multiple characters
- \_\_\_\_\_ represents a single character

'NOT LIKE' returns the results that do not match the specified pattern.

```
where column_name like 'j%'
where column_name not like 'j%'
```

28.List the names of librarians whose names start with J!

- 29. List the names of librarians whose second character is A!
- 30. List the names of librarians whose last character is R!
- 31. List the names of department that contain 'geo'!

#### Let's do these excercises

32. List the librarians whose job is Storekeeper and Team Leader and whose salary is between 60,000 and 100,000!

33. List the librarians who do not receive a salary!

34. List the publications whose title contains 'atlasz'!

35. Create a list about the catalogue items with a price between 10,000 and 30,000 and a purchase date before 2002-01-01!

36. List the publications that have no author and were published in the year 2000.

#### **Simple functions**

MySQL offers many built-in functions that help in working with different kinds of data such as text, number (integers and floats) and dates.

What does a function look like?

```
functionname(value1, value2, etc.)
```

#### **Examples:**

```
Upper('dog') \rightarrow and the result is: DOG.
Lower('CAT') \rightarrow cat
Lower \rightarrow Converts a string to lowercase,
```

Upper  $\rightarrow$  Converts a string to uppercase.

37. Display the titles of publications in lowercase, and the publishers in uppercase!

#### CONCAT

Concat() joins two or more strings into one.

#### **Example:**

#### Concat('large ','black ','dog') → large black dog

38. Create a list of borrowers from Eger! Show the name, the identifier (ID), and the date of entering. Add the word 'CUSTOMER' before each name.

#### CHAR\_LENGTH

Char\_Length() returns the number of characters in a string.

#### Char\_Length('dog') $\rightarrow$ 3

39. Show the length of each borrower's name!

#### SUBSTRING

**Substring**(string, start, length) extracts a part of a string. The length is optional; if omitted, it returns the rest of the string.

#### **Examples:**

Substring('black cat', 3, 2)  $\rightarrow$  ac Substring('black cat', 3)  $\rightarrow$  ack cat

#### **LEFT and RIGHT**

**Left**(string, n)  $\rightarrow$  Returns the first n characters of a string.

**Right**(string, n)  $\rightarrow$  Returns the last n characters of a string. **Examples:** 

```
Left('black cat', 3) → bla
Right('black cat', 3) → cat
```

#### STRPOS

In PostgreSQL, StrPos(string, substring) returns the position of a substring in a string.

```
StrPos('black cat',' ') \rightarrow 6
```

Tasks:

- 40. Print characters from 3 to 7 from the borrower's name!
- 41. Print borrower's name starting from the 5th character!
- 42. Print the first 5 characters of the borrower's name!
- 43. Print the last 5 characters of the borrower's name!
- 44. Print the position of the 'space' character from the borrower's name!
- 45 Print the borrower's first and last name in separate columns.

46. Same as 45, but display the first name in lowercase and the last name in uppercase.

47. Show the length of the publication titles using char\_length and octet\_length! Explain the difference.

#### Math functions

48. Run and explain these functions!

Abs(1000), Abs(-100) and Abs(-27.11)  $\rightarrow$  Returns the absolute value of a number

Sqrt(25), Sqrt(-36)  $\rightarrow$  square root (watch for invalid input).

Power(2,5)  $\rightarrow$  Power(2^5 = 32)

Round(-123.456,2), Round(123.456,0), Round(123.456,-2), Round(-123.4567,2)  $\rightarrow$  Rounding with different precision. It rounds a number to a specified number of decimal places (the second parameter is positive number). If the second parameter a negative number, it rounds to tens, hundreds, thousends etc. 0 round to integer.

SIGN(10), SIGN(0), SIGN(-5)  $\rightarrow$  Returns the sign of a number.

CEIL(6.1), CEIL(-6.1)  $\rightarrow$  Round up to the nearest integer

FLOOR(6.1), FLOOR(-6.1)  $\rightarrow$  Round down to the nearest integer

PI() → 3.1415...

#### Division

49. Try the following and explain:

10/3 What do you get, integer or decimal number? - Use ::decimal or ::integer to declare the data type!

10%3 What is this? (Remainder (modulus))

#### **Date functions**

You can subtract dates to get the number of days between them.

'2024-09-27'-'2024-09-20' → 7 days NOW()

Returns the current date and time with time zone:

#### 2025-02-14 12:58:02.623 +0100

#### Tasks:

- 50. Show borrowing records where the duration was less than one week!
- 51. Show borrowing records where the duration was more than one week.
- 52. How many days have passed since your birth = How old are you in days?

#### EXTRACT

Used to get specific parts of a date or time:

```
Extract(year from Now())
Extract(month from Now())
```

Extract(day from Now())
Extract(hour from Now())
Extract(minute from Now())
Extract(second from Now())
Extract(dow from Now()) or Extract(dow from DATE '1989-01-13') → day of week,
Extract(timezone\_hour from now())
53. Extract the year/month/day/hour/minute/second from today!

- 54. Extract only the days from the query in task 50.
- 55. What day of the week is today?
- 56. What day of week were you born?

#### **Group By**

The group by statement groups rows that have the same values into summary rows, like "find the number of customers in each country". An aggregate function is a function that performs a calculation on a set of values and returns a single value. Aggregate functions are often used with the group by clause of the select statement. The group by clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

#### **Common aggregate functions:**

The group by statement is often used with aggregate functions (Count(), Max(), Min(), Sum(), Avg()) to group the result set by one or more columns.

- Count()  $\rightarrow$  Returns the number of rows that match a condition.

Most aggregate functions ignore NULL values (except Count(\*), which includes all rows).

- Min()  $\rightarrow$  Returns the smallest value in a column.
- $Max() \rightarrow$  Returns the largest value in a column.
- Sum()  $\rightarrow$  Returns the total of a numeric column.
- $Avg() \rightarrow$  Returns the average of a numeric column.

#### Tasks

57. Display the minimum, maximum, average, and the total salary from the librarian table! How many librarians are there?

- 58. What was the lowest and highest price in the catalogue table?
- 59. How many borrowers are from Budapest?
- 60. Show the oldest and the most recent purchase dates in the catalogue table!
- 61. How many different jobs are there in the librarian table?

#### The order of clauses are the following:

select... from ... where... group by... having... order by ...

62. Display the minimum, maximum, average, and the total salary from the librarian table! Group the results by job title. Also show how many librarians are in each group.

63. Do the same as in Task 62, but group by department code. Order the results by department code.

64. Show the code of each publication group, and the number of publications in each group from the catalogue table!

65. Show the number of publications and the average price for each department from the catalogue table! Group the data by department!

66. What was the average borrowing duration (in days) for all borrowings before 2004-05-30?

67. What is the average borrowing duration (in days) for the publication titled *KISATLASZ*? Group by librarian.

#### HAVING

The having clause is used to filter groups created by group by. Unlike where, it can be used with aggregate functions.

68. Show the minimum, maximum, average, and total salary from the librarian table, grouped by job. How many people are in the librarian table? Show only those groups where the average salary is greater than 80,000 Forints.

69. Show the code of departments, where there are exactly 3 librarians.

70 Group librarians by the length of their names. How many librarians are in each group? Also display the average salary. Order the results by average salary.

71. For each job title, calculate the difference between the maximum and minimum salaries.

#### LIMIT

limit 1 – Returns only one row from the result set.

72. Show only the group with the biggest difference!

#### **Table Joins**

A join clause is used to combine rows from two or more tables, based on a related column between them A way to join tables, can be **primary key–foreign key** relationships definition. **Primary key** uniquely identifies each record in a table. n most cases, it is a sequential integer. **Foreign key** refers to the primary key of another table, creating a connection between the two.

Let's see an example in the library database. The department table contains a primary key (dept\_code). The librarian table has its own primary key (li\_code) and a foreign key (li\_deptcode), which refers to the department table primary key (department.dept\_code). This means that if *Noah Coleman* has li\_deptcode = 1, and department 1 is "Department of Cartography", then Noah Coleman works in the Department of Cartography.

Relational databases define relationships between tables through primary key – foreign key pairs.

2 <u>3</u>	i_code		123 li_d	eptcode	•	A-z li_name	•
		3			1 🖉	Noah Coleman	
		4			1 🖻	William Baker	
		5			1 🖻	James Butler	
		6			2 🖉	Oliver Cross	
		7			2 🖉	Dylan Carter	
		8			2 🖉	Jacob Bell	
		9			3 🗹	Luke Bennett	
	123 dept_c	od	e 🔻	A-z dept_	name	• •	
	1			Department of Cartography			
	2			Department of Geography			
			3	Departm	nent o	f Geophysics	

#### Foreign key – primary key relationships in the Library database:

dept\_code - li\_deptcode: Which department the librarian works in (department and librarian tables)

pg\_code - p\_pgcode: the group code of a publication (publication\_group and publication tables)

pg\_code - ca\_pgcode: the group code of a catalogue item (publication\_group and catalogue tables)

p\_code - ca\_pcode: Link between publication and catalogue item (publication and catalogue tables)

dept\_code - ca\_deptcode: Which department owns the catalogue item (department and catalogue tables)

dept\_code - re\_deptcode: Which department the borrower belongs to (department and borrower tables)

re\_code - bo\_recode: Which borrower made the borrowing (borrower and borrowing tables)

li\_code - bo\_licode: Which librarian handled the borrowing (librarian and borrowing tables)

ca\_code - bo\_cacode: which catalogue item was borrowed (catalogue and borrowing tables)

p\_code - bo\_pcode: which publication was borrowed (publication and borrowing tables)

The following diagram shows this structure: white diamonds represent primary keys, and filled circles represent foreign keys.

35



#### How to Write a JOIN Query

There are two standard ways to join tables. Do not mix them in one query! Implicit join syntax (older style):

select FIELD\_NAMES from TABLE1, TABLE2 where FOREIGN\_KEY=PRIMARY KEY

#### **Explicit JOIN syntax (recommended):**

select FIELD\_NAMES from TABLE1 join TABLE2 on FOREIGN\_KEY=PRIMARY KEY

#### Examples

73. Display the librarian's name, job title and the name of the department where they work. Order the results by the librarian's name.
74. Display all librarians who work in Sopron.

75. Print the title, author and publisher of publications, which price is between 3,000 and 10,000 Forints!

76. Print the names of borrowers who reserved a publication (bo type='F')!

77. Print the lowest and the highest salary, grouped by the department name!

78. Print the average salary and the number of librarians, grouped by city!

#### Joining Three or More Tables

To join three or more tables, identify the correct primary key-foreign key pairs between them.

Example syntax (implicit):
select FIELD\_NAMES from TABLE1, TABLE2, TABLE3 where FOREIGN\_KEY=PRIMARY KEY and
FOREIGN\_KEY=PRIMARY KEY

Example syntax (explicit join):

select FIELD\_NAMES from TABLE1 join TABLE2 on FOREIGN\_KEY=PRIMARY KEY join TABLE3
on FOREIGN\_KEY=PRIMARY KEY

#### More Advanced Exercises:

79. Display the inventory number and price of a publication, the name of borrower, full address (city, postal code and address), and the borrowing date but only for borrowings before 2004-05-01.

80. Print the borrower's name, the title of the publication, the name of the librarian and the borrowin code!

81. Display the borrowed publication's title, inventory number, and borrower's name and the date of borrowing!

82 Print the names of borrowers who borrowed the publication titled 'KISATLASZ'. Show the borrower's name, title of the publication, and borrowing date.

83. Show the borrower's name and the borrowing date (in- and out dates) where the librarian's name was Oliver Cross.

#### Left and Right JOINS

The left join keyword returns all records from the left table (table1), and the matching records from the right table (table2). If there is **no match**, the result will still include the row from the left table, and NULL values for the right table.

The right join keyword does the opposite: it returns all records from the right table, and the matching records from the left table.

If there is **no match**, the result will still include the row from the right table, and NULL values for the left table.

For visual examples, visit: <u>W3Schools SQL JOINs</u>

84. Print the publications and their publication groups. If the group has no publications, it should still appear in the list.

#### **Subqueries**

A **subquery** is a query nested within another SQL query. Subqueries are often used inside the WHERE clause and are placed in **round brackets** ().

Let's take a look at the next example:

85. Print the librarians, whose salary is higher than that of of Alison Porter.

This can be solved in two steps: First, you have to know how much money Alison Porter earned. Then, you can compare this value with another librarian's salary.

```
select li_salary from librarian where li_name='Alison Porter'
→The value is 50,300.
select * from librarian where li salary>50300
```

Or it can be solved in **one query using a subquery**:

```
select * from librarian where li_salary>(select li_salary from librarian where
li_name='Alison Porter')
```

#### **More Subquery Exercises**

86. Print the borrower's name and code, whose city is same as Aaron Thorsen's!

87. Print those publications, that are cheaper than the publication titled 'FÖLDRAJZI VILÁGATLASZ'

88. Print those librarians, whose job is the same as Samantha Carter's job.

89. Print those publications that were published earlier than 'Cartography Thematic Map Design'!

#### If the subquery returns multiple results, use the IN operator instead of =.

90. Print the code, name and job of the librarians whose job is the same as any librarian working in the Dept. of Cartography.

91. Print the departments that are in the same city as both the Dept. of Geophysics and the Dept. of Geodesy.

92. List borrowings where the price of the borrowed publication is greater than the average price.

93. Print the borrowings, where the borrowing time was the longest.

94. Print the borrowings, where the borrowing time was the shortest.

95. Print the publications where the title length is longer than hald of the longest title in the table.

96. Print those librarians who earn more than the average salary in Budapest.

97. Print the librarians from Budapest who earn more than the overall average salary.

# Chapter 4. KEY TO BASIC SQL Queries

```
2. select ca_inventory_number, ca_indate, ca_price from catalogue
3. select p_title, p_author, p_isbn from publications order by p_title desc
4. select p_isbn, p_year from publications
5. select ca_inventory_number, ca_price from catalogue where ca_price >3000
6. select ca_inventory_number, ca_price, ca_pgcode from catalogue where ca_price
<3000 order by ca_price desc
7. select p_code, p_title from publications where p_pgcode !=3
8. select * from catalogue where ca price>9000 and ca status ='N'
9. select * from borrowing where bo_type='F'
10. select * from catalogue where ca indate<'2004-03-01'
11. select * from borrowing where bo outdate <'2004-03-01'
12. select ca_inventory_number, ca_code, ca_status from catalogue where ca_pgcode
= 4
13. select li_name, li_job, li_enter from librarian order by li_enter
14. select * from librarian order by li_job, li_name desc
15. select distinct li_job from librarian
16. select * from librarian where li_job='Storekeeper' or li_job='Customer Service
Representative' order by li_enter desc
select * from librarian where li_job in ('Storekeeper', 'Customer Service
Representative') order by li enter desc
17. select * from librarian where li job ='Storekeeper' or li salary >50000 order
by li enter asc
18. select * from librarian where li_salary <60000 or li_salary >100000
19. select li_name from librarian where li_deptcode in (2,5)
select li_name from librarian where li_deptcode =2 or li_deptcode =5
20. select dept name from department where dept city <>'Budapest'
21. select distinct dept_city from department
22. select * from librarian where li_salary between 50000 and 100000
23. select * from librarian where li_salary not between 50000 and 100000
24. select * from librarian where li_job in ('Storekeeper','Team Leader')
25. select * from librarian where li_job not in ('Storekeeper','Team Leader')
26. select * from borrowing where bo_indate is not null
27. select li_name, li_job, li_salary+li_premium*li_salary as salary from
librarian
28. select li name from librarian where li name ilike 'j%'
29. select li name from librarian where li name ilike '_A%'
30. select li_name from librarian where li_name ilike '%R'
31. select dept name from department where dept name ilike '%geo%'
32. select * from librarian where li_job in ('Storekeeper', 'Team Leader') and
li_salary between 60000 and 90000
33. select * from librarian where li_premium is null
34. select * from publications where p_title ilike '%atlasz%'
35. select * from catalogue where ca_price between 10000 and 30000 and
ca indate<'2002-01-01'
36. select * from publications where p_author is null and p_year=2000
37. select Lower(p_title), Upper(p_publisher) from publications
38. select Concat('CUSTOMER: ', re_name), re_identifier, re_enter from borrower
where re city='Eger'
39. select re_name, Char_length(re_name) from borrower
40. select re_name, Substring(re_name,3,5) from borrower
41. select re_name, Substring(re_name, 5) from borrower
42. select re_name, Substring(re_name,1,5), Left(re_name, 5) from borrower
43. select re_name, Right(re_name, 5) from borrower
```

```
44. select re_name, StrPos(re_name,' ') from borrower
45. select Substring(re_name, 1, StrPos(re_name,' ')-1) as givenname,
Left(re_name, StrPos(re_name, ' ')-1) as givenname, Substring(re_name,
StrPos(re_name, ' ')+1) as familyname, Right(re_name, Char_Length(re_name)-
StrPos(re_name, ' ')) as familyname from borrower
46. select Lower(Substring(re_name, 1, StrPos(re_name, ' ')-1)) as givenname,
Upper(Substring(re_name, StrPos(re_name, ' ')+1)) as familyname from borrower
47. select p_title, Char_Length(p_title), Octet_Length(p_title) from publications
Octet length gives the binary length (accent characters have duplicated the
length), the char length give the real character length.
49. select 10/3 \rightarrow 3 we got an integer
   select 10/3::integer
   select 10/3::decimal \rightarrow we got the decimal number
   select 10%3 \rightarrow1 this is the modulo operation returns the remainder or signed
remainder of a division.
50. select * from borrowing where (bo_indate - bo_outdate) <7
51. select * from borrowing where (bo_indate - bo_outdate) >7
52. select now()-'1989-01-13'
53. select extract(year from now())
54. select extract(day from (now()-'1989-01-13'))
55. select extract(dow from now())
56. select extract(dow from date '1989-01-13')
57. select Min(li_salary), Max(li_salary), Avg(li_salary), Sum(li_salary),
Count(*) from librarian
58. select Min(ca_price), Max(ca_price) from catalogue
59. select Count(*) from borrower where re_city='Budapest'
60. select Min(ca_indate), Max(ca_indate) from catalogue
61. select Count(distinct li_job) from librarian
62. select li_job, Min(li_salary), Max(li_salary), Avg(li_salary), Sum(li_salary),
Count(*) from librarian group by li job
or
select li_job, Min(li_salary), Max(li_salary), Avg(li_salary), Sum(li_salary),
Count(*) from librarian group by 1
63. select li_deptcode, Min(li_salary), Max(li_salary), Avg(li_salary),
Sum(li_salary), Count(*) from librarian group by 1 order by 1
64. select ca_pgcode, Count(*) from catalogue group by 1
65. select ca_deptcode, Count(*), Avg(ca_price) from catalogue group by 1
66. select Avg(bo indate-bo outdate) from borrowing where bo outdate <'2004-05-30'
67. select bo licode, Avg(bo indate-bo outdate) from borrowing where bo pcode=1
group by 1
68. select li_job, Min(li_salary), Max(li_salary), Avg(li_salary), Sum(li_salary),
Count(*) from librarian group by li_job having Avg(li_salary)>80000
69. select li_deptcode from librarian group by 1 having count(*)=3
70. select Char Length(li name), Count(*), Avg(li salary) from librarian group by
1 order by 3
71. select li job, Max(li salary)-Min(li salary) from librarian group by 1
72. select li_job, Max(li_salary)-Min(li_salary) from librarian group by 1 order
by 2 desc limit 1
73. select li_name, li_job, dept_name, dept_city from librarian, department where
li deptcode=dept code order by 1
or
select li_name, li_job, dept_name, dept_city from librarian join department on
li deptcode=dept code order by 1
74. select li name, li job, dept name, dept city from librarian, department where
li deptcode=dept code and dept city='Sopron'
or
select li_name, li_job, dept_name, dept_city from librarian join department on
li_deptcode=dept_code where dept_city='Sopron'
```

