

Introduction of SQL

Definition

- Databases are systems that allow users to store and organize data.
- They are useful when dealing with large amounts of data.

Typical Users

- Databases have a wide variety of users!
- Analysts
 - Marketing
 - Business
 - Sales
- Technical
 - Data Scientist
 - Software Engineers
 - Web Developers
- Basically anyone needing to deal with data!

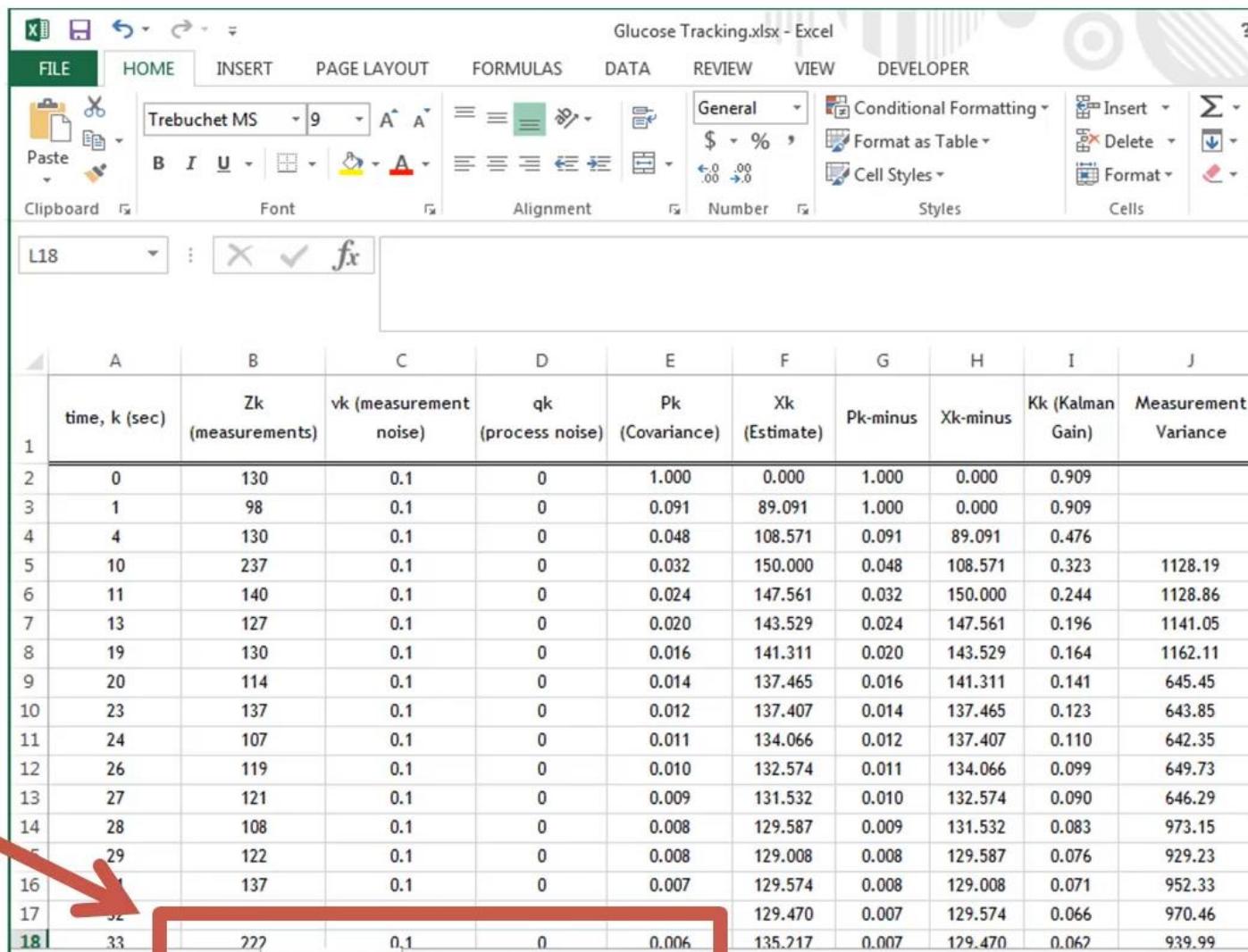
From Spreadsheets to Databases

- Spreadsheets
 - One-time analysis
 - Quickly need to chart something out
 - Reasonable data set size
 - Ability for untrained people to work with data

From Spreadsheets to Databases

- Databases
 - Data Integrity
 - Can handle massive amounts of data
 - Quickly combine different datasets
 - Automate steps for re-use
 - Can support data for websites and applications

From Spreadsheets to Databases

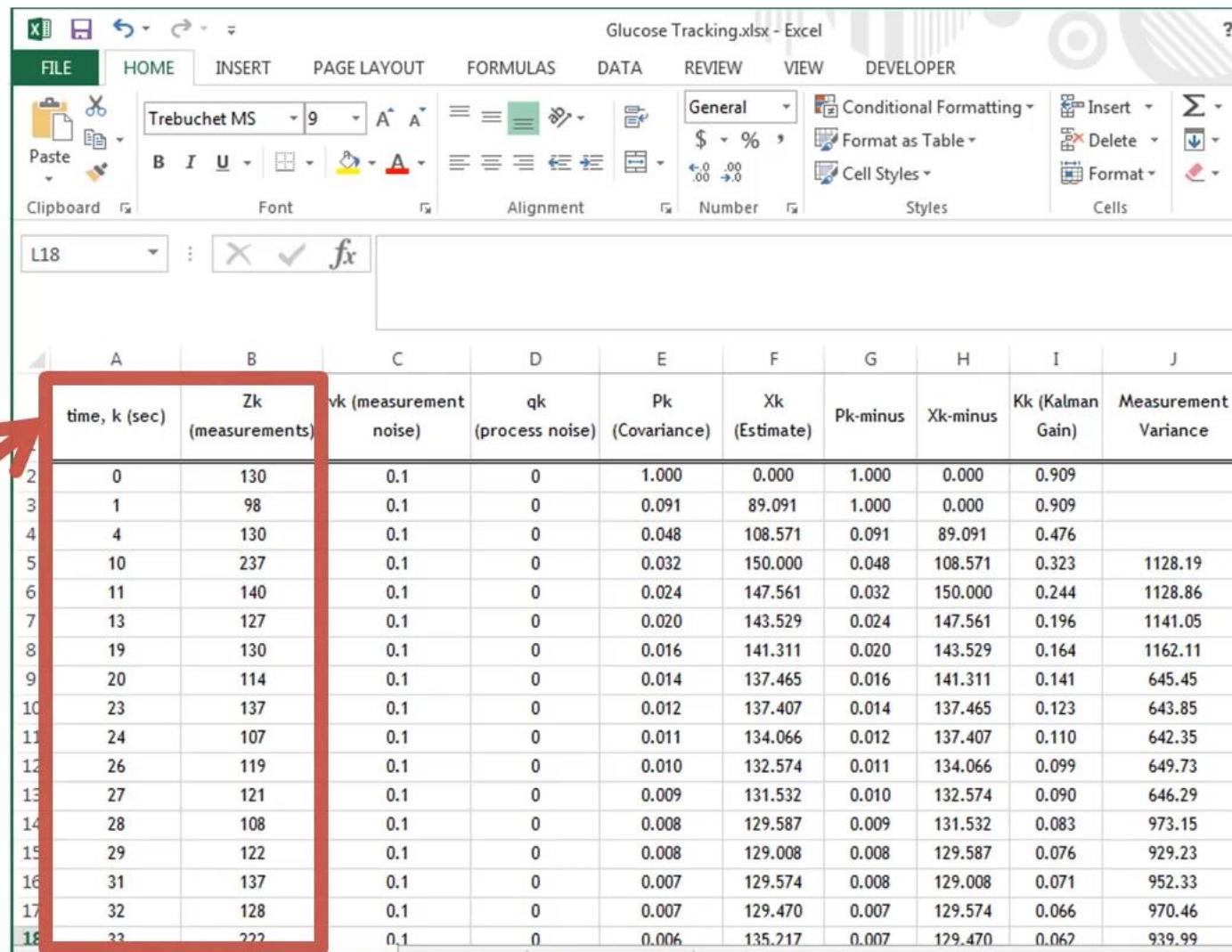


The screenshot shows a Microsoft Excel spreadsheet titled "Glucose Tracking.xlsx - Excel". The ribbon menu is visible at the top, showing tabs for FILE, HOME, INSERT, PAGE LAYOUT, FORMULAS, DATA, REVIEW, VIEW, and DEVELOPER. The HOME tab is selected. The font and number formats are set to "General". The table below contains data for glucose tracking, with columns labeled: time, k (sec), Zk (measurements), vk (measurement noise), qk (process noise), Pk (Covariance), Xk (Estimate), Pk-minus, Xk-minus, Kk (Kalman Gain), and Measurement Variance.

	A	B	C	D	E	F	G	H	I	J
1	time, k (sec)	Zk (measurements)	vk (measurement noise)	qk (process noise)	Pk (Covariance)	Xk (Estimate)	Pk-minus	Xk-minus	Kk (Kalman Gain)	Measurement Variance
2	0	130	0.1	0	1.000	0.000	1.000	0.000	0.909	
3	1	98	0.1	0	0.091	89.091	1.000	0.000	0.909	
4	4	130	0.1	0	0.048	108.571	0.091	89.091	0.476	
5	10	237	0.1	0	0.032	150.000	0.048	108.571	0.323	1128.19
6	11	140	0.1	0	0.024	147.561	0.032	150.000	0.244	1128.86
7	13	127	0.1	0	0.020	143.529	0.024	147.561	0.196	1141.05
8	19	130	0.1	0	0.016	141.311	0.020	143.529	0.164	1162.11
9	20	114	0.1	0	0.014	137.465	0.016	141.311	0.141	645.45
10	23	137	0.1	0	0.012	137.407	0.014	137.465	0.123	643.85
11	24	107	0.1	0	0.011	134.066	0.012	137.407	0.110	642.35
12	26	119	0.1	0	0.010	132.574	0.011	134.066	0.099	649.73
13	27	121	0.1	0	0.009	131.532	0.010	132.574	0.090	646.29
14	28	108	0.1	0	0.008	129.587	0.009	131.532	0.083	973.15
15	29	122	0.1	0	0.008	129.008	0.008	129.587	0.076	929.23
16	31	137	0.1	0	0.007	129.574	0.008	129.008	0.071	952.33
17	32	222	0.1	0	0.006	129.470	0.007	129.574	0.066	970.46
18	33					135.217	0.007	129.470	0.062	939.99

Tabs = Tables

From Spreadsheets to Databases



Glucose Tracking.xlsx - Excel

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER

Trebuchet MS 9 A A General Conditional Formatting Insert Σ
B I U Font Format as Table Delete
Clipboard Alignment Cell Styles
Font Number Styles Cells

L18 : X ✓ fx

A	B	C	D	E	F	G	H	I	J
time, k (sec)	Zk (measurements)	v _k (measurement noise)	q _k (process noise)	P _k (Covariance)	X _k (Estimate)	P _{k-minus}	X _{k-minus}	K _k (Kalman Gain)	Measurement Variance
2	0	130	0.1	0	1.000	0.000	1.000	0.000	0.909
3	1	98	0.1	0	0.091	89.091	1.000	0.000	0.909
4	4	130	0.1	0	0.048	108.571	0.091	89.091	0.476
5	10	237	0.1	0	0.032	150.000	0.048	108.571	0.323
6	11	140	0.1	0	0.024	147.561	0.032	150.000	0.244
7	13	127	0.1	0	0.020	143.529	0.024	147.561	0.196
8	19	130	0.1	0	0.016	141.311	0.020	143.529	0.164
9	20	114	0.1	0	0.014	137.465	0.016	141.311	0.141
10	23	137	0.1	0	0.012	137.407	0.014	137.465	0.123
11	24	107	0.1	0	0.011	134.066	0.012	137.407	0.110
12	26	119	0.1	0	0.010	132.574	0.011	134.066	0.099
13	27	121	0.1	0	0.009	131.532	0.010	132.574	0.090
14	28	108	0.1	0	0.008	129.587	0.009	131.532	0.083
15	29	122	0.1	0	0.008	129.008	0.008	129.587	0.076
16	31	137	0.1	0	0.007	129.574	0.008	129.008	0.071
17	32	128	0.1	0	0.007	129.470	0.007	129.574	0.066
18	22	???	0.1	0	0.006	135.217	0.007	129.470	0.062
19	23	???	0.1	0	0.005	135.217	0.006	129.470	0.059

Columns

From Spreadsheets to Databases

Glucose Tracking.xlsx - Excel

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER

Trebuchet MS 9 A A General Conditional Formatting Insert \sum
Clipboard Paste B I U Font Format as Table Delete S
Font Alignment Number Cell Styles Styles Cells

L18 : X ✓ fx

	A	B	C	D	E	F	G	H	I	J
1	time, k (sec)	Zk (measurements)	vk (measurement noise)	qk (process noise)	Pk (Covariance)	Xk (Estimate)	Pk-minus	Xk-minus	Kk (Kalman Gain)	Measurement Variance
2	0	130	0.1	0	1.000	0.000	1.000	0.000	0.909	
3	1	98	0.1	0	0.091	89.091	1.000	0.000	0.909	
4	4	130	0.1	0	0.048	108.571	0.091	89.091	0.476	
5	10	237	0.1	0	0.032	150.000	0.048	108.571	0.323	1128.19
7	13	127	0.1	0	0.020	143.529	0.024	147.561	0.196	1141.05
8	19	130	0.1	0	0.016	141.311	0.020	143.529	0.164	1162.11
9	20	114	0.1	0	0.014	137.465	0.016	141.311	0.141	645.45
10	23	137	0.1	0	0.012	137.407	0.014	137.465	0.123	643.85
11	24	107	0.1	0	0.011	134.066	0.012	137.407	0.110	642.35
12	26	119	0.1	0	0.010	132.574	0.011	134.066	0.099	649.73
13	27	121	0.1	0	0.009	131.532	0.010	132.574	0.090	646.29
14	28	108	0.1	0	0.008	129.587	0.009	131.532	0.083	973.15
15	29	122	0.1	0	0.008	129.008	0.008	129.587	0.076	929.23
16	31	137	0.1	0	0.007	129.574	0.008	129.008	0.071	952.33
17	32	128	0.1	0	0.007	129.470	0.007	129.574	0.066	970.46
18	33	222	0.1	0	0.006	135.217	0.007	129.470	0.062	939.99

Rows →

Database Platform Options

PostgreSQL		Free (Open Source) Widely used on internet Multi platform
MySQL MariaSQL	 	Free (Open Source) Widely used on internet Multi platform.
MS SQL Server Express		Free, but with some limitations Compatible with SQL Server Windows only
Microsoft Access		Cost (-) Not easy to use just SQL (-)
SQLite		Free (Open Source) Mainly command line (-)

SQL

- PostGreSQL is a great Database choice for learning how to use SQL
- SQL (Structured Query Language) learned in this course can be applied to a variety of Databases or SQL based software!
- SQL is the programming language used to communicate with our Database

SQL Example

- We will learn how to write and use SQL

```
SELECT customer_id, first_name, last_name  
FROM sales  
ORDER BY first_name;
```

SQL is useful for a lot of things!

MySQL

PostGreSQL

Oracle Databases

Microsoft Access

Amazon's Redshift

Looker

MemSQL

Periscope Data

Hive (Runs on top of Hadoop)

Google's BigQuery

Facebook's Presto

- We will install
 - PostgreSQL - SQL Engine that stores data and reads queries and returns information.
 - PgAdmin - Graphical User Interface for connecting with PostgreSQL



postgresql



Ad · azure.microsoft.com/Services/PostgreSQL ▾

Azure PostgreSQL Service | Get Your Free Account

Managed PostgreSQL Service for App Developers. Focus on Apps Not Infrastructure.

www.postgresql.org ▾

PostgreSQL: The world's most advanced open source database

The official site for PostgreSQL, the world's most advanced open source database.

Results from postgresql.org



Downloads

Windows installers - Mac OS -
Ubuntu - Linux downloads - ...

Documentation

Manuals - 11 - 10 - PostgreSQL 12 -
9.6 - Books - Manual Archive

People also ask

What is the use of PostgreSQL?



What is difference between SQL and PostgreSQL?



PostgreSQL



PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and technical standards compliance. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users.

[Wikipedia](#)

License: PostgreSQL License (free and open-source, permissive)

Stable release: 12.1 / 14 November 2019; 2 months ago

Initial release date: July 8, 1996

Written in: C

Developer(s): PostgreSQL Global Development Group

Operating system: FreeBSD, Linux, macOS, OpenBSD, Windows

PostgreSQL: The World's Most Advanced Open Source Relational Database

[Download →](#)[New to PostgreSQL?](#)

New to PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

There is a wealth of information to be found describing how to [install](#) and [use](#) PostgreSQL through the [official documentation](#). The [open source community](#) provides many helpful places to become familiar with PostgreSQL, discover how it works, and find career opportunities. Learn more on how to [engage with the community](#).

[Learn More](#)[Feature Matrix](#)

Latest Releases

2023-05-25 - PostgreSQL 16 Beta 1 Released!

The PostgreSQL Global Development Group announces that the [first beta release of PostgreSQL 16](#) is now [available for download](#). This release contains previews of all features that will be available when PostgreSQL 16 is made generally available, though some details of the release can change during the beta period.

You can find information about all of the features and changes found in PostgreSQL 16 in the [release notes](#).

In the spirit of the open source PostgreSQL community, we strongly encourage you to test the new features of PostgreSQL 16 on your systems to help us eliminate bugs or other issues that may exist. While



25th May 2023: [PostgreSQL 16 Beta 1 Released!](#)

Quick Links

- [Downloads](#)
 - [Packages](#)
 - [Source](#)
- [Software Catalogue](#)
- [File Browser](#)

Downloads

PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

Packages and Installers

Select your operating system family:

[Linux](#)[macOS](#)[Windows](#)[BSD](#)[Solaris](#)

Source code

The source code can be found in the main [file browser](#) or you can access the source control repository directly at [git.postgresql.org](#). Instructions for building from source can be found in the [documentation](#).

Beta/RC Releases and development snapshots (unstable)

There are source code and binary [packages](#) of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.

3rd party distributions

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
15.3	postgresql.org ↗	postgresql.org ↗			Not supported
14.8	postgresql.org ↗	postgresql.org ↗			Not supported
13.11	postgresql.org ↗	postgresql.org ↗			Not supported
12.15	postgresql.org ↗	postgresql.org ↗			Not supported
11.20	postgresql.org ↗	postgresql.org ↗			Not supported
10.23*					
9.6.24*					
9.5.25*					
9.4.26*					

postgresql-15.4-1-windows-x64.exe
Canceled

Show all downloads

Get more out of PostgreSQL with EDB

EDB builds software for teams who need to do more and go faster with PostgreSQL.



EDB Big Animal Plan
Fully managed PostgreSQL in the cloud

CATEGORIES

EDB Postgres Advanced Server

Enterprise-ready, Oracle-compatible Postgres

EDB Postgres Distributed

Extreme high availability for Postgres

EDB Postgres for Kubernetes

Kubernetes operator and container images

Postgres Enterprise Manager

Manage, monitor, and optimize PostgreSQL

Postgres Databases

Enterprise-ready PostgreSQL

EDB Postgres Migration

Schema and data migration to PostgreSQL

PostgreSQL Monitoring & Query Performance

Manage, monitor, optimize PostgreSQL

Foreign Data Wrappers

Connecting to PostgreSQL

Postgres Clustering with EDB

High availability for PostgreSQL

Replication

Advanced logical replication for PostgreSQL

PostgreSQL Database Backup Solutions

Automate backup and recovery for PostgreSQL

Training and Certification

Build, enhance and validate PostgreSQL Skills

PostgreSQL expertise and support



Upcoming Webinar: The Profitability Impact of Self-Supported PostgreSQL • Register Now

Solutions

Software & Plans

Services & Support

Resources

Company

Partners

Contact Sales

Sign in

Get Started



SUBSCRIPTION PLANS

EDB BigAnimal Plans

Fully managed Postgres in the cloud

EDB Enterprise Plan

Extending Postgres for enterprises

EDB Standard Plan

PostgreSQL with enterprise tooling

EDB Community 360 Plan

Leverage the power of PostgreSQL

EDB Subscription Plan Comparison

FEATURED SOFTWARE

EDB Postgres Advanced Server

Enterprise-ready, Oracle-compatible Postgres

EDB Postgres Distributed

Five 9s Extreme High Availability

EDB Postgres for Kubernetes

Kubernetes operator and container images

SOFTWARE PORTFOLIO

Postgres Database

Postgres for every need

Postgres High Availability Clusters

Ensure high availability for Postgres

Postgres Migration

Migrate schemas and data to Postgres easily

Postgres Replication

Advanced logical replication for Postgres

Postgres Monitoring & Management

Manage, monitor, optimize Postgres performance

Postgres Backup and Recovery

Automate backup and recovery for Postgres

View EDB Software Portfolio

32

ed

ed

ed

ed

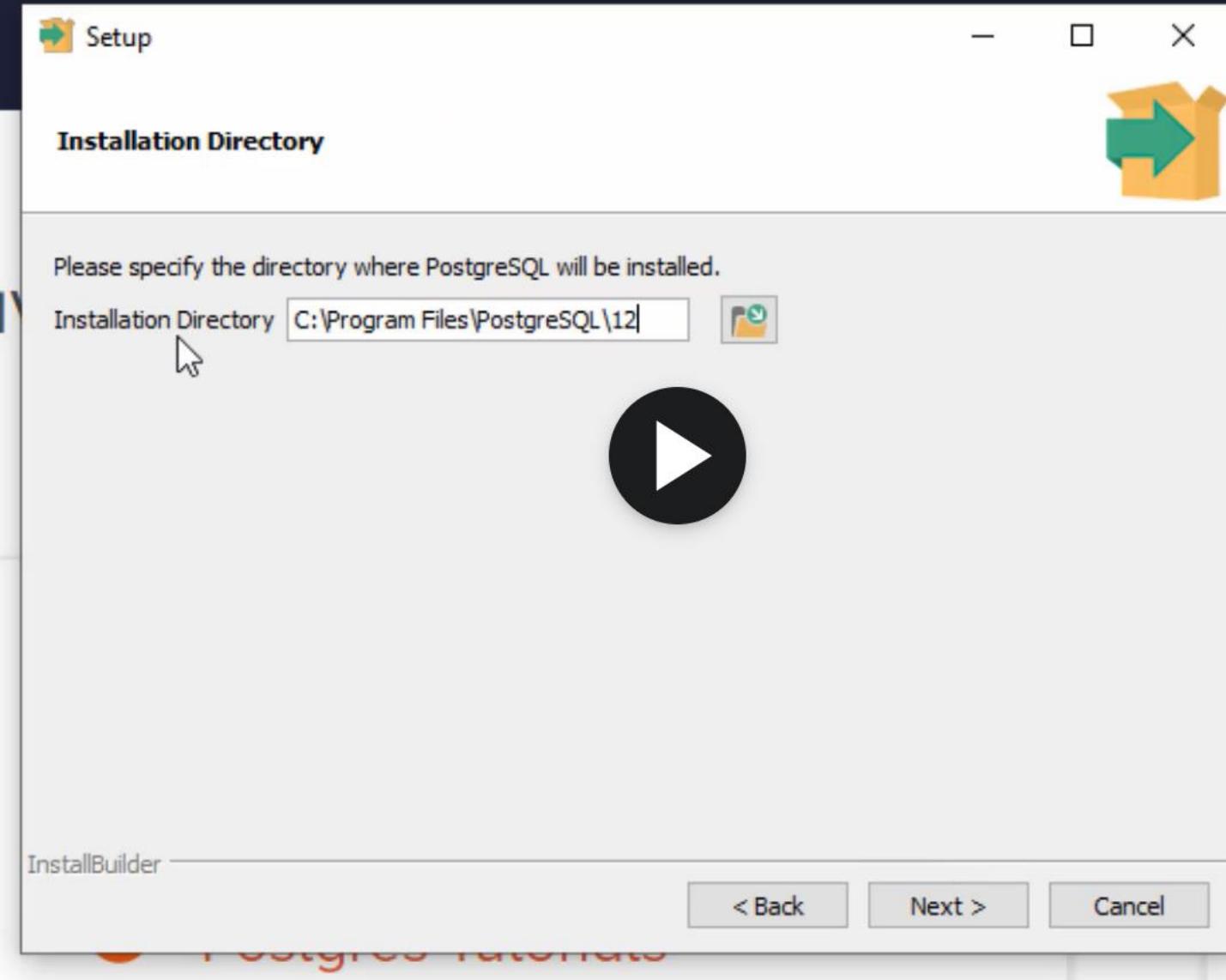
6. Windows Installation - PostGreSQL and PgAdmin with Database Setup



ted using it.

Get Postg
and Tricks

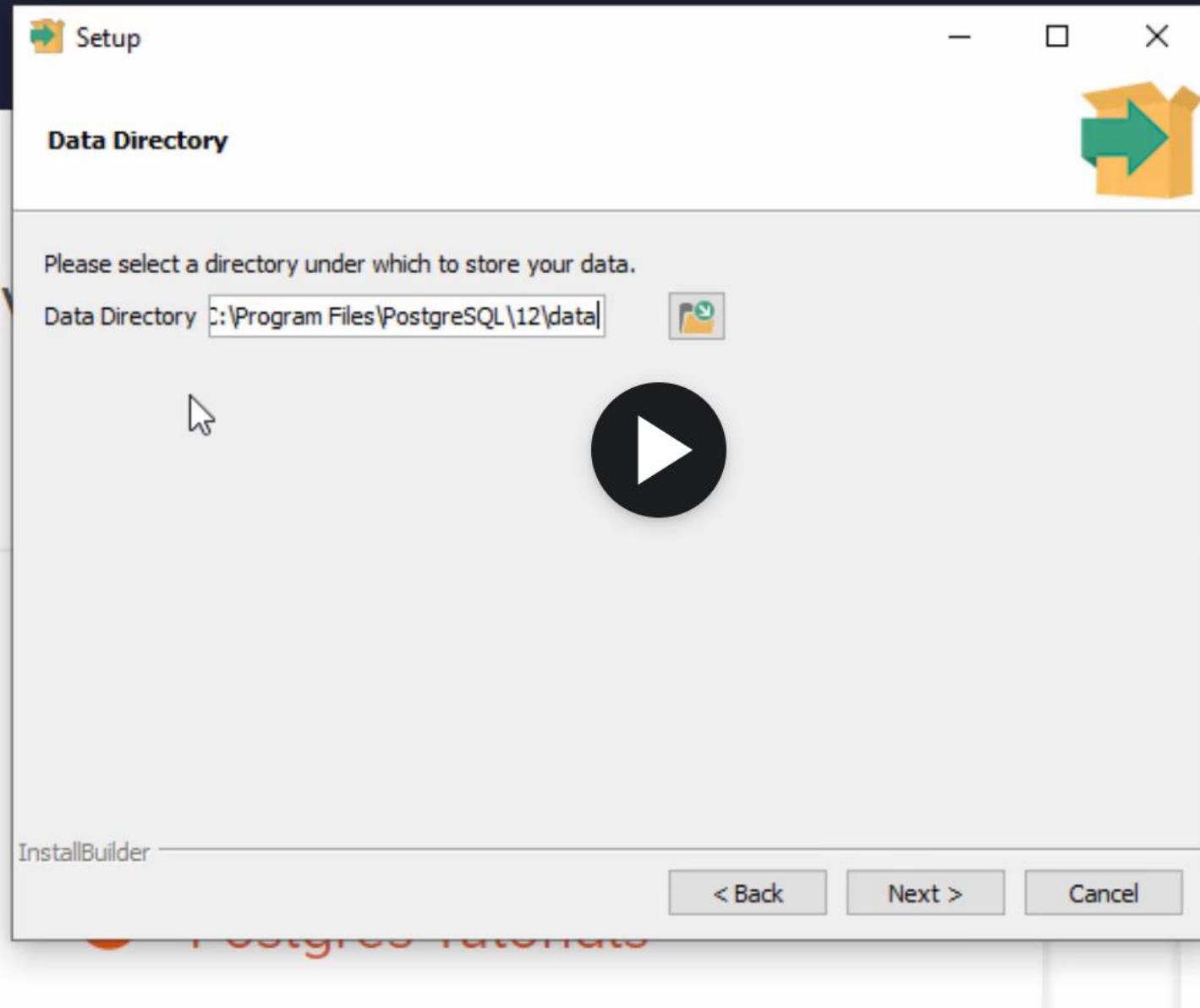
6. Windows Installation - PostGreSQL and PgAdmin with Database Setup



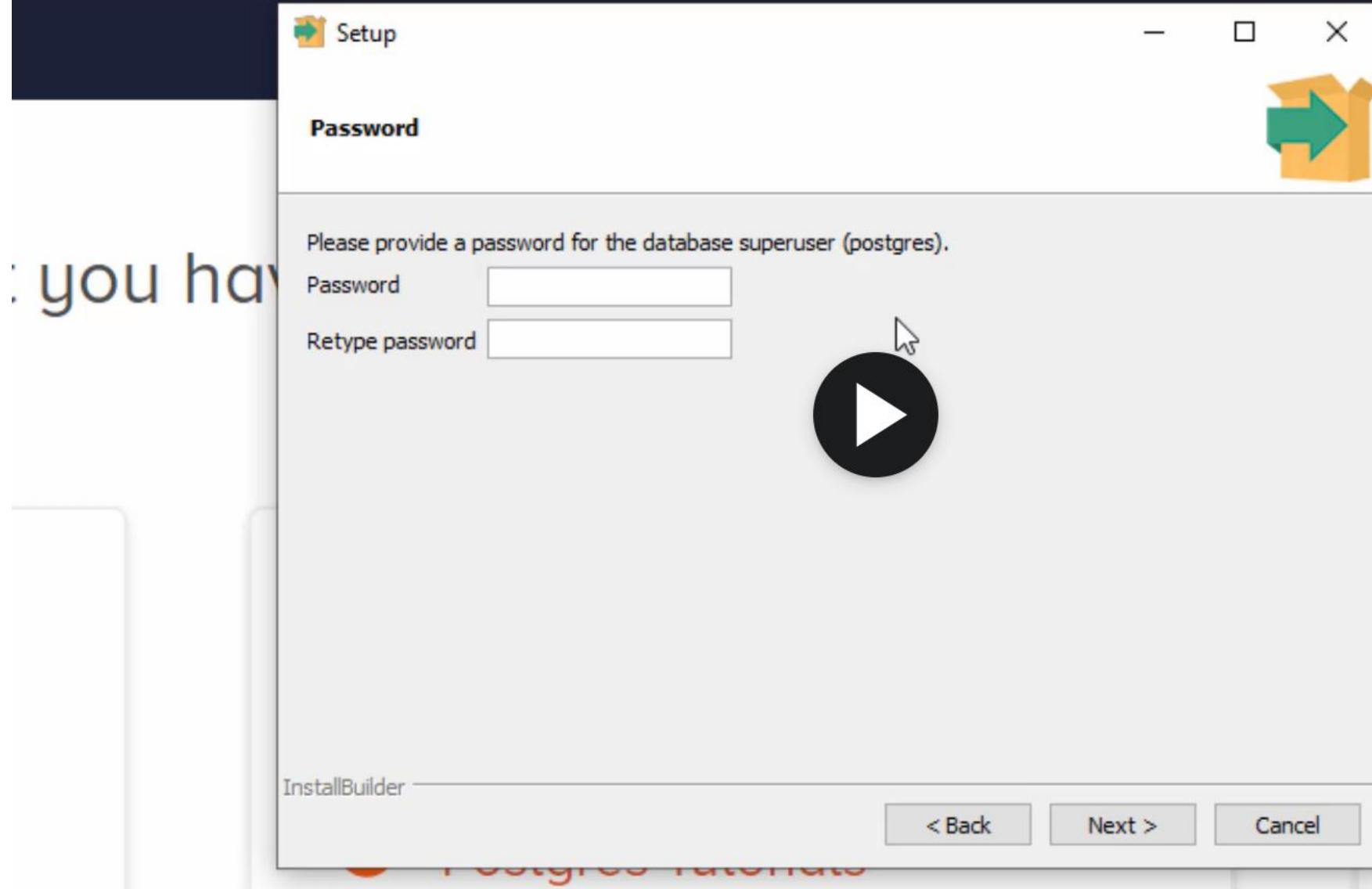
Get Postg
and Tricks

that you have

ted using it.