75. **select** p\_title, p\_author, p\_publisher **from** publications, catalogue **where** ca pcode =p code and ca price between 3000 and 10000 or select p title, p author, p publisher from publications join catalogue on ca pcode =p\_code where ca\_price between 3000 and 10000 76. select re\_name from borrower, borrowing where re\_code=bo\_recode and bo\_type='F' or select re name from borrower join borrowing on re code=bo recode where bo type='F' 77. select dept name, Min(li salary), Max(li salary) from department, librarian where li deptcode=dept code group by 1 or select dept\_name, Min(li\_salary), Max(li\_salary) from department join librarian on li\_deptcode=dept\_code group by 1 78. select dept city, Avg(li salary), Count(\*) from department, librarian where li deptcode=dept code group by 1 or select dept\_city, Avg(li\_salary), Count(\*) from department join librarian on li\_deptcode=dept\_code group by 1 79. select ca\_inventory\_number, ca\_price, re\_name, re\_address, re\_city, re\_postal\_code, bo\_indate from catalogue, borrowing, borrower where re\_code= bo\_recode and bo\_cacode=ca\_code and bo\_indate<'2004-05-01'</pre> or select ca\_inventory\_number, ca\_price, re\_name, re\_address, re\_city, re postal code, bo indate **from** catalogue **join** borrowing **on** bo cacode=ca code **join** borrower on re\_code= bo\_recode where bo\_indate<'2004-05-01'</pre> 80. select re\_name, p\_title, li\_name, bo\_code from publications, borrowing, borrower, librarian where bo pcode=p code and re code= bo recode and li\_code=bo\_licode or select re\_name, p\_title, li\_name, bo\_code from publications join borrowing on bo\_pcode=p\_code join borrower on re\_code= bo\_recode join librarian on li code=bo licode 81. select p\_title, ca\_inventory\_number, re\_name, bo\_indate, bo\_outdate from borrower, borrowing, catalogue, publications where re\_code=bo\_recode and bo\_cacode=ca\_code and bo\_pcode=p\_code or select p title, ca inventory number, re name, bo indate, bo outdate from borrower join borrowing on re code=bo recode join catalogue on bo cacode=ca code join publications on bo pcode=p code 82. select re\_name, p\_title, bo\_indate, bo\_outdate from borrower, borrowing, publications where bo recode=re code and bo pcode=p code and p title='KISATLASZ' or select re name, p title, bo indate, bo outdate from borrower join borrowing on bo\_recode=re\_code join publications on bo\_pcode=p\_code where p\_title='KISATLASZ' 83. select re\_name, bo\_indate, bo\_outdate from borrower join borrowing on bo\_recode=re\_code join librarian on bo\_licode=li\_code where li\_name='Oliver Cross' or select re\_name, bo\_indate, bo\_outdate from borrower, borrowing, librarian where bo\_recode=re\_code and bo\_licode=li\_code and li\_name='Oliver Cross' 84. select \* from publication group *left* join publications on pg code =p pgcode or select \* from publications right join publication\_group on pg\_code =p\_pgcode 85. select \* from librarian where li salary>(select li salary from librarian where li name='Alison Porter') 86. select \* from borrower where re\_city=(select re\_city from borrower where re name='Aaron Thorsen')

87. select \* from publications, catalogue where ca\_pcode=p\_code and ca\_price>(select ca\_price from publications, catalogue where ca\_pcode=p\_code and p title='FÖLDRAJZI VILÁGATLASZ') 88. select \* from librarian where li\_job=(select li\_job from librarian where li\_name='Samantha Carter') 89. select \* from publications where p\_year <(select p\_year from publications where p\_title='Cartography Thematic Map Design') 90. select \* from librarian where li\_job in (select li\_job from librarian, department where li deptcode=dept code and dept name='Department of Cartography') 91. select \* from department where dept city in (select dept city from department where dept\_name in ('Department of Geodesy', 'Department of Geophysics')) 92. select publications.\* from publications, catalogue where ca pcode=p code and ca\_price>(select Avg(ca\_price) from catalogue) 93. select \* from borrowing where (bo\_indate-bo\_outdate)=(select Max(bo\_indatebo outdate) from borrowing ) 94. select \* from borrowing where (bo\_indate-bo\_outdate)=(select Min(bo indatebo\_outdate) from borrowing ) 95. select \* from publications where Char Length( p title) > (select Max(Char\_Length(p\_title))/2 from publications) 96. select \* from librarian where li\_salary >(select Avg(li\_salary) from librarian, department where li\_deptcode=dept\_code and dept\_city='Budapest') 97. select \* from librarian, department where li\_deptcode=dept\_code and dept city='Budapest' and li salary >(select Avg(li salary) from librarian)

# Chapter 5: Comparision sheet to PostgreSQL and MySQL

If you have previously learned MySQL, you may notice some differences when working with PostgreSQL. The following table compares the two SQL dialects and highlights their key differences.

Function,		
operator, clause	MySQL Example	PostgreSQL Example
ORDER BY two fields	<pre>select * from publications order by p_year desc, p_publisher;</pre>	<pre>select * from publications order by p_year desc, p_publisher</pre>
WHERE with one condition	<pre>select * from catalogue where ca_price &gt;2000;</pre>	<pre>select * from catalogue where ca_price &gt;2000;</pre>
Text type data in SQL	<pre>select * from librarian where li_job='Team Leader' select * from librarian where li_job="Team Leader"</pre>	<pre>select * from librarian where li_job='Team Leader' Note: Use single quotes. Case sensitivity depends on collation.</pre>
Date type data in SQL	<pre>select * from librarian where li_enter&gt;='1992-01-03' select * from librarian where li_enter&gt;='1992.01.03' select * from librarian where li_enter&gt;="1992-01-03" select * from librarian where li_enter&gt;="1992.01.03"</pre>	<pre>select * from librarian where li_enter&gt;='1992-01-03' select * from librarian where li_enter&gt;='1992.01.03' Note: Use ISO format 'YYYY-MM-DD'.</pre>
IN	<pre>select * from librarian where li_job in ('Storekeeper', 'Team Leader')</pre>	<pre>select * from librarian where li_job in ('Storekeeper', 'Team Leader')</pre>
NOT IN	<pre>select * from librarian where li_job not in ('Storekeeper', 'Team Leader')</pre>	<pre>select * from librarian where li_job not in ('Storekeeper', 'Team Leader')</pre>
BETWEEN	<pre>select * from publications where p_year between 2001 and 2005</pre>	<pre>select * from publications where p_year between 2001 and 2005</pre>
NOT BETWEEN	<pre>select * from publications where p_yeaar not between 2001 and 2005</pre>	<pre>select * from publications where p_yeaar not between 2001 and 2005</pre>
LIKE Case Sensitive!	<pre>select * from publications where p_title like '%ATLASZ%'</pre>	<pre>select * from publications where p_title like '%ATLASZ%'</pre>
		case Sensitive, see the explanation above!
NOT LIKE Case Sensitive!	<pre>select * from publications where p_title not like '%ATLASZ%'</pre>	<pre>select * from publications where p_title not like '%ATLASZ%' Case Sensitive,see the explanation above!</pre>
ILIKE case- insensitive!	PostgreSQL only	<pre>select * from publications where p_title ilike '%atlas%' Use ILIKE / NOT ILIKE in PostgreSQL for case-insensitive search.</pre>
NOT ILIKE <mark>case-</mark> insensitive!	PostgreSQL only	<pre>select * from publications where p_title not ilike '%atlas%' Use ILIKE / NOT ILIKE in PostgreSQL for case-insensitive search.</pre>
IS NULL	<pre>select * from publications where p_isbn is null</pre>	<pre>select * from publications where p_isbn is null</pre>

IS NOT NULL	<pre>select * from publications where p_isbn is not null</pre>	<pre>select * from publications where p_isbn is not null</pre>
AND	<pre>select * from publications where p_isbn is not null and p_publisher='ESRI PRESS'</pre>	<pre>select * from publications where p_isbn is not null and p_publisher='ESRI PRESS'</pre>
OR	<pre>select * from publications where p_isbn is not null or p_publisher='ESRI PRESS'</pre>	<pre>select * from publications where p_isbn is not null or p_publisher='ESRI PRESS'</pre>
DISTINCT	<pre>select distinct li_job from librarian</pre>	<pre>select distinct li_job from librarian</pre>
UPPER()	<pre>select Upper(li_name) from librarian</pre>	<pre>select Upper(li_name) from librarian</pre>
LOWER()	<pre>select Lower(li_name) from librarian</pre>	<pre>select Lower (li_name) from librarian</pre>
CONCAT()	<pre>select Concat('ISBN number: ', p_isbn) from publications</pre>	<pre>select Concat('ISBN number: ', p_isbn) from publications</pre>
COALESCE()	<pre>select Coalesce(p_isbn,'****') from publications</pre>	<pre>select Coalesce(p_isbn,'****') from publications</pre>
SUBSTRING()	<pre>select Substring(li_name, 2,3) from librarian</pre>	<pre>select Substring(li_name, 2,3) from librarian</pre>
SUBSTRING()	<pre>select Substring(li_name, 2) from librarian</pre>	<pre>select Substring(li_name, 2) from librarian</pre>
SUBSTRING()	<pre>select Substring(li_name, -2,1) from librarian</pre>	<ul> <li>Negative indexes not supported in PostgreSQL, do it with RIGHT()</li> </ul>
SUBSTRING()	<pre>select Substring(li_name, -2) from librarian</pre>	<ul> <li>Negative indexes not supported in PostgreSQL, do it with RIGHT()</li> </ul>
LEFT()	<pre>select Left(li_name, 3) from librarian</pre>	<pre>select Left(li_name, 3) from librarian</pre>
RIGHT()	<pre>select Right(li_name, 3) from librarian</pre>	<pre>select Right(li_name, 3) from librarian</pre>
<mark>INSTR()</mark> STRPOS()	<pre>select Instr(li_name,' ') from librarian</pre>	<pre>select StrPos(li_name,' ') from librarian</pre>
LENGTH()	<pre>select li_name, Length(li_name) from librarian /binary length: accent character has duplicated length/</pre>	<pre>select li_name, Length(li_name) from librarian Number of characters Octet_Length(string) is the same as the Length() in MySQL: for byte length</pre>
CHAR_LENGTH()	<pre>select li_name, Char_Length(li_name) from librarian real number of characters</pre>	<pre>select li_name, Char_Length(li_name) from librarian real number of characters</pre>
DIV	<pre>select li_salary, li_salary div 5000 from librarian</pre>	<pre>select li_salary, Div(li_salary,5000) from librarian Use function in PostgreSQL</pre>
MOD(), %	<pre>select li_salary, Mod (li_salary,5000) from librarian or select li_salary, li_salary mod 5000 from librarian or select li_salary, li_salary %5000 from librarian</pre>	<pre>select li_salary, Mod (li_salary,5000) from librarian or select li_salary, li_salary %5000 from librarian</pre>
ABS()	select Abs(-2.1)	select Abs(-2.1)
SIGN()	select Sign(-2.1)	<pre>select Sign(-2.1)</pre>

POWER() vagy	<pre>select Power(2,4)</pre>	<pre>select Power(2,4)</pre>
POW()		
ROUND()	<pre>select Round(2.1234, 0)</pre>	<pre>select Round(2.1234)</pre>
	select Round(200.1234, -2)	select Round(2.1234, 0)
	<b>Select Round</b> (200.1234, 2)	select Round(200.1234, -2)
	select Truncate(200 1234 0)	select Trunc(200.1234, 2)
INDINCATE()	select Truncate(200,1234, 2)	select Trunc(200,1234, 0)
	<pre>select Truncate(200.1234, -2)</pre>	select Trunc(200.1234, 2)
	``````````````````````````````````````	select Trunc(200.1234, -2)
		PostgreSQL uses Trunc()
CEIL()	<pre>select Ceil(-2.1234)</pre>	<pre>select Ceil(-2.1234)</pre>
FLOOR()	<pre>select Floor(-2.1234)</pre>	<pre>select Floor(-2.1234)</pre>
SQRT()	select Sqrt(200)	select Sqrt(200)
DATEDIFF()	<pre>select * from borrowing where</pre>	<ul> <li>use subtraction in</li> </ul>
	<pre>Datediff(bo_outdate,bo_indate)&gt;7</pre>	PostgreSQL:
		borrowing
		borrowing
NOW()	<pre>select Now()</pre>	<pre>select now() →with timezone</pre>
YEAR()	<pre>select Year(Now())</pre>	<pre>select Extract(year from Now())</pre>
MONTH()	<pre>select Month(Now())</pre>	<pre>select Extract(month from Now())</pre>
DAY()	<pre>select Day(Now())</pre>	<pre>select Extract(day from Now())</pre>
HOUR()	<pre>select Hour(Now())</pre>	<pre>select Extract(hour from Now())</pre>
MINUTE()	<pre>select Minute(Now())</pre>	<pre>select Extract(minute from Now())</pre>
SECOND()	<pre>select Second(Now())</pre>	<pre>select Extract(second from Now())</pre>
DAYOFWEEK()	<pre>select Dayofweek(now())</pre>	<pre>select Extract(dow from Now())</pre>
timezone		<pre>select Extract(timezone_hour from Now())</pre>
MIN()	<pre>select Min(li_salary) from librarian</pre>	<pre>select Min(li_salary) from librarian</pre>
MAX()	<pre>select Max(li_salary) from librarian</pre>	<pre>select Max(li_salary) from librarian</pre>
AVG()	<pre>select Avg(li_salary) from</pre>	<pre>select Avg(li_salary) from</pre>
	librarian	librarian
SUM()	<pre>select Sum(li_salary) from </pre>	<pre>select Sum(li_salary) from </pre>
	<pre>Ilbrarlan coloct Count(*) from librarian</pre>	<pre>illorarian soloct Count(*) from librarian</pre>
COUNT()		
Group by	<pre>select li_job, Count(*) from</pre>	<pre>select li_job, Count(*) from</pre>
	librarian group by lijob or	librarian group by li_job
	librarian group by 1	or select li job Count(*) from
		librarian group by 1
HAVING	<pre>select li_job, Count(*) from</pre>	<pre>select li_job, Count(*) from</pre>
	librarian group by li_job having	librarian group by li_job having
	Count(*)>6	Count(*)>6
INNER JOIN	select dept_name, li_name from	select dept_name, li_name from
	dent code-li dentcode	dent code-li dentcode
	OR	OR
	<pre>select dept_name, li name from</pre>	<pre>select dept_name, li_name from</pre>
	librarian <mark>join</mark> department ON	librarian join department ON
	<pre>dept_code=li_deptcode</pre>	<pre>dept_code=li_deptcode</pre>
	OR	OR

	<pre>select dept_name, li_name from librarian inner join department ON dept_code=li_deptcode</pre>	<pre>select dept_name, li_name from librarian inner join department ON dept_code=li_deptcode</pre>
LEFT or RIGHT JOIN	<pre>select dept_name, li_name from librarian right join department ON dept_code=li_deptcode</pre>	<pre>select dept_name, li_name from librarian right join department ON dept_code=li_deptcode</pre>
Subqueries with relations	<pre>select li_name, li_salary from librarian where li_salary &gt;(select li_salary from librarian where li_name='Daniel Jackson')</pre>	<pre>select li_name, li_salary from librarian where li_salary &gt;(select li_salary from librarian where li_name='Daniel Jackson')</pre>
Subqueries with in	<pre>SELECT * FROM department WHERE dept_city IN (select dept_city FROM department WHERE dept_name IN ('Department of Geography','Department of Geophysics'));</pre>	<pre>SELECT * FROM department WHERE dept_city IN (select dept_city FROM department WHERE dept_name IN ('Department of Geography','Department of Geophysics'));</pre>
EXISTS, NOT EXISTS	<pre>SELECT * FROM department where NOT EXISTS (SELECT 1 FROM librarian WHERE dept_code=li_deptcode);</pre>	<pre>SELECT * FROM department where NOT EXISTS (SELECT 1 FROM librarian WHERE dept_code=li_deptcode);</pre>
LIMIT X	SELECT * FROM librarian limit 5	SELECT * FROM librarian limit 5
LIMIT X, Y LIMIT X OFFSET Y	<pre>SELECT * FROM librarian limit 5,3 OR SELECT * FROM librarian limit 5 offset 3 Write 3 results, begin from the record 6. (number of offset+1)</pre>	SELECT * FROM librarian limit 3 offset 5 Write 3 results, begin from the record 6. (number of offset+1)

# **Chapter 6. The PostGIS functions**

This chapter based on the offical PostGIS documentation. In the following sections, you'll find explanations and examples of key functions. The full documentation is available here:

https:/postgis.net/docs/manual-3.3/

# Getting familiar with PostGIS

PostGIS is a PostgreSQL extension or handling spatial data. Here's a basic example of creating a point using the ST\_Point() function:

select ST\_Point(1, 2) as MyFirstPoint; To create a point with a coordinate system (SRID):

select ST\_SetSRID(ST\_Point(-77.036548, 38.895108),4326); Or using WKT (Well-Known Text) format with ST\_GeomFromText():

select ST\_GeomFromText('POINT(-77.036548 38.895108)', 4326);

#### What is WKT format?

WKT (Well-known text format) is an OGC Standard for representing geometries as text. Functions like ST\_GeomFromText() and ST\_AsText() convert between WKT and geometry. Coordinates follow the (longitude, latitude) order.

#### What is WKB format?

WKB (Well-known binary format) is the binary representation of geometries, also standardized by OGC. Though not human-readable, it's more efficient for storage. To view WKB in readable formats, use ST\_AsEWKT() function or ST\_AsText.

```
select ST_AsEWKT('0101000020E6100000FD2E6CCD564253C0A93121E692724340');
select ST_AsText('0101000020E6100000FD2E6CCD564253C0A93121E692724340');
```

## Representing Points, Lines and Polygons in WKT

The tables below shows single geometries (geometry primitives) and the multipart geometry representations of features.

The following formats are used:

- **Point:** POINT(x y) a simple coordinate pair inside parentheses.
- LineString: LINESTRING(x1 y1, x2 y2, ...) a sequence of points separated by commas.
- **Polygon:** POLYGON((outer ring), (inner ring1), (inner ring2), ...) the first set of coordinates defines the outer ring; any additional sets define inner rings (holes). The rings must be closed (first and last points are the same).

#### Geometry primitives (2D)

Туре		Examples								
Point		POINT (30 10)								
LineString		LINESTRING (30 10, 10 30, 40 40)								
Polygon	$\checkmark$	POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))								
Polygon	1	POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))								

Image source: Wikipedia.

Multi geometries are group of geometry primitives. When do you use them? For example, the United Kingdom consists of several islands: Great Britain, Northern Island, the Hebrides, etc. If these polygons are multipolygons, when you click on one of these features, all parts of the multipolygon will be selected (try it out yourself in QGIS!)

Geometry collections are a special set of features. They can contain points, lines and polygons (as well as multipart geometries). However, their usage is not recommended because if you want to use your data in a geoinformatics software, they won't be supported. GIS software generally does not support this type of geometry.

Туре		Examples
MultiDoint		MULTIPOINT ((10 40), (40 30), (20 20), (30 10))
MultiPoint	- 0	MULTIPOINT (10 40, 40 30, 20 20, 30 10)
	٩.	MULTILINESTRING ((10 10, 20 20, 10 40),
MultiLineString	~~	(40 40, 30 30, 40 20, 30 10))
	$\mathbf{x}$	MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)),
		((15 5, 40 10, 10 20, 5 10, 15 5)))
MultiPolygon		MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)),
		((20 35, 10 30, 10 10, 30 5, 45 20, 20 35),
		(30 20, 20 15, 20 25, 30 20)))
	5	GEOMETRYCOLLECTION (POINT (40 10),
GeometryCollection		LINESTRING (10 10, 20 20, 10 40),
		POLYGON ((40 40, 20 45, 45 30, 40 40)))

#### Multipart geometries (2D)

#### What is EWKT/EWKB?

EWKT (Extended Well-know text) and EWKB (Extended Well-know binary format) are PostGIS specific file formats that store the spatial reference system (SRID). For example:

'SRID=4326;POINT(19.1 47.5)'

#### What is SRID?

SRID stands for Spatial Reference Identifier. You can easily access SRIDs and EPSG codes on this website: <u>http://epsg.io/</u>. When you add the PostGIS extension to a database, PostGIS automatically generates a spatial\_ref\_sys table. Do not delete this table, as it contains the SRIDs.

۶.	<post< th=""><th>gres&gt; Console</th><th>×</th><th></th><th></th><th>-</th><th></th></post<>	gres> Console	×			-	
•		select * f	rom pu	blic.spa	tial ref sys		~
E C							
8							
<u>*</u> +							$\sim$
8	_	<	_		▲ ▼		>
	spatia	I_ref_sys 1 $ imes$					
٥Ī	select	* from public	.spatial_	ref_sys 🕌	🕻 Enter a SQL expression to filter results (use Ctrl+Space) 🛛 🕨 💌 🛃 👯	🇞 ▼ 🗧 🔶 − =	+ -
pirid		123 srid 🛛 🕄 🕽	ABC∏‡	123 a 🟹 🛊	ABC sitext	¢ proj4text	^ 🌉
Ĭ	238	4,310	EPSG	4,310	GEOGCS["Yoff",DATUM["Yoff",SPHEROID["Clarke 1880 (IGN)",6378249.2,293.4660212936269,AUTHORITY["EPS +p	proj=longlat +a=	ane
	239	4,311	EPSG	4,311	GEOGCS["Zanderij", DATUM["Zanderij", SPHEROID["International 1924", 6378388, 297, AUTHORITY["EPSG", "7022" + p	proj=longlat +ell	5
, te	240	4,312	EPSG	4,312	GEOGCS["MGI",DATUM["Militar_Geographische_Institute",SPHEROID["Bessel 1841",6377397.155,299.1528128,4 + p	proj=longlat +da	
L,	241	4,313	EPSG	4,313	GEOGCS["Belge 1972",DATUM["Reseau_National_Belge_1972",SPHEROID["International 1924",6378388,297,AU" + p	proj=longlat +ell	
	242	4,314	EPSG	4,314	GEOGCS["DHDN",DATUM["Deutsches_Hauptdreiecksnetz",SPHEROID["Bessel 1841",6377397.155,299.1528128,/ +p	proj=longlat +da	
	243	4,315	EPSG	4,315	GEOGCS["Conakry 1905",DATUM["Conakry_1905",SPHEROID["Clarke 1880 (IGN)",6378249.2,293.466021293626! + p	proj=longlat +a=	- E
	244	4,605	EPSG	4,605	GEOGCS["St. Kitts 1955",DATUM["St_Kitts_1955",SPHEROID["Clarke 1880 (RGS)",6378249.145,293.465,AUTHORI + p	proj=longlat +ell	i i i i i i i i i i i i i i i i i i i
	245	4,316	EPSG	4,316	GEOGCS["Dealul Piscului 1930", DATUM["Dealul_Piscului_1930", SPHEROID["International 1924", 6378388, 297, AL + p	proj=longlat +ell	- E.
	246	4,317	EPSG	4,317	GEOGCS["Dealul Piscului 1970", DATUM["Dealul_Piscului_1970", SPHEROID["Krassowsky 1940", 6378245, 298.3, Al + p	proj=longlat +ell	
	247	4,318	EPSG	4,318	GEOGCS["NGN",DATUM["National_Geodetic_Network",SPHEROID["WGS 84",6378137,298.257223563,AUTHOR +p	proj=longlat +ell	
	248	4,319	EPSG	4,319	GEOGCS["KUDAMS",DATUM["Kuwait_Utility",SPHEROID["GRS 1980",6378137,298.257222101,AUTHORITY["EPS  + p	proj=longlat +ell	
	249	4,322	EPSG	4,322	GEOGCS["WGS 72",DATUM["WGS_1972",SPHEROID["WGS 72",6378135,298.26,AUTHORITY["EPSG","7043"]],TO' +p	proj=longlat +ell	
	250	4,324	EPSG	4,324	GEOGCS["WGS 72BE",DATUM["WGS_1972_Transit_Broadcast_Ephemeris",SPHEROID["WGS 72",6378135,298.26, +p	proj=longlat +ell	
	251	4,326	EPSG	4,326	GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","70] +p	oroj=longlat +da	
	252	4,463	EPSG	4,463	GEOGCS["RGSPM06",DATUM["Reseau_Geodesique_de_Saint_Pierre_et_Miquelon_2006",SPHEROID["GRS 1980" + p	proj=longlat +ell	
5	253	4,470	EPSG	4,470	GEOGCS["RGM04",DATUM["Reseau_Geodesique_de_Mayotte_2004",SPHEROID["GRS 1980",6378137,298.25722; +p	proj=longlat +ell	
eco	254	4,475	EPSG	4,475	GEOGCS["Cadastre 1997",DATUM["Cadastre_1997",SPHEROID["International 1924",6378388,297,AUTHORITY["E + p	proj=longlat +ell	
r.	255	4,483	EPSG	4,483	GEOGCS["Mexico ITRF92",DATUM["Mexico_ITRF92",SPHEROID["GRS 1980",6378137,298.257222101,AUTHORIT +p	oroj=longlat +ell	¥
						>	
	$\odot$ :	save 🔯 Cani	cel <u>a</u>	Script	=> =+ :@ =- :  < < > >    ↓ " @; : L : ; ♥ [200 ] \$\\$\\$400¥`a]Rowis: 1/VINCOWST		
	đ.	400 row(s) fet	ched - 0	ms	Aktiválja a Windows rendszert a G	sépházban.	15

#### How many dimensions does the coordinate have?

Coordinate dimensions refer to how many axes the coordinate system has. In most cases, coordinates have two (X, Y) or three (X, Y, Z, or X, Y, M). PostGIS also supports a fourth dimension, the M coordinate, which can be stored for every node.

#### What can "M" represent?

-Temporal changes: For example, a bus moving from its starting point to its current stop.

- Special IDs: For instance, an emergency phone number next to a motorway.

#### Storing special geometries in PostGIS

PostGIS does not support popular Bezier curves or other splines, but these can be approximated using polylines/linestrings. However, PostGIS supports several special curve types:

CircularString: This is a basic curve type, similar to a LINESTRING. A single segment requires three points: the start and end points (first and third) and any other point on the arc. In the case of full circle the first and the last points are the same.

CompoundCurve: it is a combination of CircularString and Linestrings.

COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0 1))

CurvePolygon: Similar to simple polygons, but the outer and inner rings can be CircularStrings or CompoundCurves or LineStrings.

MultiCurve: A collection of curves, which can be CircularStrings, CompoundCurves or LineStrings.

MultiSurfaceA special type of surface that can consist of Polygons, CurvePolygons.

PolyhedralSurface, Triangle, Tin: Additional possibilities to store geometries. See more here: <u>https://postgis.net/docs/using\_postgis\_dbmanagement.html#PolyhedralSurface</u>

#### Why do every functions start with ST?

The "ST" prefix stands for Spatial and Temporal. These are the new versions of PostGIS functions.

#### How to specify geometries and spatial reference identifier (projection)?

 $ST_Point(X,Y) \rightarrow X$ : longitude, Y: latitude, point with unknown or default SRID

 $ST_Point(X, Y, SRID) \rightarrow X$ : longitude, Y: latitude, with specified SRID.

You can also add the Z or M coordinates:

ST\_PointM(X,Y,M) or ST\_PointZ(X,Y,Z) or ST\_PointZM(X,Y,Z,M)

ST\_PointM(X,Y,M, SRID) or ST\_PointZ(X,Y,Z,SRID) or ST\_PointZM(X,Y,Z,M,SRID)

Other Functions for Geometry Creation:

 $ST_MakePoint(X,Y) \rightarrow Creates a point with two dimensions$ 

ST\_MakePoint(X,Y, Z)  $\rightarrow$  Creates a point with three dimensions

ST\_MakePoint(X,Y, Z, M)  $\rightarrow$  Creates a point with three dimensions

ST\_MakeLine(Point, Point) → Creates lines from points

ST\_MakePolygon(LineString) → Convert a linestring to polygon. (Be sure the first and last points are the same.)

# Geometry type, Dimension, Coordinate dimension and Number of Nodes

#### What is the type of the geometry?

ST\_GeometryType(geom)  $\rightarrow$  ST\_Multipolygon. Returns the type of the geometry with the ST\_ prefix

GeometryType(geom)  $\rightarrow$  Returns the type without the ST\_ prefix (e.g., MULTIPOLYGON).

select ST\_GeometryType(s\_geom), GeometryType(s\_geom) from settlement

ST\_NumGeometries(geom)  $\rightarrow$  Returns the number of geometries in a geometry collection or multi-geometry.

#### What are the dimensions?

ST\_Dimension(geom)  $\rightarrow$  Returns the topological the dimension of geometry. In case of points: 0, lines: 1, polygons: 2.

ST\_CoordDim(geom)  $\rightarrow$  Returns the coordinate dimension: 2 if only X and Y are present, 3 or 4 if Z and/or M values are included.

select ST\_Dimension(sc\_geom), ST\_CoordDim(sc\_geom) from settlement\_centroid

#### Number of Nodes

You can count the number of nodes in a geometry using the ST\_NPoints or ST\_NumPoints functions:

ST\_NPoints(geom)  $\rightarrow$  Returns the total number of nodes in a geometry, including in multi-geometries and polygons.

 $ST_NumPoints(geom) \rightarrow Returns the number of points in a LineString. For other geometry types, it returns NULL.$ 

SELECT ST\_NPoints(s\_geom) As npoints, ST\_NumPoints(s\_geom) As numpoints from
settlement

# Advanced in PostGIS Projections, Spatial Reference Identifiers

#### What is the SRID of the geometry?

SRID stands for Spatial Reference Identifier, which defines the projection used. PostGIS supports many projections and uses the EPSG identification system. The SRID is equivalent to the EPSG number. You can find all EPSG codes at <a href="https://epsg.io/">https://epsg.io/</a>.

In this course, you will use:

EPSG 23700: Hungarian EOV projection with HD72 datum.

EPSG 3857: Web Mercator projection (used in Google Maps, OpenStreetMap etc.)

EPSG 4326: WGS 84 - WGS84 - World Geodetic System 1984, used in GPS

#### To check the SRID of geometries:

 $ST_SRID(geom) \rightarrow$  returns with the SRID of each feature. In the hungary database, all features within the same table generally have the same SRID.

select ST\_Srid(sc\_geom) from settlement\_centroid

#### To update the SRID of a geometry column:

UpdateGeometrySrid('table name', 'geom ', 'EPSG number') → this function updates the SRID in a table.

select UpdateGeometrySRID('settlement','s\_geom','23700')

#### **Projection transformation**

ST\_Transform(geom, EPSG number)  $\rightarrow$  transforms a geometry into a different projection

select s\_name, ST\_Transform(s\_geom, 4326) from settlement

#### Validity of geometry

 $ST_isValid(geom) \rightarrow$  check if geometries are valid (e.g., no self-intersections): Returns true or false.

select ST\_IsValid(s\_geom) from settlement

# Measurements

Before measuring distances, lengths, or areas, consider:

- **Dimensionality (2D or 3D):** How many dimensions do the coordinates have (2D plain or 3D spatial coordinates)
- Do you use planar or 3D spatial measurement functions?
- Which coordinate system do you use?
- Whether measurements are planar (flat surface) or geodetic (curved surface, i.e., globe). In case of hundred of km distance, it can be a significant difference.

Always understand the nature of your dataset before working with measurements!

#### Measurements on a plane (2D)

Units depend on the dataset's coordinate system. If the feature coordinates are in meter, you get the result in meters or square meters. If coordinates are in degrees, you will get back the size in degrees.

#### Line length (2D)

 $ST\_Length(geom) \rightarrow length of a line (ST\_Length2D function is the same)$ 

```
select cr_name, ST_Length(cr_geom) from creek
```

#### Line length (3D)

 $ST\_Length3D(geom) \rightarrow 3D$  length, includes the Z coordinate

#### Perimeter of a polygon

 $ST_Perimeter(geom) \rightarrow$  the length of a polygon outline. If the polygon contains hole(s), it's length will be also inlcuded in the result ( $ST_Perimeter2D$  is the same).

select d\_name, ST\_Perimeter(d\_geom) from district

#### Area of a polygon

 $ST_Area(geom) \rightarrow area of the polygon$ 

select d\_name, ST\_Area(d\_geom) from district

#### Measurements using geographic coordinates

#### Length on a spheroid

ST\_LengthSpheroid(geom, shperoid data)  $\rightarrow$  returns with the length in meter, while the input coordinates are in degrees. Calculates the length or perimeter of a geometry on an ellipsoid.

```
select ST_LengthSpheroid(ST_GeomFromText('LINESTRING(19.1 47.5, 18.5 47.2)') ,
'SPHEROID["WGS_84",6378137,298.257223563]')
```

#### **Get Coordinates from Points**

If you have point data, it is possible to separate the longitude and latitude into two columns. Use

 $ST_X(geom) \rightarrow$  returns with X (longitude)

 $ST_Y(geom) \rightarrow$  returns with Y (latitude)

#### Other measurement functions

#### **Distance between geometries**

ST\_Distance(geom A, geom B)  $\rightarrow$  distance of two geometries in meter/degrees etc./depends on the initial units/)

```
select ST_Distance(ST_GeomFromText('LINESTRING(628200 215000, 629000 216000)'),
ST_GeomFromText('POINT(632500 215500)'))
```

#### **3D Distance**

ST\_3DDistance(geom A, geom B)  $\rightarrow$  returns the 3-dimensional minimum Cartesian distance between two geometries in projected units (spatial ref units).

#### Maximum distance between geometries

ST\_MaxDistance(geom A, geom B)  $\rightarrow$  returns with the maximal distance between two geometries

```
select ST_MaxDistance(ST_GeomFromText('LINESTRING(628200 215000, 629000 216000)'),
ST_GeomFromText('POINT(632500 215500)'))
```

#### **Distance on a sphere (degrees)**

ST\_DistanceSphere(point A in degree, point B in degree)  $\rightarrow$  Returns linear distance in meters between two lon/lat points. The radius of this sphere: 6 371 008 m

```
select ST_DistanceSphere(ST_GeomFromText('POINT(19.1 47.5)'),
ST_GeomFromText('POINT(18.5 47.2)'))
```

#### **Distance on a spheroid**

ST\_DistanceSpheroid(point A in degree, point B in degree, spheroid description)  $\rightarrow$  Returns linear distance between two lon/lat points given a particular spheroid. Currently only implemented for points.

```
select ST_DistanceSpheroid(ST_GeomFromText('POINT(19.1 47.5)'),
ST_GeomFromText('POINT(18.5 47.2)'),'SPHEROID["WGS 84",6378137,298.257223563]')
```

#### Shortest & Longest Lines Between Geometries

#### **Shortest line**

ST\_ShortestLine(geom A, geom B)  $\rightarrow$  returns with the shortest line between two geometries

Draw the shortest line between Somogy and Nógrád counties!

select ST\_ShortestLine((select co\_geom from county where co\_id=1), (select co\_geom
from county where co\_id=17))

#### **3D Shortest line**

ST\_3DShortestLine(geom A, geom B)  $\rightarrow$  Returns the shortest line between two geometries, including elevation (Z coordinate)

ST\_LongestLine(geom A, geom B)  $\rightarrow$  Returns the longest line between two geometries.

#### Task:

Draw the longest line between Somogy and Nógrád counties!

#### Longest line

```
select ST_LongestLine((select co_geom from county where co_id=1),(select co_geom
from county where co_id=17))
```

ST\_3DLongestLine(geom A, geom B)  $\rightarrow$  Returns the longest line between two geometries, including elevation (Z coordinate)

#### **Closest Point**

ST\_ClosestPoint(geom A, geom B)  $\rightarrow$  returns the coordinates of the closest point of the geometry B to geometry A. This point corresponds to the start of the shortest line.

select ST\_ClosestPoint((select co\_geom from county where co\_id=1),(select co\_geom
from county where co\_id=17))

ST\_3DClosestPoint(geom A, geom B)  $\rightarrow$  Returns the closest point of geometry B to geometry A, including elevation (Z coordinate).

#### Azimuth

ST\_Azimuth(point A, point B)  $\rightarrow$  Returns the azimuth of the line defined by the two points. Azimuth: north= 0; northeast=  $\pi/4$ ; east=  $\pi/2$ ; southeast =  $3\pi/4$ ; south =  $\pi$ ; southwest  $5\pi/4$ ; west=  $3\pi/2$ ; northwest =  $7\pi/4$ .

```
select degrees(ST_azimuth(ST_GeomFromText('POINT(628200 215000)'),
ST_GeomFromText('POINT(632500 215500)')))
```

#### Angles

degrees(radian)  $\rightarrow$  Converts the radians to degrees.

ST\_Angle has three forms:

- ST\_Angle(line A, line B)  $\rightarrow$  Angle between two lines.
- ST\_Angle(point A, point B, point C)  $\rightarrow$  Angle at point B formed by three points.

• ST\_Angle(point A, point B, point C, point D)  $\rightarrow$  Angle between segment AB and segment CD.

Result is given in radians.

Angle between two lines:

```
select degrees(ST_Angle(ST_GeomFromText('LINESTRING(628200 215000,632500
215500)'), ST_GeomFromText('LINESTRING(629500 212500,628200 212500)')))
```

Angle between 3 points, at point B

```
select degrees(ST_Angle(ST_GeomFromText('POINT(628200 215000)'),
ST_GeomFromText('POINT(632500 215500)'), ST_GeomFromText('POINT(629500 212500)')))
```

Angle between line AB, and CD.