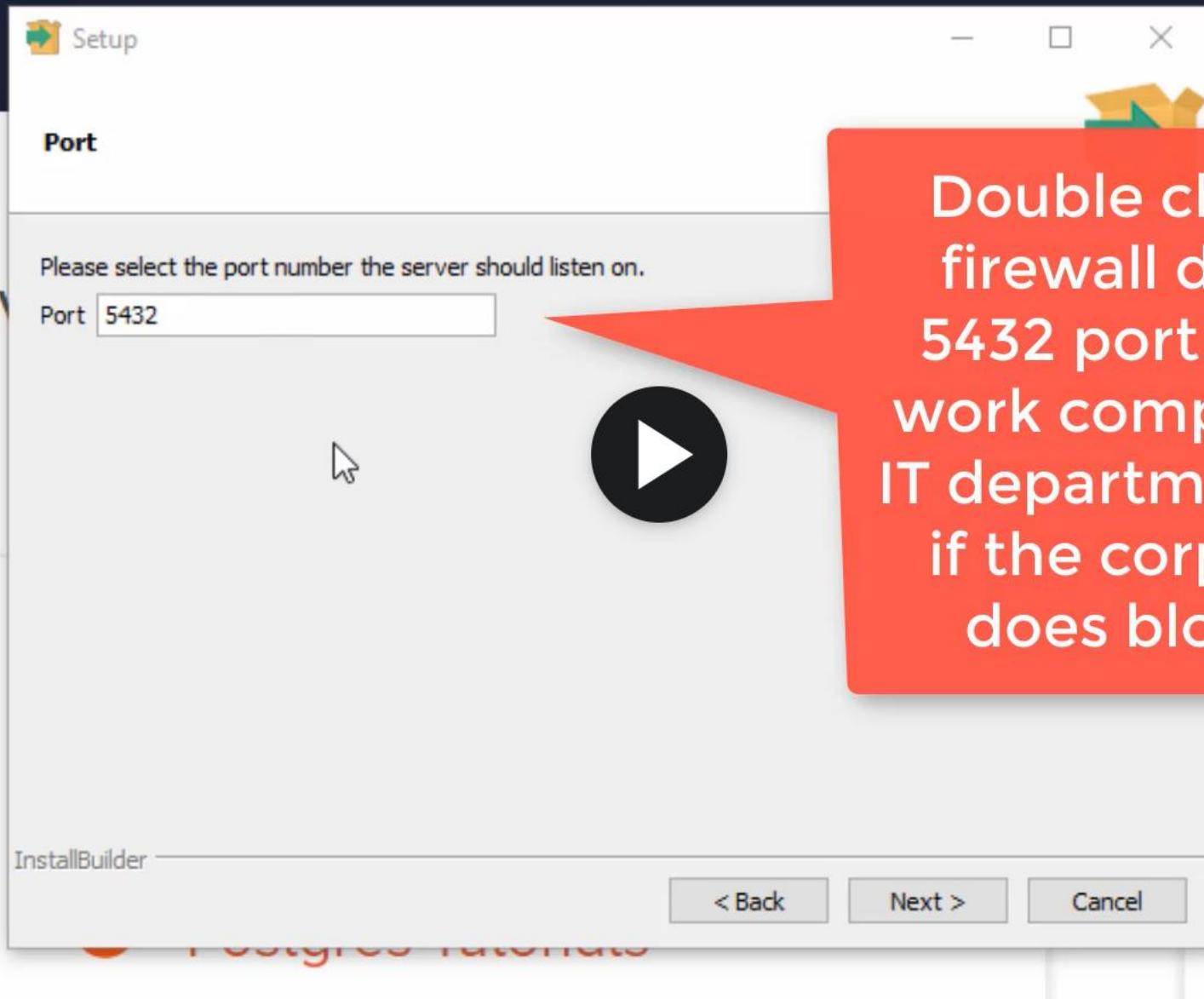
Get Postgi
and Tricks



ted using it.

Get Postgr
and Tricks

that you have



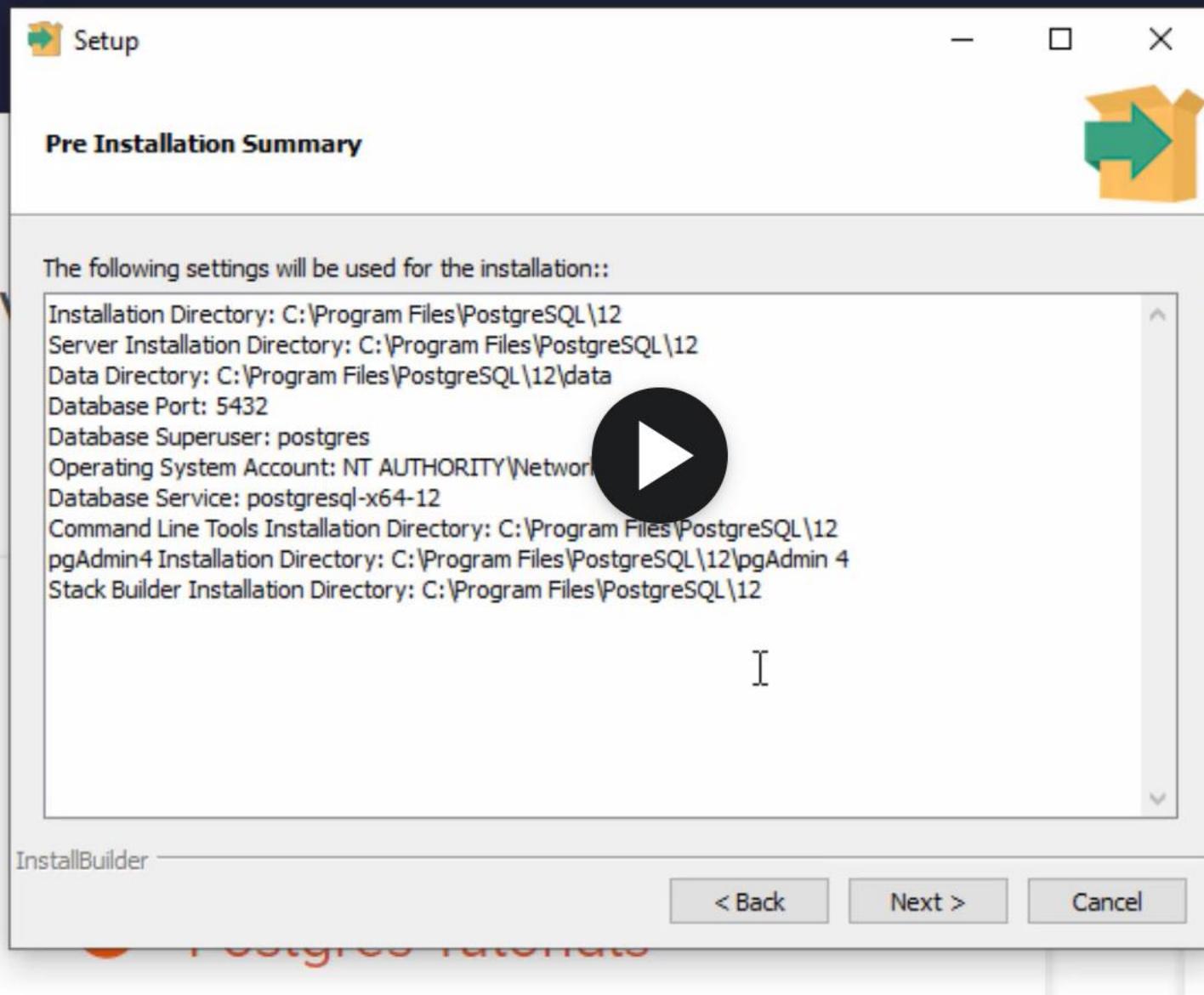
Double check that your firewall does not block 5432 port if you are on a work computer! (Ask your IT department for help just if the corporate firewall does block this port.)

Get PostgreSQL
and Tricks

that you have

ted using it.

Get Post
and Trick





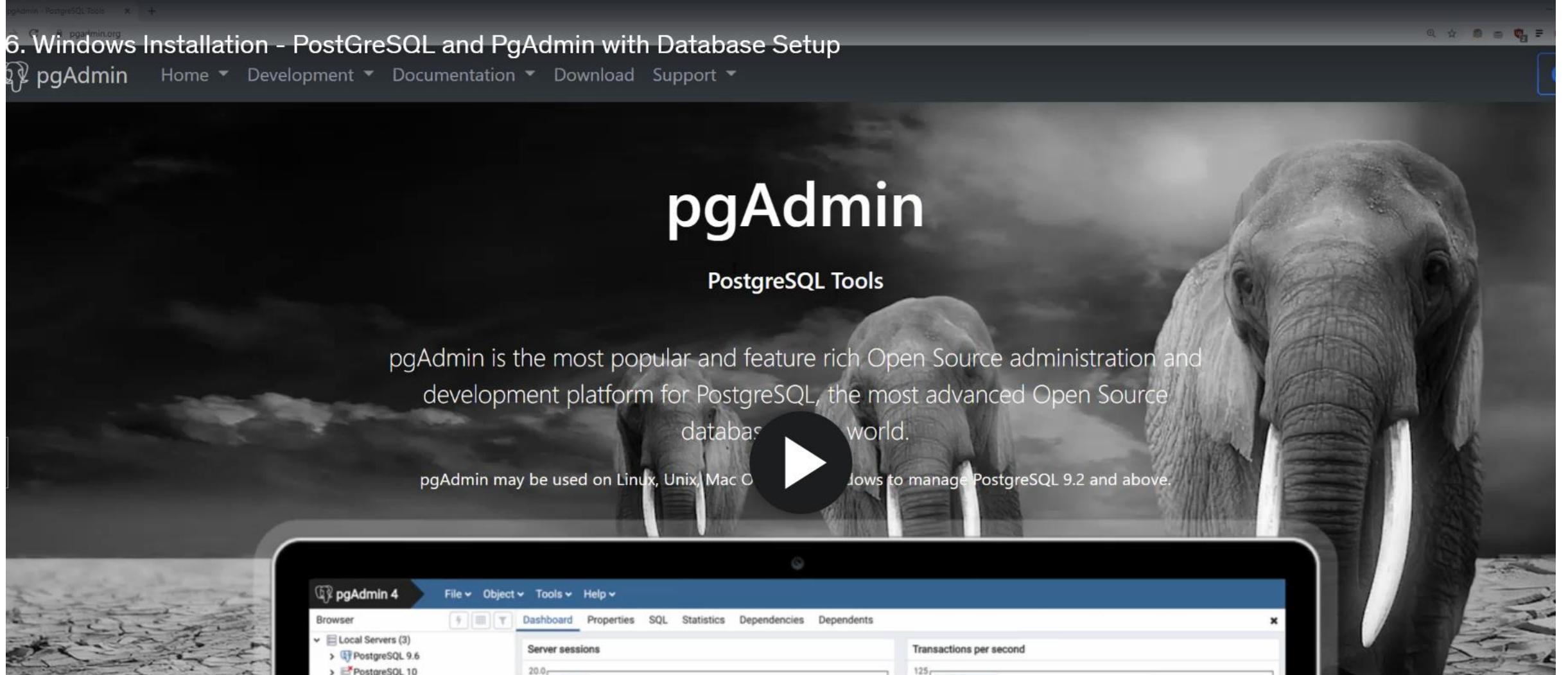
ted using it.

Get Postg
and Tricks



Step 2: Download and Install PgAdmin





Quick Links



[Online Demo](#)

News

2020-02-06 - pgAdmin 4 v4.18 Released

The pgAdmin Development Team are pleased to announce pgAdmin 4 version 4.18. This release of pgAdmin 4 includes over 23 bug fixes and new features. For more details please see the release notes [here](#).



Quick Links



[Online](#)
Open link in new tab
Open link in new window
Open link in incognito window



[Download](#)
Save link as...
Copy link address
Block element...
Inspect Ctrl+Shift+I



[FAQ](#)



[Latest Docs](#)



[Ask a Question](#)

News

2020-02-06 - pgAdmin 4 v4.18 Released

The pgAdmin Development Team are pleased to announce pgAdmin 4 version 4.18. This release of pgAdmin 4 includes over 23 bug and new features. For more details please see the release notes [here](#).

Major changes in this release include:

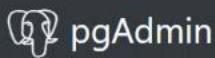
Added a Schema Diff tool to compare two schemas and generate a diff script.

- Added support for a multi-level partitioned table.
- Added labels and titles after parsing and validating all the tables in pgAdmin 4 web pages for accessibility.
- Set input controls as read-only instead of disabled will allow tab navigation in the properties tab and also allow screen readers to read it.
- Fix an issue where select, insert and update scripts on tables throwing an error.
- Logout the pgAdmin session when no user activity of mouse move, click or keypress.
- Fixed an issue where Grant wizard unable to handle multiple objects when the query string parameter exceeds its limit.
- Ensure that path file name should not disappear when changing ext from the dropdown in file explorer dialog.
- Fix an issue where setting STORAGE_DIR to empty should show all the volumes on Windows in server mode.

Download your copy [here](#).

2020-01-09 - pgAdmin 4 v4.17 Released

The pgAdmin Development Team are pleased to announce pgAdmin 4 version 4.17. This release of pgAdmin 4 includes over 20 bug fixes and new features. For more details please see the release notes [here](#).



pgAdmin Sandbox

◀ Step 2 of 2 ▶

Connect to PostgreSQL and play!

Expand the **Servers** icon on the treeview to the left of the pgAdmin user interface. As there is only one server registered in this instance of pgAdmin, it will automatically try to connect by default.



Browser

> Servers

You will be prompted for a password to connect to the server; enter **SuperSecret** and then click on the **OK** button.

Take some time to familiarise yourself with the pgAdmin user interface. You can find additional help on the **Help** menu of the application. If you have any questions, they can be asked on the pgAdmin Support [mailing list](#).

We hope you enjoy playing with PostgreSQL and pgAdmin! If you wish to install your own copy for further experimentation or to start working with PostgreSQL or EDB Postgres Advanced Server, please visit: the [pgAdmin download page](#).

pgAdmin 4



Welcome to pgAdmin 4 v4.15

Login with username: `user@domain.com` and password: `SuperSecret`



pgAdmin 4

Login

Email Address

Password

Login

[Forgotten your password?](#)

English ▾

Quick Links



Online Demo



Download



FAQ



Latest Docs



Ask a Question



Report a Bug

pgAdmin 4 (Windows)

Download

Maintainer: Dave Page

pgAdmin is available for Windows™ 7 SP1 (desktop) or 2008R2 (server) and above. The packages below include both the Desktop Runtime and Web Application:

-  [pgAdmin 4 v4.18 \(released Feb. 6, 2020\)](#)
-  [pgAdmin 4 v4.17 \(released Jan. 9, 2020\)](#)
-  [pgAdmin 4 v4.16 \(released Dec. 12, 2019\)](#)
-  [pgAdmin 4 v4.15 \(released Nov. 14, 2019\)](#)
-  [pgAdmin 4 v4.14 \(released Oct. 17, 2019\)](#)
-  [pgAdmin 4 v4.13 \(released Sept. 19, 2019\)](#)
-  [pgAdmin 4 v4.12 \(released Aug. 22, 2019\)](#)
-  [pgAdmin 4 v4.11 \(released July 25, 2019\)](#)
-  [pgAdmin 4 v4.10 \(released July 4, 2019\)](#)
-  [pgAdmin 4 v3.6 \(released Nov. 29, 2018\)](#)
-  [pgAdmin 4 v2.1 \(released Jan. 11, 2018\)](#)
-  [pgAdmin 4 v1.6 \(released July 13, 2017\)](#)



Quick Links

- Downloads
 - Binary
 - Source
- Software Catalogue
- File Browser

File Browser

Top → pgadmin → pgadmin4 → v4.18 → windows

Directories



[Parent Directory]

Files



CURRENT_MAINTAINER

2020-02-06 12:53:30

138 bytes



pgadmin4-4.18-x86.exe

2020-02-06 12:53:51

83.4 MB



pgadmin4-4.18-x86.exe.asc

2020-02-06 12:53:51

849 bytes

Current Maintainer

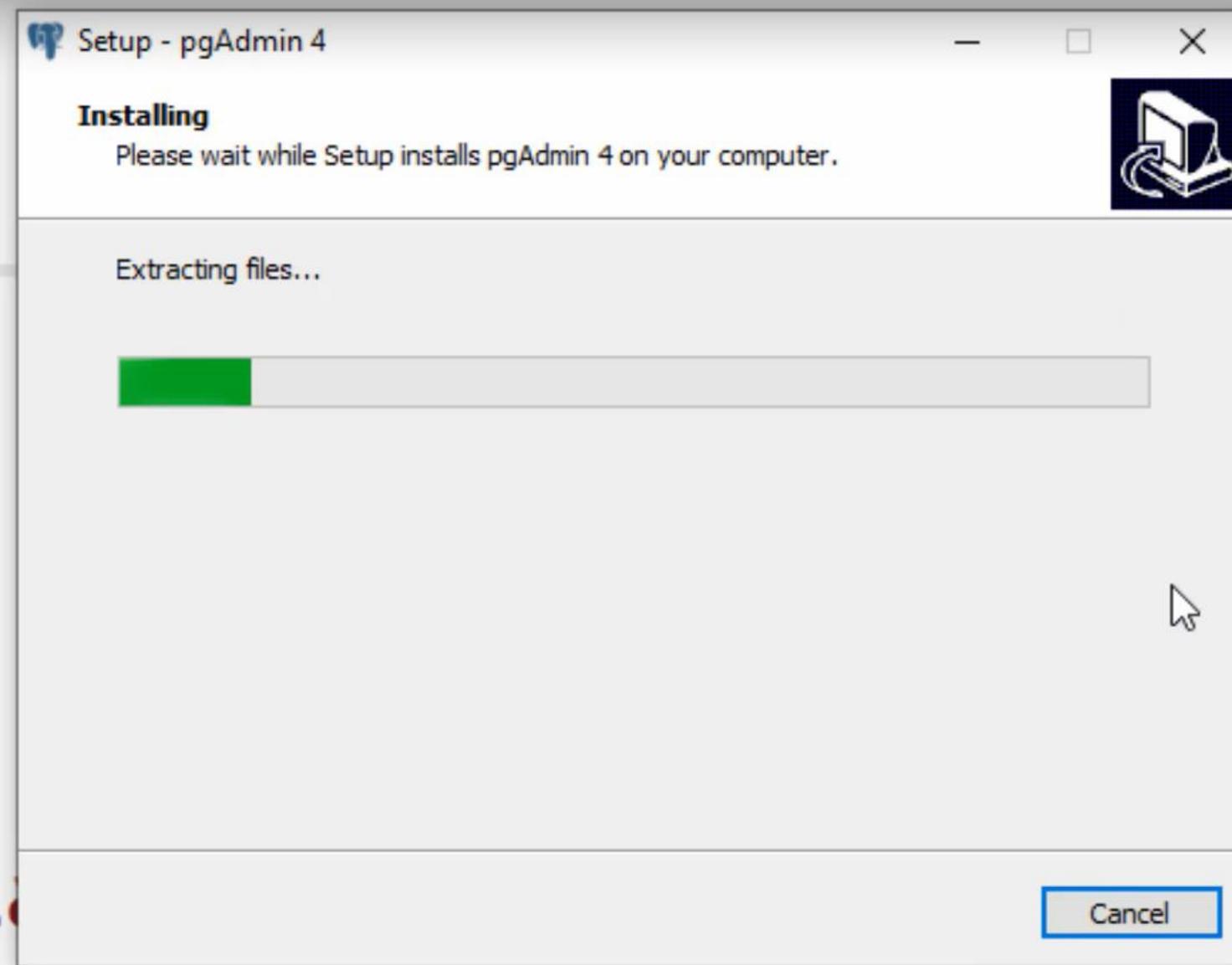
Support: pgadmin-support@lists.postgresql.org

Website: <https://www.pgadmin.org/>

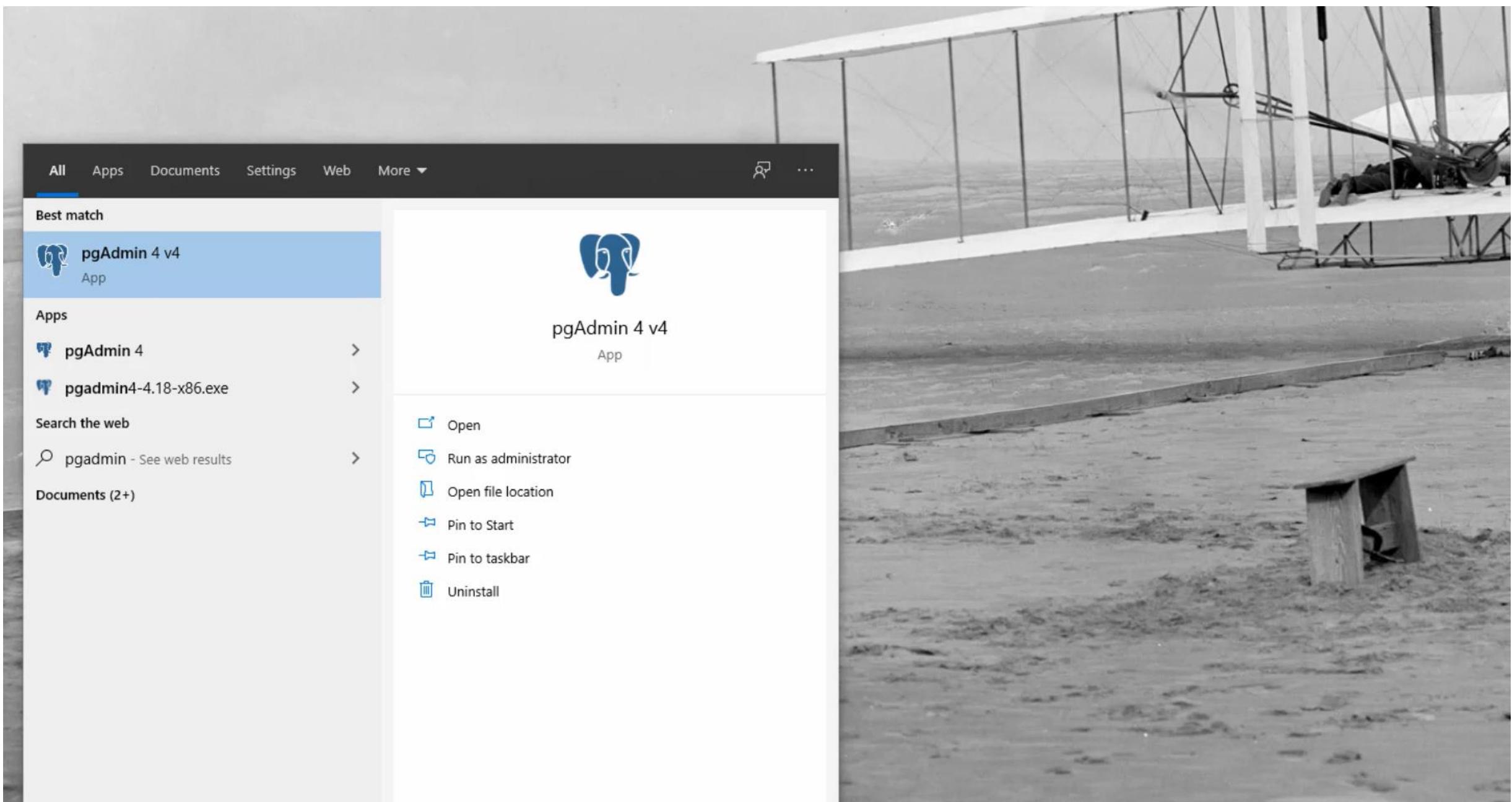
Tracker: <https://redmine.postgresql.org/projects/pgadmin4>



2-06 12:53:30
2-06 12:53:51
2-06 12:53:51



2-06 12:53:30
2-06 12:53:51
2-06 12:53:51





Set Master Password

Please set a master password for pgAdmin.

This will be used to secure and later unlock saved passwords and other credentials.

Password



Cancel

OK



Set Master Password

Please set a master password for pgAdmin.

This will be used to secure and later unlock saved passwords and other credentials.

Password



Cancel

OK

This is the password for pgAdmin, NOT PostgreSQL. But is totally OK for them to both be the same thing! We recommend you just use "password" again.



Servers (2)

> PostgreSQL 14

> mydb

Connect to Server

the selected object.

Please enter the password for the user 'postgres' to connect the server - "PostgreSQL 14"

Password

 Save Password

Cancel

OK



Servers (2)

PostgreSQL 14

Databases

Login/G

Tablespa

mydb

Create

Refresh

ⓘ No dependant information is available for the selected object.



Servers (2)

PostgreSQL 14

Databases

Login/Groups

Tablespaces

mydb

Create

Database...

Refresh



No dependant information is available for the selected object.



Servers (2)

PostgreSQL 14

Databases

Login/Group Roles

Tablespaces

mydb

Create - Database

General Definition Security Parameters Advanced SQL

Database



|

Owner

postgres

Comment

! 'Database' cannot be empty.



Close

Reset

Save





Servers (2)

PostgreSQL 14

Databases (10)

advanced-SQL

dvdrental

dvdrental2

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas

Subscriptions

excercises

learning

myDB

myDB2

pharma_sale

postgres

recruit

Login/Group Roles

Tablespaces

mydb

No dependant information is available for the selected object.

Create >

Delete/Drop

Refresh...

Restore...

Backup...

CREATE Script

Disconnect from database

Generate ERD

Maintenance...

Grant Wizard...

Search Objects...

PSQL Tool

Query Tool

Properties...



- ▼ Servers (2)
 - ▼ PostgreSQL 14
 - > Databases (10)
 - > advanced-SQL
 - > dvdrental
 - > dvdrental2
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas
 - > Subscriptions
 - > excercises
 - > learning
 - > myDB
 - > myDB2
 - > pharma_sale
 - > postgres
 - > recruit
 - > Login/Group Roles
 - > Tablespaces
- > mydb

ⓘ No dependant information is available for the selected object.

Restore (Database: dvdrental2)

General Data/Objects Options

Format ▼

Filename ! →

Number of jobs

Role name ▼

Please provide a filename. X

ⓘ ?

× Close ↻ Reset ⬆ Restore



- ▼ Servers (2)
 - ▼ PostgreSQL 14
 - > advanced-SQL
 - > dvdrental
 - ▼ dvdrental2
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas
 - > Subscriptions
 - > excercises
 - > learning
 - > myDB
 - > myDB2
 - > pharma_sale
 - > postgres
 - > recruit
- > Login/Group Roles
- > Tablespaces
- > mydb

Select file



C:\11-SQL\dvdrental.tar



Name

Size

Modified

advanced_sql_course-general_hospital_setup.sql	9.0 MB	Thu Oct 20 05:24:07 2022
dvdrental excercises	136.0 B	Fri Sep 9 20:43:32 2022
dvdrental.tar	2.7 MB	Sat Sep 3 14:50:30 2022
Excercises	173.0 B	Fri Sep 9 20:43:02 2022
exercises.tar	32.3 kB	Thu Sep 8 19:56:05 2022
Introduction of SQL.pptx	6.5 MB	Wed Jun 14 13:53:16 2023
postgresql-15-A4.pdf	13.5 MB	Fri Sep 9 06:00:33 2022
SQL The Complete Reference, 3rd Edition (PDFDrive).pdf	15.6 MB	Tue May 10 20:21:57 2022
trial.sql	406.0 B	Sat Jan 28 19:53:30 2023

Show hidden files and folders?

Format

All Files

Cancel

Select



- ▼ Servers (2)
 - ▼ PostgreSQL 14
 - > advanced-SQL
 - > dvdrental
 - ▼ dvdrental2
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas
 - > Subscriptions
 - > excercises
 - > learning
 - > myDB
 - > myDB2
 - > pharma_sale
 - > postgres
 - > recruit
- > Login/Group Roles
- > Tablespaces
- > mydb

Select file

C:\11-SQL\dvdrental.tar

Name	Size	Modified
advanced_sql_course-general_hospital_setup.sql	9.0 MB	Thu Oct 20 05:24:07 2022
dvdrental excercises	136.0 B	Fri Sep 9 20:43:32 2022
dvdrental.tar	2.7 MB	Sat Sep 3 14:50:30 2022
Excercises	173.0 B	Fri Sep 9 20:43:02 2022
exercises.tar	32.3 kB	Thu Sep 8 19:56:05 2022
Introduction of SQL.pptx	6.5 MB	Wed Jun 14 13:53:16 2023
postgresql-15-A4.pdf	13.5 MB	Fri Sep 9 06:00:33 2022
SQL The Complete Reference, 3rd Edition (PDFDrive).pdf	15.6 MB	Tue May 10 20:21:57 2022
trial.sql	406.0 B	Sat Jan 28 19:53:30 2023

Show hidden files and folders?

Format 



Servers (2)

PostgreSQL 14

Databases (10)

advanced-SQL
dvdrental
dvdrental2Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
excercises
learning
myDB
myDB2
pharma_sale
postgres
recruit
Login/Group Roles

No dependant information is available for the selected object.

Servers (2)

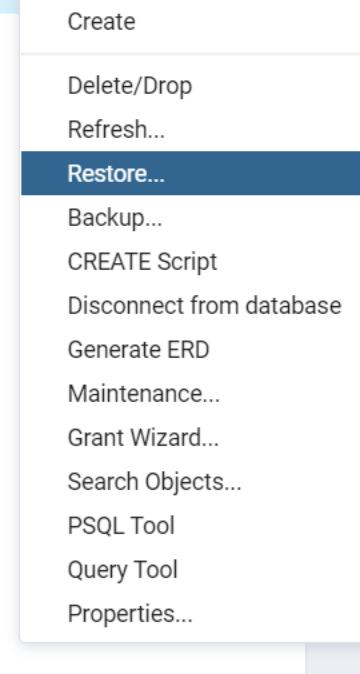
PostgreSQL 14

Databases (10)

- > advanced-SQL
- > dvdrental

dvdrental2

- > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas
 - > Subscriptions
 - > excercises
 - > learning
 - > myDB
 - > myDB2
 - > pharma_sale
 - > postgres
 - > recruit
- > Login/Group Roles



No dependant information is available for the selected object.



Servers (2)

PostgreSQL 14

Databases (10)

- > advanced-SQL
- > dvdrental

dvdrental2

- > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas
 - > Subscriptions
 - > excercises
 - > learning
 - > myDB
 - > myDB2
 - > pharma_sale
 - > postgres
 - > recruit
- > Login/Group Roles

ⓘ No dependant information is available for the selected object.

Restore (Database: dvdrental2)

General Data/Objects Options

Queries

Include CREATE DATABASE statement

Clean before restore

Single transaction

⚠ Please provide a filename. X



Close

Reset

Restore



- ▼ Servers (2)
 - PostgreSQL 14
 - Databases (10)
 - advanced-SQL
 - dvdrental
 - dvdrental2
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas
 - Subscriptions
 - excercises
 - learning
 - myDB
 - myDB2
 - pharma_sale
 - postgres
 - recruit
 - Login/Group Roles
 - Tablespaces
 - mydb

ⓘ No dependant information is available for the selected object.

Restore (Database: dvdrental2)

General Data/Objects Options

Sections

Pre-data

Data

Post-data

Type of objects

Please provide a filename.

ⓘ ? X

× Close ↻ Reset ⬆ Restore

Restoring backup on the server

Restoring backup on the server 'PostgreSQL 14 (localhost:5432)'
Wed Jun 14 2023 15:16:31 GMT+0530 (India Standard Time)

⌚ 1.58 seconds ⓘ More details... X Stop Process

✓ Successfully completed.



- ▼ Servers (2)
 - PostgreSQL 14
 - Databases (10)
 - advanced-SQL
 - dvdrental
 - dvdrental2
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas
 - Subscriptions
 - excercises
 - learning
 - myDB
 - myDB2
 - pharma_sale
 - postgres
 - recruit
 - Login/Group Roles

ⓘ No dependant information is available for the selected object.

Restore (Database: dvdrental2)

General Data/Objects Options

No data for Failed Tables

Miscellaneous / Behavior

Verbose messages

Use SET SESSION AUTHORIZATION

! Please provide a filename. X



X Close

↻ Reset

⬆ Restore



- ▼ Servers (2)
 - ▼ PostgreSQL 14
 - ▼ Databases (10)
 - > advanced-SQL
 - > dvdrental
 - ▼ dvdrental2
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas
 - > Subscriptions
 - > excercises
 - > learning
 - > myDB
 - > myDB2
 - > pharma_sale
 - > postgres
 - > recruit
 - > Login/Group Roles
 - > Tablespaces
- > mydb

ⓘ No dependant information is available for the selected object.

Restore (Database: dvdrental2)

General Data/Objects Options

Disable

Trigger

No data for Failed Tables

Miscellaneous / Behavior

Verbose messages

Use SET SESSION

Buttons



Servers (2)

PostgreSQL 14

Databases (10)

advanced-SQL

dvdrental

dvdrental2

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas

Subscriptions

excercises

learning

myDB

myDB2

pharma_sale

postgres

recruit

Login/Group Roles

Tablespaces

mydb

Create >

Delete/Drop

Refresh...

Restore...

Backup...

CREATE Script

Disconnect from database

Generate ERD

Maintenance...

Grant Wizard...

Search Objects...

PSQL Tool

Query Tool

Properties...

No dependant information is available for the selected object.

Servers (2)

PostgreSQL 14

Databases (10)

advanced-SQL

dvdrental

dvdrental2

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas
- > Subscriptions
- > excercises
- > learning
- > myDB
- > myDB2
- > pharma_sale
- > postgres
- > recruit
- > Login/Group Roles
- > Tablespaces
- > mydb

Create >

- Delete/Drop
- Refresh...
- Restore...
- Backup...
- CREATE Script
- Disconnect from database
- Generate ERD
- Maintenance...
- Grant Wizard...
- Search Objects...
- PSQL Tool
- Query Tool**
- Properties...

No dependant information is available for the selected object.