```
select degrees(ST_Angle(ST_GeomFromText('POINT(628200 215000)'),
ST_GeomFromText('POINT(632500 215500)'), ST_GeomFromText('POINT(629500 212500)'),
ST_GeomFromText('POINT(628200 212500)')))
```

#### **Distance Metrics**

**Hausdorff-distance**  $\rightarrow$  Returns the Hausdorff distance, a measure of similarity between two geometries.

ST\_HausdorffDistance(geom A, geom B)

**Fréchet distance**  $\rightarrow$  Implements algorithm for computing the Fréchet distance restricted to discrete points for both geometries, based on Computing Discrete Fréchet Distance. The Fréchet distance is a measure of similarity between **curves** that takes into account the location and ordering of the points along the curves. Therefore it is often better than the Hausdorff distance.

ST\_FrechetDistance(geom A, geom B, float densifyFrac = -1)

# Bounding boxes and bounding geometries

#### Envelope

ST\_Envelope(geom)  $\rightarrow$  returns the coordinates of bounding box as a polygon (5 points, first and the last identical).

 $Box2D(geom) \rightarrow$  returns only the lower-left and the upper-right corner points of the bounding box

select ST\_Envelope(s\_geom), Box2D(s\_geom) from settlement where s\_name ='Sopron'

Envelope result

```
POLYGON ((452818.6793498857 256385.44666137366, 452818.6793498857 271262.2491004063, 477455.0613929047 271262.2491004063, 477455.0613929047 256385.44666137366, 452818.6793498857 256385.44666137366))
```

and Box2D result

BOX(452818.6793498857 256385.44666137366,477455.0613929047 271262.2491004063)

Note: there is also a BOX3D function.

#### Hulls

ST\_ConvexHull(geom)  $\rightarrow$  returns the smallest convex polygon that encloses a geometry.

select ST\_ConvexHull(s\_geom) from settlement where s\_name ='Sopron'

ST\_ConcaveHull(geom, target\_percent, allow\_holes)  $\rightarrow$  returns with a concave hull. The target percent indicates the grade between the original object and the convex hull. If target percent is 0, it is the original object, if it is 1, this is the convex hull. Allow holes parameter indicates whether holes are allowed.

select ST\_ConcaveHull(s\_geom,0.5) from settlement where s\_name ='Sopron'

ST\_OrientedEnvelope(geom)  $\rightarrow$  Returns with the smallest area bounding box. It can be rotated, and in the most cases, it is not identical with the Envelope.

select ST\_OrientedEnvelope(s\_geom) from settlement where s\_name ='Sopron'

The image below shows these geometries. Blue is the envelope, red is the oriented envelope, and the orange is the convex hull.

select ST\_Envelope(s\_geom), ST\_OrientedEnvelope(s\_geom), ST\_Convexhull(S\_geom)
from settlement where s\_name ='Sopron'



# Accessing other geometric information

#### Boundary

 $ST_Boundary(geom) \rightarrow$  Returns the outline of a geometry as a linestring. If the polygon element has a hole/inner ring, the result will be a MultiLineString. If the original element is a polyline, the result is a MultiPoint feature. The points are the first and the last point of the line.

select ST\_Boundary(s\_geom) from settlement where s\_name ='Sopron'

SELECT ST\_Boundary(geom)FROM (SELECT 'LINESTRING( 10 130, 50 190, 110 190, 140
150, 150 80, 100 10, 20 40 )'::geometry As geom) As f;
→MULTIPOINT ((10 130), (20 40))

#### Start and end point

ST\_StartPoint(geom) and ST\_EndPoint (geom)  $\rightarrow$  returns the first or last point of a LineString.

SELECT ST\_StartPoint(geom)FROM (SELECT 'LINESTRING( 10 130, 50 190, 110 190, 140
150, 150 80, 100 10, 20 40 )'::geometry As geom) As f;
→POINT (10 130)

SELECT ST\_EndPoint(ST\_Boundary(geom)) FROM (SELECT 'POLYGON (( 10 130, 50 190, 110
190, 140 150, 150 80, 100 10, 20 40, 10 130 ), ( 70 40, 100 50, 120 80, 80 110, 50
90, 70 40 ))'::geometry As geom) As f; → returns NULL for polygons (no end-point)

#### **Exterior ring**

 $ST\_ExteriorRing(geom) \rightarrow$  returns the outer ring/outline of the polygon as a LineString. You get NULL, if the initial geometry is not polygon, or it is a MultiPolygon.

```
SELECT ST_ExteriorRing(geom) FROM (SELECT 'Polygon(( 10 130, 50 190, 110 190, 140
150, 150 80, 100 10, 20 40, 10 130 ))'::geometry As geom) As f;
  → LINESTRING (10 130, 50 190, 110 190, 140 150, 150 80, 100 10, 20 40, 10 130)
```

### Geometry validation

 $ST_IsRing(geom) \rightarrow Returns true if the LineString is closed and simple (no self-intersection).$ 

 $ST_IsSimple(geom) \rightarrow Returns true if the geometry has no self-intersections or invalid points.$ 

 $ST_IsClosed(geom) \rightarrow$  Checks if the first and last points of a LineString are the same. (It also works with CircularString, Curves and PolyhedralSurface).

```
SELECT ST_IsClosed('LINESTRING(0 0, 0 1, 1 1,0 0)');
→True
SELECT ST_IsClosed('LINESTRING(0 0, 0 1, 1 1)');
→False
```

ST\_IsPolygonCW(geom) or ST\_IsPolygonCCW(geom)  $\rightarrow$  Check if a polygon's exterior ring is clockwise (CW) or counter-clockwise (CCW).



## Creating geometries

ST\_Buffer(geom, radius of the buffer, other options)  $\rightarrow$  Creates a buffer around the geometry with the given radius (units are the same as the projection's unit). If the radius is negative, then the result is an inner (shrinking) buffer. The third parameter is optional:

quad\_segs = the number of line segments, used to approximate curves. Default is 8.

endcap = round | flat | square  $\rightarrow$  Defines the style of line endings. round (default), flat (ends at last vertex), square (extends beyond line)

join = round | mitre | bevel  $\rightarrow$  Defines how corners are joined. Default is round.

mitre\_limit $\rightarrow$  Sets the mitre ratio limit.

side = both | left | right  $\rightarrow$  Creates a one- or two-sided buffer.

select s\_name, ST\_Buffer(s\_geom, 1000) from settlement where s\_name='Sopron'

```
select cr_name, ST_Buffer(cr_geom, 500,'endcap=flat') from creek where
cr_name='Benta-patak'
```

#### Centroid

ST\_Centroid(geom)  $\rightarrow$  returns the centroid (geometric center) of the geometry.

select s\_name, ST\_Centroid(s\_geom) from settlement

#### **Delaunay Triangles**

ST\_DelaunayTriangles(geom, tolerance)  $\rightarrow$  returns a triangulated subdivision of the geometry.

select ST\_DelaunayTriangles(s\_geom) from settlement where s\_name='Sopron'

#### Line Merge

 $ST\_LineMerge(geom) \rightarrow Merges$  a collection of lines into a single continuous LineString or MultiLineString.

select ST\_LineMerge('MULTILINESTRING((10 160, 60 120), (120 140, 60 120), (120
140, 180 120))');

#### Line Simplification and Smoothing

ST\_Simplify(geom, tolerance)  $\rightarrow$  Line or polygon simplification with Douglas–Peucker algorithm.

select s\_name, ST\_Simplify(s\_geom, 500) from settlement where s\_name='Sopron'

ST\_SimplifyVW(geom, tolerance)  $\rightarrow$  Line or polygon simplification Visvalingam-Whyatt algorithm.

select s\_name, ST\_SimplifyVW(s\_geom, 500) from settlement where s\_name='Sopron'

ST\_ChaikinSmoothing(geom, number of iterations, preserve\_endPoints)  $\rightarrow$  Applies Chaikin's smoothing algorithm to the geometry. Maximum 5 iterations. The last parameter defines whether to preserve the original endpoints.

select cr\_name, ST\_ChaikinSmoothing(cr\_geom, 3, true) from creek where cr\_name='Benta-patak'

### Understanding spatial relations – Geoprocessing

These functions test spatial relationships and return a boolean value (true/false).

Please study the following website to understand these functions:

http://postgis.net/workshops/postgis-intro/spatial\_relationships.html

ST\_Intersects(geom A, geom B)  $\rightarrow$  Returns true if geometries A and B share any space. If geom A and geom B have a common part in plain or 3D, they intersect. For example: there are two points in same coordinate; a polygon touches a line, or a polygon overlaps with another one, or it overlaps with a point.

ST\_Disjoint(geom A, geom B)  $\rightarrow$  Returns true if geometries A and B do not intersect. The reverse of intersects. In this case, the two geometries do not intersect.

ST\_Equals(geom A, geom B)  $\rightarrow$  Returns true if A and B are spatially equal (identical coordinates and structure). If they are perfectly same, it it true (point-point, line-line, polygon-polygon etc).

 $ST\_Crosses(geom A, geom B) \rightarrow Returns true if A crosses B (e.g., linestring-line, point-line, etc.). Possible crossings can be between: multipoint/polygon, multipoint/linestring, linestring/linestring/polygon and linestring/multipolygon$ 

ST\_Overlaps(geom A, geom B)  $\rightarrow$  Returns true if A and B overlap and are of the same dimension. Point, line-line and polygon-polygon.

ST\_Touches(geom A geom B)  $\rightarrow$  Returns true if A and B touch at the boundary but do not overlap. For example: point is on the line or on outline of polygon; lines or polygons touch each other on outline.

 $ST\_Contains(geom A, geom B) \rightarrow$  Returns true if B is entirely within A (excluding boundaries). Geom B lays in geom A, of no points outside geom A. If a line fits on the outline (outer ring) of a polygon, it returns false. Minimum one points should be inside of the polygon. If two polygons are exactly the same, it returns true.

ST\_Within(geome A, geom B)  $\rightarrow$  Returns true if A is entirely within B (excluding boundaries). Geom A lays in geom B, of no points outside geom B. If a line fits on the outline (outer ring) of a polygon, it returns false. Minimum one points should be inside of the polygon. If two polygons are exactly the same, it returns true.

#### **Don't forget:**

#### ST\_Contains(geom A, geom B) = ST\_Within(geom B, geom A)

ST\_Covers(geom A, geom B)  $\rightarrow$  Returns true if A completely covers B (including boundaries). Geom B lays in geom A, of no points outside geom A. If a line fits on the outline (outer ring) of a polygon, it returns **true**. If two polygons are exactly the same, it returns true.

ST\_CoveredBy(geom A, geom B)  $\rightarrow$  Returns true if A is completely covered by B (including boundaries). Geom A lays in geom B, of no points outside geom B. If a line fits on the outline (outer ring) of a polygon, it returns true. If two polygons are exactly the same, it returns true.

#### **Don't forget:**

ST\_Covers(geom A, geom B) = ST\_CoveredBy(geom B, geom A)

# Combining geometries

ST\_Intersection(geom A, geom B, gridSize)  $\rightarrow$  Returns the intersecting part(s) of two geometries (A and B). Optional gridSize increases precision performance.

ST\_Difference(geom A, geom B, gridSize)  $\rightarrow$  Returns the part of A that does not intersect with B  $\rightarrow$  with other words: Geom A - ST\_Intersection(geom A, geom B). Optional gridSize increases precision performance.

ST\_SymDifference(geom A, geom B, gridSize)  $\rightarrow$  Returns the parts that are in A or B but not in both.  $\rightarrow$  with other words: ST\_Union(geom A, geom B) - ST\_Intersection(geom A, geom B).  $\rightarrow$  or with other words: ST\_SymDifference(geom A, geom B) = ST SymDifference(geom B, geom A). Optional gridSize increases precision performance.

ST\_Union(geom A, geom B, gridSize)  $\rightarrow$  Merges A and B into a single geometry (can be MultiGeometry or GeometryCollection). Optional gridSize increases precision performance.

SELECT ST\_Union(co\_geom) from county -> give the country polygon.

ST\_Split(geome A, geom B)  $\rightarrow$  Splits geometry A using geometry B (e.g., splitting a polygon with a line or point).

### Others

String\_Agg()  $\rightarrow$  A PostgreSQL aggregation function that concatenates string values with a given separator. For example: String\_Agg(d\_name, ',')

# Conversion function in PostGIS

These functions convert between PostGIS geometries (WKT or WKB) and other spatial formats (KML, GeoJSON, SVG, etc.)

 $ST_GeomFromGeoJSON(text) \rightarrow$  Creates WKT geometry from GeoJSON string.

ST\_GeomFromKML(text)  $\rightarrow$  Creates WKT geometry from a KML string

 $ST_AsGeoJSON(geom) \rightarrow$  Creates GeoJSON geometry from the database record.

 $ST_AsKML(geom) \rightarrow Creates KML geometry from the database record.$ 

select ST\_AsKML(ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

 $ST_AsSVG() \rightarrow Creates SVG$  geometry from database record.

 $ST_AsLatLonText(geom) \rightarrow$  Returns the coordinates of a point geometry as text in "latitude longitude" format. You can control the numeric format and number of decimal places.

# **Chapter 7: Practices in POSTGIS**

#### Part 0: SQL conditionals

#### CASE

### WHEN condition THEN result WHEN condition THEN result ....

ELSE result

END

- 1. Print the librarian's name and how long she or he has worked in the library.
  - If the hire date is before 1995-01-01, label it as veteran employment.
  - If the hire date is between 1995-01-01 and 2000-01-01, label it as core staff.
  - If the hire date is after 2000-01-01, label it as rookie.
- **2.** Increase the salary of the librarians.
  - Provide a 20% salary increase to those with the position Customer Service Representative.
  - Provide a 15% salary increase to those with the position Storekeeper.
  - Provide a 25% salary increase to those with the position Administrative.
  - Provide a 17% salary increase to those with the position Team Leader.

Update *hungary* database, specifically the county table. Change co\_name from 'megye' to 'vármegye'.

**3.** Add a new column to the county table and update each county name, replacing 'megye' with 'vármegye'.

#### Part 1.: Creating and querying geometry

- 1. Create a new point at Budapest using the ST\_Point function (coordinates lat: 47.5° and lon: 19.1°).
- 2. Use the same coordinates, but define a coordinate system using EPSG:4326!
- **3.** Provide this point in WKT format, and create a geometry representing Budapest (47.5°, 19.1°).
- 4. Add the spatial reference system ID (SRID) to this point (EPSG:4326)!
- 5. Create the same point with projection using the ST\_GeomFromEWKT() function.!
- Create a WKT LineString, using these cities as nodes: Budapest–Sárbogárd–Siófok (47.5° 19.1°, 46.88° 18.62°, 46.9° 18.05°).
- 7. Create a WKT polygon, using the following cities as nodes: Budapest–Sárbogárd–Siófok– Tatabánya (47.5° 19.1°, 46.88° 18.62°, 46.9° 18.05°, 47.58° 18.4°).
- 8. Add an inner ring/hole to the previous polygon. The nodes of the inner ring: (47.22° 18.66°, 47.24° 18.6°, 47.18° 18.55°)!
- 9. Convert the line (without the hole) to polygon using ST\_MakePolygon (Budapest–Sárbogárd– Siófok–Tatabánya: 47.5° 19.1°, 46.88° 18.62°, 46.9° 18.05°, 47.58° 18.4°).
- 10. Create a new line between Budapest and Székesfehérvár (47.5° 19.1°, 47.2° 18.4°) using ST\_MakePoint, ST\_MakeLine!
- 11. Get familiar with Hungary dataset! Query the geometries from the following table 'settlement\_centroid', 'creek', 'district' tables. Try to understand the geometry types based on the previous chapter.
- 12. Query the type of geometry, the dimension of geometry and the dimension of coordinates for the settlement\_centroid, creek and district tables! Let's compare what we find!
- Check the number of nodes in both simple and multi-geometries (ST\_Npoints, ST\_NumPoints). You can use the creek and district tables. For simple geometries, use those from exercises 3, 6, and 7.
- 14. Check the number of geometries in the builtup\_area table.
- 15. Query all counties. Check the spatial view!

nty ×		59.01			Warden - Data -	Tables	
erties 🚓	Data 📸	EK Diagram			🗤 postgres 🖬 Databases 👻 nungary 🧰 Schemas 👻 🔠 public 🔛	lables	
nty  25 Ent	ter a SQL	expression to filter results (use Ctrl	+Space)		▼] <b>⊘</b> ▼1	• ¥ :	÷ -
125 co_	jid 🔻	RBC co_name 🔹 👻	123 co_county_code	•	E co_geom		•
	1	Somogy megye		15	MULTIPOLYGON (((511551.58064812794 78611.19822366432, 511524.5730635522 78848.68245666656, 511404.50507310376 79514.27298077695, 511695.20717571164 795	1.44369	/05718
	2	Baranya megye		2	MULTIPOLYGON (((632476.0226387515 63351.7059950649, 632291.8635160421 63144.61388400523, 632191.7886143308 63032.57238788574, 632141.5446811111 62978.04	8073219	J2, 631
	3	Zala megye		20	MULTIPOLYGON (((510199.3185797193 144844.29071016138, 510012.45884271315 144059.77225778718, 509806.88360944943 143069.2820915488, 509648.200563174 142	48.3701	81158
	4	Tolna megye		17	MULTIPOLYGON (((593230.4829215447 163953.41263722256, 593361.767976722 163886.14090339094, 593595.6518868569 163718.57218355086, 593637.1273422661 1636	9.53803	192293
	5	Györ-Moson-Sopron megye		8	MULTIPOLYGON (((563193.7351491281 254775.0802797602, 563235.4807284032 254690.22375823697, 563319.567711701 254549.91321925496, 563362.2324071826 25447	.709628	37743,
	6	Vas megye		18	MULTIPOLYGON (((437557.96436543943 183796.79960255852, 437652.9308790002 183745.61960858738, 437752.66205741814 183752.96929945183, 437851.64765066735	83755.4	418419
	7	Komárom-Esztergom megye		12	MULTIPOLYGON (((562388.4199067777 234498.9135648153, 562261.6684378074 234837.19501010582, 562176.9375050878 234964.65471754165, 561965.4416231193 2348	9.91993	96964
	8	Fejér megye		- 7	MULTIPOLYGON (((593230.4829215447 163953.41263722256, 593200.4530204294 163890.96744346243, 593073.3830135342 163765.8503972597, 592993.2858669555 1637	4.90655	325664
	9	Pest megye		- 14	MULTIPOLYGON (((645983.0479518177 292912.7747399573, 646025.4168517716 292870.81376406905, 645875.2575046101 292757.1679120057, 645855.7009703794 29274	.393152	37204,
	10	Bács-Kiskun megye		1	MULTIPOLYGON (((721235.9596732428 180270.62561517794, 721193.9595363426 181540.37886608837, 721151.2574615948 182090.85654687666, 721151.2669191458 182	14.5648	12905
	11	Csongrád-Csanád megye		6	MULTIPOLYGON (((783199.5236198354 104964.09640242404, 782205.0826431303 103546.39133745001, 781209.8643900712 102128.39738283269, 781723.8491678527 997	5.81800	75943
	12	Békés megye		3	MULTIPOLYGON (((763645.0773162139 174052.5283574945, 763742.0256191518 174173.92215709874, 764074.6559858592 174492.81706998416, 764142.5541053662 1746	2.96938	<i>)</i> 3162,
	13	Hajdú-Bihar megye		9	MULTIPOLYGON (((802842.3351557947 285586.78389777755, 802884.3569143176 285671.52708743967, 803011.1246671273 285967.77737600834, 803222.6206938793 286	48.7617	/3703
	14	Szabolcs-Szatmár-Bereg megye		16	MULTIPOLYGON (((932215.7262588663 311288.87415373453, 932200.3463246274 311261.59784291434, 932193.1833724217 311235.32122672215, 932188.6296611563 311.	04.6176	537432
	15	Borsod-Abaúj-Zemplén megye		4	MULTIPOLYGON (((724903.5877398818 314728.6253726068, 724935.1357454632 314808.9941962851, 724964.1201759559 314868.86688404693, 724987.2429366717 31491	.985824	16726,
	16	Heves megye		10	MULTIPOLYGON (((718952.6845884114 304255.1775234808, 718910.7940213192 304380.10165060544, 718911.2469001273 304807.59913847275, 718994.3155202835 3048	1.45365	13036,
	17	Nógrád megye		13	MULTIPOLYGON (((718952.6845884114 304255.1775234808, 718995.0809418652 304214.8550170477, 719164.8789909694 304213.2641142627, 719334.467878333 304298.1	9493474	i, 7193
	18	Veszprém megye		19	MULTIPOLYGON (((520315.6058774801 165204.90989149077, 520188.15424976626 165204.91593882258, 519891.92913774383 165501.165190409, 519680.4429155146 165	13.3746	323858
	19	Jász-Nagykun-Szolnok megye		11	MULTIPOLYGON (((793867.1174377839 206556.4528527984, 793825.0944693526 206514.4220441991, 793825.0927104566 206471.71007959705, 793783.068551851 206429	6906882	<del>1</del> 886, 7
	20	Budapest		5	MULTIPOLYGON (((641076.7720314066 247623.38572116045, 641107.0062474224 247635.8359571133, 641739.6367178678 247896.3072804478, 641876.141871774 247978	1870806	17, 642
Refresh	- : Q :	Save 👻 🖂 Cancel : 🎫 📼 🔻	a == 11< < > 1	N n	] : .↑, Export data 👻 👷 200 🔍 20 : … 20 row(s) fetched - 253ms (177ms fetch), on 2024-10-09 at 11:21:14		

### Data view - Grid view

# Spatial View



#### Part 2. Measurements, coordinates, projection

- 1. What is the projection of the features in county table? Print the name of the county and the EPSG code of the projection.
- 2. What is the area of each county? Print the county name and the area size! What is the unit of measurement?
- 3. Print the area of all counties in square kilometers (km<sup>2)</sup> sorted in descending order.
- 4. Print the area of each county in the layer (table)'s original projection (Hungarian EOV, EPSG: 23700) and in Web Mercator projection (EPSG: 3857)! Sort the data in ascending order based on EOV projection.
- 5. What is the perimeter of each county? What is the unit of measurement?
- 6. Print the perimeter of each county in the layer/table's original projection (Hungarian EOV, EPSG: 23700) and in Web Mercator projection (EPSG: 3857)! Sort the data in ascending order based on EOV projection.
- 7. What is the length of each creek? Print their names and lengths!
- 8. Print the length of each creek in the layer/table's original projection (Hungarian EOV, EPSG: 23700) and in Web Mercator projection (EPSG: 3857)! Sort the data in ascending order based on EOV projection.
- 9. Print the X and Y coordinates of the centroid of the settlements in separate columns!
- 10. What is the distance between Aggtelek and Barcs using the settlement layer?
- 11. What is the distance between the centroid of Aggtelek and Barcs? Use the settlement layer. (Note: ST\_Centroid calculates the centroid of a feature)
- 12. What is the EOV coordinates (EPSG: 23700) and the geographic coordinates (EPSG: 4326) of the centroid of Barcs?
- 13. What is the distance in meters between Barcs and the point at X=600 000 and Y=200 000 (EPSG: 23700)?
- 14. What is the distance in meters between Barcs and the point Lat=47.5° and Lon=19.1° (EPSG: 4326)?
- 15. What is the distance in meters between the points: 47.5°, 19,1° and 48.2°, 20.2° (EPSG:4326)?
- 16. What is the shortest and longest distance between the polygons of Barcs and Aggtelek?
- 17. Display these (shortest and the longest) lines on the map! Check them on the map!
- 18. If you travel along the shortest path between Lake Balaton and city of Barcs, which node of the Balaton polygon is the closest to Barcs?
- 19. Which settlement contains this point? (Use St\_Contains(geomA, geomB) → where geom A is the settlement, geomB is the point)
- 20. Which settlement is the farthest settlement from Barcs? What is the distance? (Use settlement\_centroid table)
- 21. What is the azimuth of the line drawn from Barcs to Magosliget?
- 22. And what is the azimuth of the line drawn from Magosliget to Barcs (the reversed line)

#### Part 3. Accessing, editing, and processing of geometry

- 1. Print the bounding box of Barcs! Check it the Spatial View.
- 2. Print the coordinates of the bounding box of Barcs.
- 3. Print the bounding box of Barcs with geographic coordinates.
- 4. Print the minimum bounding rectangle (MBR) of Barcs. Check it the Spatial View.
- 5. Print the convex hull of Barcs. Check it the Spatial View.
- 6. What is the area and perimeter of the convex hull?
- 7. Print the concave hull of Barcs using the following parameters for the second argument: 0,1, 0.25,0.5, 0.75. Check it the Spatial View.
- 8. Print the centroid of Barcs.
- 9. Transform the centroid coordinates to the Mercator projection (3857)!
- 10. Print the outline (boundary) of Pest and Fejér counties.
- Create a WKT polygon using the following cities as vertices: Budapest–Sárbogárd–Siófok– Tatabánya (47.5° 19.1°, 46.88° 18.62°, 46.9° 18.05°, 47.58° 18.4°). Add a hole: the nodes of the ring: (47.22° 18.66°, 47.24° 18.6°, 47.18° 18.55°)! Print the outer ring of this polygon.
- 12. Check whether the ring of the previously created polygon is clockwise or counterclockwise.
- 13. Create a line using the cities Budapest–Székesfehérvár–Siófok–Tatabánya–Budapest. (47.5° 19.1°, 47.2° 18.4°, 46.9° 18.05°, 47.58° 18.4°, 47.5° 19.1°). Is the line closed? Is it a ring?
- 14. Check the line created in the previous task has any self-intersections.
- 15. Create a 1km buffer zone around 'Dera-patak'.
- 16. Ensure the buffer has square end caps, and beveled joins.
- 17. Display only the left side of the buffer zone!
- 18. Simplify the path of Dera-patak's using Douglas-Peucker and Visvalingam-Whyatt algorithms! Compare the results. Check how each algorithm works.
- 19. Apply Chaikin-smoothing to Dera-patak. Check how the Chaikin algorithm works!
- 20. Create a triangulated irregular network (TIN) subdivision for Barcs!

#### Part 4.: Understanding spatial relations, geoprocessing

- 1. Check whether the Danube (Duna) and Tisza intersect (use river\_polygon table).
- 2. Check whether the Danube (Duna) and Tisza are disjoint.
- 3. Check whether the point (616100, 206300 in Hungarian EOV projection) is located within Fejér county ('Fejér megye').
- 4. Check whether the point (47.2° N, 18.6° E) is located within Fejér county ('Fejér megye').
- 5. Check whether Fejér county touches Pest county? What are the results (true or false), for the following spatial relationships: intersects, touches, disjoint, crosses, overlaps, within, contains. Explain why?
- 6. Check whether Fejér county contains the polygon of Székesfehérvár?
- 7. Check whether Fejér county overlap or crosses the polygon of Székesfehérvár?
- 8. Test the ST\_COVERS/ST\_CONTAINS and the ST\_COVEREDBY/ST\_WITHIN functions. Observe their subtle differences.

-Does this polygon 'POLYGON((0 0,0 1,1 1,1 0,0 0))' contains/covers an identical polygon POLYGON((0 0,0 1,1 1,1 0,0 0)) ?

-Does this linestring 'LINESTRING(0 0,0 1,1 1,1 0)' contains/covers the polygon POLYGON((0 0,0 1,1 1,1 0,0 0))?

Does this linestring 'LINESTRING(0 0,0.5 0.5,0 1,1 1,1 0)' ' contains/covers the polygon? POLYGON((0 0,0 1,1 1,1 0,0 0))

-Does the point 'POINT(0 0)' contains/covers the polygon POLYGON((0 0,0 1,1 1,1 0,0 0)) ? -Run the same tests using ST\_COVEREDBY/ST\_WITHIN functions!

9. Draw section of the Danube (Duna) that lies within Fejér county!

10. Draw all national parks that are outside Veszprém county! If a national park is partially within the county, only display the part outside it.

- 11. Print the names of settlements in Fejér county! (use settlement\_centorid table)
- 12. Print the names and geometry of settlements in Fejér county!
- 13. Print the names of all settlements not located in Fejér county!
- 14. How many settlements are not located in Fejér county!
- 15. Draw the section of M7 motorway that passses through Fejér county!
- 16. What is the total length of this road?

17. What is the length of one direction (either backward or forward) in kilometers?

- 18. What is the total area of the protected regions in Veszprém county (in km<sup>2</sup>)?
- 19. What percentage of Veszprém County is protected?
- 20. What percentage of the **entire country** is protected?
- 21. Which geoboxes are located in Fejér county?
- 22. How many geoboxes are located within Fejér county?
- 23. Which geoboxes are located outside Fejér county?
- 24. Which geoboxes are located in the 10 km buffer zone around the Danube (Duna)?
- 25. Which geoboxes are located in the 10 km buffer zone around the M7 motorway?

26. Which geoboxes are located in the 10 km buffer zone around M7 motorway in Fejér county?

27. Which creek is the closest to Hollókő castle. What is this distance?

28. Which node of this creek is the closest to Hollókő? (Print its coordinates.)

29. Split Hungary (polygon) into two halves along the 47° latitude (between 16° and 22°).

30. Draw the areas that belong to either the 'Budai Tájvédelmi Körzet' or Budapest, but exclude the areas where these features intersect.

- 31. Print the district name and and associated castle(s), ordered in ascending order.
- 32. Formulate a question for the following query!

select b\_name, string\_agg(va\_name,', ') from belterulet, var where st\_contains(st\_intersection((select m\_geom from megyek where m\_name='Nógrád megye'), b\_geom),st\_transform(va\_geom, 23700)) group by b\_name order by 1

33. Print all projection pairs, where the sum of their EPSG code is 23700!

34. Which districts border 'Székesfehérvári járás'?

Create a list of districts, with each row containing the district name and a second column listing its neighboring districts.

# Chapter 8: Key

#### Part 0: The CASE conditional

```
1. select li_name, li_job,
case
      when li_enter between '1995-01-01' and '2000-01-01' then 'Core staff'
      when li_enter >'2000-01-01' then 'rookie'
      else 'Veteran employment'
end jobrange
from librarian
2. select li_name, li_job, li_salary,
case
      when li_job='Customer Service Representative' then li_salary*1.2
      when li_job='Storekeeper' then li_salary*1.15
      when li_job='Administrator' then li_salary*1.25
      when li_job='Team Leader' then li_salary*1.17
end as newsalary
from librarian
3. ALTER TABLE public.county ADD new_name varchar NULL;
```

update county set new\_name =(concat(substring(co\_name,1,(Length(co\_name)-5)),'vármegye')) where Substring(co\_name,(Length(co\_name)-5))=' megye' Part 1.: Creating and querying geometry

1. select ST\_Point(19.1, 47.5);

2. select ST\_SetSRID(ST\_Point(19.1, 47.5), 4326);

3. select ST\_GeomFromText('POINT(19.1 47.5)');

4. select ST\_GeomFromText('POINT(19.1 47.5)', 4326);

5. select ST GeomFromEWKT('SRID=4326;POINT(19.1 47.5)');

6. select ST\_GeomFromText('LINESTRING(19.1 47.5, 18.62 46.88, 18.05 46.9)', 4326);

7. select ST\_GeomFromText('POLYGON((19.1 47.5, 18.62 46.88, 18.05 46.9, 18.4
47.58, 19.1 47.5))', 4326);

8. select ST\_GeomFromText('POLYGON((19.1 47.5, 18.62 46.88, 18.05 46.9, 18.4 47.58, 19.1 47.5),(18.66 47.22,18.6 47.24, 18.55 47.18,18.66 47.22))', 4326);

9. select ST\_MakePolygon('LINESTRING(19.1 47.5, 18.62 46.88, 18.05 46.9, 18.4 47.58, 19.1 47.5)');

10. select ST\_MakeLine(ST\_MakePoint(19.1, 47.5), ST\_MakePoint(18.4, 47.2));

11.

select sc\_geom from settlement\_centroid;

select cr\_geom from creek;

select d\_geom from district;

12. select ST\_GeometryType(sc\_geom), GeometryType(sc\_geom), ST\_Dimension(sc\_geom),
ST\_CoordDim(sc\_geom) from settlement\_centroid

select ST\_GeometryType(cr\_geom), GeometryType(cr\_geom), ST\_Dimension(cr\_geom), ST\_CoordDim(cr\_geom) from creek

select ST\_GeometryType(d\_geom), GeometryType(d\_geom), ST\_Dimension(d\_geom), ST\_CoordDim(d\_geom) from district

13. select ST\_NPoints(d\_geom) from district;

select ST\_NumPoints(d\_geom) from district; → NULL

select ST\_NPoints(cr\_geom) from creek;

select ST\_NumPoints(cr\_geom) from creek; → NULL

select ST\_NumPoints(ST\_GeomFromText('LINESTRING(19.1 47.5, 18.4 47.2, 18.05
46.9)', 4326));

select ST\_NPoints(ST\_GeomFromText('LINESTRING(19.1 47.5, 18.4 47.2, 18.05 46.9)',
4326));

select ST\_NPoints(ST\_GeomFromText('POLYGON((19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5))', 4326));

select ST\_NumPoints(ST\_GeomFromText('POLYGON((19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5))', 4326));→ NULL

14. select ST\_NumGeometries(ba\_geom) from builtup\_area;

15. select \* from county;

#### Part 2. Measurements, coordinates, projection

- 1. select co\_name, ST\_Srid(co\_geom) from county;
- 2. select co\_name, ST\_Area(co\_geom) from county; /Unit in square meter/
- 3. select co\_name, ST\_Area(co\_geom)/1000000 from county order by 2 desc
- 4. select co\_name, ST\_Area(co\_geom)/1000000 as eov, ST\_Area(ST\_Transform(co\_geom, 3857))/1000000 as webmerc from county order by 2 asc
- 5. select co\_name, ST\_Perimeter(co\_geom) from county; /meter/
- 6. select co\_name, ST\_Perimeter(co\_geom) as eov,
   ST\_Perimeter(ST\_Transform(co\_geom, 3857)) as webmerc from county order by 2
- 7. select cr\_name, ST\_Length(cr\_geom) from creek; /meter/
- 8. select cr\_name, ST\_Length(cr\_geom) as eov, ST\_Length(ST\_Transform(cr\_geom, 3857)) as webmerc from creek;
- 9. select ST\_X(sc\_geom), ST\_Y(sc\_geom) from settlement\_centroid;
- 10.select ST\_Distance((select s\_geom from settlement where
   s\_name='Aggtelek'),(select s\_geom from settlement where s\_name='Barcs'))
- 11.select ST\_Distance((select ST\_Centroid(s\_geom) from settlement where
   s\_name='Aggtelek'),(select ST\_Centroid(s\_geom) from settlement where
   s\_name='Barcs'))
- 12.select ST\_Centroid(s\_geom) as eov, ST\_Transform(ST\_Centroid(s\_geom),4326)
   as geog from settlement where s\_name='Barcs'

#### or

```
select ST_Distance((select s_geom from settlement where s_name='Barcs'),
ST_GeomFromText('POINT(600000 200000)',23700))
```

- 14.select ST\_Distance((select s\_geom from settlement where s\_name='Barcs'), ST\_Transform(ST\_GeomFromText('POINT(19.1 47.5)',4326),23700)) or select ST\_Distance((select sc\_geom from settlement\_centroid sc where sc\_name='Barcs'), ST\_Transform(ST\_GeomFromText('POINT(19.1 47.5)',4326),23700))
- 15.select ST\_DistanceSphere(ST\_GeomFromText('POINT(19.1 47.5)',4326), ST\_GeomFromText('POINT(20.2 48.2)',4326)); /Unit is meter/

16.select ST\_Distance((select s\_geom from settlement where
 s\_name='Aggtelek'),(select s\_geom from settlement where s\_name='Barcs')),
 ST\_MaxDistance((select s\_geom from settlement where
 s\_name='Aggtelek'),(select s\_geom from settlement where s\_name='Barcs'))

- 17.select ST\_ShortestLine((select s\_geom from settlement where
   s\_name='Aggtelek'), (select s\_geom from settlement where s\_name='Barcs')),
   ST\_LongestLine((select s\_geom from settlement where s\_name='Aggtelek'),
   (select s\_geom from settlement where s\_name='Barcs'))
- 18.select ST\_ClosestPoint((select la\_geom from lake where la\_name='Balaton'),(select s\_geom from settlement where s\_name='Barcs'))
- 19.select \* from settlement where ST\_Contains(s\_geom, ST\_ClosestPoint((select la\_geom from lake where la\_name='Balaton'),(select s\_geom from settlement where s\_name='Barcs')))
```
20.select sc_name, ST_Distance((select sc_geom from settlement_centroid where
   sc_name='Barcs'), sc_geom) from settlement_centroid order by 2 desc limit
   1
21.select ST_Azimuth((select sc_geom from settlement_centroid sc where
   sc_name='Barcs'), (select sc_geom from settlement_centroid where
   sc_name='Magosliget') )*180/pi()
   or
   select degrees(ST_Azimuth((select sc_geom from settlement_centroid sc where
   sc_name='Barcs'), (select sc_geom from settlement_centroid where
   sc_name='Magosliget')))
22. select ST_Azimuth((select sc_geom from settlement_centroid sc where
    sc_name='Magosliget'), (select sc_geom from settlement_centroid where
    sc_name='Barcs') )*180/pi()
    or
    select degrees(ST_Azimuth((select sc_geom from settlement_centroid sc
    where sc_name='Magosliget'), (select sc_geom from settlement_centroid
    where sc_name='Barcs')))
```

Part 3. Accessing, editing, and processing of geometry