- ▼ Servers (2)
 - ▶ PostgreSQL 14
 - ▶ Databases (10)
 - > advanced-SQL
 - > dvrental
 - > dvrental2
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas
 - > Subscriptions
 - > excercises
 - > learning
 - > myDB
 - > myDB2
 - > pharma_sale
 - > postgres
 - > recruit
 - > Login/Group Roles
 - > Tablespaces
 - > mydb

dvrental2/postgres@PostgreSQL 14 *

Query Editor Query History

1 SELECT * FROM film

Data Output Explain Messages Notifications

	film_id [PK] integer	title character varying (255)	description text
1	133	Chamber Italian	A Fateful Reflection of a Moose And a Husband who must Overcome a Monkey in Nigeria
2	384	Grosse Wonderful	A Epic Drama of a Cat And a Explorer who must Redeem a Moose in Australia
3	8	Airport Pollock	A Epic Tale of a Moose And a Girl who must Confront a Monkey in Ancient India
4	98	Bright Encounters	A Fateful Yarn of a Lumberjack And a Feminist who must Conquer a Student in A Jet Boat
5	1	Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies
6	2	Ace Goldfinger	A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China
7	3	Adaptation Holes	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Baloon Factory
8	4	Affair Prejudice	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank
9	5	African Egg	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologist in The Gulf of Mexico
10	6	Agent Truman	A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo Wrestler in Ancient China

pgAdmin 4 x + 127.0.0.1:55376/browser/

pgAdmin File Object Tools Help

Browser

Servers (4)
PostgreSQL 9.5
PostgreSQL 10
PostgreSQL 11
PostgreSQL 12
Databases (2)
dvrental
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas

dvrental/postgres@PostgreSQL 12 *

Dashboard Properties SQL Statistics Dependencies Dependents dvrental/postgres@PostgreSQL 12 *

File Properties SQL Statistics Dependencies Dependents dvrental/postgres@PostgreSQL 12 *

dvrental/postgres@PostgreSQL 12

Query Editor Query History

1 SELECT * FROM film;

Make sure you have selected Data Output. If its empty or says "film not found or doesn't exist", try restarting your computer or trying a lower version of PostgreSQL

Data Output Explain Messages Notifications

	title character varying (255)	description text	release_year integer	language_id smallint	rental_duration smallint	rental_rate numeric (4,2)	length smallint	replace numer
133	Chamber Italian	A Fateful Reflec...	2006	1	7	4.99	117	
384	Grosse Wonderful	A Epic Drama of...	2006	1	5	4.99	49	
8	Airport Pollock	A Epic Tale of a ...	2006	1	6	4.99	54	
98	Bright Encounters	A Fateful Yarn o...	2006	1	4	4.99	73	
1	Academy Dinosaur	A Epic Drama of...	2006	1	6	0.99	86	
2	Ace Goldfinger	A Astounding E...	2006	1	3	4.99	48	
3	Adaptation Holes	A Astounding R...	2006					

Successfully run. Total query runtime: 248 msec. 1000 rows affected



Servers (2)

PostgreSQL 14

Databases (10)

advanced-SQL

dvdrental

dvdrental2

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas

Subscriptions

excercises

learning

myDB

myDB2

pharma_sale

postgres

recruit

Login/Group Roles

Tablespaces

mydb

Welcome



pgAdmin

Management Tools for PostgreSQL

Feature rich | Maximises PostgreSQL | Open Source

pgAdmin is an Open Source administration and management tool for the PostgreSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.

Quick Links



Add New Server



Configure pgAdmin

Getting Started



PostgreSQL Documentation



pgAdmin Website



Planet PostgreSQL



Community Support



▼ Servers (2)

PostgreSQL 14

Databases (10)

> advanced-SQL

> dvdrental

dvdrental2

> Casts

> Catalogs

> Event Triggers

> Extensions

> Foreign Data Wrappers

> Languages

> Publications

> Schemas

> Subscriptions

> excercises

> learning

> myDB

> myDB2

> pharma_sale

> postgres

> recruit

> Login/Group Roles

> Tablespaces

> mydb

Welcome



pgAdmin

Management Tools for PostgreSQL

Feature rich | Maximises PostgreSQL | Open Source

pgAdmin is an Open Source administration and management tool for the PostgreSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.

Quick Links



Add New Server



Configure pgAdmin

Getting Started



PostgreSQL Documentation



pgAdmin Website



Planet PostgreSQL



Community Support

Preferences



Dashboard

Properties

SQL

Statistics

Dependencies

Dependents

dvdrental2/pos...



Servers (2)

- PostgreSQL
- Database
 - adv
 - dvdrental
 - dvdrental2
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas
 - Subscriptions
 - excercises
 - learning
 - myDB
 - myDB2
 - pharma_sale
 - postgres
 - recruit
 - Login/Group Roles
 - Tablespaces
 - mydb

Welcome



pgAdmin

Management Tools for PostgreSQL

Feature rich | Maximises PostgreSQL | Open Source

pgAdmin is an Open Source administration and management tool for the PostgreSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.

Quick Links



Add New Server



Configure pgAdmin

Getting Started



PostgreSQL Documentation



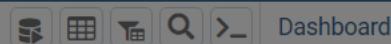
pgAdmin Website



Planet PostgreSQL



Community Support



Servers (2)

PostgreSQL 14

Databases (10)

> advanced-SQL

> dvdrental

> dvdrental2

> Casts

> Catalogs

> Event Triggers

> Extensions

> Foreign Data Wrappers

> Languages

> Publications

> Schemas

> Subscriptions

> excercises

> learning

> myDB

> myDB2

> pharma_sale

> postgres

> recruit

Login/Group Roles

Tablespaces

> mydb

Welcome

Preferences

Browser

Display

Auto-expand sole children



If a treeview node is expanded and has only a single child, automatically expand the child node as well.

30

Browser tree state saving interval in seconds. Use -1 to disable the tree saving mechanism.



Confirm before closing or resetting the changes in the properties dialog for an object if the changes are not saved.



Confirm closure or refresh of the browser or browser tab is intended before proceeding.



Enable browser tree animation?



Enable dialogue/notification animation?

Cancel

Save



administration interface, an SQL query tool, a
administrators alike.

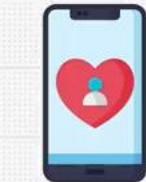


Configure pgAdmin



Community Support

WHAT WE WILL LEARN



1. How to put data in DB
2. How to use/update/learn from data
3. How to remove data

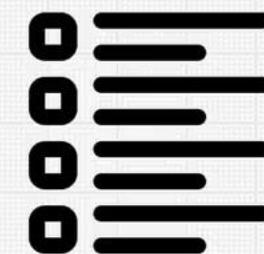
CONFUSING ACRONYMS

DataBase Management System

**Relational DataBase Management
System**

Structured Query Language

CONFUSING ACRONYMS



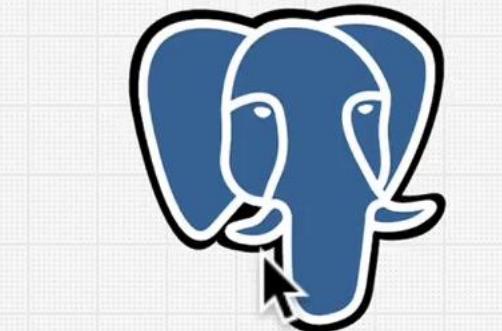
1. How to put data in DB
2. How to use/update/learn from data
3. How to remove data

CONFUSING ANAGRAMS

SQL

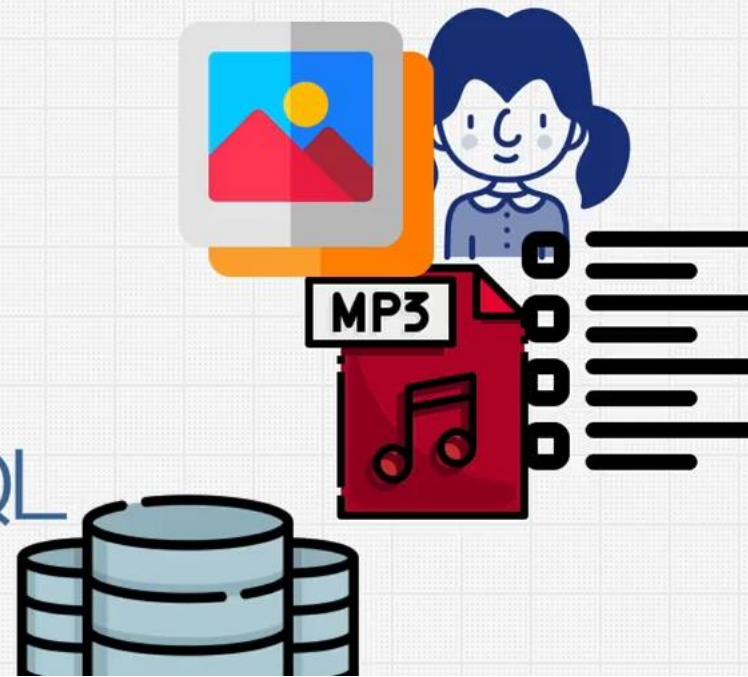


DBMS
RDBMS

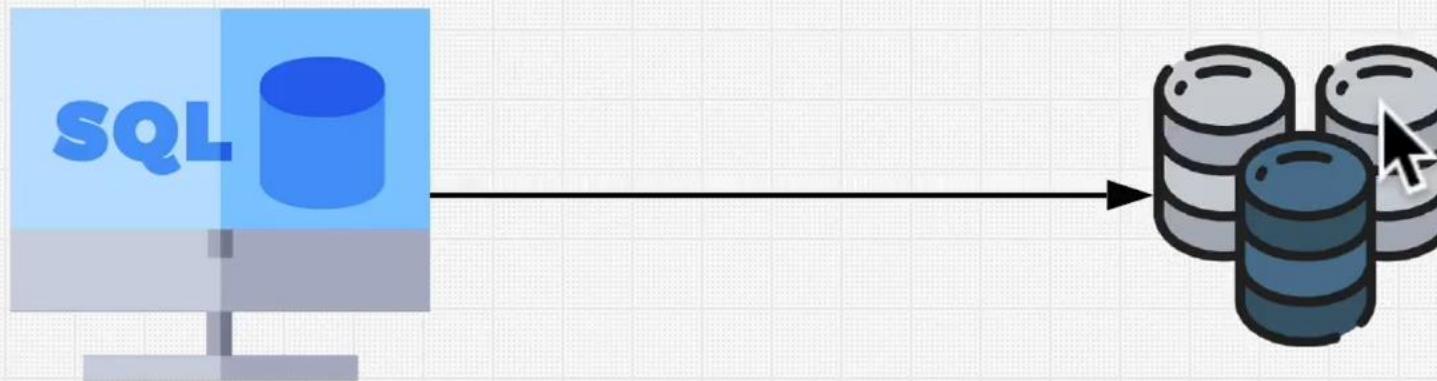


PostgreSQL

Data



WHAT IS SQL

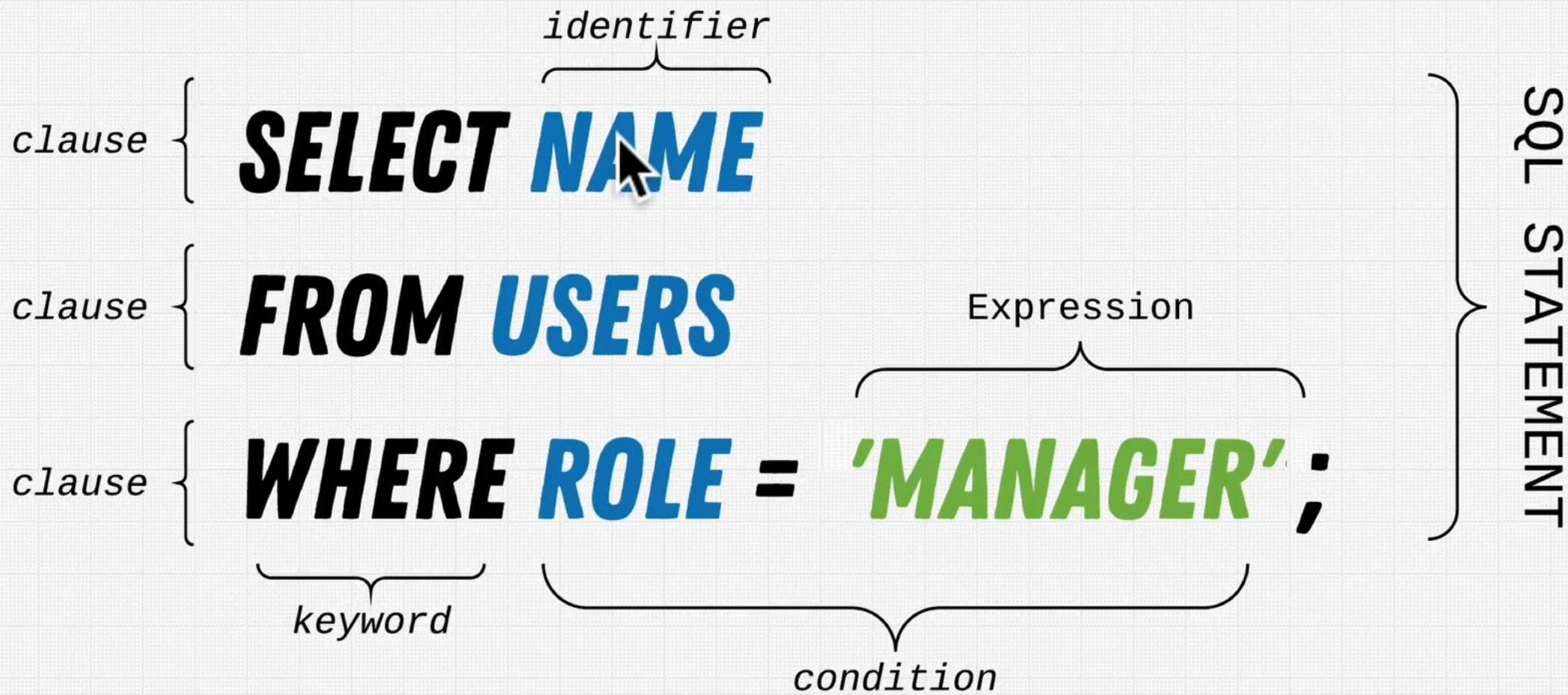


QUERY

SELECT * FROM USERS

SQL STATEMENT

QUERY BREAKDOWN



DECLARA-WHAT?

DECLARATIVE

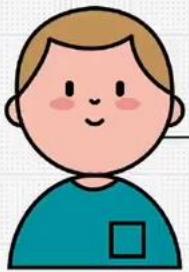
IMPERATIVE 

DECLARA-WHAT?

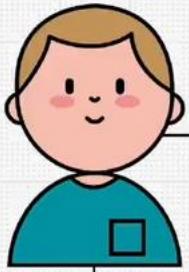
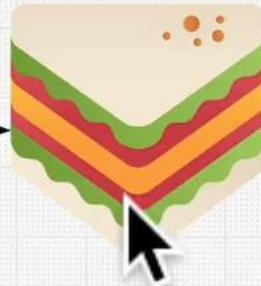
WHAT WILL HAPPEN

HOW IT WILL HAPPEN

DECLARA-WHAT?



make a sandwich
with ham



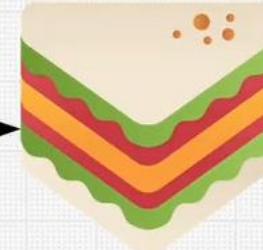
slice bread



slice ham



make
sandwich



DECLARA-WHAT?



DECLARATIVE

IMPERATIVE

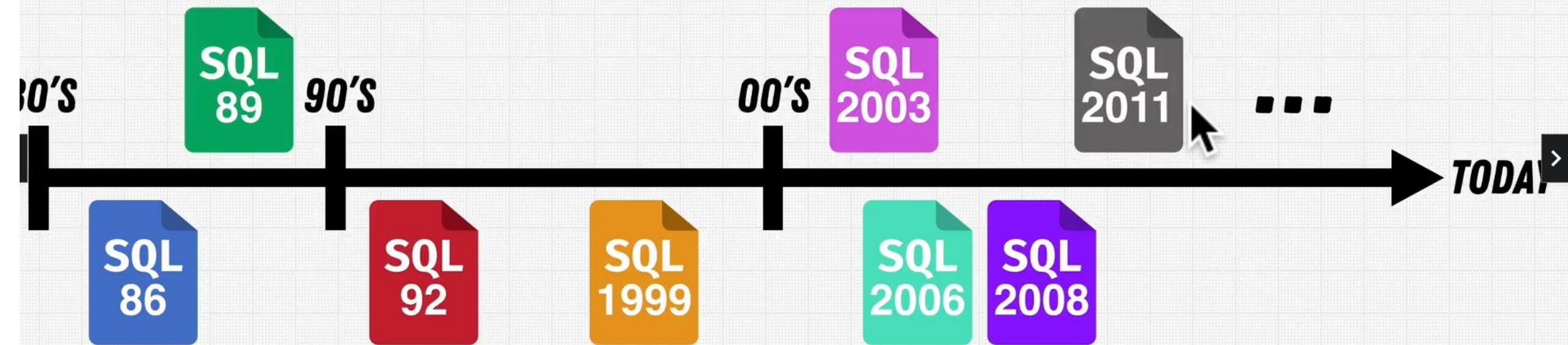


HISTORY OF SQL

SQL?

SEQUEL?

SQL STANDARDS



DATABASE MODELS?

**HIERARCHICAL
NETWORKING**

ENTITY-RELATIONSHIP

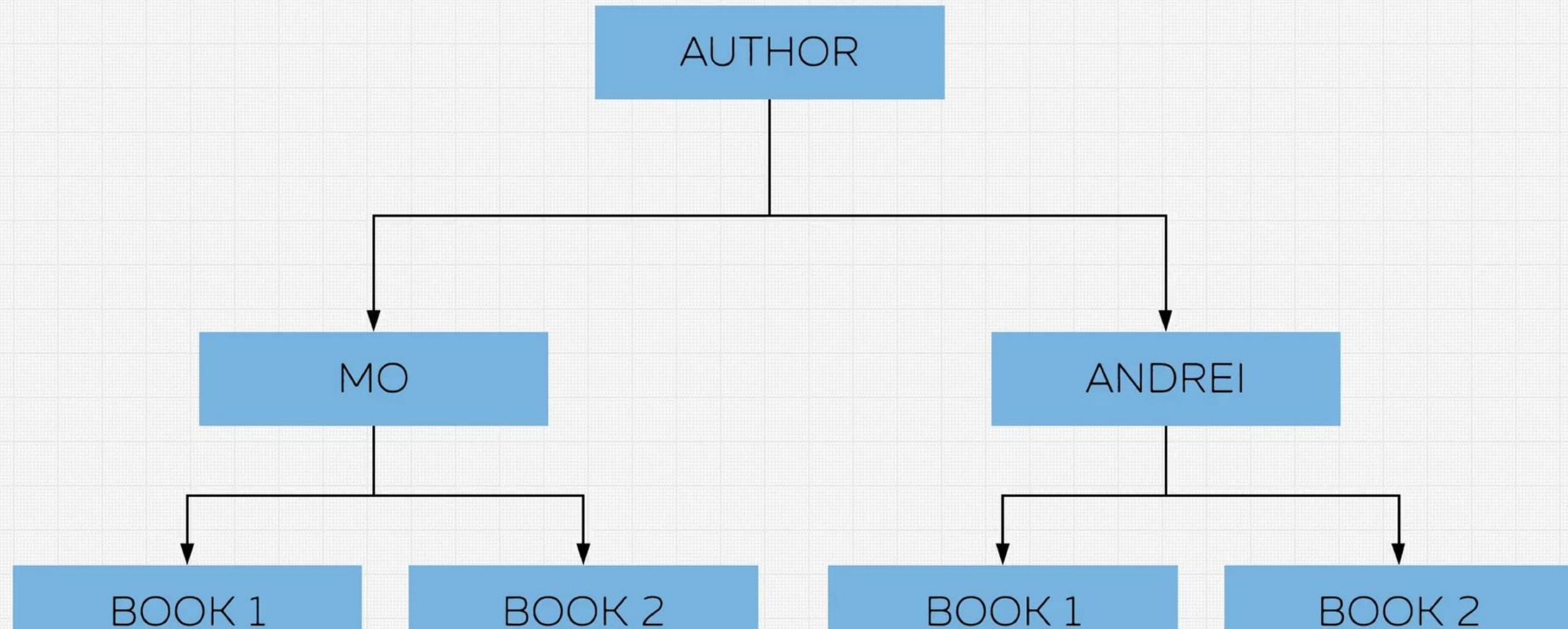
RELATIONAL

OBJECT ORIENTED

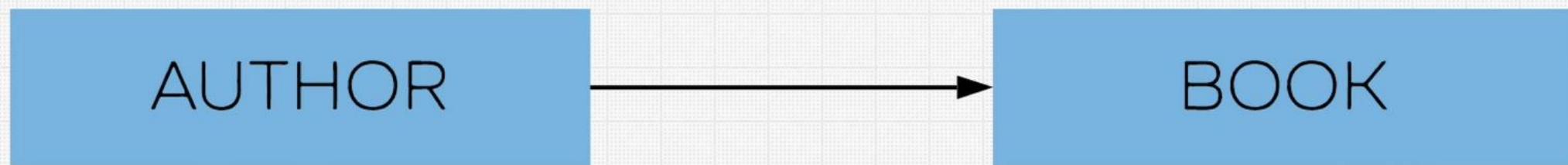
**FLAT
SEMI-STRUCTURED**

...

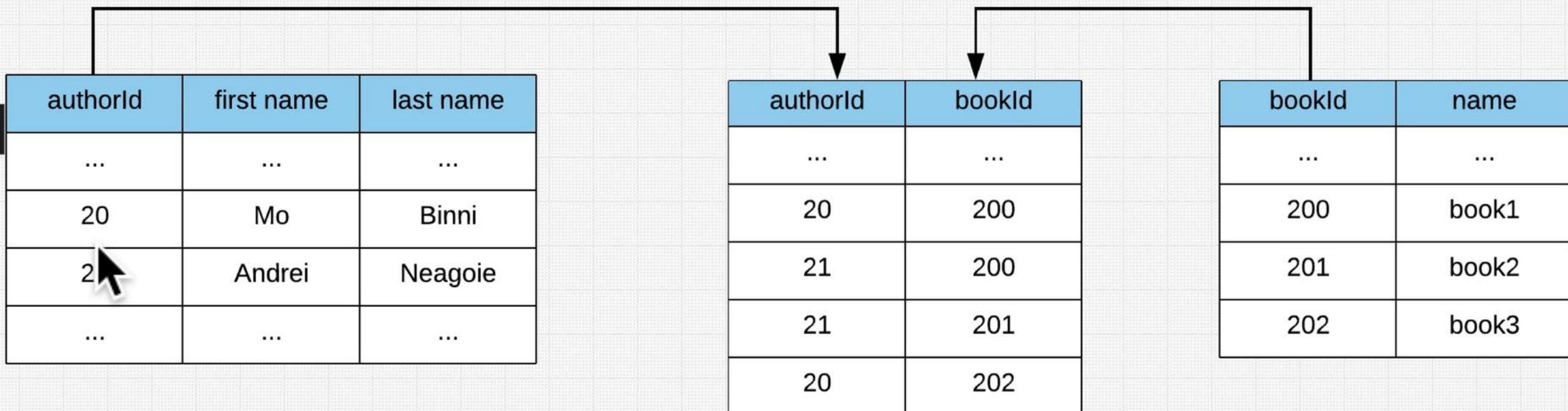
HIERARCHICAL MODEL



RELATIONAL MODEL



RELATIONAL MODEL



PRIMARY KEY

PRIMARY KEY



id	firstName	lastName	sex	dob	email
1	Mo	Binni	m	1992-11-13	mo.binni@yahoo.com
2	Charles	Barker	m	1982-01-01	charles.barker@yahoo.com
3	Jimmy	Neutron	m	1968-04-28	jimmy.neutron@yahoo.com
4	Charlize	Theron	f	1975-08-07	charlize.theron@yahoo.com
5	Will	Smith	m	1968-09-25	will.smith@yahoo.com

FOREIGN KEY

PRIMARY KEY

id	firstName	lastName	sex	dob	email	managerId
1	Mo	Binni	m	1992-11-13	mo.binni@yahoo.com	m1
2	Charles	Barker	m	1982-01-01	charles.barker@yahoo.com	m2
3	Jimmy	Neutron	m	1968-04-28	jimmy.neutron@yahoo.com	m3
4	Charlize	Theron	f	1975-08-07	charlize.theron@yahoo.com	m2
5	Will	Smith	m	1968-09-25	will.smith@yahoo.com	m1

FOREIGN KEY

managerId	firstName	lastName	sex
m1	Clyde	Bowie	m
m2	Connor	Little	m
...

RELATIONAL DATABASES

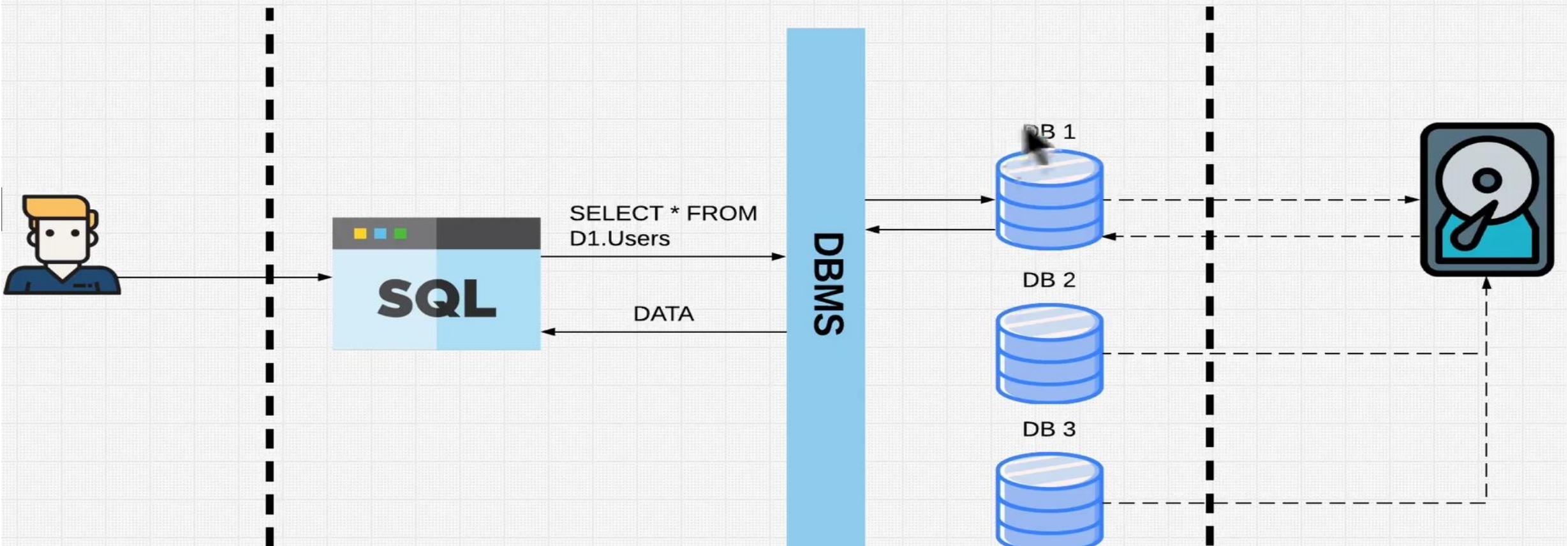


SUPPORT DAY TO DAY

SUPPORT ANALYSIS

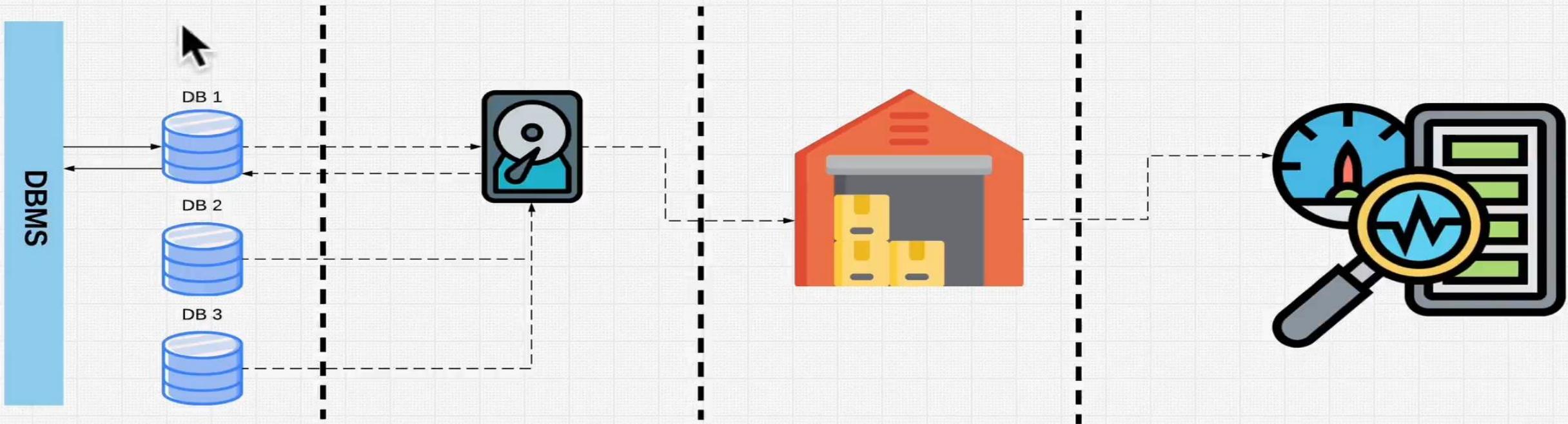
Online transaction processes

OLTP



Online analytical processing

OLAP

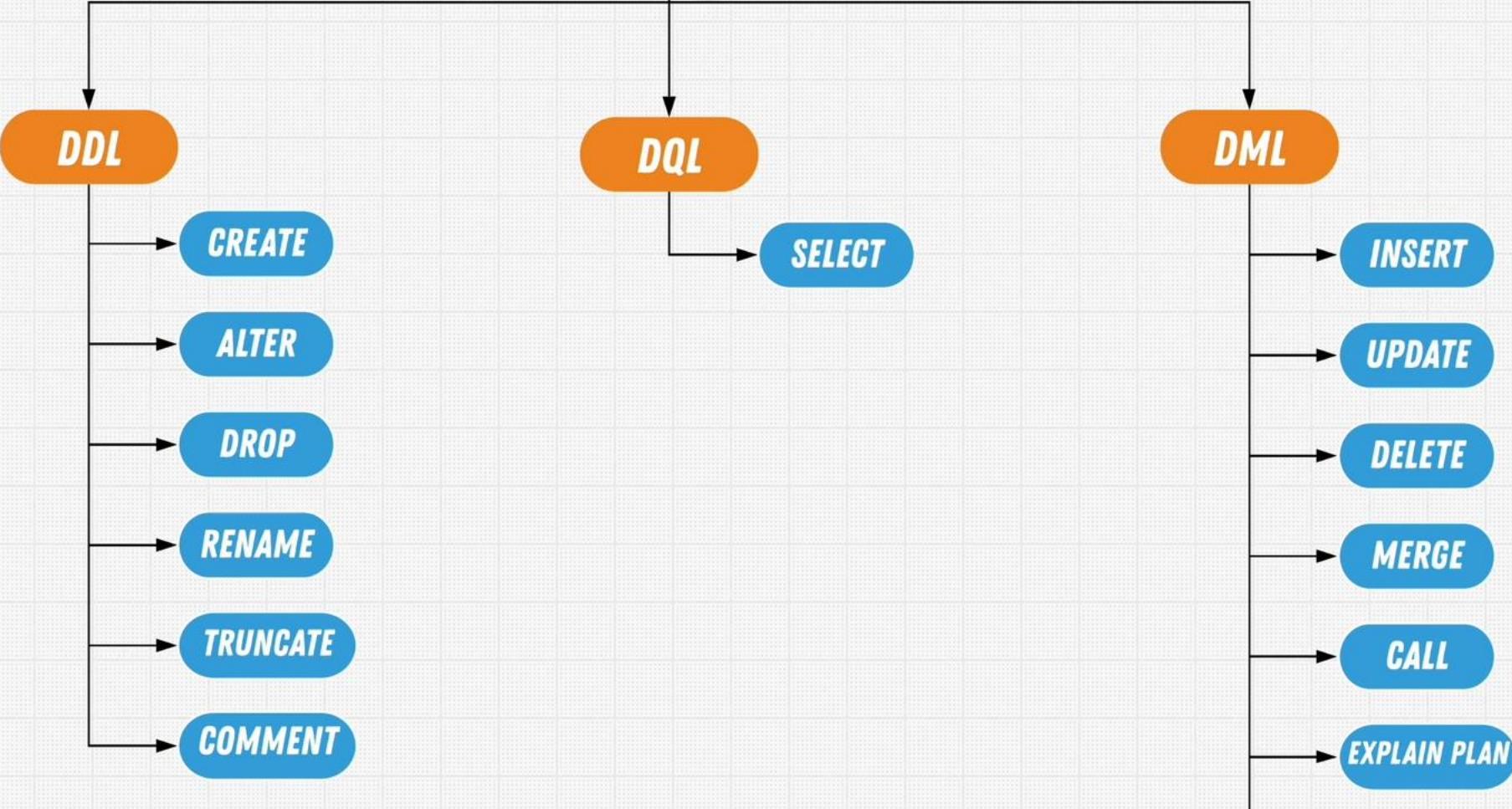


Starting With Query

SQL COMMANDS

DCL

GRANT
REVOKE



- DCL:Data control language
- DDL:Data Definition language
- DQL:Data Query language
- DML:Data Modification language

EXERCISE

QUESTION

GIVE ME A LIST OF ALL EMPLOYEES IN THE COMPANY

Give me a list of all films available on DVD

The screenshot shows a database management application window with the following layout:

- Left Sidebar:** Contains sections for "Bonjour", "Bookmarks" (localhost:5432), and "Connections" (localhost:5432, SQLite Files, Valentina Files).
- Top Bar:** Features four dropdown menus: "Databases", "Schemas", "Tables", and "Fields".
- Databases View:** Shows a list of databases: education, Employees (selected), France, Store, and World.
- Schemas View:** Shows the "public" schema with tables: departments, dept_emp, dept_manager, employees, salaries (selected), and titles.
- Fields View:** Displays fields for the selected table: emp_no, from_date, salary, and to_date.

A large black arrow points to the "salaries" table in the Schemas view.

The screenshot shows a database management interface with a sidebar and a main query editor.

Left Sidebar (Schema Browser):

- Notification channels(0)
- Schemas(1)
 - public
 - Domains(0)
 - Functions(0)
 - Links(6)
 - Sequences(0)
 - Tables(6)
 - departments
 - dept_emp
 - dept_manager
 - employees** (highlighted in blue)
 - salaries
 - titles
 - Types(1)
 - Views(0)
- Triggers(0)

Main Area (Query Editor):

1 | `SELECT * FROM "public"."employees"`

The query editor displays the SQL command `SELECT * FROM "public"."employees"`. The word `employees` is highlighted in red, indicating it is a keyword or has been previously used in the session.



- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- ▼ Publications
- ▼ Schemas (1)
 - ▼ public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - ▼ Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types
 - > Views

1 SELECT * FROM employees

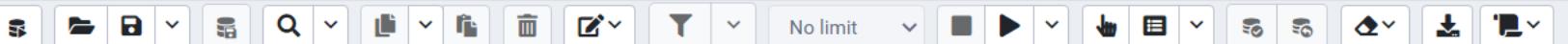
	emp_no [PK] integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender gender	hire_date date	
1	10001	1953-09-02	Georgi	Facello	M	1986-06-26	
2	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	
3	10003	1959-12-03	Porto	Bamford	M	1986-08-28	
4	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	
5	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12	
6	10006	1953-04-20	Anneke	Preusig	F	1989-06-02	
7	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10	
8	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15	
9	10009	1952-04-19	Sumant	Peac	F	1985-02-18	
10	10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24	



- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
- > Tables (15)

- > actor
- > address
- > category
- > city
- > country
- > customer
- > film
- > film_actor
- > film_category
- > inventory
- > language



```
1 SELECT * FROM film
```

	film_id [PK] integer	title character varying (255)	description text
1	133	Chamber Italian	A Fateful Reflection of a Moose And a Husband who must Overcome a Monkey in Nigeria
2	384	Grosse Wonderful	A Epic Drama of a Cat And a Explorer who must Redeem a Moose in Australia
3	8	Airport Pollock	A Epic Tale of a Moose And a Girl who must Confront a Monkey in Ancient India
4	98	Bright Encounters	A Fateful Yarn of a Lumberjack And a Feminist who must Conquer a Student in A Jet Boat
5	1	Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies
6	2	Ace Goldfinger	A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China
7	3	Adaptation Holes	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Baloon Factory
8	4	Affair Prejudice	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank
9	5	African Egg	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologist in The Gulf of Mexico
10	6	Agent Truman	A Intriguing Panorama of a Robot And a Boy who must Escape a Sumo Wrestler in Ancient China

EXERCISE

QUESTION

HOW MANY DEPARTMENTS ARE THERE IN THE COMPANY?

How many actors are there in the film?



- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- ▼ Publications
- ▼ Schemas (1)
 - ✓ public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - ▼ Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types
 - > Views



1 SELECT * FROM departments

	dept_no [PK] character (4)	dept_name character varying (40)
1	d001	Marketing
2	d002	Finance
3	d003	Human Resources
4	d004	Production
5	d005	Development
6	d006	Quality Management
7	d007	Sales
8	d008	Research
9	d009	Customer Service



- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (15)
 - > actor
 - > address
 - > category
 - > city
 - > country
 - > customer
 - > film
 - > film_actor
 - > film_category
 - > inventory
 - > language

Query Editor Query History

1 `SELECT * FROM actor`

Data Output Explain Messages Notifications

	actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	1	Penelope	Guiness	2013-05-26 14:47:57.62
2	2	Nick	Wahlberg	2013-05-26 14:47:57.62
3	3	Ed	Chase	2013-05-26 14:47:57.62
4	4	Jennifer	Davis	2013-05-26 14:47:57.62
5	5	Johnny	Lollobrigida	2013-05-26 14:47:57.62
6	6	Bette	Nicholson	2013-05-26 14:47:57.62
7	7	Grace	Mostel	2013-05-26 14:47:57.62
8	8	Matthew	Johansson	2013-05-26 14:47:57.62
9	9	Joe	Swank	2013-05-26 14:47:57.62
10	10	Christian	Gable	2013-05-26 14:47:57.62

EXERCISE

QUESTION

WHAT TITLE DOES 10006 HAVE?

Ahmednagar has which city id?



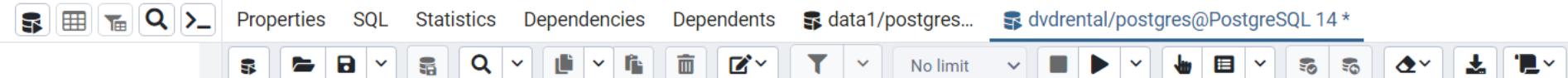
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- ▼ Publications
- ▼ Schemas (1)
 - ▼ public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - ▼ Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types
 - > Views

1 `SELECT * FROM titles`

No limit



	emp_no [PK] integer	title [PK] character varying (50)	from_date [PK] date	to_date date
1	10001	Senior Engineer	1986-06-26	9999-01-01
2	10002	Staff	1996-08-03	9999-01-01
3	10003	Senior Engineer	1995-12-03	9999-01-01
4	10004	Engineer	1986-12-01	1995-12-01
5	10004	Senior Engineer	1995-12-01	9999-01-01
6	10005	Senior Staff	1996-09-12	9999-01-01
7	10005	Staff	1989-09-12	1996-09-12
8	10006	Senior Engineer	1990-08-05	9999-01-01
9	10007	Senior Staff	1996-02-11	9999-01-01
10	10007	Staff	1989-02-10	1996-02-11



- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (15)
 - > actor
 - > address
 - > category
 - > city
 - > country
 - > customer
 - > film
 - > film_actor
 - > film_category
 - > inventory
 - > language

Properties SQL Statistics Dependencies Dependents data1/postgres... dvdrental/postgres@PostgreSQL 14 *

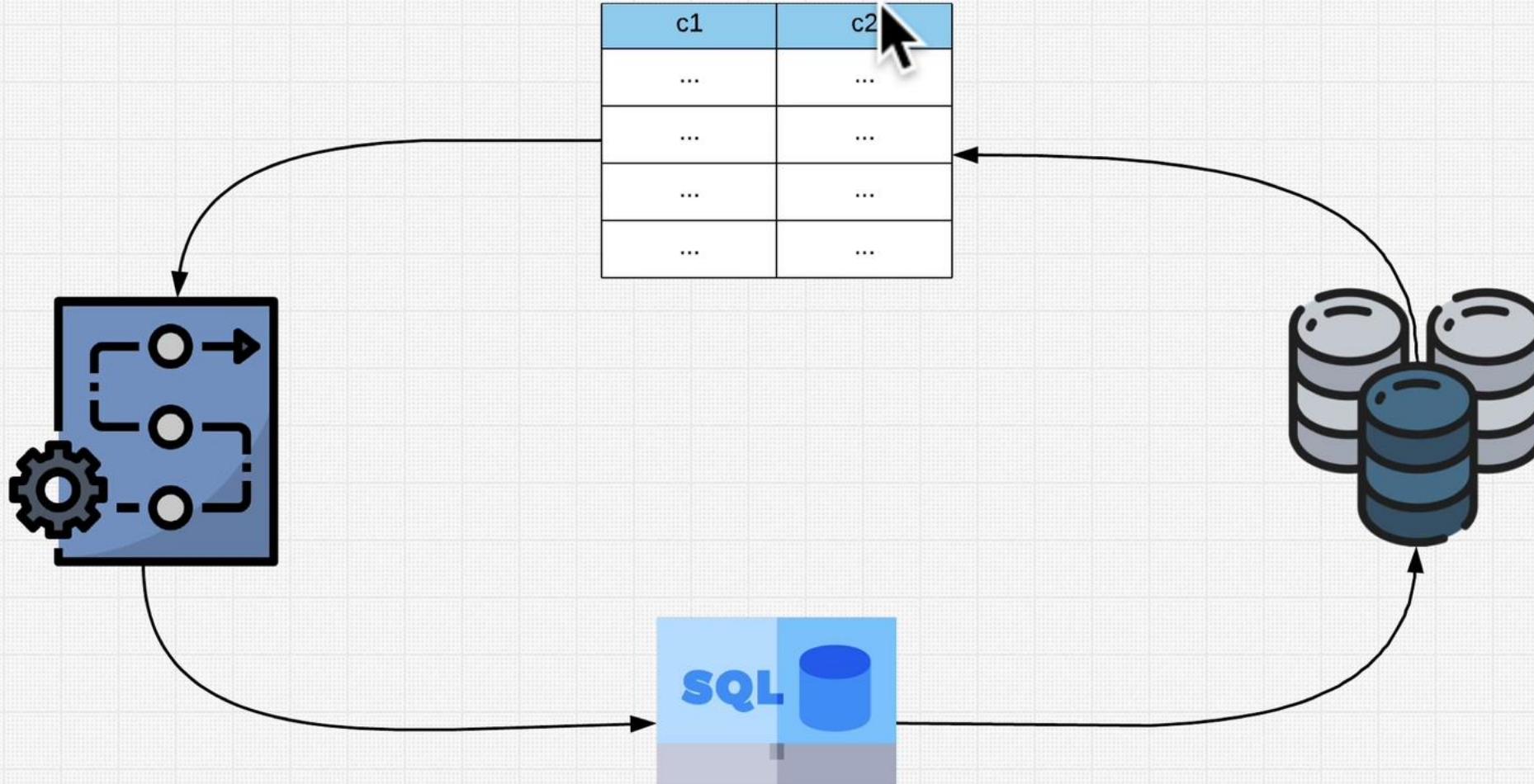
Query Editor Query History

1 SELECT * FROM city

Data Output Explain Messages Notifications

	city_id [PK] integer	city character varying (50)	country_id smallint	last_update timestamp without time zone
1	1	A Corua (La Corua)	87	2006-02-15 09:45:25
2	2	Abha	82	2006-02-15 09:45:25
3	3	Abu Dhabi	101	2006-02-15 09:45:25
4	4	Acua	60	2006-02-15 09:45:25
5	5	Adana	97	2006-02-15 09:45:25
6	6	Addis Abeba	31	2006-02-15 09:45:25
7	7	Aden	107	2006-02-15 09:45:25
8	8	Adoni	44	2006-02-15 09:45:25
9	9	Ahmadnagar	44	2006-02-15 09:45:25
10	10	Akishima	50	2006-02-15 09:45:25

WHAT IS A FUNCTION?



WHAT IS A FUNCTION?

**A FUNCTION IS A SET OF STEPS
THAT CREATES A SINGLE VALUE**

TYPES OF FUNCTIONS

AGGREGATE

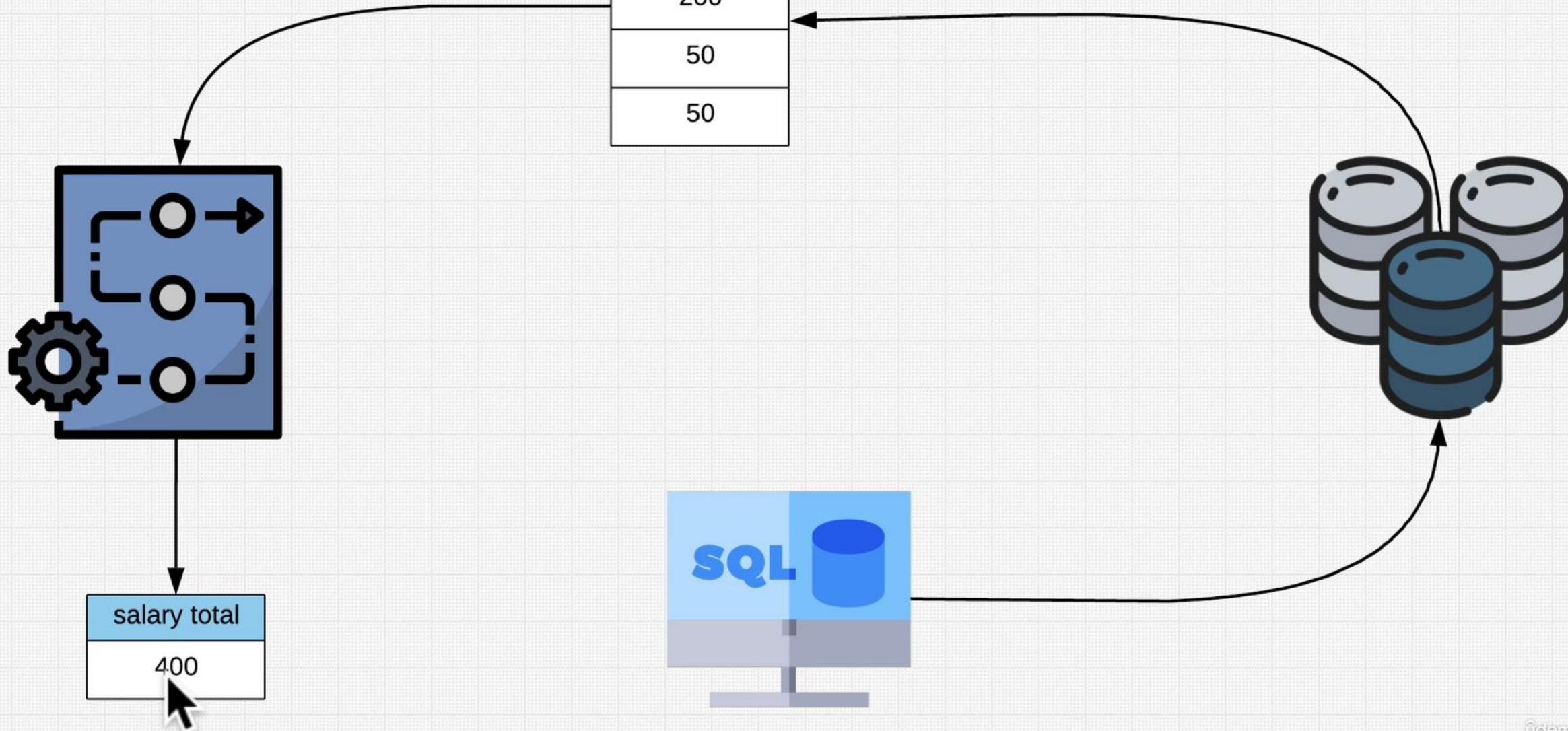
SCALAR

AGGREGATE

***OPERATE ON MANY RECORDS
TO PRODUCE 1 VALUE***



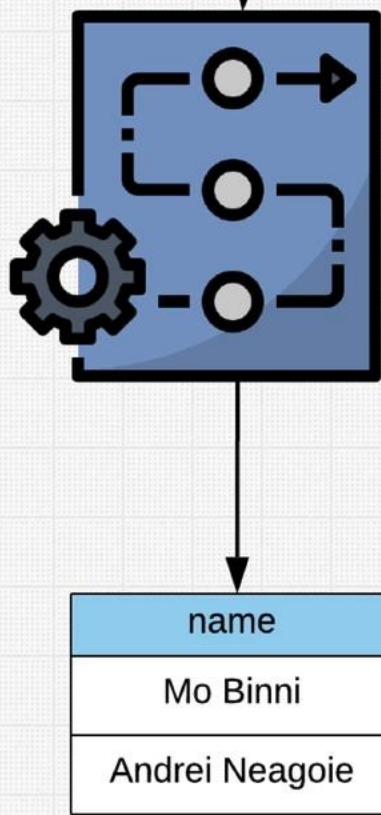
AGGREGATE



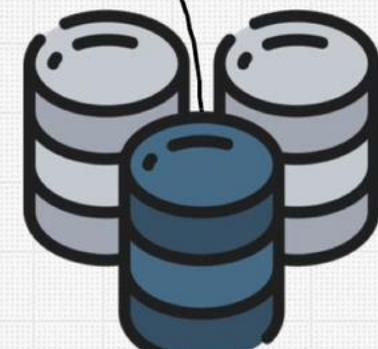
Scalar functions

- 1.[LCASE\(\)](#)
- 2.[UCASE\(\)](#)
- 3.[LEN\(\)](#)
- 4.[MID\(\)](#)
- 5.[ROUND\(\)](#)
- 6.[NOW\(\)](#)
- 7.[FORMAT\(\)](#)

SCALAR



firstName	lastName
Mo	Binni
Andrei	Neagoie

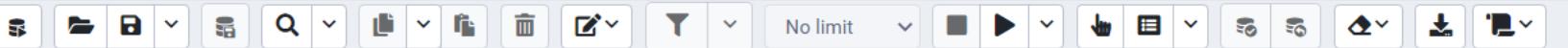


AGGREGATE FUNCTIONS

- AVG()** -- calculates the average of a set of values.
- COUNT()** -- counts rows in a specified table or view.
- MIN()** -- gets the minimum value in a set of values.
- MAX()** -- gets the maximum value in a set of values.
- SUM()** -- calculates the sum of values.



- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (15)
 - > actor
 - > address
 - > category
 - > city
 - > country
 - > customer
 - > film
 - > film_actor
 - > film_category
 - > inventory
 - > language



```
1 SELECT COUNT(category_id) FROM film_category
```

count	bigint
-------	--------

1	1000
---	------



Employees



2 schemas

Execute

Execute Current



Notification channels(0)

Schemas(1)

public

Domains(0)

Functions(0)

Links(6)

Sequences(0)

Tables(6)

departments

dept_emp

dept_manager

employees

Checks(0)

Fields(6)

birth_date

emp_no

first_name

gender

1

SELECT max(emp_no) FROM employees;

max

1

499999

QUESTION

GET THE HIGHEST SALARY AVAILABLE

QUESTION

Get the max payment got from rental

The screenshot shows a database management interface with the following components:

- Top Bar:** Includes icons for edit, refresh, connection status (Employees, 2 schemas), and execution buttons (Execute, Execute Current).
- Toolbar:** Contains icons for database, table, view, and calendar.
- Left Sidebar (Schema Browser):**
 - Notification channels (0)
 - Schemas (1)
 - public
 - Domains (0)
 - Functions (0)
 - Links (6)
 - Sequences (0)
 - Tables (6)
 - departments
 - dept_emp
 - dept_manager
 - employees
 - salaries
 - Checks (0)
 - Fields (4)
 - emp_no
 - from_date
- Query Editor:** Shows a single query window with the following content:

```
1 | SELECT max(salary) FROM salaries;
```
- Result Panel:** Displays the result of the executed query:

	max
1	158220

Browser



Properties SQL Statistics Dependencies Dependents

data1/postgres...

dvd

< > xgr

Servers (3)

PostgreSQL 14

Databases (17)

advanced-SQL

data1

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

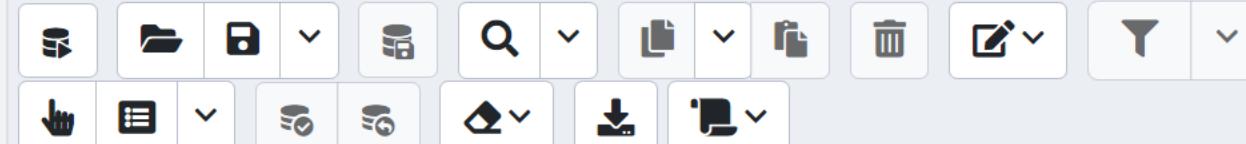
Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries



Query Editor

Query History

1 SELECT max(amount) FROM payment

Data Output

Explain

Messages

Notifications

	max	
	numeric	🔒
1	11.99	



Employees



2 schemas

Execute

Execute Current



Notification channels(0)

Schemas(1)

public

Domains(0)

Functions(0)

Links(6)

Sequences(0)

Tables(6)

departments

dept_emp

dept_manager

employees

salaries

Checks(0)

Fields(4)

emp_no

from_date

salary

1

SELECT sum(salary) FROM salaries;

sum

181480757419

ADDING COMMENTS



```
-- Comment all the things
-- I like having single line comments
/*
 * Multi line comments are also
 * pretty sweet
*/
SELECT firstName, lastName FROM users;
```



- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types
 - > Views

```
1 SELECT first_name, last_name FROM employees
2 WHERE first_name='Mayumi' and last_name='Schueller'
```

	first_name	last_name
1	Mayumi	Schueller

Browser



- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (15)
 - > actor
 - > address
 - > category
 - > city
 - > country
 - > customer
 - > film
 - > film_actor
 - > film_category
 - > inventory
 - > language
 - > payment
 - > rental

Properties

SQL

Statistics

Dependencies

Dependents

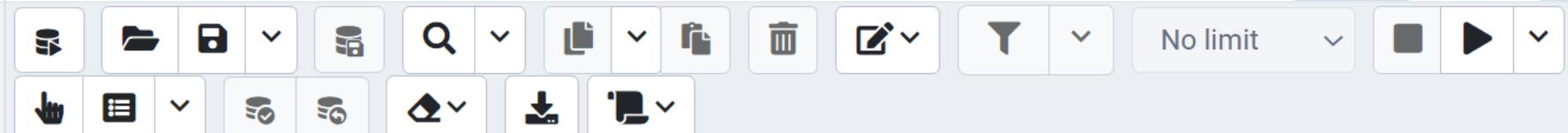
data1/postgres...

dvd

<

>

xgr



Query Editor

Query History

```
1   SELECT first_name, last_name| FROM actor
```

Data Output

Explain

Messages

Notifications

	first_name character varying (45)	last_name character varying (45)
1	Penelope	Guiness
2	Nick	Wahlberg
3	Ed	Chase
4	Jennifer	Davis
5	Johnny	Lollobrigida

```
1 SELECT first_name, last_name FROM "public"."employees"
2 -- filter on first name AND last name to limit the amount of data returned
3 -- and focus the filtering on a single person
4 WHERE first_name = 'Mayumi' AND last_name = 'Schueller';
```



first_name	last_name
Mayumi	Schueller

The screenshot shows a PostgreSQL terminal window with the following content:

```
1 -- select statement to filter Mayumi Schueler
2 SELECT first_name, last_name FROM "public"."employees"
3 /*
4   filter on first name AND last name to limit the amount of data returned
5   and focus the filtering on a single person
6 */
7 WHERE first_name = 'Mayumi' AND last_name = 'Schueler';
```

A mouse cursor is visible over the code at line 3.

first_name	last_name
Mayumi	Schueler

COMMON SELECT MISTAKES



COMMON SELECT MISTAKES

```
SLEECT fullName as "full name"  
FORM usres;
```



COMMON SELECT MISTAKES

USING ; INSTEAD OF ,

OR VICE VERSA



COMMON SELECT MISTAKES

```
SELECT firstName, lastName FROM users;
```

COMMON SELECT MISTAKES

USING ↵" INSTEAD OF '

COMMON SELECT MISTAKES

“ IS FOR TABLES

‘ IS FOR TEXT

Case sensitive

COMMON SELECT MISTAKES

```
SELECT fullName as "full name" FROM "Users";
```



COMMON SELECT MISTAKES

```
SELECT fullName as "full name" FROM users;
SELECT fullName as "full name" FROM "Users";
SELECT fullName as "full name" FROM "USERS";
SELECT fullName as "full name" FROM "My Users";
```

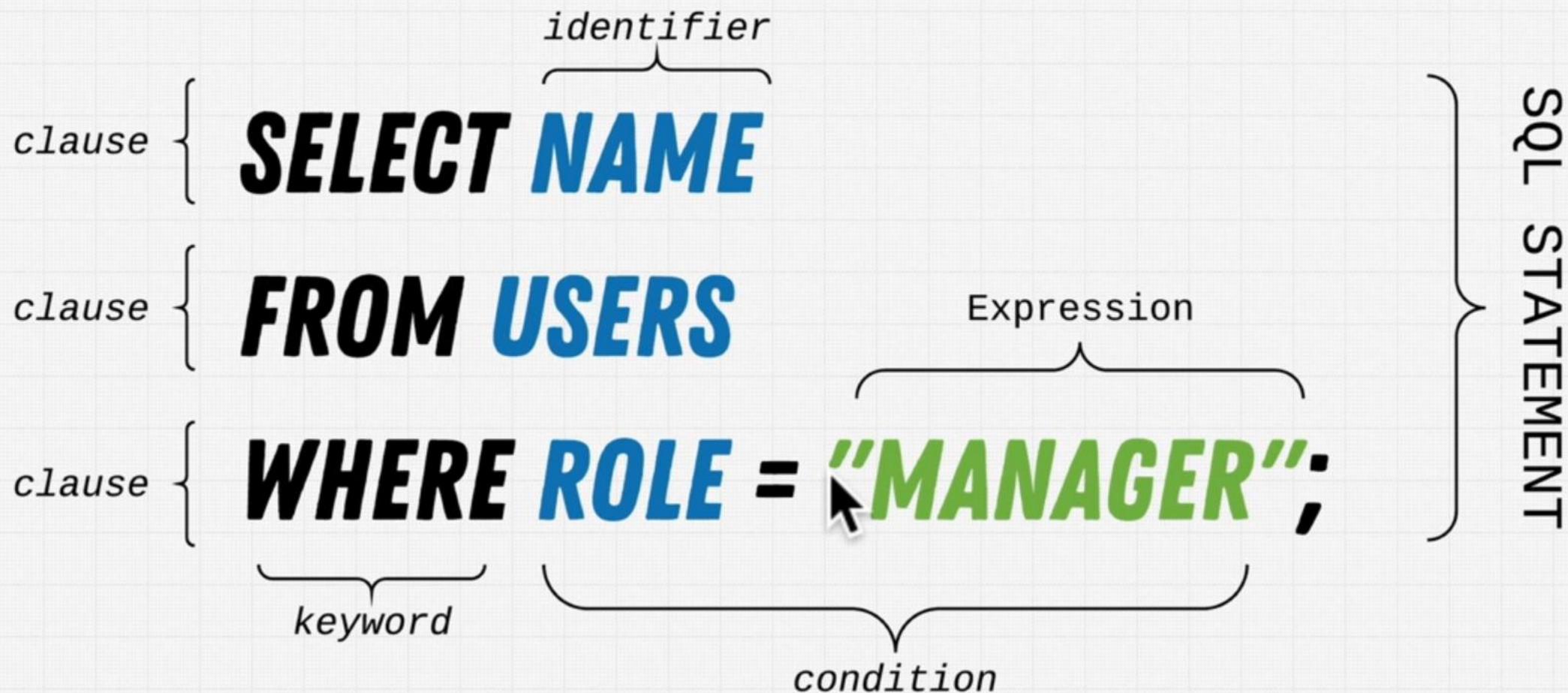
COMMON SELECT MISTAKES

INVALID COLUMN NAME

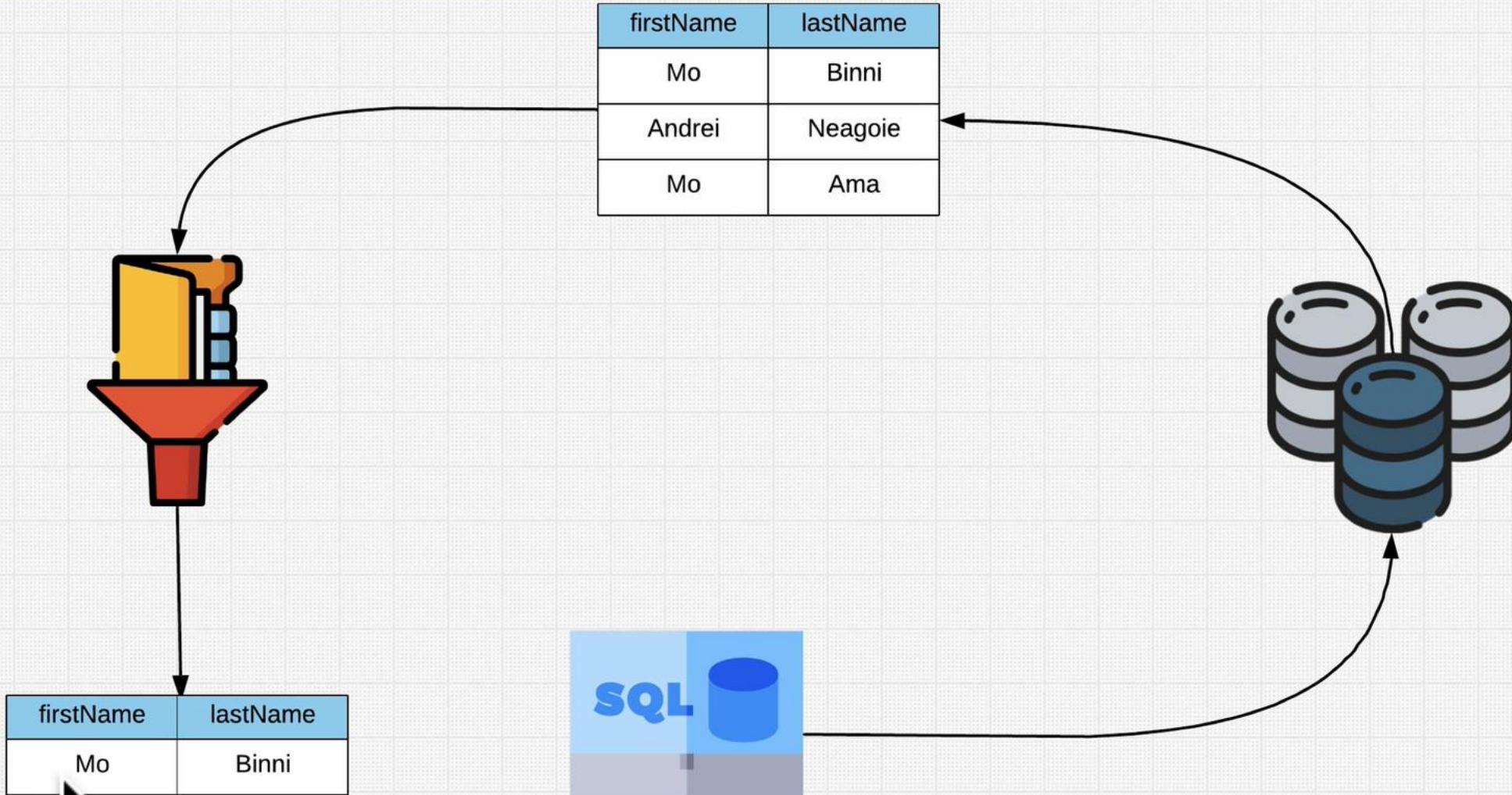
HOW TO FILTER DATA



SIMPLE SELECT STATEMENT



WHERE IS WHAT?



Employees | 2 schemas | Execute | Execute Current | fx Functions... | Client/NoLock | A |

Notification channels(0) | Schemas(1) | Triggers(0)

```
1 | SELECT first_name FROM employees
2 | WHERE gender = 'F';
```

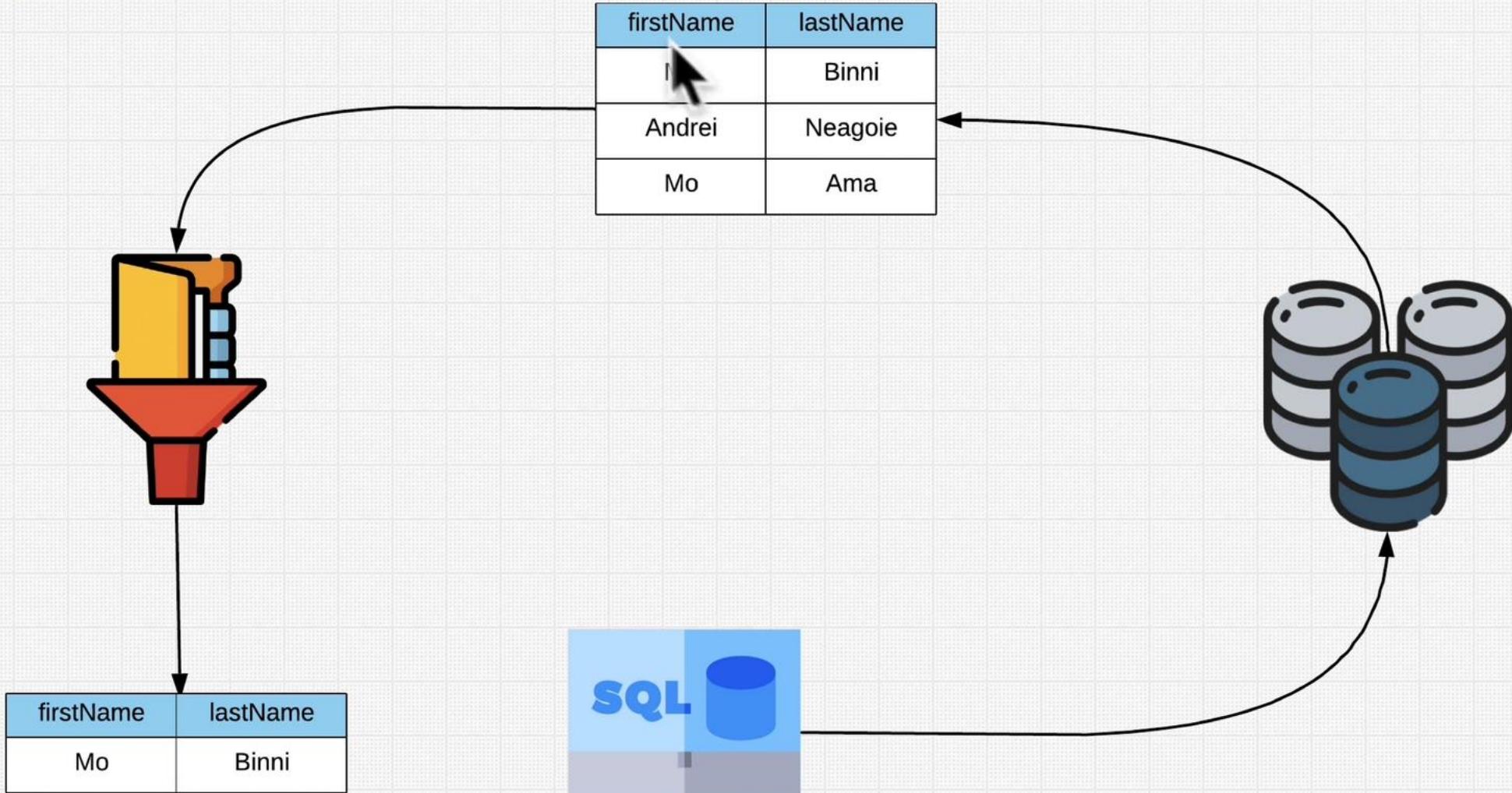
	first_name
1	Bezalel
2	Anneke
3	Tzvetan
4	Suman
5	Duangkaew
6	Mary
7	Cristinel
8	Kazuhide
9	Bojan
10	Suzette
11	Divier
12	Irena

Number of records: 120051 Number of fields: 1 Query time: 174 millisecond(s) Read-Only

THE “AND” KEYWORD

```
SELECT firstName, lastName  
FROM users  
WHERE firstName = 'Mo' AND lastName = 'Binni';
```

THE “AND” KEYWORD



WHAT IF

YOU WANT TO FILTER

ON 2 FIRST NAMES?



- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types
 - > Views

```
1 SELECT first_name, last_name FROM employees  
2 WHERE first_name='Georgi' and last_name='Facello'
```

	first_name	last_name
	character varying (14)	character varying (16)
1	Georgi	Facello
2	Georgi	Facello

EXERCISE

QUESTION

***HOW MANY FEMALE CUSTOMERS DO WE HAVE
FROM THE STATE OF OREGON (OR) AND NEW
YORK (NY)***

Store 2 schemas

Execute Execute Current

Links(3) Sequences(4) Tables(8)

categories cust_hist customers

Checks(0) Fields(10)

address1 address2 age city country creditcard creditcardexpir... creditcardtype

1 | SELECT firstname, lastname FROM customers;

	firstname	lastname
1	VKUUXF	ITHOMQJNYX
2	HQNMZH	UNUKXHJVXB
3	JTNRNB	LYYSHTQJRE
4	XMFYXD	WQLQHUHLFE
5	PGDTDU	ETBYBNNEGUT
6	FXDZBW	BAXPEEKXVJ
7	WVZTXZ	RMEVXCQGQF
8	LIWLAI	PVGRMMHSEQ
9	NCGWRC	CJOPRHUHIE
10	FUOHXX	WMOEHWMWWM
11	XQVVMI	KRPGDBCQJH

Number of records: 20000 Number of fields: 2 Query time: 21 millisecond(s) Read-Only

Store 2 schemas

Execute Execute Current

creditcardexpirydate creditcardtype customerid email firstname gender income lastname password phone region state username zip

Indexes(1) Links(2)

```
1 SELECT firstname, lastname FROM customers
2 WHERE state = 'OR' OR state = 'NY';
```

	firstname	lastname
416	IHCCIK	RHHIXCZNZ
417	TEBJWB	DKQFJGLNYS
418	GJIMKC	XEIJSBGNVV
419	NUKAZB	KMNRSLGBQH
420	DEEXJL	LYXESTFZF
421	UOQGNI	BJBVBKZMOX
422	RFESKI	NAJDXIDGCY
423	DLUQGA	YYQAJIYOCN
424	CMMHAF	DMJCFNOTRG
425	OVNEQC	OLSRYCDQAJ
426	KXFFIM	QZLFAHLUCT

Number of records: 426 Number of fields: 2 Query time: 9 millisecond(s) Read-Only

Store 2 schemas

Execute Execute Current

creditcardexpirydate creditcardtype customerid email firstname gender income lastname password phone region state username zip

Indexes(1) Links(2)

```
1 SELECT firstname, lastname, gender FROM customers
2 WHERE state = 'OR' OR state = 'NY' AND gender = 'F';
```

	firstname	lastname	gender
1	NPSFGQ	NAYJFNKHRQ	F
2	CMYWQG	FHLLHPWNZQ	M
3	NYLUDS	GPAVMNUDEU	M
4	OANHVR	CAOUMZIUEZ	M
5	KOZJVV	YLGBTFHGFZ	M
6	IIGRBQ	OUCWDXMWAB	F
7	YFHHCI	VZVOWBWKTG	F
8	LQPAAP	PHCMXEWHV	F
9	RSCFDI	RXUAFLXMNX	F
10	XJKGGQ	ZETDBZXNXE	F
11	LIMGDN	WGTJIHRLQI	M

Number of records: 314 Number of fields: 3 Query time: 13 millisecond(s) Read-Only

Store 2 schemas

Execute Execute Current

creditcardexpirydate creditcardtype customerid email firstname gender income lastname password phone region state username zip

Indexes(1) Links(2)

```
1 SELECT firstname, lastname, gender, state FROM customers
2 WHERE state = 'OR' AND gender = 'F' OR state = 'NY' AND gender = 'F';
```

	firstname	lastname	gender	state
190	NVEZKO	TSVMHZRUAI	F	NY
191	UTTZLA	WTSGHCIUSC	F	OR
192	NCLHPY	VKGWMWFZLXW	F	NY
193	SVPSNK	ABZGKLSDEA	F	OR
194	CTESZI	CNXBQZPTXA	F	NY
195	DKMFDP	BSNZJDJPIS	F	OR
196	IHCCK	RHHIXZCZNZ	F	NY
197	TEBJWB	DKQFJGLNYS	F	NY
198	RFESKI	NAJDXIDGCY	F	NY
199	CMMHAF	DMJCFNOTRG	F	OR
200	KXFFIM	QZLFAHLUCT	F	OR

Number of records: 200 Number of fields: 4 Query time: 12 millisecond(s) Read-Only



Store 2 schemas

Execute Execute Current

creditcardexpirydate creditcardtype customerid email firstname gender income lastname password phone region state username zip

Indexes(1) Links(2)

```
1 SELECT firstname, lastname, gender, state FROM customers
2 WHERE (state = 'OR' OR state = 'NY') AND gender = 'F';
```

	firstname	lastname	gender	state
103	VMHKAG	BUBOXSJDEM	F	NY
104	NLGXBB	ICMXTANDHM	F	OR
105	ECICCF	PORRWFWRYS	F	OR
106	NYQNOK	AWYJSFNKFN	F	OR
107	PLAORW	DLRDIKXYUO	F	OR
108	OAEJKG	OQBNDHULAM	F	OR
109	RWSPZL	DYSFJPJBUK	F	OR
110	YQOMML	THHAXAXQCV	F	OR
111	WYSZOY	TAQHREJEST	F	NY
112	TFARZA	IGRMPVEIQI	F	NY
113	OVPMOP	LIZZSSPEUH	F	OR

Number of records: 200 Number of fields: 4 Query time: 13 millisecond(s) Read-Only



Store 2 schemas

Execute Execute Current

creditcardexpirydate creditcardtype customerid email firstname gender income lastname password phone region state username zip

Indexes(1) Links(2)

```
1 SELECT COUNT(firstname) FROM customers
2 WHERE (state = 'OR' OR state = 'NY') AND gender = 'F';
```

	count
1	200

Number of records: 1 Number of fields: 1 Query time: 12 millisecond(s) Read-Only

WHAT IF

YOU WANT TO FILTER

ON EVERYTHING BUT...?

THE “NOT” KEYWORD



```
SELECT firstName, gender FROM users  
WHERE NOT gender = 'm';
```



COMPARISON OPERATORS

QUESTION

**WHO OVER THE AGE OF 44 HAS AN INCOME
OF 100 000?**

COMPARISON OPERATORS



```
10 > 20 -- false
10 < 20 -- true
10 <= 20 -- true
10 >= 9 --true
0 = 0 -- true
1 != 0 -- true
```

EQUAL TO

CHECK IF THE GIVEN VALUES ARE EQUAL.

IF IT'S EQUAL, THEN THE CONDITION WILL BE TRUE.

```
0 = 0 -- true
```

```
'Mo' = 'mo' -- false
```



NOT EQUAL TO

CHECK IF THE GIVEN VALUES ARE *NOT EQUAL*.

IF IT'S NOT EQUAL, THEN THE CONDITION WILL BE TRUE.

```
0 != 1 -- true
```

```
0 <> 1 -- true
```

GREATER THAN

CHECK IF THE GIVEN VALUE IS LARGER THAN THE OTHER VALUE.

IF IT'S GREATER, THEN THE CONDITION WILL BE TRUE.

```
10 > 0 -- true
```

```
0 > 10 -- false
```

```
'abc' > 'ace' -- true
```

GREATER THAN OR EQUAL TO

CHECK IF THE GIVEN VALUE IS LARGER THAN OR EQUAL THE OTHER VALUE.
IF IT'S GREATER OR EQUAL, THEN THE CONDITION WILL BE TRUE.

```
2 >= 1 -- true
```

```
2 >= 2 -- true
```

LESS THAN OR EQUAL TO

CHECK IF THE GIVEN VALUE IS SMALLER THAN OR EQUAL TO THE OTHER VALUE.

IF IT'S LESS OR EQUAL, THEN THE CONDITION WILL BE TRUE.

```
0 <= 1 -- true
```

```
1 <= 1 -- true
```

- -- How many female customers do we have from the state of Oregon (OR)?
- -- Who over the age of 44 has an income of 100 000 or more? (excluding 44)
- -- Who between the ages of 30 and 50 has an income less than 50 000?
- -- What is the average income between the ages of 20 and 50? (Excluding 20 and 50)

- SELECT COUNT(firstName)
- FROM customers
- WHERE gender = 'F' and state = 'OR';
- --106
- #####
- SELECT COUNT(income)
- FROM customers
- WHERE age > 44 and income >= 100000;
- -- Result: 2497

- SELECT COUNT(income)
- FROM customers
- WHERE age >= 30 and age <= 50 AND income < 50000;
- -- Result: 2362
- -----
- SELECT AVG(income)
- FROM customers
- WHERE age > 20 and age < 50;
- --Result: 59409.926240780098

LOGICAL OPERATORS



LOGICAL OPERATORS

AND

OR

NOT

DEFINITIONS

AND

**IF BOTH BOOLEAN EXPRESSIONS ARE TRUE
THEN IT WILL RETURN RESULTS**

DEFINITIONS

```
SELECT firstName, lastName  
FROM users  
WHERE firstName = 'Mo' AND lastName = 'Binni';
```



DEFINITIONS

OR

**IF ANY BOOLEAN EXPRESSION IS TRUE
THEN IT WILL RETURN RESULTS**

DEFINITIONS

```
SELECT firstName, lastName  
FROM users  
WHERE firstName = 'Mo' OR firstName = 'Andrei';
```



DEFINITIONS

NOT

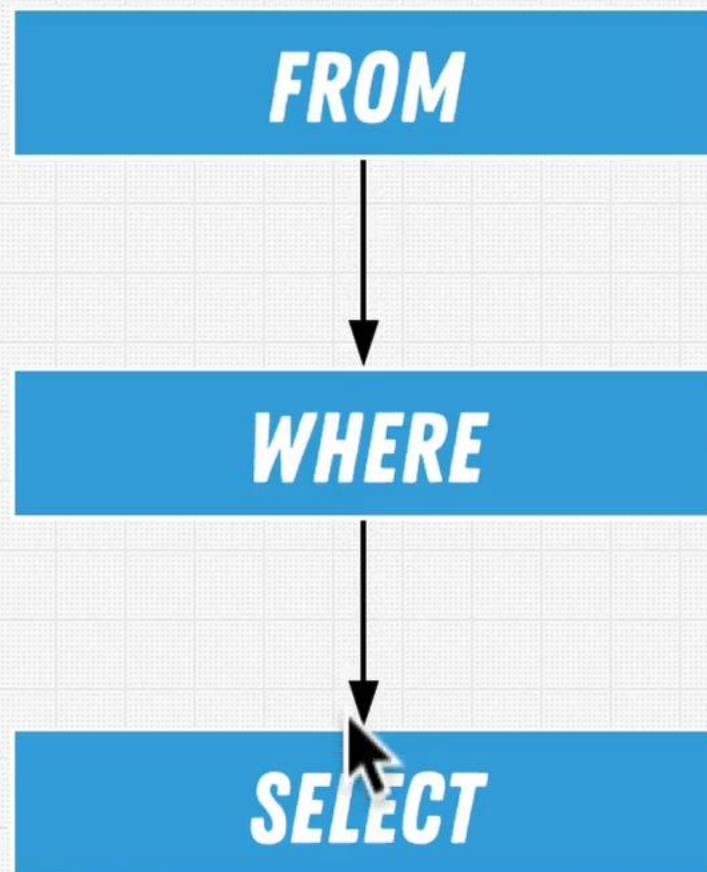
**I IF ANY BOOLEAN EXPRESSION IS NOT TRUE
 THEN IT WILL RETURN RESULTS**

DEFINITIONS

```
SELECT firstName, lastName  
FROM users  
WHERE NOT firstName = 'Mo';
```



ORDER OF OPERATIONS

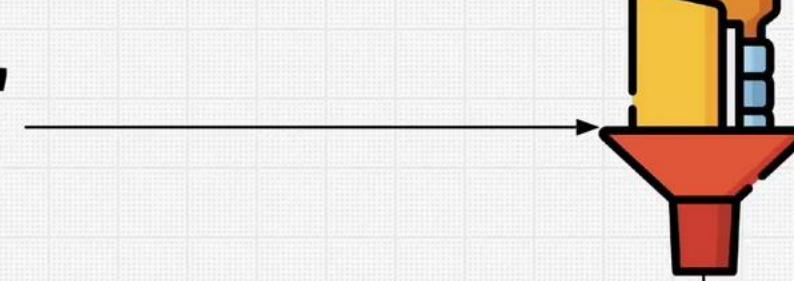


ORDER OF OPERATIONS

FROM



WHERE



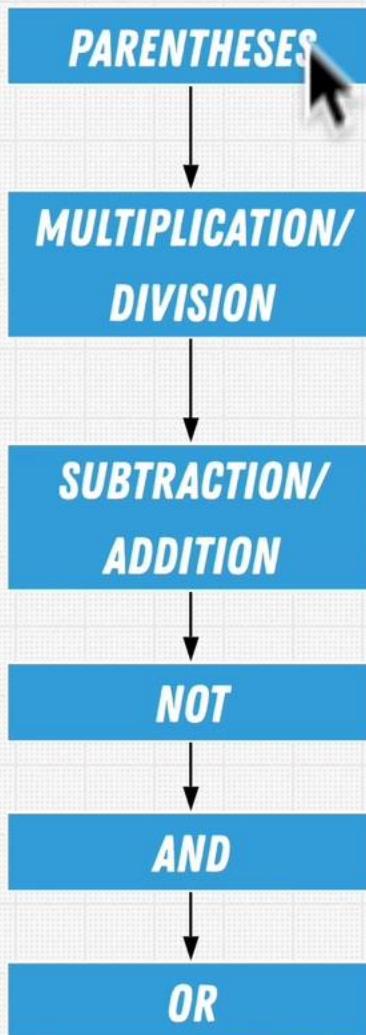
SELECT



OPERATOR PRECEDENCE

**A STATEMENT HAVING MULTIPLE OPERATORS IS EVALUATED
BASED ON THE PRIORITY OF OPERATORS.**

OPERATOR PRECEDENCE



OPERATOR PRECEDENCE

IF THE OPERATORS HAVE EQUAL PRECEDENCE, THEN THE OPERATORS ARE EVALUATED DIRECTIONALLY, FROM LEFT TO RIGHT OR RIGHT TO LEFT.



OPERATOR PRECEDENCE

Operator/Element	Associativity	Description
.	left	table/column name separator
::	left	PostgreSQL-style typecast
[]	left	array element selection
+ -	right	unary plus, unary minus
^	left	exponentiation
* / %	left	multiplication, division, modulo
+ -	left	addition, subtraction
(any other operator)	left	all other native and user-defined operators
BETWEEN IN LIKE ILIKE SIMILAR		range containment, set membership, string matching
<> = <= >= <>		comparison operators
IS ISNULL NOTNULL		IS TRUE, IS FALSE, IS NULL, IS DISTINCT FROM, etc
NOT	right	logical negation
AND	left	logical conjunction
OR	left	logical disjunction



```
SELECT state, gender FROM customers  
WHERE gender = 'F' AND state = 'OR' OR state = 'NY';
```



EXAMPLES

GET ME STATE AND GENDER WHERE YOU ARE:

A FEMALE FROM OREGON

OR

YOU ARE FROM NY

EXAMPLES



```
SELECT state, gender FROM customers  
WHERE gender = 'F' AND state = 'OR' OR state = 'NY';
```

EXAMPLES

GET ME STATE AND GENDER WHERE YOU ARE:

A FEMALE FROM OREGON

OR A FEMALE FROM NY



```
SELECT state, gender FROM customers  
WHERE gender = 'F' AND (state = 'OR' OR state = 'NY');
```

EXAMPLES

***GET ME STATE AND GENDER WHERE YOU ARE:
A FEMALE AND
YOU ARE FROM OREGON OR NY***

PRIORITY

**A STATEMENT HAVING MULTIPLE OPERATORS IS EVALUATED
BASED ON THE PRIORITY OF OPERATORS.**

PRIORITY



```
SELECT state, gender FROM customers  
WHERE gender = 'F' AND state = 'OR'  
OR gender = 'F' AND state = 'NY';
```

ORDER

1. GENDER FEMALE

2. FROM OREGON

1. GENDER FEMALE

2. FROM NY



```
SELECT state, gender FROM customers  
WHERE gender = 'F' AND (state = 'OR' OR state = 'NY');
```



ORDER

- 1. FROM OREGON**
- 2. FROM NY**
- 3. GENDER FEMALE**



DIRECTION

IF THE OPERATORS HAVE EQUAL PRECEDENCE, THEN THE OPERATORS ARE EVALUATED DIRECTIONALLY, FROM LEFT TO RIGHT OR RIGHT TO LEFT.

DIRECTION



```
age > 20  
AND salary > 1000  
AND gender = 'f'  
AND NOT state = 'NY'
```



ORDER

- 1. NOT FROM NY**
- 2. OLDER THAN 20**
- 3. SALARY > 1000**
- 4. GENDER FEMALE**

DIRECTION



```
(age > 20 OR age < 30)  
AND salary > 1000  
AND NOT state = 'NY'  
AND NOT state = 'OR'
```

ORDER

- 1. AGES 21 AND 29**
- 2. NOT FROM NY**
- 3. NOT FROM OR**
- 4. SALARY > 1000**



DIRECTION



```
age > 20
OR age < 30
AND salary > 1000
AND NOT state = 'NY'
AND NOT state = 'OR'
```

ORDER

FILTER 1

- 1. YOUNGER THAN 30**
- 2. NOT FROM NY**
- 3. NOT FROM OR**
- 4. SALARY > 1000**

FILTER 2

- 1. OLDER THAN 20**

DIRECTION



age > 20

OR age < 30

OR salary > 1000



ORDER

FILTER 1

1. YOUNGER THAN 30

FILTER 2

1. OLDER THAN 20

FILTER 3



1. SALARY > 1000

PRIORITY AND DIRECTION



```
(  
    salary > 10000 AND state = 'NY'  
    OR (  
        ( age > 20 AND age < 30 )  
        AND salary <= 20000  
    )  
)  
AND gender = 'F'
```

ORDER

FILTER 1

- 1. SALARY > 10 000**
- 2. FROM NY**
- 3. FEMALE**

FILTER 2

- 1. BETWEEN 21 AND 29**
- 2. SALARY LOWER THAN OR EQUAL TO 20 000**
- 3. FEMALE**

- Always remember the following: **HIGHEST to LOWEST**
- 1. Parenthesis
- 2. Arithmetic Operators
- 3. Concatenation Operators
- 4. Comparison Conditions
- 5. IS NULL, LIKE, NOT IN, etc.
- 6. NOT
- 7. AND
- 8. OR

1. Select people either under 30 or over 50 with an income above 50000 that are from either Japan or Australia
2. What was our total sales in June of 2004 for orders over 100 dollars?

- SELECT firstname, income, age from customers
- WHERE income > 50000 AND (age < 30 OR age >= 50)
- and (country = 'Japan' OR country = 'Australia')
- -----
- SELECT SUM(totalamount) from orders
- WHERE (orderdate >= '2004-06-01' AND orderdate <= '2004-06-30')
- AND totalamount > 100

CHECKING FOR EMPTY VALUES

***WHEN A RECORD DOES NOT HAVE
A VALUE IT IS CONSIDERED EMPTY***

CHECKING FOR EMPTY VALUES

customerID	firstName
1	Mark
2	<NULL>
3	Josh
4	<NULL>
5	Christen

```
1 CREATE TABLE "Student" (          Schema SQL
2     id serial PRIMARY KEY,        CREATE TABLE, INSERT,
3     name varchar(255),           UPDATE etc.
4     lastName varchar(255),
5     age int
6 );
7
8
9 INSERT INTO "Student" (name, lastName,
10    age) VALUES (
11        'STUDENT 1',
12        NULL,
13        NULL
14    );
15
16 INSERT INTO "Student" (name, lastName,
17    age) VALUES (
18        'STUDENT 2',
19        NULL,
20        25
21    );
22
23 INSERT INTO "Student" (name, lastName,
24    age) VALUES (
25        null,
26        'LAST NAME 3',
```



75. Checking For NULL Values

Empty space

```
9 INSERT INTO "Student" (name, lastName,
10   age) VALUES (
11     '          ',
12     NULL,
13     NULL
14 );
15 INSERT INTO "Student" (name, lastName,
16   age) VALUES (
17     'STUDENT 2',
18     NULL,
19     25
20 );
```

Text to DDL

Copy as Markdown

Results

1		null	null
2	STUDENT 2	null	25

THE “IS” OPERATOR



```
SELECT * FROM <table>  
WHERE <field> IS [NOT] NULL;
```



THE “IS” OPERATOR

***ALLOWS YOU TO FILTER ON VALUES
THAT ARE NULL, NOT NULL,
TRUE OR FALSE***

THE “IS” OPERATOR

```
SELECT * FROM users  
WHERE age = 20 IS FALSE;
```



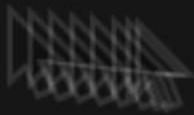
EVERYONE THAT IS NOT 20

NULL VALUE SUBSTITUTION



***ABILITY TO REPLACE NULL VALUES
TO OPERATE ON THE DATA***

COALESCE



```
SELECT coalesce(<column>, 'Empty') AS column_alias  
FROM <table>
```

NULL VALUE SUBSTITUTION

**COALESCE RETURNS THE FIRST
NON-NULL VALUE IN A LIST**

COMBINE COALESING

```
SELECT coalesce(  
    <column1>,  
    <column2>,  
    <column3>,  
    'Empty' ) AS combined_columns  
FROM <table>
```

```
1 CREATE TABLE "Student" (          Schema SQL
2     id serial PRIMARY KEY,          CREATE TABLE, INSERT,
3     name varchar(255),           UPDATE etc.
4     lastName varchar(255),
5     age int
6 );
7
8
9 INSERT INTO "Student" (name, lastName,
10    age) VALUES (
11        'STUDENT 1',
12        NULL,
13        NULL
14    );
15
16 INSERT INTO "Student" (name, lastName,
17    age) VALUES (
18        'STUDENT 2',
19        NULL,
20        25
21    );
22
23 INSERT INTO "Student" (name, lastName,
24    age) VALUES (
25        null,
```

```
1 CREATE TABLE "Student" (           Schema SQL
2   id serial PRIMARY KEY,          CREATE TABLE, INSERT,
3   name varchar(255),             UPDATE etc.
4   lastName varchar(255),
5   age int
6 );
7
8
9 INSERT INTO "Student" (name, lastName,
10   age) VALUES (
11     'STUDENT 1',
12     NULL,
13     NULL
14 );
15
16 INSERT INTO "Student" (name, lastName,
17   age) VALUES (
18     'STUDENT 2',
```

Text to DDL

CREATE TABLE, INSERT,
UPDATE etc.

```
1
2
3
4
5 SELECT name, lastName from "Student";
```

Have any feedback?  Query SQ



Results

Query #1 Execution time: 1ms

name	lastname
STUDENT 1	null
STUDENT 2	null
null	LAST NAME 3
null	null
STUDENT 5	null

```
1 CREATE TABLE "Student" (          Schema SQL
2   id serial PRIMARY KEY,          CREATE TABLE, INSERT,
3   name varchar(255),           UPDATE etc.
4   lastName varchar(255),
5   age int
6 );
7
8
9 INSERT INTO "Student" (name, lastName,
10  age) VALUES (
11    'STUDENT 1',
12    NULL,
13    NULL
14  );
15 INSERT INTO "Student" (name, lastName,
16  age) VALUES (
17    'STUDENT 2',
```

Text to DDL

Schema SQL

1

4

```
5 SELECT coalesce(name, 'no name
available'), lastName from "Student";
```

1

Query successfully executed in 7ms

X



Results

Query #1 **Execution time: 1ms**

coalesce	lastname
STUDENT 1	null
STUDENT 2	null
 name available	LAST NAME 3
no name available	null
STUDENT 5	null

1.Assuming a student's minimum age for the class is 15, what is the average age of a student?

SELECT avg(coalesce(age, 15)) FROM "Student";

1.Replace all empty first or last names with a default

**SELECT id, coalesce(name, 'fallback'), coalesce(lastName, 'lastName'),
age FROM "Student";**

SELECT <column>
FROM <table>
WHERE <column> BETWEEN X AND Y



- -- Who between the ages of 30 and 50 has an income less than 50 000?
 - -- (include 30 and 50 in the results)
-
- -- What is the average income between the ages of 20 and 50?
(Including 20 and 50)

- -- Who between the ages of 30 and 50 has an income less than 50 000?
- SELECT *
- FROM customers
- WHERE age BETWEEN 30 AND 50 AND income < 50000;
- -- What is the average income between the ages of 20 and 50? (Including 20 and 50)
- SELECT AVG(income)
- FROM customers
- WHERE age BETWEEN 20 AND 50;



```
SELECT * FROM <table>  
WHERE <column> IN (value1,value2,...)
```



The screenshot shows a database interface with the following details:

- Top Bar:** Includes icons for file operations (New, Open, Save, Close), a clock, a user profile, and a database connection labeled "Employees".
- Top Bar:** Shows "2 schemas".
- Top Bar:** Buttons for "Execute" and "Execute Current".
- Left Sidebar:** A tree view with three items:
 - Notification channels (0)
 - Schemas (1) - This item is selected, indicated by a blue background.
 - Triggers (0)
- Right Panel:** A code editor with the following SQL query:

```
1 SELECT * FROM employees
2 WHERE emp_no IN (100001, 100006, 11008);
3 -WHERE emp_no = 100001 OR emp_no = 100006
```

A black cursor arrow is positioned over the number "1" at the start of the first line of the query.

- Question: How many orders were made by customer 7888, 1082, 12808, 9623
- SELECT COUNT(orderid)
- FROM orders
- WHERE customerid IN (7888, 1082, 12808, 9623)
- Question: How many cities are in the district of Zuid-Holland, Noord-Brabant and Utrecht?
- SELECT COUNT(id)
- FROM city
- WHERE district IN ('Zuid-Holland', 'Noord-Brabant', 'Utrecht');



```
SELECT first_name FROM employees  
WHERE first_name LIKE 'M%';
```

GET EVERYONE WHO'S NAME START WITH 'M'



Use Cases	Meaning
LIKE '%2'	Fields that end with 2
LIKE '%2%'	Fields that have 2 anywhere in the value 
LIKE '_00%'	Fields that have 2 zero's as the second and third character and anything after
LIKE '%200%'	Fields that have 200 anywhere in the value
LIKE '2_%_%	Finds any values that start with 2 and are at least 3 characters in length
LIKE '2___3'	Finds any values in a five-digit number that start with 2 and end with 3

CASTING TO TEXT

```
CAST(salary AS text);  
salary::text
```



CASE INSENSITIVE MATCHING

Employees 2 schemas Execute



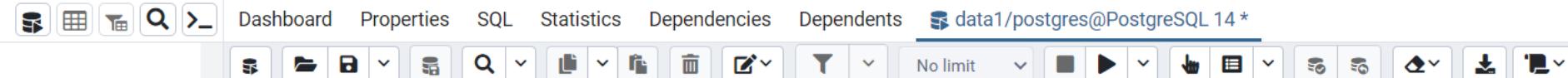
- Notification channels (0)
- Schemas (1)
- Triggers (0)

```
1 SELECT * FROM employees  
2 WHERE first_name LIKE 'G%';
```

The screenshot shows a database interface with a toolbar at the top. The toolbar includes icons for edit, refresh, schema browser, and two schemas (Employees and 2 schemas). The 'Employees' schema is selected. There are also 'Execute' and 'Execute Current' buttons. Below the toolbar is a navigation bar with icons for back, forward, and search. On the left, a sidebar lists 'Notification channels (0)', 'Schemas (1)' (selected), and 'Triggers (0)'. The main area contains a code editor with the following SQL query:

```
1 | SELECT * FROM employees
2 | WHERE first_name ILIKE 'G%GER';
```

A large black cursor arrow points to the 'Execute' button in the toolbar.



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

```
1 SELECT * FROM employees
2 WHERE first_name ILIKE 'G%GER'
```

	emp_no [PK] integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender gender	hire_date date
1	10202	1956-01-05	Greger	Lichtner	M	1991-10-06
2	10422	1954-06-02	Greger	Rubsam	F	1990-12-07
3	11084	1955-07-03	Greger	Senzako	M	1990-05-31
4	13902	1954-04-20	Greger	Kilgore	M	1993-07-04
5	13935	1961-07-20	Greger	Gopalakrishnan	F	1988-05-16
6	14848	1957-06-14	Greger	Tagansky	M	1997-05-27
7	16740	1957-08-17	Greger	Oehlmann	M	1986-03-06
8	18171	1957-11-26	Greger	Litzler	M	1995-05-24
9	25772	1952-03-11	Greger	Marreeve	M	1988-04-09
10	26293	1963-02-06	Greger	Peyn	F	1994-12-24



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

```
1 SELECT * FROM employees  
2 WHERE first_name LIKE 'G%'
```

	emp_no [PK] integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender gender	hire_date date
1	10001	1953-09-02	Georgi	Facello	M	1986-06-26
2	10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02
3	10055	1956-06-06	Georgy	Dredge	M	1992-04-27
4	10063	1952-08-06	Gino	Leonhardt	F	1989-04-08
5	10075	1960-03-09	Gao	Dolinsky	F	1987-03-19
6	10121	1962-07-14	Guoxiang	Ramsay	M	1989-05-03
7	10124	1962-05-23	Geraldo	Marwedel	M	1991-09-05
8	10133	1963-12-12	Giri	Isaak	M	1985-12-15
9	10202	1956-01-05	Greger	Lichtner	M	1991-10-06
10	10207	1955-05-28	Girolamo	Anandan	F	1992-10-11

- Question: Find the age of all employees who's name starts with M.
- `SELECT emp_no, first_name, EXTRACT (YEAR FROM AGE(birth_date)) as "age" FROM employees`
- `WHERE first_name ILIKE 'M%';`



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
- > Tables (6)

- > departments
- > dept_emp
- > dept_manager
- > employees
- > salaries
- > titles
- > Trigger Functions
- > Types

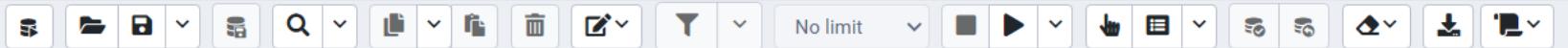
```
1 SELECT emp_no, first_name, EXTRACT (YEAR FROM AGE(birth_date)) as "age" FROM employees
2 WHERE first_name ILIKE 'M%';
3
```

	emp_no [PK] integer	first_name character varying (14)	age numeric
1	10011	Mary	69
2	10020	Mayuko	70
3	10042	Magy	67
4	10044	Mingsen	61
5	10045	Moss	65
6	10054	Mayumi	66
7	10069	Margareta	62
8	10074	Mokhtar	67
9	10077	Mona	59
10	10109	Mariusz	64

- SELECT count(emp_no) FROM employees
- WHERE first_name ILIKE 'A%R';



- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types



No limit

data1/postgres@PostgreSQL 14 *

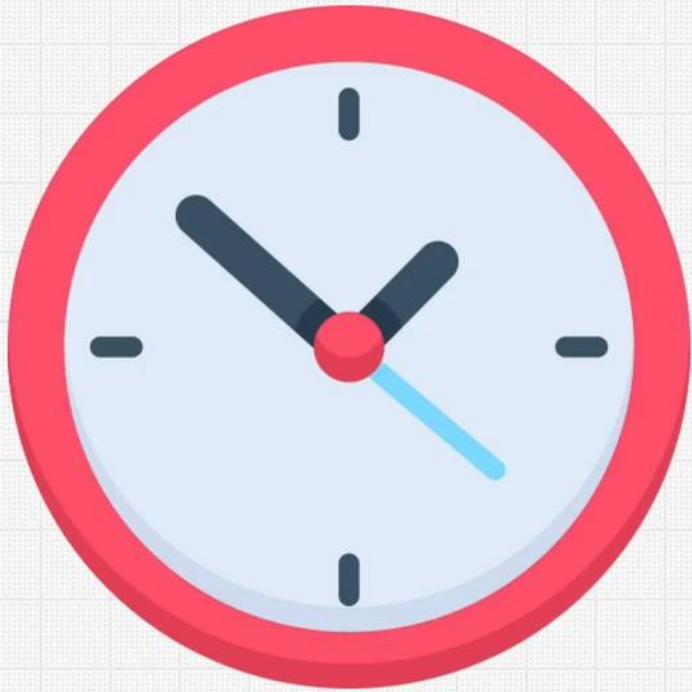
Query Editor Query History

```
1 SELECT count(emp_no) FROM employees
2 WHERE first_name ILIKE 'A%R';
3
```

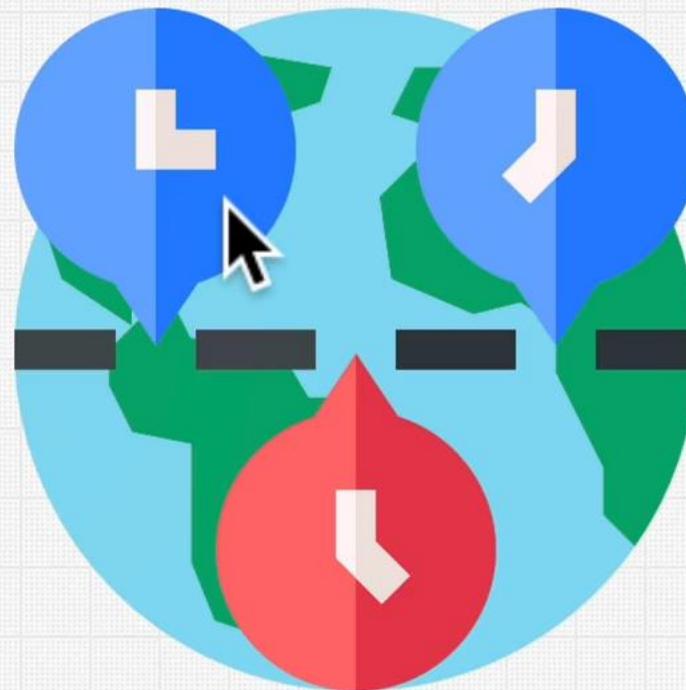
Data Output Explain Messages Notifications

	count	bigint
1	1846	

DATE FILTERING



WHAT IS A TIMEZONE



SIR SANFORD FLEMING

WHAT IS UTC?