```
1. select ST_Envelope(s_geom) from settlement where s_name='Barcs'
 2. select Box2D(s geom) from settlement where s name='Barcs'
     or select Box(s geom) from settlement where s name='Barcs'
 3. select ST_Transform(ST_Envelope(s_geom),4326) from settlement where
     s name='Barcs'
 4. select ST_OrientedEnvelope(s_geom) from settlement where s_name='Barcs'
 5. select ST_ConvexHull(s_geom) from settlement where s_name='Barcs'
 6. select ST_Area(ST_ConvexHull(s_geom)), ST_Perimeter(ST_ConvexHull(s_geom))
     from settlement where s name='Barcs'
 7. select ST_ConcaveHull(s_geom,0) from settlement where s_name='Barcs'
      Change the second parameter to 0.25, 0.5, 0.75 and 1.
 8. select ST_Centroid(s_geom) from settlement where s_name='Barcs'
 9. select ST Transform(ST Centroid(s geom), 3857) from settlement where
 s name='Barcs'
10. select ST Boundary(co geom) from county where co name in ('Fejér megye', 'Pest
megye')
-> Fejér is a LineString, Pest is a MultiLineString because of the ring of
Budapest
11. select ST_ExteriorRing(ST_GeomFromText('POLYGON((19.1 47.5,18.62 46.88,18.05
46.9, 18.4 47.57, 19.1 47.5), (18.66 47.22, 18.6 47.24, 18.55 47.18, 18.66
47.22))'));
12. select ST IsPolygonCW(ST GeomFromText('POLYGON((19.1 47.5, 18.62 46.88, 18.05
46.9, 18.4 47.57, 19.1 47.5), (18.66 47.22, 18.6 47.24, 18.55 47.18, 18.66
47.22))'));
\rightarrow true
select ST_IsPolygonCCW(ST_GeomFromText('POLYGON((19.1 47.5, 18.62 46.88, 18.05 46.9,
18.4 \ 47.57, 19.1 \ 47.5), (18.66 \ 47.22, 18.6 \ 47.24, \ 18.55 \ 47.18, \ 18.66 \ 47.22))'));
→ false
13. select ST_IsRing(ST_GeomFromText('LINESTRING(19.1 47.5, 18.62 46.88, 18.05 46.9,
18.4 47.57,19.1 47.5)'));
-> true
14. select st_IsSimple(ST_GeomFromText('LINESTRING(19.1 47.5,18.62 46.88,18.05
46.9, 18.4 47.57, 19.1 47.5)'));
->true, no self-intersection (try to do alone a self-intersected object and test
it!)
15. select ST_Buffer(cr_geom, 1000) from creek where cr_name='Dera-patak'
16. select ST_Buffer(cr_geom, 1000, 'endcap=square') from creek where cr_name='Dera-
patak'
17. select ST_Buffer(cr geom, 1000, 'side=left') from creek where cr name='Dera-
patak'
18. select ST_Simplify(cr_geom, 100), ST_SimplifyVW(cr_geom, 5000), cr_geom from
creek where cr name='Dera-patak'
Click here to read more about Douglas-Peucker algorithm and Visvalingam-Whyatt
algorithm
```

```
74
```

Both algorithms are point reduction algorithms. Read the articles in the url above. In cartography, we often use them to simplify or generalize lines or polygons. Read the PostGIS Topology Module, before you use these algorithms in polygon simplification.

Check and understand the differences between the tolerance values!

19. select ST\_Chaikinsmoothing(cr\_geom,5), cr\_geom from creek where cr\_name='Derapatak'

#### Chaikin's smoothing

Smoothing is used to smooth a polyline. Smoothing is often used after simplification to make the new line with less nodes. Th appperance is much more curve-like.

20. select ST\_DelaunayTriangles(s\_geom), s\_geom from settlement where
s\_name='Barcs'

<u>Delaunay Triangulation</u> is used to tessellate or subdivide a plane.

Part 4.: Understanding spatial relations, geoprocessing.

- 1. select ST\_Intersects((select rp\_geom from river\_polygon where
   rp\_name='Duna'),(select rp\_geom from river\_polygon where rp\_name='Tisza')) →
   False
- 2. select ST\_Disjoint((select rp\_geom from river\_polygon where
   rp\_name='Duna'),(select rp\_geom from river\_polygon where rp\_name='Tisza')) →
   True
- 3. select ST\_Intersects(ST\_SetSRID(ST\_Point(616100, 206300),23700), (select co\_geom from county where co\_name='Fejér megye'))
- 4. select ST\_Intersects(ST\_Transform(ST\_SetSRID(ST\_Point(18.6, 47.2),4326),23700), (select co\_geom from county where co\_name='Fejér megye'))
- 5. select ST\_Intersects(ST\_Transform(ST\_GeomFromText('POINT(18.6
   47.2)',4326),23700), (select co\_geom from county where co\_name='Fejér megye'))
   select ST\_Touches((select co\_geom from county where co\_name='Fejér
   megye'),(select co\_geom from county where co\_name='Pest megye')) → True

Use other spatial relations!

select ST\_Contains((select co\_geom from county where co\_name='Fejér
megye'),(select co\_geom from county where co\_name='Pest megye')) → False

select ST\_Disjoint((select co\_geom from county where co\_name='Fejér
megye'),(select co\_geom from county where co\_name='Pest megye')) → False

select ST\_Within((select co\_geom from county where co\_name='Fejér
megye'),(select co\_geom from county where co\_name='Pest megye')) → False

select ST\_Crosses((select co\_geom from county where co\_name='Fejér
megye'),(select co\_geom from county where co\_name='Pest megye')) → False

select ST\_Overlaps((select co\_geom from county where co\_name='Fejér
megye'),(select co\_geom from county where co\_name='Pest megye')) → False

- 6. select ST\_Contains((select co\_geom from county where co\_name='Fejér
  megye'),(select s geom from settlement where s name='Székesfehérvár')) → True
- 7. select ST\_Overlaps((select co\_geom from county where co\_name='Fejér megye'),(select s\_geom from settlement where s\_name='Székesfehérvár')) → False select ST\_Crosses((select co\_geom from county where co\_name='Fejér megye'),(select s\_geom from settlement where s\_name='Székesfehérvár')) → False
- 8. select ST\_Within((select ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))')),(select ST\_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_CoveredBy((select ST\_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))')),(select ST\_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_Within((select ST\_GeomFromText('LINESTRING(0 0,0 1,1 1,1 0)')),(select ST\_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → False select ST\_CoveredBy((select ST\_GeomFromText('LINESTRING(0 0,0 1,1 1,1 0)')),(select ST\_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_Within((select ST\_GeomFromText('LINESTRING(0 0,0.5 0.5,0 1,1 1,1 0)')),(select ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_CoveredBy((select ST\_GeomFromText('LINESTRING(0 0,0.5 0.5,0 1,1 1,1 0)')),(select ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_CoveredBy((select ST\_GeomFromText('LINESTRING(0 0,0.5 0.5,0 1,1 1,1 0)')),(select ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_CoveredBy((select ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True select ST\_Within((select ST\_GeomFromText('POINT(0 0)')),(select ST\_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → False

```
select ST_CoveredBy((select ST_GeomFromText('POINT(0 0)')),(select
ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True
```

```
To run the queries above with ST_COVERS or ST_CONTAINS reverse the order of geometries!
```

ST\_COVERS(geomA, geomB) <-> ST\_CoveredBy(geomB, geomA)

```
ST_CONTAINS(geomA, geomB) <-> ST_Within(geomB, geomA)
```

- 9. select ST\_Intersection((select co\_geom from county where co\_name='Fejér megye'),(select rp\_geom from river\_polygon where rp\_name='Duna'))

- 12. select sc\_name, sc\_geom from settlement\_centroid where
   ST\_Intersects(sc\_geom,(select co\_geom from county where co\_name='Fejér
   megye'))

- 15. select ST\_Intersection((select co\_geom from county where co\_name='Fejér megye'), m\_geom ) from motorway where m\_ref ilike '%M7%' and ST\_Intersects((select co\_geom from county where co\_name='Fejér megye'), m\_geom)

this is why M7 text can not be found everywhere in the name column

or do it with st\_union

```
select ST_Intersection((select co_geom from county where co_name='Fejér
megye'), ST_Union(m_geom)) from motorway where m_ref ilike '%M7%'
```

```
16. select ST_Length(ST_Intersection((select co_geom from county where
co_name='Fejér megye'), ST_Union(m_geom))) from motorway where m_ref ilike '%M7%'
17. select ST_Length(ST_Intersection((select co_geom from county where
co_name='Fejér megye'), ST_Union(m_geom)))/1000 from motorway where m_ref ilike
'%M7%'and m_hu_ed_dire='forward'
```

```
or
```

select ST\_Length(ST\_Intersection((select co\_geom from county where co\_name='Fejér megye'), ST\_Union(m\_geom)))/1000 from motorway where m\_ref ilike '%M7%'and m\_hu\_ed\_dire='backward'

```
18. select ST_Area(ST_Union(ST_Intersection((select co_geom from county where
co_name='Veszprém megye'),np_geom)))/1000000 from nature_protection
```

```
19. select ST_Area(ST_Union(ST_Intersection((select co_geom from county where co_name='Veszprém megye'),np_geom)))/1000000/(select ST_Area(co_geom)/1000000 from county where co_name='Veszprém megye')*100 from nature_protection
```

```
20. select ST_Area(ST_Union(np_geom))/1000000/(select
ST_Area(ST_Union(co_geom))/1000000 from county)*100 from nature_protection
```

```
21. select * from geobox where ST_Contains((select co_geom from county where
co_name='Fejér megye'), gl_geom)
22. select count(*) from geobox where ST Contains((select co geom from county
where co_name='Fejér megye'), gl_geom)
23. select * from geobox where ST_Disjoint((select co_geom from county where
co_name='Fejér megye'), gl_geom)
or select * from geobox where not ST_Contains((select co_geom from county where
co_name='Fejér megye'), gl_geom)
24. select * from geobox where ST Intersects((select ST Buffer(rp geom, 10000))
from river_polygon where rp_name='Duna'), gl_geom)
25. select * from geobox where ST_Intersects((select ST_Union(ST_Buffer(m_geom,
10000)) from motorway where m_ref ilike '%M7%'), gl_geom)
26. select * from geobox where ST_Intersects(ST_Intersection((select
ST_Union(ST_Buffer(m_geom, 10000)) from motorway where m_ref ilike '%M7%'),(select
co geom from county where co name='Fejér megye')), gl geom)
27. select cr name, ST Distance((select ST Transform(ca geom, 23700) from castles
where ca name='Hollókői vár'), cr geom) from creek order by 2 asc limit 1
28. select ST_Endpoint(ST_ShortestLine((select ST_Transform(ca_geom, 23700) from
castles where ca name='Hollókői vár'), cr geom)) from creek order by 1 asc limit 1
or
select ST_ClosestPoint((select ST_Transform(ca_geom, 23700) from castles where
ca name='Hollókői vár'), cr geom) from creek order by 1 asc limit 1
29. select ST_Split(ST_Union(co_geom),
ST_Transform(ST_GeomFromText('LINESTRING(16.0 47.0, 22.0 47.0)',4326),23700)) from
county
30. select ST SymDifference((select ST Union(np geom) from nature protection where
np_name='Budai Tájvédelmi Körzet'), (select co_geom from county where co_name=
'Budapest'))
31. select d name, ca name from district, castles where ST Contains(d geom,
ST_Transform(ca geom, 23700)) order by d name
or
select d name, ca name from district, castles where d geom ~
ST_Transform(ca_geom, 23700) and ST_Contains(d_geom, ST_Transform(ca_geom, 23700))
order by d_name
32. Draw the section of Route 8 that lies between Bory Castle in Székesfehérvár
and Thury Castle in Várpalota. When trimming the section, make sure it ends at the
point on the road closest to each castle.
33. select SRS1.srid, SRS2.srid from spatial_ref_sys SRS1, spatial_ref_sys SRS2
where (SRS1.srid+SRS2.srid)=23700
divide by 2, because it repeated twice
34. select d2.d_name from district d1, district d2 where ST_Touches(d1.d_geom,
d2.d_geom) and d1.d_name='Székesfehérvári járás'
```

```
35. select d1.d_name, count(*), string_agg(d2.d_name,', ') from district d1, district d2 where ST_Touches(d1.d_geom, d2.d_geom) group by 1 order by 1
```

## Chapter 9. Data import

## Creating a new database, and importing data from SQL files

Right-click on **Database**, and create a new empty database, e.g. *library* (**Create**  $\rightarrow$  **Database**).

😨 Create data				×				
General								
Database name:	atabase name: ko							
Owner:	$\sim$							
Definition								
Template databas	e:			$\sim$				
Encoding:		UTF8	}	$\sim$				
Tablespace:		pg_c	lefault	$\sim$				
OK		Cance	I					

Open the *library.sql* file via **SQL editor** menu  $\rightarrow$  **Open SQL script**.

Execute the script  $\rightarrow$  Execute script (use the 3rd icon on the left of the SQL window) then press F5

(refresh the database).

If your database is in MySQL, you cannot directly use SQL files to migrate it to PostgreSQL, as MySQL and PostgreSQL dumps are not compatible. In such cases, use CSV text files for database migration instead.

## Adding extensions (PostGIS, PostGIS Raster, PostGIS Topology)

A spatial database requires at least the PostGIS extension. Don't forget to add this to your database before importing data. Go to **Database**  $\rightarrow$  **Schemas**  $\rightarrow$  **Extensions**, and right-click  $\rightarrow$  **Create New Extension**  $\rightarrow$  Search for the extension by name (e.g., PostGIS) and add it.



Uploading Data into a PostgreSQL Database from CSV – Using an Existing table structure

The first step is to create a new empty table. In **Database Navigator** right-click  $\rightarrow$  **Tables**  $\rightarrow$  *Create New Table*.



In the Properties window, define the table columns, data types, collations, and other settings. To add a new column: right-click under Columns  $\rightarrow$  Create New Column.

Once ready, don't forget to save the table (Save icon is at the bottom of the window).

#### Spatial Databases

Auto	<b>⊙</b> •	🥵 bos	stgres	• 🗉	public@kony	rvtar 🔻 🧧	2 ₽ -	۹ •							Q i 🖻 🞯	
∑ <postgres></postgres>	Conso	le	== kia	dvany_o	sop ×											
\equiv Properties	民 Dat	ta 📅 E	R Diag	Iram			🖁 postgres	📴 Data	ibases 🔻	🔩 konyvtai	r 📅 Schen	nas 🔻 ፤ p	ublic 臣	Tables	🔻 🖽 newtable	
Table Name:	kiadva	any_csop	p					0	Object ID: 19246							
Tablespace:	pg_def	fault		~	*			C	wner:	postgre	5					
Partition by:										Partit	Partitions					
Comment:								~ E	xtra Option	tra Options:					~	
							~	$\sim$							~	
	L	<u>.</u>			<b>D</b>	1.1	<b>.</b>			- ·						
Columns		Colum	n Nam	e #	Data type	Identity	Collation	Not Null	Default	Comment						
Constraints	;		- Cr	este Ne	w Column											
Foreign Key	ys		9 Ci	ew Colu	mns		F4									
Dependenc	iec		e Br	owse fro	m here											
References	.103	۲	<b>Fil</b>	ter			>									
Partitions		, i	Vi	ew Diag	ram											
Triggers		E	Vie	ew Data												
Rules		6	🤉 Co	mpare/	Migrate		>									
i Statistics		E	Ex	port Dat	a											
Permission:	s	-2	lm	port Da	ta											
ODL		×	to 🕈	ols			>									
Ge Mittaal		5	Ge	enerate S	QL		>									
		1	Re	ad data	in SQL editor											
		0	C	ру			Ctrl+C									
		Ć	] Pa	ste			Ctrl+V			-	\ktiválja	a Windo	owst			
No items			Co	opy Adv	anced Info	Ctrl+S	hift+C			<b>Q</b> ∉ <sup>∆</sup>	🖌 þið 🖓 🕅	Vi <b>n</b> do <b>n</b> /s r	en dazerit	alitiketu	lấ₹b 🍪 Refresh	
		C 🧐	Re	fresh			F5				1					
			0			1						_	~			
				colt at	ribute colu	mnı							^			
			Na	me:	colum	n1										
			Pro	pertie	s: Name		Value									
					Dat	a type	varch	ar								
					lder	ntity										
					Coll	ation										
					Not	Null	[]									
					Defa	ault										
					Con	nment										
										0	K	Cance	el			

Columns can have constrainst, indexes and keys.

A primary key is a column that contains unique, non-null values (e.g., a sequence like 1, 2, 3...n).

To define it: go to the **Constraints** submenu, right-click  $\rightarrow$  **Create New Constraint**  $\rightarrow$  select **Primary Key**.

😨 Add	constraint for ta	ble "l	ciad —		×			
Table:	Table: public.kiadvany_csop							
Name:	kiadvany_csor	o_pk						
Туре:	PRIMARY KEY				~			
Columns:								
Colum	n	#	Туре					
123	cs_kod		int4					
ABC	cs_nev		varchar					
ABC	cs_kateg_k		varchar					
ABC	cs_kiadas_k		varchar					
S	elect All							
			ОК	Car	ncel			

After any modification, always save your changes.

If you already have a CSV file and a predefined table structure, you can import data from the CSV file.

Beaver 21.3.2 - kiadvany_csop				
<u>File Edit Navigate Search SQL Editor I</u>	<u>D</u> ataba	se <u>W</u> indov	v <u>H</u> elp	
🕴 👫 👻 🔍 🕂 🛄 🖓 👘 👫 🖓 👘	mmit	[ Rollbac	k 🏋 🔻	Auto 🕓 🔻
🔁 Database Navigator 🗙 📫 🔻		- 🗧 🖇		🔀 <postgres> Console</postgres>
Enter a part of object name here			7	😑 Properties 🔣 Data
✓ № postgres - localhost:5432			^	·
✓ 📑 Databases				Table Name: kiadvar
V Schemar				Tablespace: pg_defa
v i schemas				Destition has
V 🔁 Tables				Partition by:
> 🚍 katalogus			32K	Comment:
> 🚍 kiadvany			48K	
> 🚍 kiadvany_csoport			32K	
> 🚍 kolcsonzes			32K	PE Colores
> 🖪 kolcsonzo		<u> </u>	32K	
> 🔜 konyvtaros		Create		,
> 🖶 spatial_ref_sys		View Table	2	F4
	7	Filter		>
✓ ■ Columns		View Diagr	am	
123 cs_kod (int4)		View Data		
ABC cs_nev (varchar)	-0			
RBC cs_kateg_k (varc	h 🛱	Compare/I	Migrate	>
ABC cs_kiadas_k (var	ch 🖪	Export Dat	а	
✓ Image: Constraints		Import Dat	a	
Kiadvany_csop_p Eoreign Keys	* 🔀	Tools		>
> Indexes		Generate S	01	>
> Dependencies	<u>а</u>	Read data	in SOL edit	· ·
> 🦰 References	-	Neau uata	in SQL Cuit	.01
> 🛅 Partitions	D	Сору		Ctrl+C
> 🔚 Triggers		Paste		Ctrl+V
> Rules		Copy Adva	inced Info	Ctrl+Shift+C
VIEWS Materialized Views		Delete		Delete
> Indexes	2	Rename		F2
kiadvany csop	E	Refresh		F5
2	×			

Right-click on the **Database Navigator**  $\rightarrow$  **Import data**.

Select the file format: CSV. **Next**.

<b>ansfer targets</b> Configure data transfer s	ource type and f	ormat					
<ul> <li>Import source Input file(s) Tables mapping Data load settings Confirm</li> </ul>		/ le	Import from Database ta	n CSV file(s) ble(s)	Expo	ted blic.kiadvany_csop	Descrip
Save task 👪					<		<b>````</b>

Then configure the CSV properties:

*Extension*: (CSV, TSV,TXT)

*Encoding*: character encoding

Column delimiter: semicolon, comma, TAB etc.

Header position: does the file include a header? If so, specify how many rows it spans.

Quote char: e.g." if used around values

*Escape char*: usually  $\setminus$ 

*Null value mark*: what is the NULL value?, Null or ,," (empty string)

Set empty strings to NuLL

Date Time format: specify the format used in the file

*Timezone ID:* set if applicable

Column length: optional column width

Click Next.

😰 Data Transfer		-		
nput file(s)				
Configure input files or di	rectories			
<ul> <li>Import source</li> </ul>	Input files:			
<ul> <li>Input file(s)</li> </ul>	Source Target			
Data load settings	KIADVANY CSOPORT csv	public kiadvany csop [Existing]		
Confirm		tuny_coop (existing)		
Commit				
	Importer settings:			
	Name Value			
	Extension csv,tsv,txt			
	Encoding utf-8			
	Column delimiter ;			
	Header position none			
	Quote char "			
	NIII Value mark			
	Set empty strings to NULL []			
	Date/time format vvvv-MM-dd			
	Timezone ID			
	Column samples count 100			
	<ul> <li>Column minimal length</li> <li>1</li> </ul>			
	Column use byte length [ ]			
Save task				
		Aletio	vália a W	
		Aktiv	lia a Wind	

In the following window, map the columns in your CSV to the corresponding columns in the database table.

Man tables and columns t	transfe					
<ul> <li>Import source</li> </ul>			15.7			Characteristics
<ul> <li>Input file(s)</li> </ul>		larget container: 📑 konyvtar.p	ublic [postgres]			
<ul> <li>Tables mapping</li> </ul>		Source	Target	Mapping	Transform	⇒ Auto assi
<ul> <li>Data load settings</li> </ul>		✓ == KIADVANY_CSOPORT.csv	kiadvany_csop	existing		
Confirm		123 Column1	cs_kod	create		📰 Browse
		RBC Column2	cs_nev	create		
		ABC Column3	cs_kateg_k	create		To New
		ABC Column4	cs_kiadas_k	create		Column
						o∏ Target [
						• Preview
						∧ Up
						V Dowr
						-
						-
						-
	~					
Save task 📇		* DEL - skip column(s) SPACE - r	map column(s)			
					Δ	ktivália a V
					~	Kuvaija a
					Ak	ctivalja a wind

Final settings: Check the option "Truncate target table before load" if you want to delete all existing data from the table before importing new data. Click **Next. Proceed**.

#### Spatial Databases

😰 Data Transfer		-		×
Data load settings Configuration of table dat	a load			
<ul> <li>Import source</li> <li>Input file(s)</li> <li>Tables mapping</li> <li>Data load settings Confirm</li> </ul>	<ul> <li>▲ Data load</li> <li>▲ Transfer auto-generated columns</li> <li>▲ Truncate target table(s) before load</li> <li>▲ Disable referential integrity checks during the transfer</li> <li>Replace method: ON CONFLICT DO NOTHING</li> <li>▲ Replace/Ignore method documentation</li> <li>Performance</li> <li>④ Open new connection(s)</li> <li>④ Use transactions</li> <li>Do Commit after row insert: 10000</li> <li>■ Use multi-row value insert</li> <li>■ Disable batches</li> <li>■ Use bulk load</li> </ul>	on finish Ige		
		Akti	válja a	W
		Aktiv	álja a W	
	< <u>B</u> ack Next > <u>P</u> roceed	Ca	ncel	

After the import, check the table to confirm the data has been loaded (refresh the table if necessary).

# Uploading Data to a PostgreSQL database from a CSV – Missing table structure

In this task, you need to upload a CSV file and the table structure will be inferred from the file itself. The example used here is the europe.csv file.

Right-click on the **Schemas** section of your database and select  $\rightarrow$  **Import Data**.



Select the format **CSV** and configure the CSV settings as shown in the previous example.

If the **Table Mapping** settings appear, you can define a new table structure here. In the mapping settings, choose **Create**, which will generate the columns. Then click the **Configure** button.

Input file(s)	î	Target container: 🧾 postg	gres.public [postgres]			🗸 📙 Choo
Tables mapping		Source	Target	Mapping	Transform	🛛 🗏 Auto as
Data load settings		✓	europa	create		
Confirm		ABC Albánia	orszag	create		🕀 Browse
		123 "28748"	ter	create		
		123 "3600000"	lakossag	create		lo New
		123 "131"	nepsur	create		
		ABC Tirana	tirana	create		Configu
						Preview
						∧ Up
						√ Dow
						-
	v					
						]
	~					

Here, you can review and adjust the data types for each column if needed.

Configure metad	ata structure						×
Columns mapping	Table properties Ta	rget DDL					
Source container:	europa.csv						
Source entity:	europa.csv						0
larget container:	postgres						
larget entity:	europa						
Source Column	Source Type	Target Column	Target Type	Mapping	Transform		
ABC Albánia	VARCHAR(32)	orszag	VARCHAR(32)	new			
123 "28748"	INTEGER	ter	INTEGER	new			
123 "3600000"	INTEGER	lakossag	INTEGER	new			
123 "131"	INTEGER	nepsur	INTEGER	new			
ABC Tirana	VARCHAR(32)	tirana	VARCHAR(32)	new			
	1	1		1			_

All other settings should be the same as in the previous example. Finally, check the updated table!

## Creating a backup

If you want to back up your database or migrate it, use the Backup option.

#### Right-click on the **Database** $\rightarrow$ **Tools** $\rightarrow$ **Backup**.

Select the tables you want to include in the backup.

Backup				- C	X נ
Choose objects to export					
Schemas/tables/views which w	ill be exported				
<ul> <li>Choose objects to export Backup settings Backup progress</li> </ul>	Objects				
	🗹 🚍 kolcsonzes	A.¥	All	N	lone
	<ul> <li>✓ == kolcsonzo</li> <li>✓ == konyvtaros</li> <li>□ == spatial_ref_sys</li> <li>✓ == tanszek</li> </ul>				*
Save task 💾	Show views		All	N	lone
	1				
Local Client		< Back Next >	Start	C	ancel

#### **Recommended settings:**

- *Format: Plain.* (can be edited/viewed in a text editor).
- Encoding: UTF-8

- Use SQL INSERT instead of copy of rows: Inserts the rows as SQL statements rather than copying raw data
- Output folder: Select the destination folder
- $\rightarrow$  Click **Start** to begin the backup process.

#### Upload a table from an another one.

Create a new empty table named 'newtable'. Add the following columns: n\_id serial, n\_name varchar and a n\_geom geometry columns. The primary key is n\_id. SAVE the changes.

```
create table public.newtable ();
alter table public.newtable add n_id serial NOT NULL;
alter table public.newtable add n_name varchar NULL;
alter table public.newtable add n_geom public.geometry NULL;
```

Insert data from the geobox table, only include rows where the geometry intersects with the county named Fejér megye.

Use: an INSERT INTO + SELECT SQL Statement:

```
insert into newtable (n_name, n_geom) select gl_full_name, gl_geom from geobox
where ST_Intersects(gl_geom, (select co_geom from county where co_name='Fejér
megye'))
```

#### Data conversion

You can export geometry to GeoJSON format using the ST\_AsGeoJSON function. This function can convert individual geometries or full rows (with geometry).

Example – convert only the geometry column:

select emp\_name, ST\_AsGeoJSON(emp\_geom) from empty\_table

Example – convert entire records (including all attributes):

select ST\_AsGeoJSON(empty\_table.\*) from empty\_table

#### **Exporting to a GeoJSON File**

Run the query. Below the result table, click on the **Export Data** button.

۶	<post< th=""><th>gres&gt;Console X</th><th></th></post<>	gres>Console X	
•		select st_asgeojson(ures.*) from ures	~
▶.			
Ð			
E			
۶.			
*			
B		<	>
T	Result	is 1 X	
۰T	select	st_asgeojson(ures.*) from ures 🎼 A Enter a SQL expression to filter results (use Ctrl+Space) 🕨 💌 🖉 🏹 🗄 (T )	*
9	•	APC st asgeoison	. 📖
5	1	"Tune": "Feature": "neometric: "tune":"Doint: "crs:"/tune":"name":"nronerfies:"/"name":"EDSG:03700"\\ "coordinates":(640502.682482180.18	Par
ш	2	Chore: "Feature" "geometry:: Chore: "Point" "crs:"("hype:"name" "properties": "mame": "FPSG:23700") "coordinates": 630219.486206184.18	lels
봆	3	("type": "Feature", "geometry": ("type": "Point", "crs": ("type": "name", "properties": ("name", "EPSG:23700"), "coordinates": 628914.03107912.199	
E.	4	"type": "Feature", "geometry": ("type": "Point", "crs": ("type": "name", "properties": ("name": "EPSG:23700")), "coordinates": (642462.345891477, 17	
Ť	5	{"type": "Feature", "geometry": {"type":"Point","crs":{"type":"name","properties":{"name":"EPSG:23700"}},"coordinates":{634709.156737781,19	
	6	{"type": "Feature", "geometry": {"type": "Point", "crs": {"type": "name", "properties": {"name": "EPSG:23700"}}, "coordinates": [631723.13487744, 195	, in the second
	7	{"type": "Feature", "geometry": {"type": "Point", "crs": {"type": "name", "properties": {"name": "EPSG:23700"}}, "coordinates": {642099.083642779, 18	
	8	{"type": "Feature", "geometry": {"type":"Point", "crs":{"type":"name", "properties":{"name":"EPSG:23700"}}, "coordinates":{638506.680816417,19	LH
	9	{"type": "Feature", "geometry": {"type": "Point", "crs": {"type": "name", "properties": {"name": "EPSG:23700"}}, "coordinates": {626012.672744951,23	
	10	("type": "Feature", "geometry": {"type": "Point", "crs": {"type": "name", "properties": {"name": "EPSG:23700"}}, "coordinates": {625375.615012578,23	
Ę	11	("type": "Feature", "geometry": ("type": "Point", "crs": ("type": "name", "properties": {"name": "EPSG:23700"}}, "coordinates": {625682.093727173,23	
Reco	12	{"type": "Feature", "geometry": {"type":"Point","crs":{"type":"name","properties":{"name":"EPSG:23700"}},"coordinates":[628638.683968291,22	,
C	Ø	Refresh ▼ : ② Save ▼ ⊠ Cancel 🗦 ☶ :: 🛛 드 : : K < > >  [1] : L Export date/// : 🏩 1200 ndd 🔀 t150 :	
		150 row(s) fetched - 980ms, on 2023-02-14 at 10:48:43 Export data align a Windows rendszert a Gépházban.	12

Choose TXT as the export format. Disable delimiters (refer to the image). Leave the remaining settings as default. After exporting, manually change the file extension to GeoJSON.

😨 Data Transfer		- 0	×
Format settings			
File format settings			
<ul> <li>Export target</li> <li>General</li> </ul>			
Extraction settings     Formatting: <co< th=""><th>onnection's default&gt;</th><th></th><th></th></co<>	onnection's default>		
Output Disasion	and the formation frame and		
Confirm	Te v Encoding: Binary v		
Value display format: Disp	olay (default) 🗸		
Configure Columns			
Exporter settings			
Name	Value		
File extension	txt		
Batch size	200		
Max column length	3		
Show NULLs	[]		
Show header delimite	er []		
Show leading delimite	er []		
Show trailing delimite	er []		
Show in-between del	imiter []		
Print header	LJ		
Save task			
< Back	Next > Proceed	Cancel	

{"type": "Feature", "geometry": {"type": "Point", "crs": {"type": "name", "properties": {"name": "EPSG:23700"}}, "coordinates": [642593.682 483189,180088.555515703]}, "properties": {"ur\_id": 1, "ur\_name": "Dunaújvárosi Madár Liget"}} {"type": "Feature", "geometry": {"type": "Point", "crs": {"type": "name", "properties": {"name": "EPSG:23700"}}, "coordinates": [630219.486 206184,189545.599921337]}, "properties": {"ur id": 2, "ur name": "Kastélykutató"}}

This file can be used in GIS software such as QGIS. Be careful with the coordinate reference system (CRS): GeoJSON expects EPSG:4326 by default.

In this case, the correct CRS is EPSG:23700, so after importing the file into QGIS, go to:

]	Layer Properties -> Source tab	<b>)</b> , and	set the	correct projection.	
I	Q Layer Properties — ures_202302141100 — Source				

Q Layer Properties — ures_20230.	2141100 — Source	×
Q	▼ Settings	
information	Layer name ures_202302141100 Data source encoding UTF-8 *	
Source	Assigned Coordinate Deference System (CDS)	-
. ኛ Symbology		
	EPSG:23700 - HD72 / EOV 👻 👘	
abc Labels	Changing this option does not modify the original data source or perform any reprojection of features.	
abc Masks	Rather, it can be used to override the layer's CRS within this project if it could not be detected or has been incorrectly detected.	
SD View	The Processing " <i>Reproject Layer</i> " tool should be used to reproject features and permanently change a data source's CRS.	
🛉 Diagrams	▼ Geometry	
Fields	Create Spatial Index Update Extents	
🔡 Attributes Form	▼ Provider Feature Filter	
• 🔰 Joins		
Auxiliary Storage		
Actions		
🧭 Display		
🞸 Rendering		
🕓 Temporal	A Duery Brilder	L
S Variables	Query builder	٣
- Vallables	F Style T OK Cancel Apply Help	

#### **Exporting to KML Format**

PostGIS also provides an ST ASKML function. However, it only works with individual geometry columns-it does not support full table rows (i.e., table\_name.\* is not allowed). This means, you can convert geometries to KML, but attribute data will not be included.

select ST\_AsKML(emp\_geom) from empty\_table

## The Geography data type

The geography data type does not use planar (flat) coordinates—it works with spherical coordinates: geographic latitude and longitude.

If you want to convert a geometry to geography, you first need to use the ST Transform function to project the geometry into EPSG:4326 (WGS84), which uses geographic coordinates.

The most significant differences between geometric and geographic coordinates appear when calculating distances using functions like ST Distance and ST DistanceSphere.

The following article explains these differences in detail: http://postgis.net/workshops/postgis-intro/geography.html Unless strictly necessary, it's often better to avoid using the geography type and stick with geometry, unless you're working with data covering the entire globe.

Add a geography Column to the empty table Table:

Create a new column in the ures table with the geography data type:

alter table public.empty\_table add emp\_geog public.geography NULL;

Populate the new column using transformed geometry data:

update empty\_table set emp\_geog = ST\_Transform(enp\_geom,4326);

or

update empty\_table set emp\_geog = ST\_Transform(emp\_geom,4326)::geography;

Which geoboxes (geocaches) in Fejér County are within a 10 km radius of Lake Velence?

select emp\_name from empty\_table where ST\_Intersects(emp\_geog,(select
ST\_Transform(ST\_Buffer(la\_geom, 10000),4326) from lake where la\_name='Velenceitó'))

#### **Distance Calculations Example**

Let's calculate the distance between the geocache "Madár Liget in Dunaújváros" (emp\_id = 1) and "Bakonykúti" (emp\_id = 28) using different methods.

When using ST\_Distance on geometry, you get a planar (projected) distance. When using ST Distance on geography, you get a distance over the ellipsoid (WGS84).

When using ST\_DistanceSphere (after transforming to EPSG:4326), you get the spherical distance in meters. The assumed Earth radius is 6,371,008 meters.

```
select ST_Distance((select emp_geom from empty_table where emp_id=1),(select
emp_geom from empty_table where emp_id=28)),ST_Distance((select emp_geog from
empty_table where emp_id=1),(select emp_geog from empty_table where emp_id=28)) →
65 527.41 m and 65 531.6 m
```