**UNIVERSAL COORDINATE
TIME**

WHAT'S THE DIFFERENCE?

ONE IS A TIME ZONE

ONE IS A TIME STANDARD



WHAT'S THE SAME?

***THEY SHARE THE SAME
CURRENT TIME***

WHAT SHOULD I USE?



"ALWAYS" USE UTC



- > [Tables](#)
- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- > [Schemas \(1\)](#)
 - > [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - > [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)

data1/postgres@PostgreSQL 14

Query EditorQuery History

1 SHOW TIMEZONE

Data OutputExplainMessagesNotifications

TimeZone	text	
1	Asia/Calcutta	

MANIPULATING DATES?



HOW DO DATES LOOK?

POSTGRESQL USES ISO-8601



HOW DO DATES LOOK?

YYYY-MM-DDTHH:MM:SS



2017-08-17T12:47:16+02:00

WHAT IS A FORMAT?

**A FORMAT IS A WAY OF
REPRESENTING A DATE AND TIME**

TIMESTAMPS

***A TIMESTAMP IS A DATE
WITH TIME AND TIMEZONE INFO***

90. Timestamps

Start Page Schema Editor SQL Employees SQL Employees

Employees 2 schemas Execute Execute Current

Notification channels (0) Schemas (1) Triggers (0)

```
1 -- INSERT INTO timezones VALUES(  
2 --     TIMESTAMP WITHOUT TIME ZONE '2000-01-01 10:00:00-05' ,  
3 --     TIMESTAMP WITH TIME ZONE '2000-01-01 10:00:00-05'  
4 -- );  
5  
6 CREATE TABLE timezones (  
7     ts TIMESTAMP WITHOUT TIME ZONE ,  
8     tz TIMESTAMP WITH TIME ZONE  
9 )
```

1x 2:01 / 10:27



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications

- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)

- > departments
- > dept_emp
- > dept_manager
- > employees
- > salaries
- > titles
- > Trigger Functions
- > Types

```
1 CREATE TABLE Timezones(  
2     ts TIMESTAMP WITHOUT TIME ZONE,  
3     tz TIMESTAMP with TIME ZONE  
4  
5  
6  
7 )  
8
```

CREATE TABLE

Query returned successfully in 78 msec.

Start Page

Schema Editor

SQL Employees

SQL Employees

Employees 2 schemas Execute

Notification channels (0) Schemas (1) Triggers (0)

```
1| INSERT INTO timezones VALUES(
2|   TIMESTAMP WITHOUT TIME ZONE '2000-01-01 10:00:00-05',
3|   TIMESTAMP WITH TIME ZONE '2000-01-01 10:00:00-05'
4| );
5|
6| -- CREATE TABLE timezones (
7| --   ts TIMESTAMP WITHOUT TIME ZONE,
8| --   tz TIMESTAMP WITH TIME ZONE
9| -- )
```



- > [Tables](#)
- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- > [Schemas \(1\)](#)
 - > [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - > [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)

```
1 INSERT INTO TIMEZONES VALUES(  
2  
3 TIMESTAMP WITHOUT TIME ZONE '2023-03-07 10:50',  
4 TIMESTAMP WITH TIME ZONE '2023-03-07 10:50'  
5 )
```

INSERT 0 1

Query returned successfully in 61 msec.



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

1 SELECT * FROM timezones

	ts	tz
	timestamp without time zone	timestamp with time zone
1	2023-03-07 10:50:00	2023-03-07 10:50:00+05:30

DATE OPERATORS

**POSTGRES GIVES US
OPERATORS TO HELP SIMPLIFY
DATES**



No limit



data1/postgres@PostgreSQL 14 ▾

Query Editor Query History

```
1 SELECT now()::date;  
2 SELECT CURRENT_DATE
```

Data Output Explain Messages Notifications

	current_date	🔒
	date	
1	2023-07-03	

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
- > Tables (6)

- > departments
- > dept_emp
- > dept_manager
- > employees
- > salaries
- > titles
- > Trigger Functions
- > Types

pgAdmin File Object Tools Help

Browser

Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (6)
departments
dept_emp
dept_manager
employees
salaries
titles
Trigger Functions
Types

Dashboard Properties SQL Statistics Dependencies Dependents data1/postgres@PostgreSQL 14 *

No limit

Query Editor Query History

```
1 SELECT TO_CHAR(CURRENT_DATE, 'dd/mm/yy')
```

Data Output Explain Messages Notifications

	to_char	text
1	03/07/23	

FORMAT MODIFIERS



IDENTIFIER	MEANING
D	Day
M	Month
Y	Year

POSTGRES DOCS

Pattern	Description
HH	hour of day (01-12)
HH12	hour of day (01-12)
HH24	hour of day (00-23)
MI	minute (00-59)
SS	second (00-59)
MS	millisecond (000-999)
US	microsecond (000000-999999)
SSSS	seconds past midnight (0-86399)
AM, am, PM or pm	meridiem indicator (without periods)
A.M., a.m., P.M. or p.m.	meridiem indicator (with periods)
Y, YYY	year (4 or more digits) with comma
YYYY	year (4 or more digits)
YY	last 3 digits of year

Day of the year

The screenshot shows the pgAdmin 4 interface. The top navigation bar includes File, Object, Tools, and Help. The main window has tabs for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The SQL tab is active, showing a connection to 'data1/postgres@PostgreSQL 14 *'. The Query Editor contains the following SQL code:

```
1 SELECT TO_CHAR(CURRENT_DATE, 'DDD')
```

The Data Output tab displays the result of the query:

to_char	text
1	184

The Browser panel on the left lists various database objects under the 'Schemas (1)' section, specifically the 'public' schema, and the 'Tables (6)' section, which includes 'departments', 'dept_emp', 'dept_manager', 'employees', 'salaries', 'titles', and 'Trigger Functions'.



> Casts
> Catalogs
> Event Triggers
> Extensions
> Foreign Data Wrappers
> Languages
> Publications
> Schemas (1)
< public
> Aggregates
> Collations
> Domains
> FTS Configurations
> FTS Dictionaries
> FTS Parsers
> FTS Templates
> Foreign Tables
> Functions
> Materialized Views
> Operators
> Procedures
> Sequences
< Tables (6)
> departments
> dept_emp
> dept_manager
> employees
> salaries
> titles
> Trigger Functions
> Types

```
1 SELECT TO_CHAR(CURRENT_DATE, 'WW')
```

No limit



to_char	text
1	27

CALCULATE AGE



```
select AGE('1800/01/01')
```

?column?

ERROR



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

data1/postgres@PostgreSQL 14 *

Query Editor Query History

1 `SELECT AGE(date'1800-01-01')`

No limit



Data Output Explain Messages Notifications

age	interval	lock
1	223 years 6 mons 2 days	



- > [Tables](#)
- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- > [Schemas \(1\)](#)
 - > [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - > [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)

1 `SELECT AGE(date'1800/01/01')`

age	interval
1	223 years 6 mons 2 days

CALCULATE AGE BETWEEN



```
select AGE(date '1992/11/13', date '1800/01/01')
```

?column?

192 years 10 mons 12 days



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

```
1 SELECT AGE(date '1992/11/13',date'1800/01/01')
```

No limit



age	interval
1	192 years 10 mons 12 days

EXTRACT DAY



```
SELECT EXTRACT (DAY FROM date '1992/11/13') AS DAY
```



DAY



- > [Tables](#)
- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- > [Schemas \(1\)](#)
 - > [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - > [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)

```
1 SELECT EXTRACT(DAY FROM date'1992/11/13')AS DAY
```

No limit

day	numeric	lock
1	13	

pgAdmin File Object Tools Help

Browser

Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (6)
departments
dept_emp
dept_manager
employees
salaries
titles
Trigger Functions
Types

Dashboard Properties SQL Statistics Dependencies Dependents data1/postgres@PostgreSQL 14 *

No limit

Query Editor Query History

1 SELECT EXTRACT(MONTH FROM date'1992/11/13')AS MONTH

Data Output Explain Messages Notifications

	month	
	numeric	lock
1	11	

EXTRACT YEAR



```
SELECT EXTRACT (YEAR FROM date '1992/11/13') AS YEAR
```



DAY

1992



- > [Tables](#)
- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- > [Schemas \(1\)](#)
 - > [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - > [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)

```
1 SELECT EXTRACT(YEAR FROM date'1992/11/13')AS YEAR
```

year	numeric
1	1992

ROUND A DATE



```
SELECT DATE_TRUNC('year', date '1992/11/13');
```

?column?

1992-01-01



- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- > [Schemas \(1\)](#)
 - > [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - > [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)



```
1 SELECT DATE_TRUNC('year', date'1992/11/13')
```

	date_trunc
1	timestamp with time zone

1 1992-01-01 00:00:00+05:30



```
SELECT *  
FROM orders  
WHERE purchaseDate <= now( ) - interval '30 days'
```



DAY

<30 days before given date>

- Question: Get me all the employees above 60, use the appropriate date functions
 - Question: How many employees were hired in February?
 - `SELECT AGE(birth_date), * FROM employees`
 - `WHERE (`
 - `EXTRACT (YEAR FROM AGE(birth_date))`
 - `)>60;`
-

```
SELECT count(birth_date) FROM employees  
WHERE birth_date < now() - interval '61 years'
```



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

data1/postgres@PostgreSQL 14 ▾

Query Editor Query History

```
1 SELECT AGE(birth_date), * FROM employees
2 WHERE (
3   EXTRACT (YEAR FROM AGE(birth_date))
4 )>60;
```

Data Output Explain Messages Notifications

	age interval	emp_no [PK] integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender gender	hire_date date
1	69 years 10...	10001	1953-09-02	Georgi	Facello	M	1986-06-26
2	63 years 7 ...	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	69 years 2 ...	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	68 years 5 ...	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
5	70 years 2 ...	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
6	66 years 1 ...	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
7	65 years 4 ...	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
8	71 years 2 ...	10009	1952-04-19	Sumant	Peac	F	1985-02-18
9	69 years 7 ...	10011	1953-11-07	Mary	Sluis	F	1990-01-22
10	62 years 8 ...	10012	1960-10-04	Patricia	Bridgland	M	1992-12-18



- > [Tables](#)
- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- ▽ [Schemas \(1\)](#)
 - ▽ [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - ▽ [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)

```
1 SELECT count(birth_date) FROM employees
2 WHERE birth_date < now() - interval '61 years'
3
```

	count	bigint
1	240711	

- Question: How many employees were hired in February?
- `SELECT count(emp_no) FROM employees`
- `where EXTRACT (MONTH FROM hire_date) = 2;`



```
1 SELECT count(emp_no) FROM employees  
2 where EXTRACT (MONTH FROM hire_date) = 2;  
3
```

count	bigint
1	24448

- > Cascades
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions

- Question: How many employees were born in november?
- `SELECT COUNT(emp_no) FROM employees`
- `WHERE EXTRACT (MONTH FROM birth_date) = 11;`



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications

▼ Schemas (1)

- < public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences

▼ Tables (6)

- > departments
- > dept_emp
- > dept_manager
- > employees
- > salaries
- > titles
- > Trigger Functions
- > Types

```
1 SELECT COUNT(emp_no) FROM employees
2 WHERE EXTRACT (MONTH FROM birth_date) = 11;
3
```

	count	bigint
1	24500	

- Question: Who is the oldest employee?

```
SELECT MAX(AGE(birth_date)) FROM employees;
```



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

```
1 SELECT MAX(AGE(birth_date)) FROM employees;
```

	max	
	interval	
1	71 years 5 mons 2 days	

DISTINCT KEYWORD



REMOVE DUPLICATES



DISTINCT KEYWORD

***DISTINCT CLAUSE KEEPS ONE ROW
FOR EACH GROUP OF DUPLICATES***



DISTINCT KEYWORD

```
SELECT  
    DISTINCT <col1>, <col2>  
FROM  
    <table>;
```

**MULTIPLE COLUMNS WILL EVALUATE BASED ON THE
COMBINATION OF THE COLUMNS**



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

data1/postgres@PostgreSQL 14 *

Query Editor Query History

```
1 SELECT DISTINCT salary FROM salaries  
2  
3
```

Data Output Explain Messages Notifications

	salary	lock
	integer	
1	71772	
2	53347	
3	79163	
4	86036	
5	73130	
6	47778	
7	53885	
8	68453	
9	73152	
10	44342	

pgAdmin File Object Tools Help

Browser

Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (6)
departments
dept_emp
dept_manager
employees
salaries
titles
Trigger Functions
Types

Dashboard Properties SQL Statistics Dependencies Dependents data1/postgres@PostgreSQL 14 *

No limit

Query Editor Query History

```
1 SELECT DISTINCT salary,from_date FROM salaries
```

Data Output Explain Messages Notifications

	salary integer	from_date date
1	48229	2001-03-17
2	75524	1997-05-08
3	63265	1993-05-21
4	50343	1993-12-07
5	43141	1991-08-09
6	40182	1987-11-19
7	59812	1999-06-06
8	44833	1996-08-12
9	50415	1999-09-09
10	71657	1999-10-08

- * Question: What unique titles do we have?
- SELECT DISTINCT title FROM titles;



- > [Tables](#)
- > [Catalogs](#)
- > [Event Triggers](#)
- > [Extensions](#)
- > [Foreign Data Wrappers](#)
- > [Languages](#)
- > [Publications](#)
- > [Schemas \(1\)](#)
 - > [public](#)
 - > [Aggregates](#)
 - > [Collations](#)
 - > [Domains](#)
 - > [FTS Configurations](#)
 - > [FTS Dictionaries](#)
 - > [FTS Parsers](#)
 - > [FTS Templates](#)
 - > [Foreign Tables](#)
 - > [Functions](#)
 - > [Materialized Views](#)
 - > [Operators](#)
 - > [Procedures](#)
 - > [Sequences](#)
 - > [Tables \(6\)](#)
 - > [departments](#)
 - > [dept_emp](#)
 - > [dept_manager](#)
 - > [employees](#)
 - > [salaries](#)
 - > [titles](#)
 - > [Trigger Functions](#)
 - > [Types](#)

```
1 SELECT DISTINCT title FROM titles;
```

title	
	character varying (50)
1	Engineer
2	Senior Engineer
3	Manager
4	Assistant Engineer
5	Staff
6	Senior Staff
7	Technique Leader



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
- > Tables (6)

- > departments
- > dept_emp
- > dept_manager
- > employees
- > salaries
- > titles
- > Trigger Functions
- > Types

```
1 --Question: How many unique birth dates are there?
2 SELECT COUNT(DISTINCT birth_date)
3 from employees;
4
```

	count	bigint
1	4750	

SORTING DATA

A ↑
Z ↓

ORDER BY



```
SELECT * FROM CUSTOMERS  
ORDER BY <column> [ASC/DESC]
```

**SORT DATA EITHER ASCENDING OR DESCENDING
BY COLUMN**



```
SELECT * FROM customers  
ORDER BY name DESC
```



Buy Pro Version [Read About Pro Features](#)

Start Page Schema Editor salaries-employees SQL Employees employees

Employees 2 schemas Execute Execute Current

Notification channels (0) Schemas (1) Triggers (0)

```
1 | SELECT first_name, last_name FROM employees
2 | ORDER BY first_name, last_name DESC;
```

	first_name	last_name
1	Aamer	Zongker
2	Aamer	Zizka
3	Aamer	Zedlitz
4	Aamer	Zallococo
5	Aamer	Zaccaria
6	Aamer	Yurov
7	Aamer	Yavatkar
8	Aamer	Wendel

Number of records: 300024 Number of fields: 2 Query time: 5.039 second(s)



- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

```
1 SELECT first_name, last_name FROM employees
2 ORDER BY first_name DESC, last_name DESC
```

	first_name character varying (14)	last_name character varying (16)
1	Zvonko	Zuberek
2	Zvonko	Zobel
3	Zvonko	Zambonelli
4	Zvonko	Yurov
5	Zvonko	Yoshizawa
6	Zvonko	Yoshizawa
7	Zvonko	Yetto
8	Zvonko	Wroclawski
9	Zvonko	Wilharm
10	Zvonko	Wielonsky

USING EXPRESSIONS



```
SELECT * FROM customers  
ORDER BY LENGTH(name)
```



> Schemas
> Catalogs
> Event Triggers
> Extensions
> Foreign Data Wrappers
> Languages
> Publications
> Schemas (1)
> public
> Aggregates
> Collations
> Domains
> FTS Configurations
> FTS Dictionaries
> FTS Parsers
> FTS Templates
> Foreign Tables
> Functions
> Materialized Views
> Operators
> Procedures
> Sequences
> Tables (6)
> departments
> dept_emp
> dept_manager
> employees
> salaries
> titles
> Trigger Functions
> Types

```
1 SELECT first_name, last_name FROM employees
2 ORDER BY LENGTH(first_name) DESC;
```

	first_name character varying (14)	last_name character varying (16)
1	Chandrasekaran	Oehlmann
2	Gopalakrishnan	Uludag
3	Chandrasekaran	Curless
4	Chandrasekaran	Schoegge
5	Chandrasekaran	Jumpertz
6	Chandrasekaran	Perelgut
7	Gopalakrishnan	Cyne
8	Chandrasekaran	Farrag
9	Chandrasekaran	Ramsay
10	Chandrasekaran	Fiebach



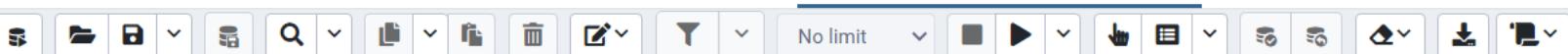
- > Schemas
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trigger Functions
 - > Types

```
1 --Question: Sort employees by age
2 SELECT * FROM employees
3 ORDER BY birth_date;
```

	emp_no [PK] integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender gender	hire_date date
1	65308	1952-02-01	Jouni	Pocchiola	M	1985-03-10
2	87461	1952-02-01	Moni	Decaestecker	M	1986-10-06
3	237571	1952-02-01	Ronghao	Schaad	M	1988-07-10
4	91374	1952-02-01	Eishiro	Kuzuoka	M	1992-02-12
5	207658	1952-02-01	Kiyokazu	Whitcomb	M	1988-07-26
6	406121	1952-02-01	Supot	Remmele	M	1989-01-27
7	409040	1952-02-02	Mayuko	Pardalos	M	1991-07-03
8	422556	1952-02-02	Ayakannu	Assaf	M	1989-04-03
9	427703	1952-02-02	Hyuckchul	Loncour	M	1986-09-19
10	421547	1952-02-02	Rasiah	Sudkamp	M	1988-04-30



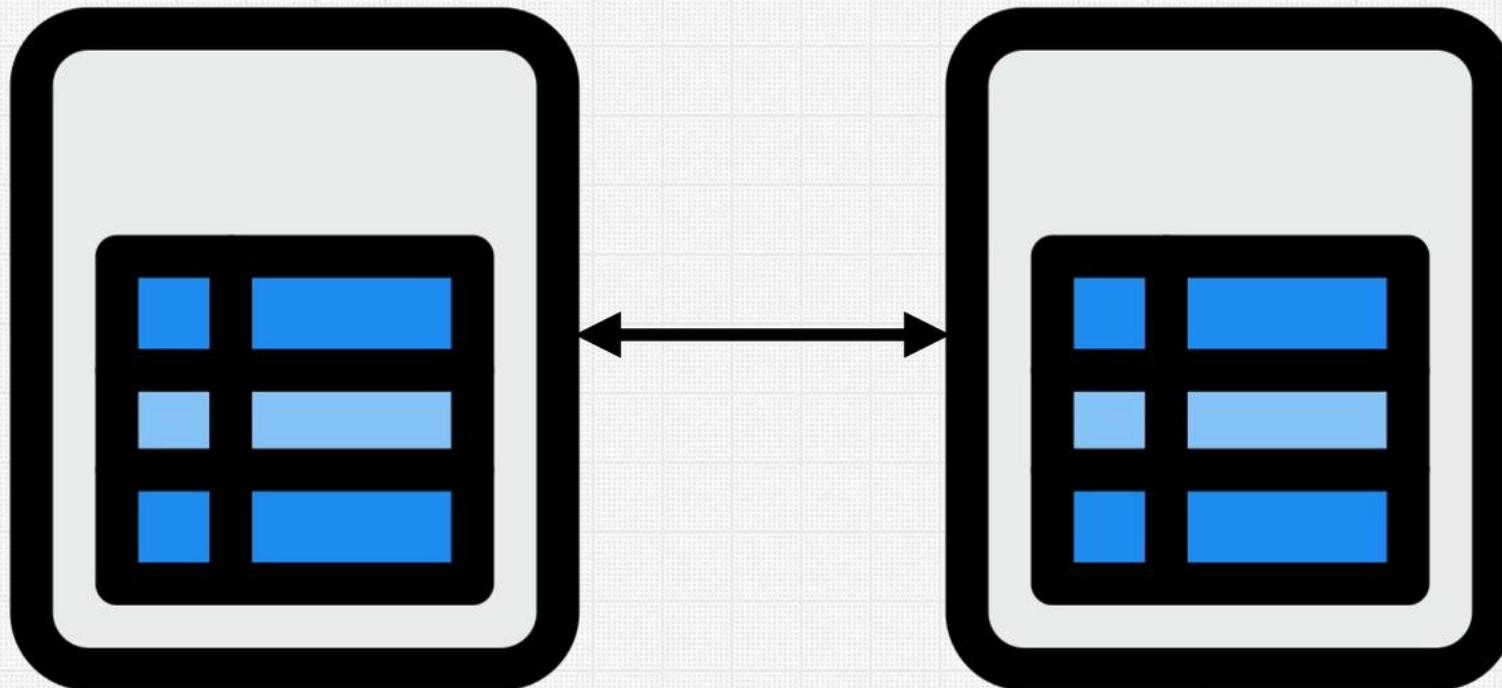
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > Sequences
 - > Tables (6)
 - > departments
 - > dept_emp
 - > dept_manager
 - > employees
 - > salaries
 - > titles
 - > Trinner Functions



```
1 --Question: Sort employees who's name starts with a "k" by hire_date
2 SELECT * FROM employees
3 WHERE first_name ILIKE 'k%'
4 ORDER BY hire_date;
```

	emp_no [PK] integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender gender	hire_date date
1	110303	1956-06-08	Krassimir	Wegerle	F	1985-01-01
2	98636	1952-12-15	Kannan	Mahnke	M	1985-02-01
3	494045	1963-09-26	Kasidit	Nitta	F	1985-02-02
4	448924	1963-02-08	Khalid	Siochi	F	1985-02-02
5	89618	1962-12-28	Keung	Peak	F	1985-02-02
6	235752	1964-08-02	Kwangsub	Selenyi	F	1985-02-02
7	297022	1960-02-19	Kousuke	Swist	M	1985-02-02
8	82019	1962-04-14	Kazuhito	Lupu	M	1985-02-03
9	431976	1962-01-15	Karlis	Maksimenko	M	1985-02-03
10	307675	1956-06-08	1985-02-03

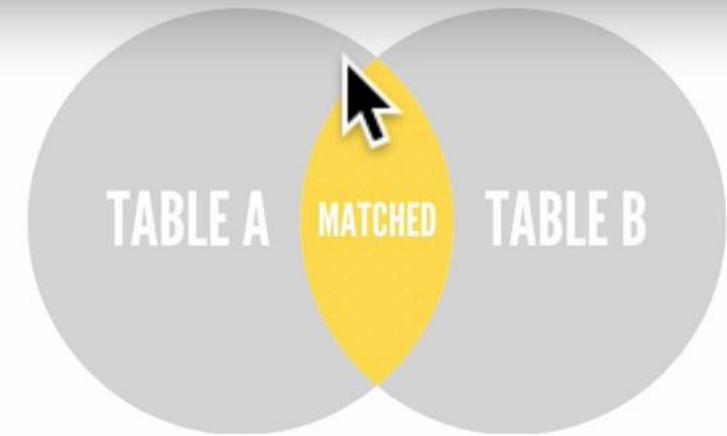
MULTI TABLE SELECT



COMBINING DATA

***WHAT IF YOU WANT TO COMBINE
DATA FROM MULTIPLE TABLES?***

INNER JOINS



**IF YOU HAVE A COLUMNS THAT LINKS THE COLUMN FROM
TABLE A TO TABLE B YOU CAN USE AN INNER JOIN**

The diagram illustrates an inner join between two tables, TABLE A and TABLE B. A curved arrow connects the "emp_no" column in TABLE A to the "emp_no" column in TABLE B, indicating that the join is based on this common key column.

TABLE A		
emp_no	first_name	last_name
...
20	Mo	Binni
21	Andrei	Neagoie
...

TABLE B	
emp_no	salary
...	...
20	100
21	200
22	300



- Before we learn about JOINS , let's quickly cover the **AS** clause which allows us to create an “alias” for a column or result.
- Let's see the example syntax



- **SELECT column AS new_name
FROM table**



SQL

- **SELECT SUM(column) AS new_name
FROM table**



SQL

- Example

Query Editor Query History

```
1 SELECT amount AS rental_price  
2 FROM payment;
```

Data Output

Explain

Messages

Notifications

	rental_price	lock	
	numeric (5,2)		
1	7.99		
2	1.99		

- Example

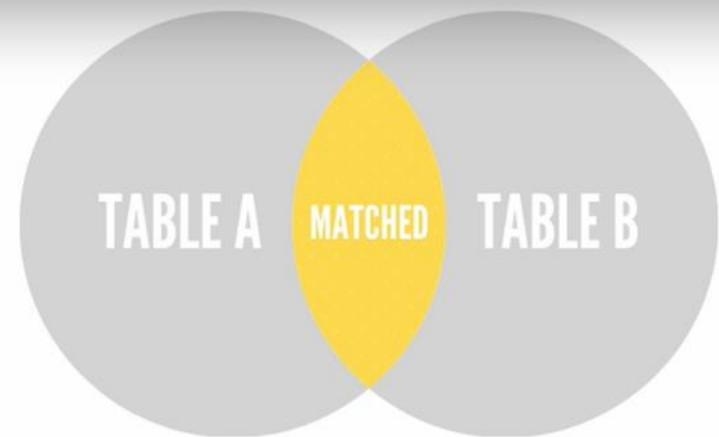
Query Editor Query History

```
1 SELECT SUM(amount) AS net_revenue
2 FROM payment;
```

Data Output Explain Messages Notifications

	net_revenue	
	numeric	🔒
1	61312.04	

INNER JOINS



```
SELECT a.emp_no,  
       CONCAT(a.first_name, a.last_name) as "name",  
       b.salary  
  FROM employees as a  
INNER JOIN salaries as b ON b.emp_no = a.emp_no;
```



- Servers (2)
 - PostgreSQL 14
 - Databases (11)
 - advanced-SQL
 - data1
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (7)
 - departments
 - dept_emp

Query Editor Query History

```
1 SELECT a.emp_no,
2        CONCAT(a.first_name,a.last_name)as "name",
3        b.salary
4        FROM employees as a
5        INNER JOIN salaries as b
6        ON b.emp_no=a.emp_no
```

Data Output Explain Messages Notifications

	emp_no integer	name text	salary integer
1	10005	KyoichiMaliniak	78228
2	10005	KyoichiMaliniak	82621
3	10005	KyoichiMaliniak	83735
4	10005	KyoichiMaliniak	85572
5	10005	KyoichiMaliniak	85076
6	10005	KyoichiMaliniak	86050
7	10005	KyoichiMaliniak	88448
8	10005	KyoichiMaliniak	88063
9	10005	KyoichiMaliniak	89724
10	10005	KyoichiMaliniak	90392

- Let's imagine a simple example.
- Our company is holding a conference for people in the movie rental industry.
- We'll have people register online beforehand and then login the day of the conference.



- An INNER JOIN will result with the set of records that match in both tables.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

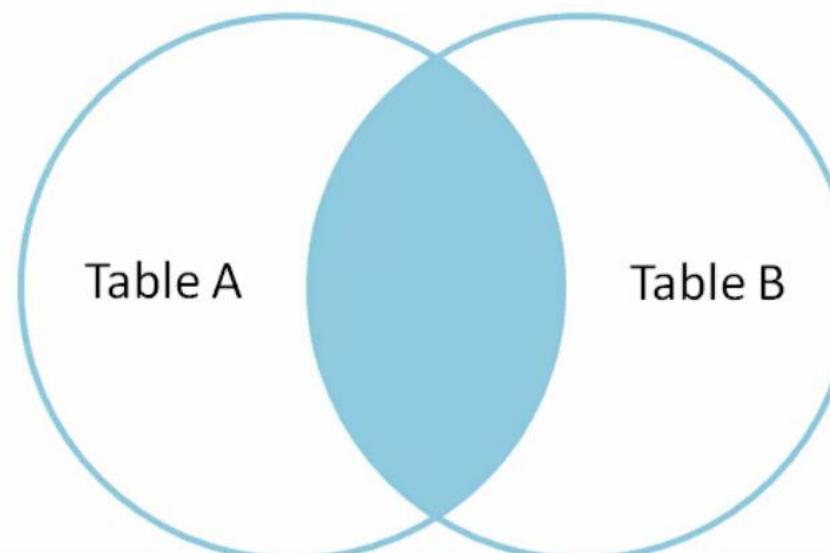
LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

- An INNER JOIN will result with the set of records that match in both tables.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

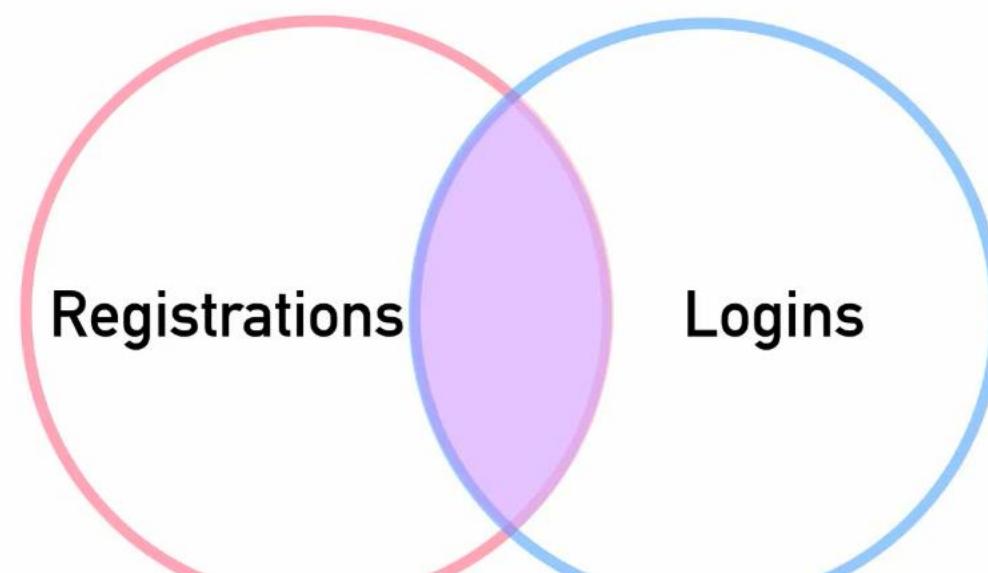
LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

- **SELECT * FROM TableA
INNER JOIN TableB
ON TableA.col_match = TableB.col_match**



- **SELECT * FROM Registrations
INNER JOIN Logins
ON Registrations.name = Logins.name**

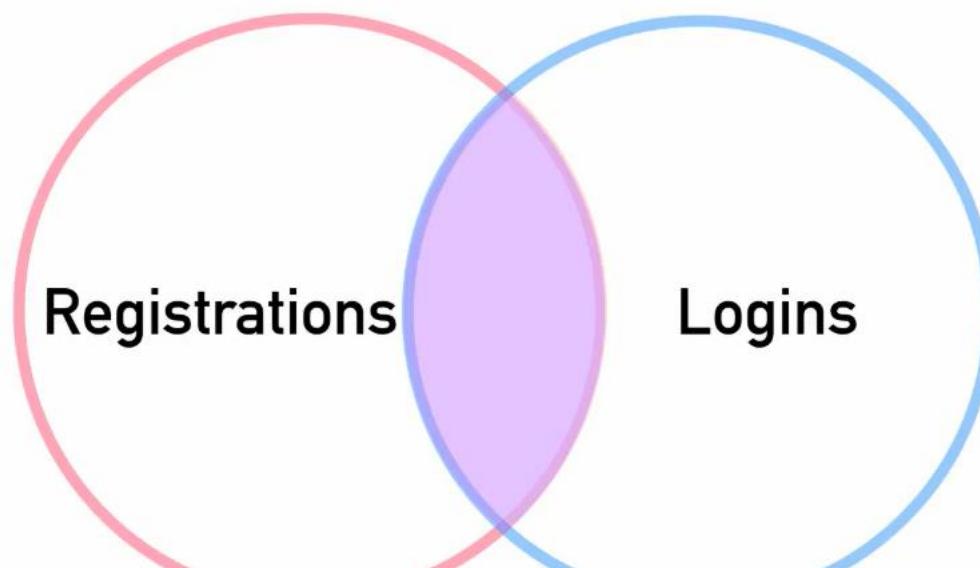
REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David



LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

- **SELECT * FROM Registrations
INNER JOIN Logins
ON Registrations.name = Logins.name**

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David



LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

- **SELECT * FROM Registrations
INNER JOIN Logins
ON Registrations.name = Logins.name**

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

- **SELECT reg_id, Logins.name, log_id
FROM Registrations
INNER JOIN Logins
ON Registrations.name = Logins.name**

RESULTS		
reg_id	name	log_id
1	Andrew	2
2	Bob	4

File ▾ Object ▾ Tools ▾ Help ▾

Dashboard Properties SQL Statistics Dependencies Dependents dvdrental/postgres@PostgreSQL 12 *