```
select ST_DistanceSphere((select ST_Transform(emp_geom,4326) from empty_table
where emp_id=1),(select ST_Transform(emp_geom,4326) from empty_table where
emp_id=28)),ST_Distance((select emp_geog from empty_table where emp_id=1),(select
emp_geog from empty_table where emp_id=28)) → 65 387.86 m and 65 531.6 m
```

Expected results (approximate):

- Planar vs. ellipsoidal: 65,527.41 m vs. 65,531.6 m
- Spherical vs. ellipsoidal: 65,387.86 m vs. 65,531.6 m

### Shapefile import with PostGIS Bundle

The PostGIS Bundle is available on Windows. (On macOS and Linux/Unix systems, the necessary tools are available, but there is no graphical user interface—you must use the command line.)

Start the PostGIS Bundle from the Start menu. This tool allows you to import and export Shapefiles to and from a PostgreSQL + PostGIS database.

First step is to set the connection details. (PostgreSQL user name+password and port number, and the database name)

Try using the *stops.shp* file to test the import process.

	View	connection de	tails	
nport E	xport			
Import l	.ist			
Shapefi	ile Schema Table	Geo Column	SRID Mode Rm	
_				_
	PostGIS conn	ection –	- 🗆 X	
	PostGIS Connect	tion		
	Username:	postgres		
	Password:	•••••		
	Server Host:	localhost	5432	
	Database:	mo		F
Onti				el
Opti				
opu				

Click **Add File** to select the Shapefile. Here, you can define: the table name, the geometry column name and the SRID (Spatial Reference Identifier). In the Import options, you can also set the character encoding of the DBF file, and enable the creation of a spatial index, etc

hapefile			Schema	Table	Geo Column	SRID	Mode	Rm	
G:\ADATOK\BENTI\zsuzsi\oktata	s\terbeli_adat	tbazisok\ove	public	epuletek2	geom	4326	Create		
<u></u>									
🥥 Imp	ort Options					×			
UTF-8		DBF file c	haracter en	coding					
		Preserve	case of colu	mn names					
		Do not cr	eate 'bigint'	columns					
	$\checkmark$	Create sp	atial index a	automatically a	fter load				
		Load only	attribute (	dbf) data					
		Load data	a using COP	Y rather than II	NSERT				
		Load into	GEOGRAPH	HY column					
		Generate	simple geo	metries instead	l of MULTI geomet	tries			

Once everything is configured, click the **Import** button.

Check the Log Window to confirm whether the import was successful.

#### **Exporting Shapefiles**

You can also export Shapefiles from the database. To do this, select the table, then specify the geometry column and set the output file name.

port Export				
Schema	Table	Geo Column	Filename	Rm
public	autout	au_geom	autout	
		Add Table		

## Troubleshooting FAQ

Here are solutions to common issues:

#### Issue: I can't see the database, even though I created it.

Right-click on the username (e.g., postgres)  $\rightarrow$  Edit connection

Go to the **PostgreSQL tab**  $\rightarrow$  check the box **Show all databases**.

Connection "postgres"	configuration	— 🗆 X
Connection settings PostgreSQL connection set	ttings	PostgreSQL
<ul> <li>Connection settings Initialization Shell Commands Client identification Transactions General Metadata Errors and timeouts</li> <li>Data editor</li> <li>SQL Editor</li> </ul>	Main       PostgreSQL       Driver properties       SSH       Proxy       SSL         Settings       Show all databases       Show vallabases       Show template databases       Show template databases       Show databases not available for connection       Show database statistics       Read all data types       Read all data types         Read all data types       Read table keys with columns       SQL       Show SS quote as:       Code block       V         Show StagNameS quote as:       Code block       V         Performance       Use prepared statements       V	
Test Connection		OK Cancel

Issue: The font in the window is too small. I want to increase it.

Navigate to: Window  $\rightarrow$  Preferences  $\rightarrow$  User Interface  $\rightarrow$  Colors and Fonts  $\rightarrow$  Basic  $\rightarrow$  Text Font  $\rightarrow$  Click Edit to increase the font size.

Issue: A long SQL query appears in one line in the SQL Console. I want to format it.

Use the shortcut: **CTRL** + **SHIFT** + **F** (Auto-format)

# **CHAPTER 10: POSTGIS Topology**

PostGIS allows us to built topological data structure and work with them. To use these features, add the PostGIS Topology Extension. A detailed description of this module can be found here:

https://postgis.net/docs/Topology.html

Why use a topological data structure instead of standard geometry?

- To store spatial relationships explicitly
- Reduced storage size every edge is stored once  $\rightarrow$  more compact geometry representation
- Ensures topological integrity: every intersection becomes a node
- Shared edges are stored once, rather than being duplicated for each polygon

## Topological integrity

• Every intersection is a node



Sandro Santilli <strk@keybit.net>

http://strk.keybit.net

Source of image: http://strk.kbt.io/projects/postgis/Paris2011\_TopologyWithPostGIS\_2\_0.pdf

Topological data structure uses four data type for storing data:

- nodes (marked with red dots in the image above)
- edges (lines between nodes, marked green for shared edges and black individual ones.
- faces (entities built from edges, similar to polygons in geometry)
- relation (define relationships between topological elements)

#### When should you use topological data structures?

Topological structures are especially useful in certain cases. The example below demonstrates such a case. The hungary database contains a county table with data suited for a map scale of 1:300,000–500,000. Our goal is to **generalize** these county borders using a **line simplification** algorithm to make the resulting map suitable for a **1:2,000,000** scale. Among various generalization algorithms, the most commonly used is the **Douglas–Peucker** (or Ramer–Douglas–Peucker) algorithm. This recursive algorithm reduces the number of nodes while preserving the vertices that contribute to the line's overall shape.

If you apply a simplification algorithm directly to geometries:

- Every feature is simplified **independently**
- The algorithm does not account for shared borders
- This can cause topological errors, like gaps or overlaps between polygons

So, **geometry simplification is not suitable** for small-scale maps if shared borders must remain consistent.

#### Try it yourself:

Even when using ST\_SimplifyPreserveTopology, shared borders can still break:

select co\_name, ST\_Simplify(co\_geom, 1000) from county

select co\_name, ST\_SimplifyPreserveTopology(co\_geom, 1000) from county



Upper-left: Original layer: counties of Hungary. Upper-right: Topological errors of the feature simplification. Bottom-center image: yellow dots show the nodes of topology

The solution is to introduce the topological data structure. In this case, you will simplify the edges (single or shared shared edges are stored only once, along with which **faces** they belong to. The result maintains **topological correctness**.

How to do this with PostGIS?

#### Outline

- Create an empty topology table and populate it with the transformed geometry from county table
- Create one more empty topology table. Call the Douglas–Peucker algorithm on the previous table, and upload the simplified topology to this table.
- Convert the simplified topolpgy back to geometry.

Douglas–Peucker algorithm will be described at the end of this lesson. Now, let's do the task step by step.

Add a topology Schema and the postgis.topology extension to the database. **create schema IF NOT EXISTS** topology;

create extension IF NOT EXISTS postgis\_topology SCHEMA topology;

Copy the original county table and add a new field for a simplified geometry.

create table new\_county as (select co\_id, co\_name, co\_geom from county);

alter table new\_county add column simplified\_county geometry(POLYGON, 23700);

Create an empty topology table.

select topology.CreateTopology('topo1',23700)

Populate the empty topology table with the transformed data from county table.

select ST\_CreateTopoGeo('topo1', ST\_Collect(co\_geom)) from county

Create another empty topology for the simplified result.

```
select topology.CreateTopology('topo2',23700);
```

Simplify the edges with Douglas–Peucker algorithm and populate the second table.

```
select ST_CreateTopoGeo('topo2', geom) from (select
ST_Collect(ST_SimplifyPreserveTopology(geom, 1000)) as geom from topo1.edge_data)
As foo;
```

Reconstruct the simplified geometry from the simplified topology. Use with clause.

with simple\_face As (select ST\_GetFaceGeometry('topo2',face\_id) as geom from topo2.face where face\_id>0) update new\_county d set simplified\_county=sf.geom from simple\_face sf where ST\_Intersects(d.co\_geom, sf.geom) and ST\_Area(ST\_Intersection(sf.geom,d.co\_geom))/ST\_Area(sf.geom)>0.5;

Detailed explanation: an Update statement uses the with clausule to reconstruct the polygons from faces. ST\_GetFaceGeometry returns the polygon in the given topology with the specified face id. Builds the polygon from the edges making up the face.

Select ST\_GetFaceGeometry('topo2',face\_id) as geom from topo2.face where face\_id>0

The simplified\_county table is populated from the result of select statement. The new – simplified – geometry is inherited from

```
update new_county d set simplified_county=sf.geom from simple_face sf where
ST_Intersects(d.co_geom, sf.geom) and
ST_Area(ST_Intersection(sf.geom,d.co_geom))/ST_Area(sf.geom)>0.5;
```

To restore the geometry, the original geometry of the new county and the polygons reconstructed from the faces must intersect, and the ratio of the area of their intersection to the area of the reconstructed polygons must be greater than 0.5. This condition is included because we did not store descriptive data for the topologies. Therefore, I identify the simplified county polygons based on which original polygon

they significantly overlap with (after simplification, there may be minor overlaps with neighboring counties, which must be excluded).

Other functions in the task:

 $ST_Collect() \rightarrow$  create geometry collection from other data types

ST\_CreateTopoGeo()  $\rightarrow$  convert topology from geometry

 $ST_GetFaceGeometry() \rightarrow$  create faces from edges with similar id-s

How WITH clause works:

A **CTE** (Common Table Expression) is a temporary, named result set in SQL that you define at the start of a query using the WITH clause. You can use it just like a table or subquery in the main part of your SQL statement. In PostgreSQL, the WITH clause is used to define **Common Table Expressions** (**CTEs**). A CTE is a temporary result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement. It helps structure complex queries and improves readability.

Syntax:

```
with temporaryTable (averageValue) as
 (select Avg(Attr1)
 from table)
 select Attr1
 from Table, temporaryTable
 where table.Attr1 > temporaryTable.averageValue;
```

Lets see an example from *library* database. Which salaries are bigger than the average salary?

```
with libr(avgsalary) as
   (select Avg(li_salary)
   from librarian)
   select li_job from librarian, libr
   where librarian.li_salary > libr.avgsalary;
```

One more example: the average salary in which position is bigger than the average salary of librarian table?

```
with libr(avgsalary) as
(select Avg(li_salary)
    from librarian),
    job(avgsalary, job) as (select Avg(li_salary), li_job from librarian group by
li_job)
    select job.job from job,libr where job.avgsalary>Libr.avgsalary;
```

Other tutorials about PostGIS Topology:

http://strk.kbt.io/projects/postgis/Paris2011\_TopologyWithPostGIS\_2\_0.pdf https://trac.osgeo.org/postgis/wiki/UsersWikiPostgisTopology

#### What is ST\_Simplify and ST\_SimplifyPreserveTopology?

The most geoinformatics software offers algorithm(s) for line generalization. The most popular algorithm is the Douglas–Peucker, which can be found in every software with different naming. This algorithm was find out to reduce the number of nodes of a line, but it can be applied for polygons as well. In this case, it simplify the ring (exterior and interior) of the polygon. If you simplify a polygon layer, be careful: do not simplify the features one by one, because you will get topological errors (gaps or overlaps). Use a topology data structure instead as in this chapter you have seen.

#### How it works (step-by-step):

#### Input:

- A curve made up of a series of ordered points.
- A tolerance ( $\epsilon$ ) the maximum allowed deviation between the original and simplified curve.

#### **Basic Steps:**

- Start with the first and last point of the polyline. These will always be part of the simplified result.
- Find the point farthest from the line segment between the first and last points. This is the point that causes the most "bending."
- If this distance is greater than  $\varepsilon$ :
  - Keep this point.
  - Recursively apply the same process to the two segments:
  - From the first point to the farthest point.
  - From the farthest point to the last point.
- If the maximum distance  $\leq \varepsilon$ , remove all intermediate points the entire curve segment can be approximated by the straight line.

#### **Recursive Nature:**

The algorithm uses divide and conquer, splitting the polyline wherever the maximum deviation is too large, until all segments are within tolerance.



Source of image: Wikipédia. (https://it.wikipedia.org/wiki/Algoritmo\_Ramer-Douglas-Peucker)

# Chapter 10: Appendix – databases and files in these lecture notes

The CSV (Comma Separated Values) files stores data in a table-like structure as a plain text. Each line represents a database record, and the columns are separated by a delimiter (such as a comma, semicolon, tab, etc.).

#### europa.csv

This file contains the names, area, polulation, population density and the capital city of European countries.

#### SQL

.*sql* scripts are used for backing up and the migrating databases. They contains the whole table structure and data as well. Be careful, MySQL and PostgreSQL scripts are incompatible with each other.

#### library.sql

It contains the *library* database. The tables are *catalogue*, *publication*, *publication\_group*, *borrowing*, *borrower*, *librarian*, *department*.

#### hungary.sql

Table structure of hungary database. EPSG/SRID 23700 for all tables, except castles.

		Field name	Data type	
	county	co_id	serial4	Primary key
		co_name	varchar(254)	Name of county (=vármegye)
		co_county_code	int4	A custom code for the county
y		co_geom	geometry (MultiPolygon)	Geometry field
ngar	district	d_id	serial4	Primary key
f dataset hur		d_name	varchar(254)	District name (járás)
		d_county_code	int4	the code of the county, which contains the district
les o		d_district_code	int4	A custom code for district
ı tabl		d_geom	geometry (MultiPolygon)	Geometry field
t,	settlement	s_id	serial4	Primary key
		s_name	varchar(254)	Name of the settlement
		s_name_de	varchar(254)	german name
		s_jaras_kod	int4	the code of the district , which contains the settlement
		s_geom	geometry (MultiPolygon)	Geometry field

	settlement_centroid	sc_id	int4	Primary key
		sc_name	varchar(254)	Name of the settlement
		sc_geom	geometry	Geometry field (the centorid of built up areas)
	builtup_area	ba_id	int4	Primary key
		ba_name	varchar(254)	Name of the settlement
		ba_geom	geometry (MultiPolygon)	Geometry field (only the built up areas)
	motorway	mo_id	int4	Primary key
		mo _wayid	varchar(254)	OSM identifier
		mo _hu_ed_dire	varchar(254)	direction of the motorway from Budapest
		mo _alt_name	varchar(254)	Old name of the motorway (in the most cases is NULL)
		mo_alt_bridge	varchar(254)	is it a bridge?
		mo _highway	varchar(254)	type: motorway
		mo _int_ref	varchar(254)	Other number of the road
		mo _lanes	varchar(254)	Number of lanes
		mo _maxspeed	varchar(254)	Maximum speed
		mo _oneway	varchar(254)	Is it a one way road?
		mo _ref	varchar(254)	Number of the road
		mo_surface	varchar(254)	covering material
		mo _toll	varchar(254)	is there a toll?
		mo _geom	geometry(multilinestring)	Geometry field
	highway	hi_id	int4	Primary key
		hi _wayid	varchar(254)	OSM identifier
		hi _hu_ed_dire	varchar(254)	direction of the road from Budapest
		hi _bridge	varchar(254)	is it a bridge?
		hi _foot	varchar(254)	can people walk on road?
		hi _highway	varchar(254)	road type
		hi _horse	varchar(254)	Can people use horse carriage?
		hi _int_ref	varchar(254)	Other road numbering
		hi _maxspeed	varchar(254)	Maximum speed
		A	*	• • • • • • • • • • • • • • • • • • • •

	hi _name	varchar(254)	road name
	hi _oneway	varchar(254)	One way?
	hi _ref	varchar(254)	Number of the road
	hi _surface	varchar(254)	covering material
	hi _toll	varchar(254)	is there a toll?
	hi _tunnel	varchar(254)	is it a tunnel?
	hi _geom	geometry(multilinestring)	Geometry field
primary highway	ph_id	int4	Primary key
	ph _wayid	varchar(254)	OSM identifier
	ph _hu_ed_dire	varchar(254)	Road direction from Budapest
	ph_bicycle	varchar(254)	Can peole ride a bicycle?
	ph_bridge	varchar(254)	Is it a bridge?
	ph_foot	varchar(254)	Can people walk there?
	ph_highway	varchar(254)	road type
	ph_horse	varchar(254)	can people ride a horse carriage?
	ph_int_ref	varchar(254)	other road numbering
	ph_maxspeed	varchar(254)	maximum speed
	ph_name	varchar(254)	Name of the road section
	ph_oneway	varchar(254)	One way road?
	ph_ref	varchar(254)	Number of the primary road
	ph_surface	varchar(254)	covering material
	ph_toll	varchar(254)	Is there a toll?
	ph_tunnel	varchar(254)	Is it a tunnel?
	ph_geom	geometry(multilinestring)	Geometry field
secondary highway	sh_id	int4	Primary key
	sh_wayid	varchar(254)	OSM identifier
	sh_bicycle	varchar(254)	can people ride a bicycle?
	sh_bridge	varchar(254)	is it a bridge?
	sh_foot	varchar(254)	can people walk there?
	sh_highway	varchar(254)	road type
	sh_horse	varchar(254)	can people ride a horse carriage?

 	-1		
	sn_maxspeed	varchar(254)	maximum speed
	sh_name	varchar(254)	name of the road section
	sh_oneway	varchar(254)	one way?
	sh_ref	varchar(254)	Number of the secondary road
	sh_surface	varchar(254)	covering material
	sh_toll	varchar(254)	is there a toll?
	sh_tunnel	varchar(254)	is there a tunnel?
	sh_geom	geometry(multilinestring)	Geometry field
tertiary highway	th_id	int4	Primary key
	th_wayid	varchar(254)	OSM identifier
	th_bicycle	varchar(254)	can people ride a bicycle?
	th_bridge	varchar(254)	is it a bridge?
	th_foot	varchar(254)	can people walk there?
	th_highway	varchar(254)	road type
	th_horse	varchar(254)	can people ride a horse carriage?
	th_maxspeed	varchar(254)	maximum speed
	th_name	varchar(254)	name of the road section
	th_oneway	varchar(254)	one way?
	th_ref	varchar(254)	Number of the tertiary road
	th_surface	varchar(254)	coverint material
	th_toll	varchar(254)	is there a toll?
	th_tunnel	varchar(254)	is it a tunnel?
	th_geom	geometry(multilinestring)	Geometry field
railway	r_id	serial4	Primary key
			5 5
	r_name	varchar(48)	Nam of the section, if there is
	r_name r_type	varchar(48) varchar(16)	Nam of the section, if there is type
	r_name r_type r_linenumber	varchar(48) varchar(16) varchar(10)	Nam of the section, if there is type Number of the line
	r_name r_type r_linenumber r_status	varchar(48) varchar(16) varchar(10) varchar(15)	Nam of the section, if there is type Number of the line Current status of the section.
	r_name r_type r_linenumber r_status r_agency	varchar(48) varchar(16) varchar(10) varchar(15) varchar(5)	Nam of the section, if there istypeNumber of the lineCurrent status of the section.Agency name
	r_name r_type r_linenumber r_status r_agency r_geom	<pre>varchar(48) varchar(16) varchar(10) varchar(15) varchar(5) geometry(multilinestring)</pre>	Nam of the section, if there is type Number of the line Current status of the section. Agency name Geometry field

river_polygon	rp_id	int4	Primary key
	rp_name	varchar(254)	Name
	rp_geom	geometry(multipolygon)	Geometry field
river_line	rl_id	int4	Primary key
	rl_name	varchar(254)	River name
	rl_category	varchar(50)	River category
	rl_geology	varchar(50)	The geological environment of the river
	rl_steepness	varchar(50)	Steepness
	rl_crossing	varchar(6)	Crossing identifier
	rl_desc	varchar(250)	Detaild description
	rl_geom	geometry(multilinestring)	Geometry field
river	ri_id	int4	Primary key
	ri_name	varchar(254)	Primary name of the river
	ri_name_hu	varchar(254)	Hungarian name
	ri_geom	geometry(multilinestring)	Geometry field
creek	cr_id	int4	Primary key
	cr_name	varchar(254)	Name
	cr_geom	geometry(multilinestring)	Geometry field
lake	la_id	serial4	Primary key
	la_name	varchar(100)	Lak name
	la_category	varchar(30)	Lake category
	la_elevation	numeric	Elevation above sea level
	la_general	varchar(50)	general environment of the lake
	la_depth	numeric	lake depth
	la_desc	varchar(150)	description
	la_geom	geometry(multipolygon)	Geometry field
hungary_border	mo_id	int4	Primary key
	mo_name	varchar(254)	Name of the county
	mo_geom	geometry(multipolygon)	Geometry field
geobox	gid	serial(4)	Primary key
	gl_id	float(8)	other id
	gl_lat_d	float(8)	latitude in degree

		gl lat m	numeric	latitude in minute
		0	<b>G</b> (0)	
		gl_lon_d	float(8)	longitude in degree
		gl_lon_m	numeric	longitude in minute
		gl_elevation	numeric	elevation above sea level
		gl_full_name	varchar(254)	name of the geobox
		gl_type	varchar(254)	type of the geobox
		gl_county_country	varchar(254)	Name of the county and country where it can be found
		gl_terep	numeric	difficulty of the road
		gl_waypoint	varchar(254)	short id
		gl_description	varchar(254)	geobox description
		gl_felhaszn	varchar(254)	owner
		gl_email	varchar(254)	owner email
		gl_field_30	numeric	latitude in degrees
		gl_field_31	numeric	longitude in degress
		gl_geom	geometry(point(23700))	Geometry field
	nature_protection	np_id	in(4)	Primary key
		np_name	varchar(254)	Name of the protected area
		np_type	varchar(50)	National park (nemzeti park) or local protected area (tájvédelmi körzet)
		np_geom	geometry(MultiPolygon)	Geometry field
	castles	ca_id	int(4)	Primary key
		ca_name	varchar(254)	Name of the castle
		ca_geom	geometry(geometry,4326)	Geometry field (EPSG: 4326)
	spatial_ref_sys			Built in PostGIS table, it stores the projections and spatial reference system.