No limit

Query Editor Query History Scratch Pad

```
1 SELECT payment_id,payment.customer_id,first_name FROM payment
2 INNER JOIN customer
3 ON payment.customer_id = customer.customer_id
4
```

Data Output Explain Messages Notifications

customer_id integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	last_update timestamp without time zone
341	1	Peter	Menard	peter.menard@sakilacustom...	346	true	2006-02-14	2013-05-26 14:4
341	1	Peter	Menard	peter.menard@sakilacustom...	346	true	2006-02-14	2013-05-26 14:4
341	1	Peter	Menard	peter.menard@sakilacustom...	346	true	2006-02-14	2013-05-26 14:4
341	1	Peter	Menard	peter.menard@sakilacustom...	346	true	2006-02-14	2013-05-26 14:4
341	1	Peter	Menard	peter.menard@sakilacustom...	346	true	2006-02-14	2013-05-26 14:4
341	1	Peter	Menard	peter.menard@sakilacustom...	346	true	2006-02-14	2013-05-26 14:4
341	1	Peter	Menard	peter.menard@sakilacustom...	346	true	2006-02-14	2013-05-26 14:4
341	1	Peter	Menard	peter.menard@sakilacustom...	347	true	2006-02-14	2013-05-26 14:4

pgAdmin File ▾ Object ▾ Tools ▾ Help ▾

Dashboard Properties SQL Statistics Dependencies Dependents dvrental/postgres@PostgreSQL 12 *

Query Editor Query History Scratch Pad

```
1 SELECT payment_id,payment.customer_id,first_name
2 FROM payment
3 INNER JOIN customer
4 ON payment.customer_id = customer.customer_id
5
```

Data Output Explain Messages Notifications

	payment_id integer	customer_id smallint	first_name character varying (45)
1	17503	341	Peter
2	17504	341	Peter
3	17505	341	Peter
4	17506	341	Peter
5	17507	341	Peter
6	17508	341	Peter
7	17509	342	Harold

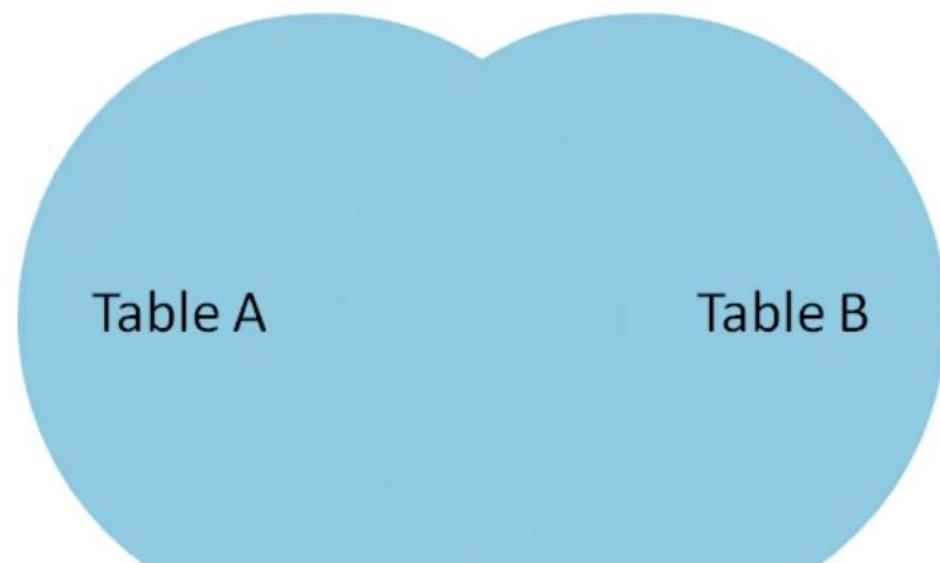
Udemy

- Recall we match Andrew and Bob in both tables

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

- **SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.col_match = TableB.col_match**

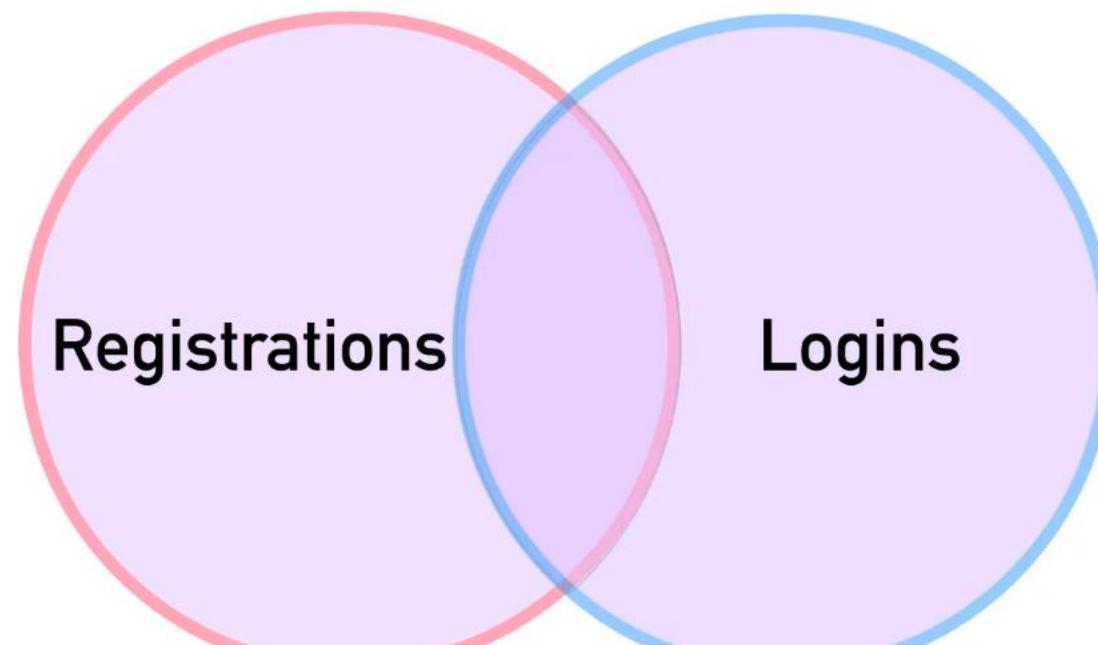




SQL

- `SELECT * FROM Registrations
FULL OUTER JOIN Logins
ON Registrations.name = Logins.name`

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David



LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

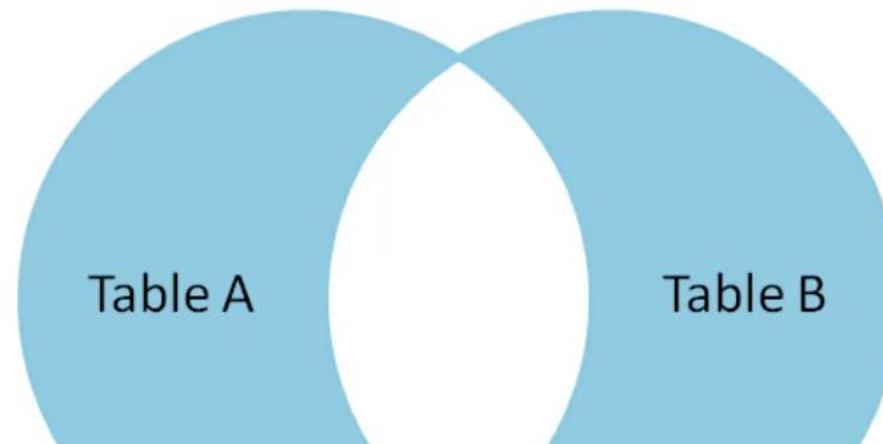
- SELECT * FROM Registrations FULL OUTER JOIN Logins
ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie	null	null
4	David	null	null
null	null	1	Xavier
null	null	2	Yolanda

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

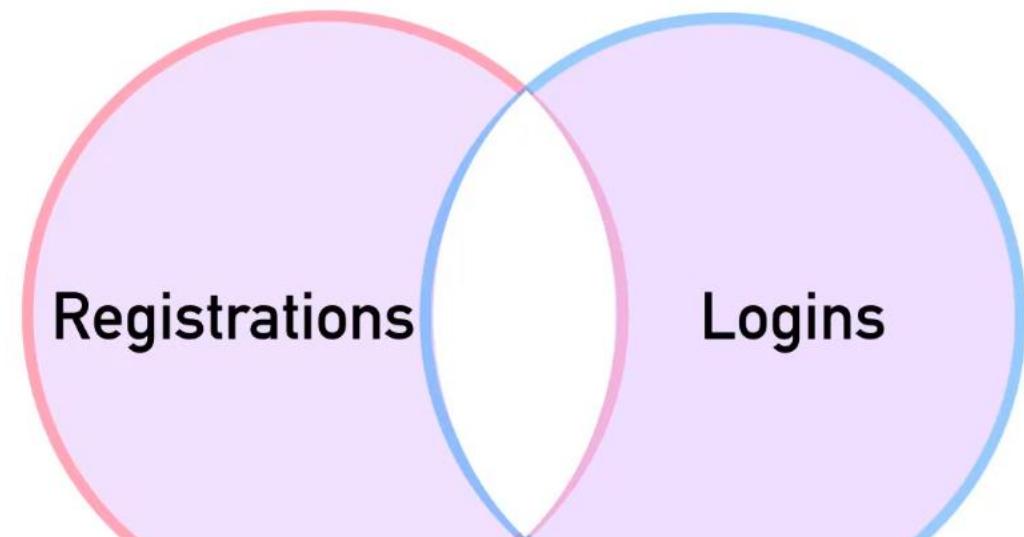
- ```
SELECT * FROM TableA
 FULL OUTER JOIN TableB
 ON TableA.col_match = TableB.col_match
 WHERE TableA.id IS null OR
 TableB.id IS null
```





SQL

```
SELECT * FROM Registrations FULL OUTER
JOIN Logins
ON Registrations.name = Logins.name
WHERE Registrations.reg_id IS null OR
Logins.log_id IS null
```





- `SELECT * FROM Registrations FULL OUTER JOIN Logins  
ON Registrations.name = Logins.name`

| REGISTRATIONS |         |
|---------------|---------|
| reg_id        | name    |
| 1             | Andrew  |
| 2             | Bob     |
| 3             | Charlie |
| 4             | David   |

| RESULTS |         |        |        |
|---------|---------|--------|--------|
| reg_id  | name    | log_id | name   |
| 1       | Andrew  | 2      | Andrew |
| 2       | Bob     | 4      | Bob    |
| 3       | Charlie | null   | null   |
| 4       | David   | null   | null   |
|         | null    | 1      | Xavier |

| LOGINS |         |
|--------|---------|
| log_id | name    |
| 1      | Xavier  |
| 2      | Andrew  |
| 3      | Yolanda |
| 4      | Bob     |

127.0.0.1:49738/browser/

Admin File ▾ Object ▾ Tools ▾ Help ▾

Dashboard Properties SQL Statistics Dependencies Dependents **dvdrental/postgres@PostgreSQL 12 \***

No limit

Query Editor Query History Scratch Pad

```
1 SELECT * FROM customer
2 FULL OUTER JOIN payment
3 ON customer.customer_id = payment.customer_id
4 WHERE customer.customer_id IS null
5 OR payment.payment_id IS null
```

Data Output Explain Messages Notifications

| re_date    | last_update             | active | payment_id | customer_id | staff_id | rental_id | amount | payment_date               |
|------------|-------------------------|--------|------------|-------------|----------|-----------|--------|----------------------------|
| 2013-05-26 | 2013-05-26 14:49:45.738 | 1      | 17503      | 341         | 2        | 1520      | 7.99   | 2007-02-15 22:25:46.996577 |
| 2013-05-26 | 2013-05-26 14:49:45.738 | 1      | 17504      | 341         | 1        | 1778      | 1.99   | 2007-02-16 17:23:14.996577 |
| 2013-05-26 | 2013-05-26 14:49:45.738 | 1      | 17505      | 341         | 1        | 1849      | 7.99   | 2007-02-16 22:41:45.996577 |
| 2013-05-26 | 2013-05-26 14:49:45.738 | 1      | 17506      | 341         | 2        | 2829      | 2.99   | 2007-02-19 19:39:56.996577 |
| 2013-05-26 | 2013-05-26 14:49:45.738 | 1      | 17507      | 341         | 2        | 3130      | 7.99   | 2007-02-20 17:31:48.996577 |

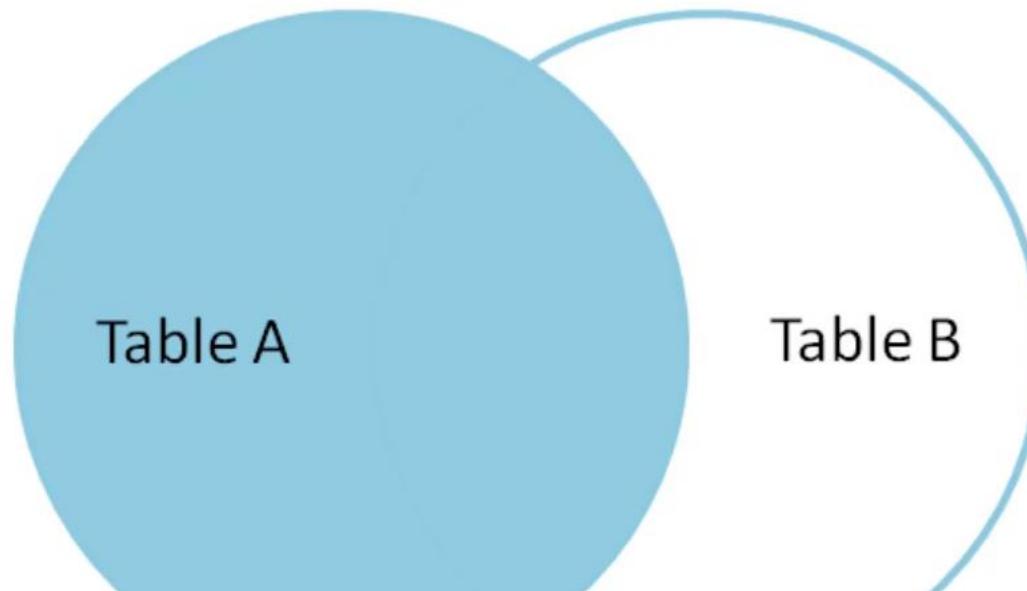


- A LEFT OUTER JOIN results in the set of records that are in the left table, if there is no match with the right table, the results are null.
- Later on we will learn how to add WHERE statements to further modify a LEFT OUTER JOIN



SQL

- **SELECT \* FROM **TableA**  
LEFT OUTER JOIN **TableB**  
ON **TableA.col\_match = TableB.col\_match****





Let's explore a **LEFT OUTER JOIN** with our two example tables.

| REGISTRATIONS |         |
|---------------|---------|
| reg_id        | name    |
| 1             | Andrew  |
| 2             | Bob     |
| 3             | Charlie |
| 4             | David   |

| LOGINS |         |
|--------|---------|
| log_id | name    |
| 1      | Xavier  |
| 2      | Andrew  |
| 3      | Yolanda |
| 4      | Bob     |



- **SELECT \* FROM Registrations  
LEFT OUTER JOIN Logins  
ON Registrations.name = Logins.name**

**REGISTRATIONS**

| reg_id | name    |
|--------|---------|
| 1      | Andrew  |
| 2      | Bob     |
| 3      | Charlie |
| 4      | David   |

**RESULTS**

| reg_id | name    | log_id | name   |
|--------|---------|--------|--------|
| 1      | Andrew  | 2      | Andrew |
| 2      | Bob     | 4      | Bob    |
| 3      | Charlie | null   | null   |
| 4      | David   | null   | null   |

**LOGINS**

| log_id | name    |
|--------|---------|
| 1      | Xavier  |
| 2      | Andrew  |
| 3      | Yolanda |
| 4      | Bob     |



- **SELECT \* FROM Registrations  
LEFT OUTER JOIN Logins  
ON Registrations.name = Logins.name**

**REGISTRATIONS**

| reg_id | name    |
|--------|---------|
| 1      | Andrew  |
| 2      | Bob     |
| 3      | Charlie |
| 4      | David   |

**RESULTS**

| reg_id | name    | log_id | name   |
|--------|---------|--------|--------|
| 1      | Andrew  | 2      | Andrew |
| 2      | Bob     | 4      | Bob    |
| 3      | Charlie | null   | null   |
| 4      | David   | null   | null   |

**LOGINS**

| log_id | name    |
|--------|---------|
| 1      | Xavier  |
| 2      | Andrew  |
| 3      | Yolanda |
| 4      | Bob     |



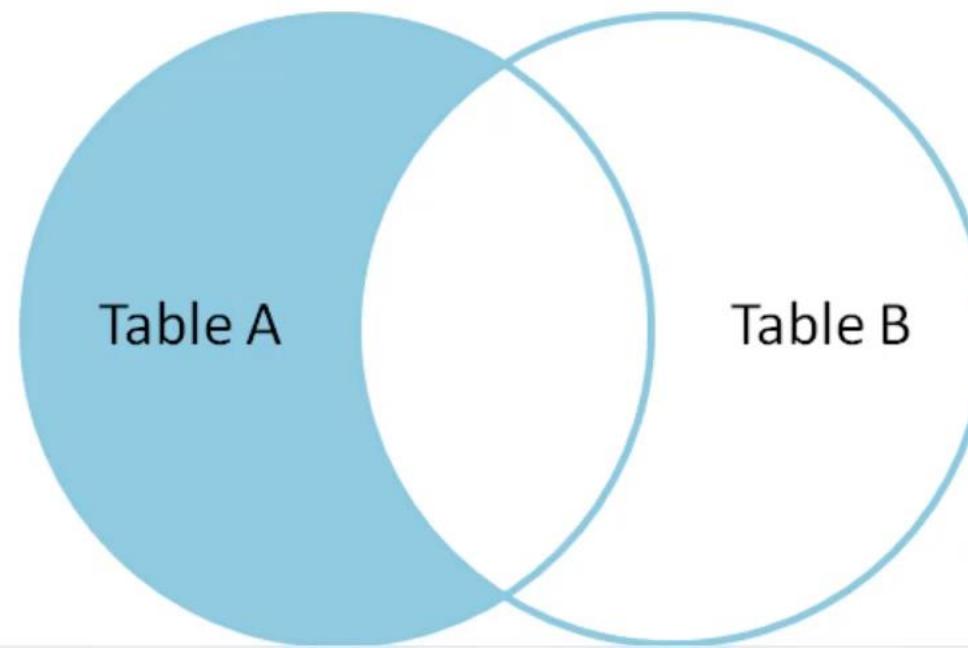
# LEFT OUTER JOIN With WHERE

Get rows unique to left table





- `SELECT * FROM TableA  
LEFT OUTER JOIN TableB  
ON TableA.col_match = TableB.col_match  
WHERE TableB.id IS null`





- **SELECT \* FROM Registrations  
LEFT OUTER JOIN Logins  
ON Registrations.name = Logins.name**

**REGISTRATIONS**

| reg_id | name    |
|--------|---------|
| 1      | Andrew  |
| 2      | Bob     |
| 3      | Charlie |
| 4      | David   |

**RESULTS**

| reg_id | name    | log_id | name   |
|--------|---------|--------|--------|
| 1      | Andrew  | 2      | Andrew |
| 2      | Bob     | 4      | Bob    |
| 3      | Charlie | null   | null   |
| 4      | David   | null   | null   |

**LOGINS**

| log_id | name    |
|--------|---------|
| 1      | Xavier  |
| 2      | Andrew  |
| 3      | Yolanda |
| 4      | Bob     |



```
SELECT * FROM Registrations
LEFT OUTER JOIN Logins
ON Registrations.name = Logins.name
WHERE Logins.log_id IS null
```

| REGISTRATIONS |         |
|---------------|---------|
| reg_id        | name    |
| 1             | Andrew  |
| 2             | Bob     |
| 3             | Charlie |
| 4             | David   |

| RESULTS |         |        |      |
|---------|---------|--------|------|
| reg_id  | name    | log_id | name |
| 3       | Charlie | null   | null |
| 4       | David   | null   | null |

| LOGINS |         |
|--------|---------|
| log_id | name    |
| 1      | Xavier  |
| 2      | Andrew  |
| 3      | Yolanda |
| 4      | Bob     |

## 42. Left Outer Join

File ▾ Object ▾ Tools ▾ Help ▾

Dashboard Properties SQL Statistics Dependencies Dependents dvrental/postgres@PostgreSQL 12 \*

No limit

dvrental/postgres@PostgreSQL 12

Query Editor Query History Scratch P: 

```
1 SELECT film.film_id, title, inventory_id, store_id
2 FROM film
3 LEFT JOIN inventory ON
4 inventory.film_id = film.film_id
```

Data Output Explain Messages Notifications

|   | film_id<br>integer | title<br>character varying (255) | inventory_id<br>integer | store_id<br>smallint |
|---|--------------------|----------------------------------|-------------------------|----------------------|
| 1 | 1                  | Academy Dinosaur                 | 1                       | 1                    |
| 2 | 1                  | Academy Dinosaur                 | 2                       | 1                    |
| 3 | 1                  | Academy Dinosaur                 | 3                       | 1                    |
| 4 | 1                  | Academy Dinosaur                 | 4                       |                      |
|   |                    |                                  | 5                       |                      |

Successfully run. Total query runtime: 68 msec. 4623 rows affected.

1x 8:35 / 11:18 1 Academy Dinosaur

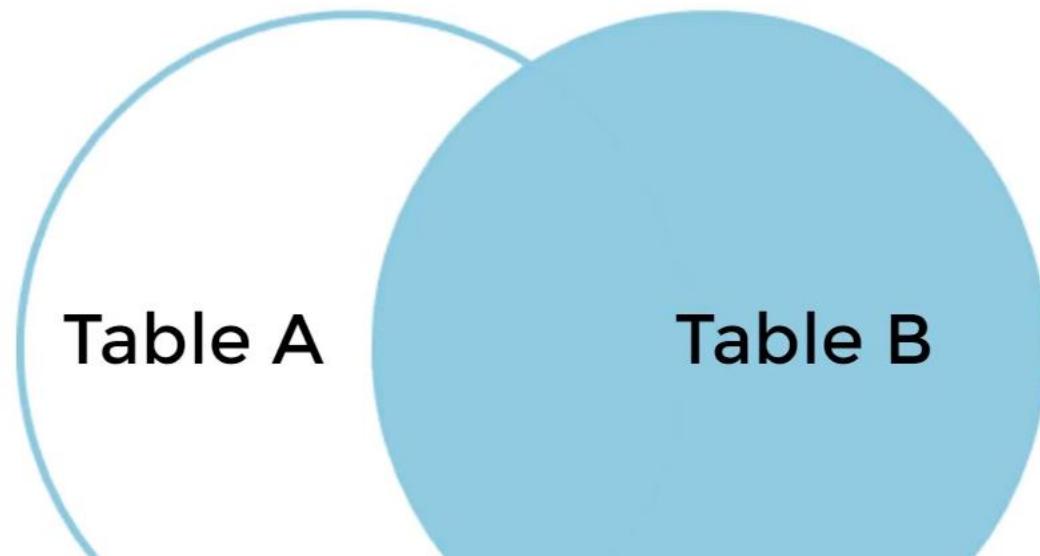


```
1 SELECT film.film_id,title,inventory_id,store_id
2 FROM film
3 LEFT JOIN inventory ON
4 inventory.film_id = film.film_id
5 WHERE inventory.film_id IS null
```

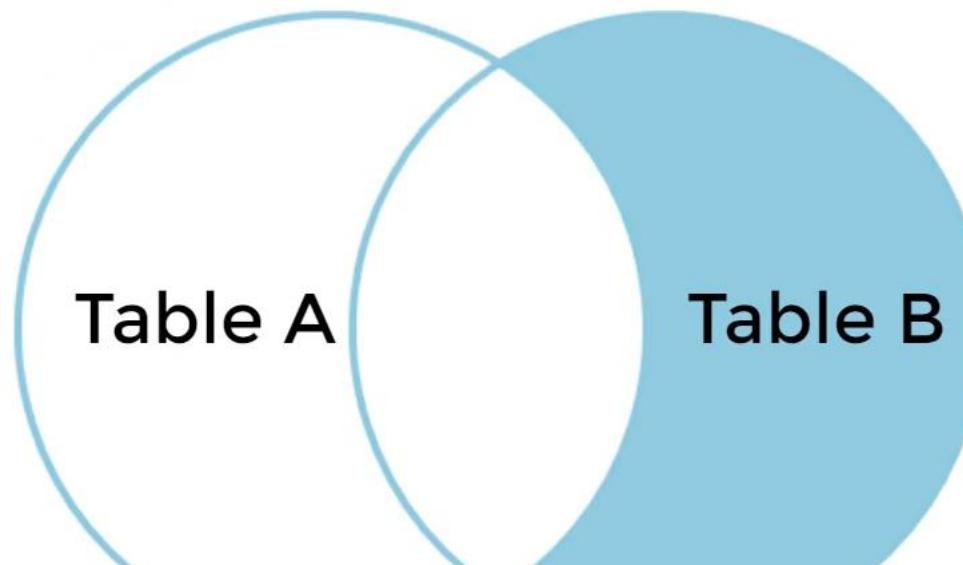
|   | film_id<br>integer | title<br>character varying (255) | inventory_id<br>integer | store_id<br>smallint |  |
|---|--------------------|----------------------------------|-------------------------|----------------------|--|
| 1 | 14                 | Alice Fantasia                   | [null]                  | [null]               |  |
| 2 | 33                 | Apollo Teen                      | [null]                  | [null]               |  |
| 3 | 36                 | Argonauts Town                   | [null]                  | [null]               |  |

- A RIGHT JOIN is essentially the same as a LEFT JOIN, except the tables are switched.
- This would be the same as switching the table order in a LEFT OUTER JOIN.
- Let's quickly see some examples of a RIGHT JOIN.

- **SELECT \* FROM TableA  
RIGHT OUTER JOIN TableB  
ON TableA.col\_match = TableB.col\_match**

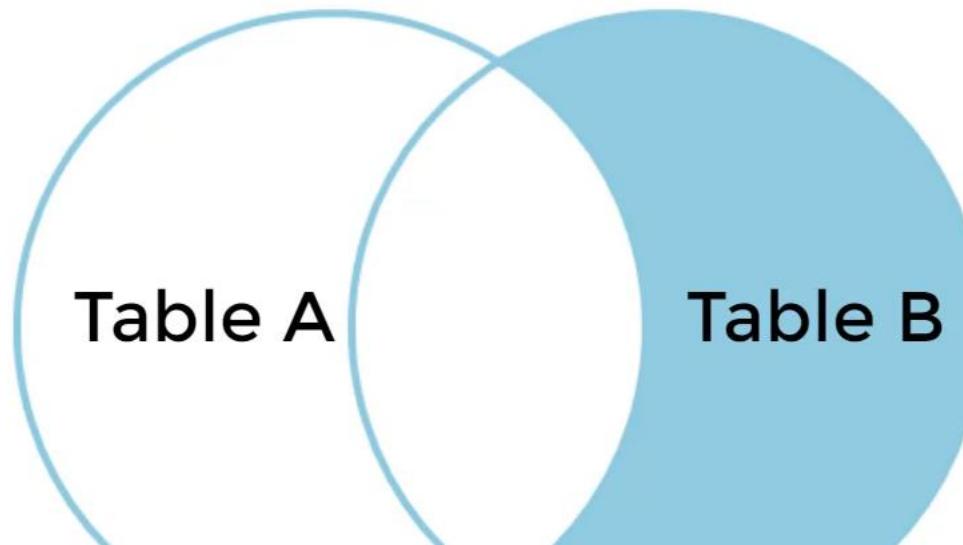


- **SELECT \* FROM TableA  
RIGHT OUTER JOIN TableB  
ON TableA.col\_match = TableB.col\_match  
WHERE TableA.id IS null**



- It is up to you and how you have the tables organized “in your mind” when it comes to choosing a LEFT vs RIGHT join, since depending on the table order you specify in the JOIN, you can perform duplicate JOINs with either method.

- **SELECT \* FROM TableA  
RIGHT OUTER JOIN TableB  
ON TableA.col\_match = TableB.col\_match  
WHERE TableA.id IS null**



# GROUP BY



# **GROUP BY**

***TO GET IN-DEPTH INFORMATION  
BY GROUP***

# **GROUP BY**

***WHAT IF WE WANTED TO KNOW  
HOW MANY EMPLOYEES WORKED IN EACH  
DEPARTMENT?***

# GROUP BY

```
SELECT dept_no, COUNT(emp_no)
FROM dept_emp;
```



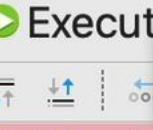
2 schemas

Execute

Execute Current

```
1| SELECT dept_no, COUNT(emp_no)
2| FROM dept_emp;
3|
```

|   | emp_no | first_name | dept_name          |
|---|--------|------------|--------------------|
| 1 | 10005  | Kyoichi    | Human Resources    |
| 2 | 10010  | Duangkaew  | Production         |
| 3 | 10010  | Duangkaew  | Quality Management |
| 4 | 10011  | Mary       | Customer Service   |
| 5 | 10018  | Kazuhide   | Production         |

2 schemas     Execute     Execute Current    Functions...    Client

! 1 `SELECT dept_no, COUNT(emp_no)`  
    ERROR: column "dept\_emp.dept\_no" must appear in the GROUP BY clause or be used in an aggregate function  
2 `FROM dept_emp;`

All    Logs (82)    Tunes (49)    Warnings    Analyzer

:time: 736 millisecond(s), Number of affected records: 331603

e.emp\_no, e.first\_name, de.dept\_no...

:time: 874 millisecond(s), Number of affected records: 331603

e.emp\_no, e.first\_name, de.dept\_no...

:time: 973 millisecond(s), Number of affected records: 331603

# **GROUP BY**

**THEY AGGREGATE RIGHT?**

**SO IF THEY'RE RETURNING 1 RECORD**

**WHAT IS GROUP BY FOR?**



# **GROUP BY**

***GROUP BY SPLITS DATA INTO GROUPS  
OR CHUNKS SO WE CAN APPLY FUNCTIONS  
AGAINST THE GROUP RATHER THAN THE ENTIRE  
TABLE***

2 schemas

Execute

Execute Current

```
1 | SELECT dept_no
2 | FROM dept_emp
3 | ORDER BY dept_no
4 | -- GROUP BY dept_no.
```

dept\_no

|     | dept_no |
|-----|---------|
| 185 | d001    |
| 178 | d001    |
| 186 | d001    |
| 179 | d001    |
| 187 | d001    |
| 180 | d001    |
| 188 | d001    |
| 181 | d001    |
| 189 | d001    |
| 182 | d001    |
| 190 | d001    |
| 183 | d001    |
| 191 | d001    |
| 184 | d001    |
| 192 | d001    |

Number of records: 331603 Number of fields: 1 Query time: 224 millisecond(s)  Read-Only

**SELECT dept\_no FROM dept\_emp ORDER by dept\_no -- GROUP BY**

All

Logs (86)

Tunes (52)

Warnings

Analyzer

2 schemas

Execute

Execute Current



```
1 | SELECT dept_no
2 | FROM dept_emp
3 | GROUP BY dept_no;
```

dept\_no

1 d001

2 d002

3 d003

4 d004

5 d005

6 d006

7 d007

8 d008

Number of records: 9 Number of fields: 1 Query time: 98 millisecond(s)  Read-Only



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
  - > Aggregates
  - > Collations
  - > Domains
  - > FTS Configurations
  - > FTS Dictionaries
  - > FTS Parsers
  - > FTS Templates
  - > Foreign Tables
  - > Functions
  - > Materialized Views
  - > Operators
  - > Procedures
  - > Sequences
- > Tables (7)
  - > departments
  - > dept\_emp

## Columns (4)

- emp\_no
  - dept\_no
  - from\_date
  - to\_date
- > Constraints
- > Indexes

```
1 SELECT dept_no,emp_no
2 FROM dept_emp
3 GROUP BY dept_no
```

ERROR: column "dept\_emp.emp\_no" must appear in the GROUP BY clause or be used in an aggregate function

LINE 1: SELECT dept\_no,emp\_no

^

SQL state: 42803

Character: 16

- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
  - > Aggregates
  - > Collations
  - > Domains
  - > FTS Configurations
  - > FTS Dictionaries
  - > FTS Parsers
  - > FTS Templates
  - > Foreign Tables
  - > Functions
  - > Materialized Views
  - > Operators
  - > Procedures
  - > Sequences

- > Tables (7)

- > departments
- > dept\_emp

- > Columns (4)

- emp\_no
- dept\_no
- from\_date
- to\_date

- > Constraints

- > Indexes

Query Editor   Query History

```
1 SELECT dept_no,emp_no
2 FROM dept_emp
3 GROUP BY dept_no,emp_no
4 ORDER BY dept_no
```

Data Output   Explain   Messages   Notifications

|    | dept_no<br>[PK] character (4) | emp_no<br>[PK] integer |
|----|-------------------------------|------------------------|
| 1  | d001                          | 10017                  |
| 2  | d001                          | 10055                  |
| 3  | d001                          | 10058                  |
| 4  | d001                          | 10108                  |
| 5  | d001                          | 10140                  |
| 6  | d001                          | 10175                  |
| 7  | d001                          | 10208                  |
| 8  | d001                          | 10228                  |
| 9  | d001                          | 10239                  |
| 10 | d001                          | 10259                  |



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
  - > Aggregates
  - > Collations
  - > Domains
  - > FTS Configurations
  - > FTS Dictionaries
  - > FTS Parsers
  - > FTS Templates
  - > Foreign Tables
  - > Functions
  - > Materialized Views

- > Operators
- > Procedures
- > Sequences
- > Tables (7)
  - > departments
  - > dept\_emp

↳ Columns (4)

- emp\_no
- dept\_no
- from\_date
- to\_date

- > Constraints
- > Indexes

```
1 SELECT dept_no, count(emp_no)
2 FROM dept_emp
3 GROUP BY dept_no
4
```

|   | dept_no<br>character (4) | count<br>bigint |
|---|--------------------------|-----------------|
| 1 | d001                     | 20211           |
| 2 | d002                     | 17346           |
| 3 | d003                     | 17786           |
| 4 | d004                     | 73485           |
| 5 | d005                     | 85707           |
| 6 | d006                     | 20117           |
| 7 | d007                     | 52245           |
| 8 | d008                     | 21126           |
| 9 | d009                     | 23580           |

# **GROUP BY**

***GROUP BY SPLITS DATA INTO GROUPS  
OR CHUNKS SO WE CAN APPLY FUNCTIONS  
AGAINST THE GROUP RATHER THAN THE ENTIRE  
TABLE***

# **GROUP BY**

***WE USE GROUP BY ALMOST EXCLUSIVELY  
WITH AGGREGATE FUNCTIONS***

# **GROUP BY**

***WHEN WE “GROUP BY” WE APPLY THE  
FUNCTION PER GROUP, NOT ON THE ENTIRE  
DATA SET***

## **THINGS TO REMEMBER**

***GROUP BY IS STRICTER THAN IT LOOKS***

# THINGS TO REMEMBER

***EVERY COLUMN NOT IN THE GROUP-BY CLAUSE  
MUST APPLY A FUNCTION.***

pgAdmin File ▾ Object ▾ Tools ▾ Help ▾

Browser

- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
  - > public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > Sequences
  - > Tables (7)
    - > departments
    - > dept\_emp
      - > Columns (4)
        - emp\_no
        - dept\_no
        - from\_date
        - to\_date
      - > Constraints
      - > Indexes

data1/postgres@PostgreSQL 14 ▾

Query Editor Query History

```
1 SELECT dept_no,emp_no
2 FROM dept_emp
3 GROUP BY dept_no
4
```

Data Output Explain Messages Notifications

ERROR: column "dept\_emp.emp\_no" must appear in the GROUP BY clause or be used in an aggregate function

LINE 1: SELECT dept\_no,emp\_no

^

SQL state: 42803

Character: 16

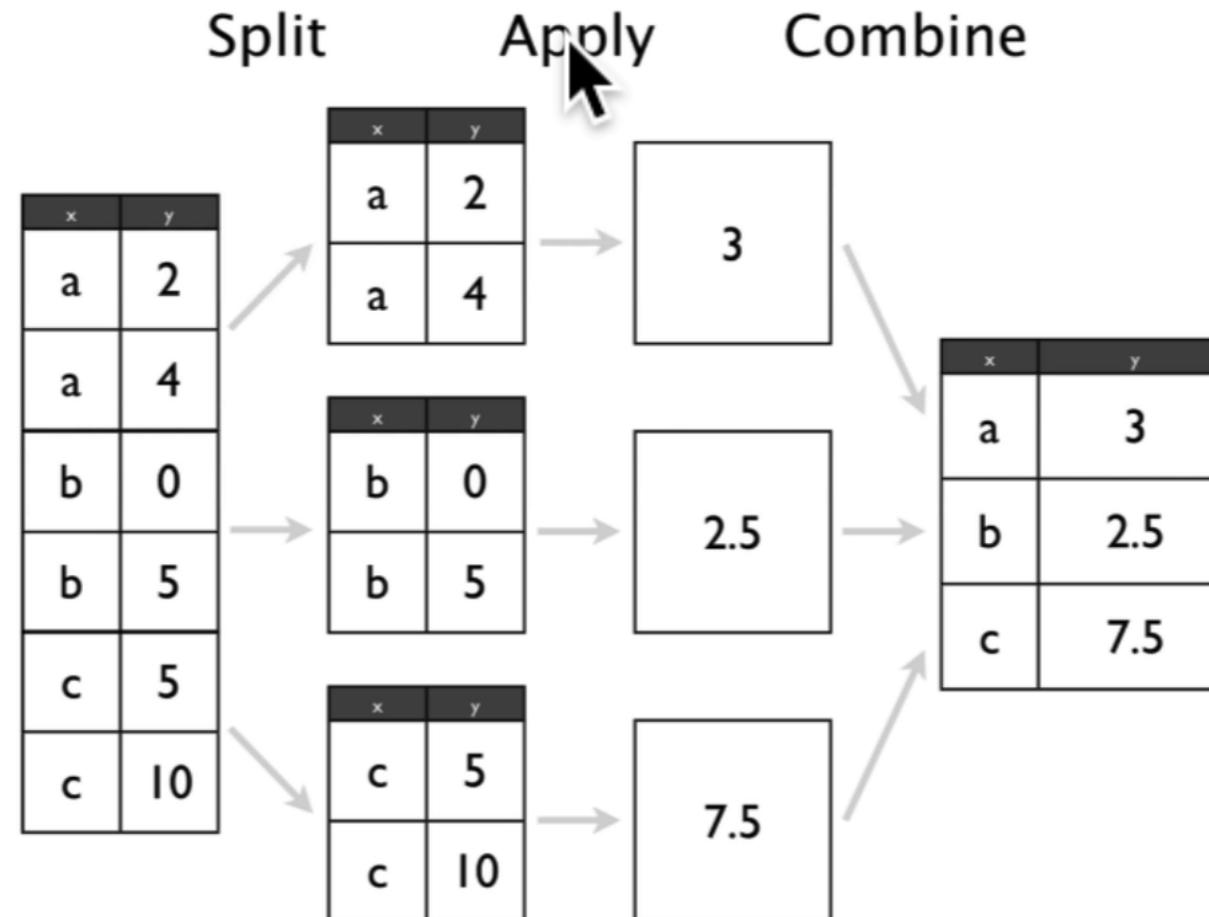
**WHY?**

***TO REDUCE ALL RECORDS FOUND FOR THE  
MATCHING “GROUP” TO A SINGLE RECORD.***

# **HOW DOES IT WORK?**

**GROUP BY UTILIZES A SPLIT-APPLY-COMBINE  
STRATEGY**

# SPLIT-APPLY-COMBINE



# **SPLIT PHASE**

**DIVIDE INTO GROUPS WITH VALUES**

# **COMBINE PHASE**

**COMBINES GROUPS WITH A SINGLE VALUE  
INTO SINGLE VALUE**

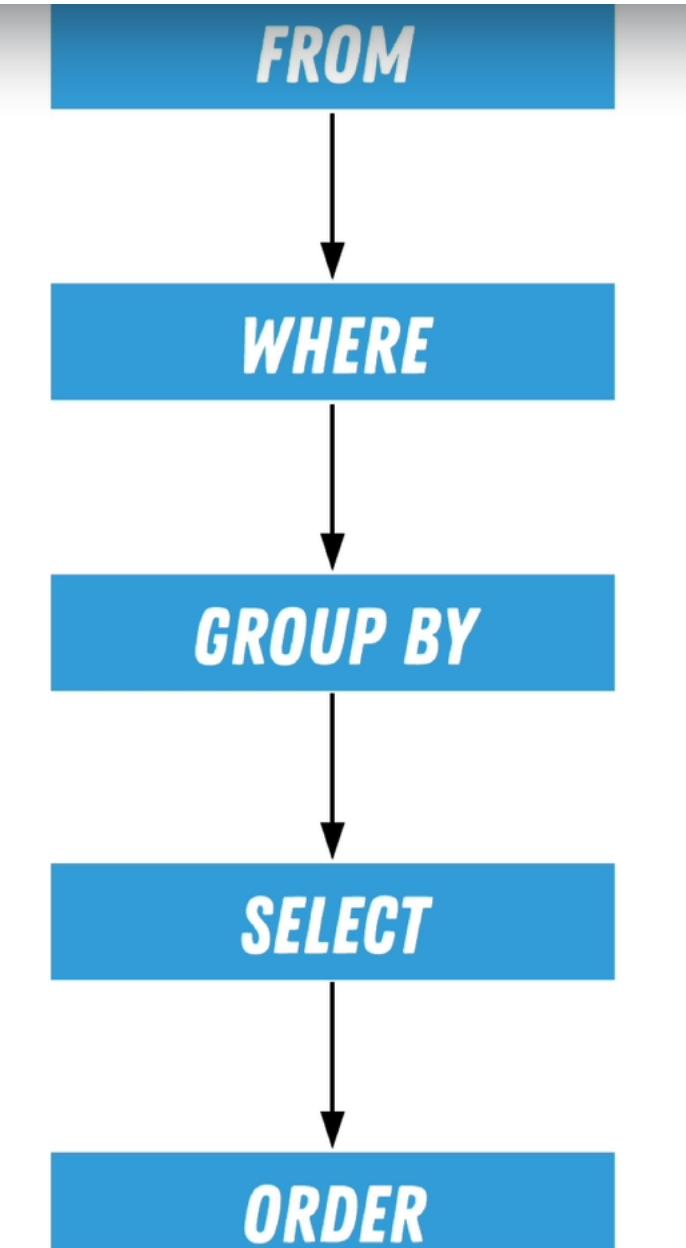
# **COMBINE PHASE**

**COMBINES GROUPS WITH A SINGLE VALUE  
INTO SINGLE VALUE**

# **ORDER OF OPERATIONS**

**GROUP BY**  
**HAPPENS AFTER**

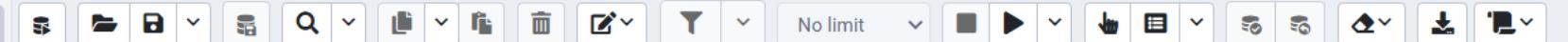
**WHERE/FROM**





Servers (2)

- PostgreSQL 14
  - Databases (11)
    - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (7)
        - departments
        - dept\_emp
        - Columns (4)



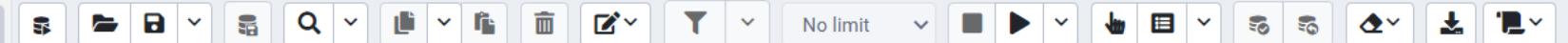
```
1 SELECT emp_no,salary from_date
2 FROM salaries
3
4
```

|    | emp_no  | from_date |
|----|---------|-----------|
|    | integer | integer   |
| 1  | 10001   | 60117     |
| 2  | 10001   | 62102     |
| 3  | 10001   | 66074     |
| 4  | 10001   | 66596     |
| 5  | 10001   | 66961     |
| 6  | 10001   | 71046     |
| 7  | 10001   | 74333     |
| 8  | 10001   | 75286     |
| 9  | 10001   | 75994     |
| 10 | 10001   | 76884     |



Servers (2)

- PostgreSQL 14
  - Databases (11)
    - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (7)
        - departments
        - dept\_emp



```
1 SELECT emp_no,MAX(from_date)
2 FROM salaries
3 GROUP BY emp_no
4
5
```

|    | emp_no | integer | max | date       |
|----|--------|---------|-----|------------|
| 1  |        | 10001   |     | 2002-06-22 |
| 2  |        | 10002   |     | 2001-08-02 |
| 3  |        | 10003   |     | 2001-12-01 |
| 4  |        | 10004   |     | 2001-11-27 |
| 5  |        | 10005   |     | 2001-09-09 |
| 6  |        | 10006   |     | 2001-08-02 |
| 7  |        | 10007   |     | 2002-02-07 |
| 8  |        | 10008   |     | 2000-03-10 |
| 9  |        | 10009   |     | 2002-02-14 |
| 10 |        | 10010   |     | 2001-11-23 |

| emp_no | first_name | last_name | from_date  |
|--------|------------|-----------|------------|
| 1      | Name       | Name      | 1980-01-01 |
| 1      | Name       | Name      | 2002-01-01 |
| 3      | Name       | Name      | 1980-01-01 |
| 3      | Name       | Name      | 2000-01-01 |
| 5      | Name       | Name      | 1980-01-01 |
| 5      | Name       | Name      | 1999-01-01 |

| emp_no | first_name |
|--------|------------|
| 1      | Name       |
| 1      | Name       |
| 3      | Name       |
| 3      | Name       |
| 5      | Name       |
| 5      | Name       |



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
- data1
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
    - Tables (7)
      - departments
      - dept\_emp
      - Columns (4)

data1/postgres@PostgreSQL 14 \*

Query Editor Query History

```
1 SELECT emp_no,from_date
2 FROM salaries
3 GROUP BY emp_no,from_date
4
5
```

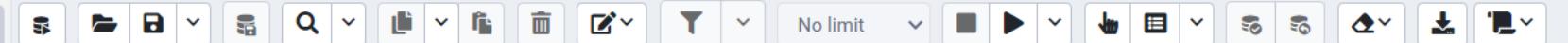
Data Output Explain Messages Notifications

|    | emp_no<br>[PK] integer | from_date<br>[PK] date |
|----|------------------------|------------------------|
| 1  | 10001                  | 1986-06-26             |
| 2  | 10001                  | 1987-06-26             |
| 3  | 10001                  | 1988-06-25             |
| 4  | 10001                  | 1989-06-25             |
| 5  | 10001                  | 1990-06-25             |
| 6  | 10001                  | 1991-06-25             |
| 7  | 10001                  | 1992-06-24             |
| 8  | 10001                  | 1993-06-24             |
| 9  | 10001                  | 1994-06-24             |
| 10 | 10001                  | 1995-06-24             |



Servers (2)

- PostgreSQL 14
  - Databases (11)
    - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (7)
        - departments
        - dept\_emp
        - Columns (4)



```
1 SELECT emp_no,MAX(from_date)
2 FROM salaries
3 GROUP BY emp_no,from_date
4
5
```

|    | emp_no<br>integer | max<br>date |
|----|-------------------|-------------|
| 1  | 10001             | 1986-06-26  |
| 2  | 10001             | 1987-06-26  |
| 3  | 10001             | 1988-06-25  |
| 4  | 10001             | 1989-06-25  |
| 5  | 10001             | 1990-06-25  |
| 6  | 10001             | 1991-06-25  |
| 7  | 10001             | 1992-06-24  |
| 8  | 10001             | 1993-06-24  |
| 9  | 10001             | 1994-06-24  |
| 10 | 10001             | 1995-06-24  |

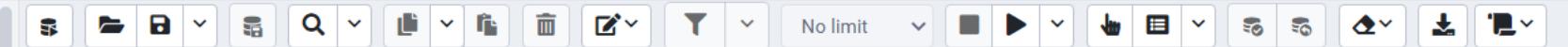


| emp_no | first_name | last_name | from_date  | salary |
|--------|------------|-----------|------------|--------|
| 1      | Name       | Name      | 1980-01-01 | 88000  |
| 1      | Name       | Name      | 2002-01-01 | 99000  |
| 3      | Name       | Name      | 1980-01-01 | 90000  |
| 3      | Name       | Name      | 2000-01-01 | 70000  |
| 5      | Name       | Name      | 1980-01-01 | 50000  |
| 5      | Name       | Name      | 1999-01-01 | 60000  |



Servers (2)

- PostgreSQL 14
  - Databases (11)
    - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (7)
        - departments
        - dept\_emp
        - Columns (4)



No limit

## Query Editor Query History

```
1 SELECT emp_no,MAX(from_date),MAX(salary)
2 FROM salaries
3 GROUP BY emp_no,from_date
4
5
```

## Data Output Explain Messages Notifications

|    | emp_no<br>integer | max<br>date | max<br>integer |
|----|-------------------|-------------|----------------|
| 1  | 10001             | 1986-06-26  | 60117          |
| 2  | 10001             | 1987-06-26  | 62102          |
| 3  | 10001             | 1988-06-25  | 66074          |
| 4  | 10001             | 1989-06-25  | 66596          |
| 5  | 10001             | 1990-06-25  | 66961          |
| 6  | 10001             | 1991-06-25  | 71046          |
| 7  | 10001             | 1992-06-24  | 74333          |
| 8  | 10001             | 1993-06-24  | 75286          |
| 9  | 10001             | 1994-06-24  | 75994          |
| 10 | 10001             | 1995-06-24  | 76884          |

- --How many people were hired on did we hire on any given hire date?
- --Database: Employees
- SELECT hire\_date, COUNT(emp\_no) as "amount"
- FROM employees
- GROUP BY hire\_date
- ORDER BY "amount" DESC;



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
  - ✓ public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > Sequences
  - > Tables (7)
    - > departments
    - > dept\_emp
      - > Columns (4)
        - emp\_no
        - dept\_no
        - from\_date
        - to\_date
      - > Constraints
      - > Indexes

## Query Editor Query History

```
1 --How many people were hired on did we hire on any given hire date?
2 --Database: Employees
3 SELECT hire_date, COUNT(emp_no) as "amount"
4 FROM employees
5 GROUP BY hire_date
6 ORDER BY "amount" DESC;
```

## Data Output Explain Messages Notifications

|    | hire_date  | amount |
|----|------------|--------|
|    | date       | bigint |
| 1  | 1985-06-20 | 132    |
| 2  | 1985-03-21 | 131    |
| 3  | 1985-03-24 | 128    |
| 4  | 1985-08-08 | 128    |
| 5  | 1985-05-11 | 127    |
| 6  | 1985-12-12 | 127    |
| 7  | 1985-05-24 | 127    |
| 8  | 1985-03-07 | 126    |
| 9  | 1985-09-23 | 126    |
| 10 | 1985-03-30 | 126    |

- --Show me all the employees, hired after 1991 and count the amount of positions they've had
- --Database: Employees
- SELECT e.emp\_no, count(t.title) as "amount of titles"
- FROM employees as e
- JOIN titles as t USING(emp\_no)
- WHERE EXTRACT (YEAR FROM e.hire\_date) > 1991
- GROUP BY e.emp\_no
- ORDER BY e.emp\_no;



- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)

- > public
  - > Aggregates
  - > Collations
  - > Domains
  - > FTS Configurations
  - > FTS Dictionaries
  - > FTS Parsers
  - > FTS Templates
  - > Foreign Tables
  - > Functions
  - > Materialized Views
  - > Operators
  - > Procedures
  - > Sequences
- > Tables (7)
  - > departments
  - > dept\_emp
- > Columns (4)

- emp\_no
  - dept\_no
  - from\_date
  - to\_date
- > Constraints
- > Indexes

```
1 --Show me all the employees, hired after 1991 and count the amount of positions they've had
2 --Database: Employees
3 SELECT e.emp_no, count(t.title) as "amount of titles"
4 FROM employees as e
5 JOIN titles as t USING(emp_no)
6 WHERE EXTRACT (YEAR FROM e.hire_date) > 1991
7 GROUP BY e.emp_no
8 ORDER BY e.emp_no;
```

|    | emp_no<br>[PK] integer | amount of titles<br>bigint |
|----|------------------------|----------------------------|
| 1  | 10008                  | 1                          |
| 2  | 10012                  | 2                          |
| 3  | 10016                  | 1                          |
| 4  | 10017                  | 2                          |
| 5  | 10019                  | 1                          |
| 6  | 10022                  | 1                          |
| 7  | 10024                  | 1                          |
| 8  | 10026                  | 2                          |
| 9  | 10030                  | 2                          |
| 10 | 10036                  | 1                          |

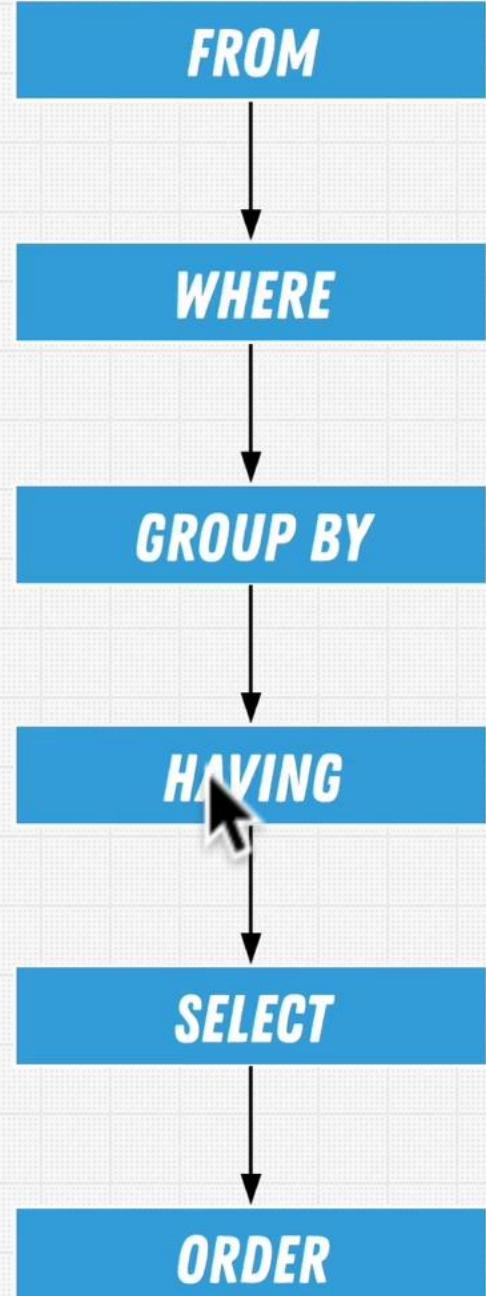
- -- Show me all the employees that work in the department d005 development and the from and to date.
- -- Database: Employees
- SELECT e.emp\_no, de.from\_date, de.to\_date
- FROM employees as e
- JOIN dept\_emp AS de USING(emp\_no)
- WHERE de.dept\_no = 'd005'
- GROUP BY e.emp\_no, de.from\_date, de.to\_date
- ORDER BY e.emp\_no, de.to\_date;

**HAVING**

***WHAT IF I WANT TO FILTER  
GROUPS?***

# HAVING

```
SELECT col1, COUNT(col2)
FROM <table>
WHERE col2 > X
GROUP BY col1
HAVING col1 === Y;
```



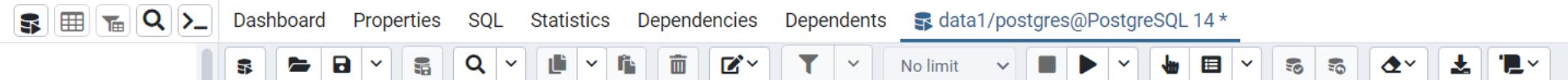
# HAVING

***“WHERE” APPLIES FILTERS  
TO INDIVIDUAL ROWS***

# **HAVING**

**“HAVING” APPLIES FILTERS  
TO A GROUP AS A WHOLE**

- SELECT d.dept\_name,COUNT(e.emp\_no)
- FROM employee AS e
- INNER JOIN dept\_emp AS de ON de.emp\_no=e.emp\_no
- INNER JOIN departments AS d ON de.dept\_no=d.dept\_no
- GROUP BY d.dept\_name



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
- data1
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
    - Tables (7)
      - departments
      - dept\_emp

## Query Editor Query History

```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name
6
```

## Data Output Explain Messages Notifications

|   | dept_name          | # of employees |
|---|--------------------|----------------|
| 1 | Customer Service   | 23580          |
| 2 | Development        | 85707          |
| 3 | Finance            | 17346          |
| 4 | Human Resources    | 17786          |
| 5 | Marketing          | 20211          |
| 6 | Production         | 73485          |
| 7 | Quality Management | 20117          |
| 8 | Research           | 21126          |
| 9 | Sales              | 52245          |



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
- data1
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
    - Tables (7)
      - departments
      - dept\_emp

data1/postgres@PostgreSQL 14 \*

Query Editor Query History

```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name
6 HAVING COUNT(e.emp_no)>25000
7
```

Data Output Explain Messages Notifications

|   | dept_name   | # of employees |
|---|-------------|----------------|
| 1 | Development | 85707          |
| 2 | Production  | 73485          |
| 3 | Sales       | 52245          |



## Servers (2)

PostgreSQL 14

Databases (11)

advanced-SQL

data1

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (7)

departments

dept\_emp

```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name
6 HAVING COUNT(e.emp_no)>25000 AND e.gender='F'
```

7

8

9

ERROR: column "e.gender" must appear in the GROUP BY clause or be used in an aggregate function  
LINE 6: HAVING COUNT(e.emp\_no)>25000 AND e.gender='F'  
^

SQL state: 42803

Character: 230

pgAdmin File Object Tools Help

Browser

Servers (2)  
PostgreSQL 14  
Databases (11)  
advanced-SQL  
data1  
Casts  
Catalogs  
Event Triggers  
Extensions  
Foreign Data Wrappers  
Languages  
Publications  
Schemas (1)  
public  
Aggregates  
Collations  
Domains  
FTS Configurations  
FTS Dictionaries  
FTS Parsers  
FTS Templates  
Foreign Tables  
Functions  
Materialized Views  
Operators  
Procedures  
Sequences  
Tables (7)  
departments  
dept\_emp  
Other (1)

Dashboard Properties SQL Statistics Dependencies Dependents data1/postgres@PostgreSQL 14 \*

No limit

Query Editor Query History

```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name,e.gender
6 HAVING COUNT(e.emp_no)>25000 AND e.gender='F'
```

Data Output Explain Messages Notifications

|   | dept_name   | # of employees |
|---|-------------|----------------|
| 1 | Development | 34258          |
| 2 | Production  | 29549          |



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
- data1
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
    - Tables (7)
      - departments
      - dept\_emp

data1/postgres@PostgreSQL 14

Query Editor Query History

```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 WHERE e.gender='F'
6 GROUP BY d.dept_name,e.gender
7 HAVING COUNT(e.emp_no)>25000
8
9
10
```

Data Output Explain Messages Notifications

|   | dept_name   | # of employees |
|---|-------------|----------------|
| 1 | Development | 34258          |
| 2 | Production  | 29549          |

- --Show me all the employees, hired after 1991, that have had more than 2 titles
- SELECT e.emp\_no, count(t.title) as "amount of titles"
- FROM employees as e
- JOIN titles as t USING(emp\_no)
- WHERE EXTRACT (YEAR FROM e.hire\_date) > 1991
- GROUP BY e.emp\_no
- HAVING count(t.title) > 2
- ORDER BY e.emp\_no;

pgAdmin File Object Tools Help

Browser

Servers (2)  
PostgreSQL 14  
Databases (11)  
advanced-SQL  
data1  
Casts  
Catalogs  
Event Triggers  
Extensions  
Foreign Data Wrappers  
Languages  
Publications  
Schemas (1)  
public  
Aggregates  
Collations  
Domains  
FTS Configurations  
FTS Dictionaries  
FTS Parsers  
FTS Templates  
Foreign Tables  
Functions  
Materialized Views  
Operators  
Procedures  
Sequences  
Tables (7)  
departments  
dept\_emp  
Columns (4)

Dashboard Properties SQL Statistics Dependencies Dependents data1/postgres@PostgreSQL 14 \*

data1/postgres@PostgreSQL 14

Query Editor Query History

```
1 --Show me all the employees, hired after 1991, that have had more than 2 titles
2 SELECT e.emp_no, count(t.title) as "amount of titles"
3 FROM employees as e
4 JOIN titles as t USING(emp_no)
5 WHERE EXTRACT (YEAR FROM e.hire_date) > 1991
6 GROUP BY e.emp_no
7 HAVING count(t.title) > 2
8 ORDER BY e.emp_no;
```

Data Output Explain Messages Notifications

|    | emp_no<br>[PK] integer | amount of titles<br>bigint |
|----|------------------------|----------------------------|
| 1  | 13251                  | 3                          |
| 2  | 21695                  | 3                          |
| 3  | 25429                  | 3                          |
| 4  | 27121                  | 3                          |
| 5  | 27803                  | 3                          |
| 6  | 29106                  | 3                          |
| 7  | 30222                  | 3                          |
| 8  | 32507                  | 3                          |
| 9  | 33040                  | 3                          |
| 10 | 46776                  | 3                          |

- --Show me all the employees that have had more than 15 salary changes that work in the department development
- SELECT e.emp\_no, count(s.from\_date) as "amount of raises"
- FROM employees as e
- JOIN salaries as s USING(emp\_no)
- JOIN dept\_emp AS de USING(emp\_no)
- WHERE de.dept\_no = 'd005'
- GROUP BY e.emp\_no
- HAVING count(s.from\_date) > 15
- ORDER BY e.emp\_no;

Servers (2)

- PostgreSQL 14
  - Databases (11)
    - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (7)
        - departments
        - dept\_emp

data1/postgres@PostgreSQL 14

Query Editor Query History

```
1 --Show me all the employees that have had more than 15 salary changes that work in the department development
2 SELECT e.emp_no, count(s.from_date) as "amount of raises"
3 FROM employees as e
4 JOIN salaries as s USING(emp_no)
5 JOIN dept_emp AS de USING(emp_no)
6 WHERE de.dept_no = 'd005'
7 GROUP BY e.emp_no
8 HAVING count(s.from_date) > 15
9 ORDER BY e.emp_no;
```

10  
11  
12  
13  
14

Data Output Explain Messages Notifications

|    | emp_no<br>[PK] integer | amount of raises<br>bigint |
|----|------------------------|----------------------------|
| 1  | 10001                  | 17                         |
| 2  | 10018                  | 16                         |
| 3  | 10066                  | 17                         |
| 4  | 10070                  | 17                         |
| 5  | 10150                  | 16                         |
| 6  | 10191                  | 17                         |
| 7  | 10198                  | 17                         |
| 8  | 10258                  | 18                         |
| 9  | 10271                  | 17                         |
| 10 | 10318                  | 16                         |

- --Show me all the employees that have worked for multiple departments
- SELECT e.emp\_no, count(de.dept\_no) as "worked for # departments"
- FROM employees as e
- JOIN dept\_emp AS de USING(emp\_no)
- GROUP BY e.emp\_no
- HAVING count(de.dept\_no) > 1
- ORDER BY e.emp\_no;



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
- data1
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
    - Tables (7)
      - departments
      - dept\_emp

data1/postgres@PostgreSQL 14 \*

Query Editor Query History

```
1 --Show me all the employees that have worked for multiple departments
2 SELECT e.emp_no, count(de.dept_no) as "worked for # departments"
3 FROM employees as e
4 JOIN dept_emp AS de USING(emp_no)
5 GROUP BY e.emp_no
6 HAVING count(de.dept_no) > 1
7 ORDER BY e.emp_no;
```

Data Output Explain Messages Notifications

|    | emp_no<br>[PK] integer | worked for # departments<br>bigint |
|----|------------------------|------------------------------------|
| 1  | 10010                  | 2                                  |
| 2  | 10018                  | 2                                  |
| 3  | 10029                  | 2                                  |
| 4  | 10040                  | 2                                  |
| 5  | 10050                  | 2                                  |
| 6  | 10060                  | 2                                  |
| 7  | 10070                  | 2                                  |
| 8  | 10080                  | 2                                  |
| 9  | 10088                  | 2                                  |
| 10 | 10098                  | 2                                  |

# Ordering grouped data

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

Servers (2)  
PostgreSQL 14  
Databases (11)  
advanced-SQL  
data1  
Casts  
Catalogs  
Event Triggers  
Extensions  
Foreign Data Wrappers  
Languages  
Publications  
Schemas (1)  
public  
Aggregates  
Collations  
Domains  
FTS Configurations  
FTS Dictionaries  
FTS Parsers  
FTS Templates  
Foreign Tables  
Functions  
Materialized Views  
Operators  
Procedures  
Sequences  
Tables (7)  
departments  
dept\_emp  
Columns (4)

Query Editor Query History

```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name
6 ORDER BY dept_name
7
8
```

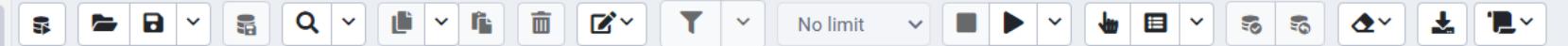
Data Output Explain Messages Notifications

| dept_name          | # of employees |
|--------------------|----------------|
| Customer Service   | 23580          |
| Development        | 85707          |
| Finance            | 17346          |
| Human Resources    | 17786          |
| Marketing          | 20211          |
| Production         | 73485          |
| Quality Management | 20117          |
| Research           | 21126          |
| Sales              | 52245          |



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (7)
        - departments
        - dept\_emp
        - employees
        - functions
        - languages
        - operator\_classes
        - operator\_inherit



```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name
6 ORDER BY dept_name,COUNT(e.emp_no)
7
8
```

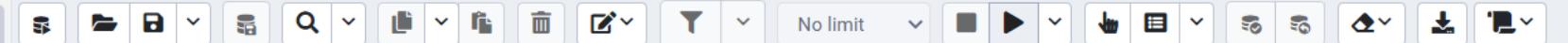
|   | dept_name        | # of employees |
|---|------------------|----------------|
| 1 | Customer Service | 23580          |

|   |                    |       |
|---|--------------------|-------|
| 1 | Customer Service   | 23580 |
| 2 | Development        | 85707 |
| 3 | Finance            | 17346 |
| 4 | Human Resources    | 17786 |
| 5 | Marketing          | 20211 |
| 6 | Production         | 73485 |
| 7 | Quality Management | 20117 |
| 8 | Research           | 21126 |
| 9 | Sales              | 52245 |



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (7)
        - departments
        - dept\_emp
        - employees
        - functions
        - languages
        - operator\_classes
        - operator\_inherit



```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name
6 ORDER BY COUNT(e.emp_no)
7
8
```

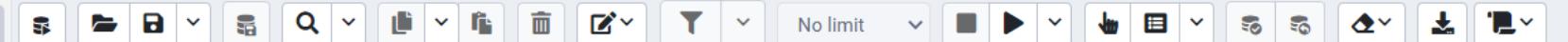
|   | dept_name              | # of employees |
|---|------------------------|----------------|
| 1 | character varying (40) | bigint         |

|   |                    |       |
|---|--------------------|-------|
| 1 | Finance            | 17346 |
| 2 | Human Resources    | 17786 |
| 3 | Quality Management | 20117 |
| 4 | Marketing          | 20211 |
| 5 | Research           | 21126 |
| 6 | Customer Service   | 23580 |
| 7 | Sales              | 52245 |
| 8 | Production         | 73485 |
| 9 | Development        | 85707 |



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
- data1
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
    - Tables (7)
      - departments
      - dept\_emp
      - employees
      - jobs
      - locations
      - regions
      - titles



No limit

Query Editor Query History

```
1 SELECT d.dept_name,COUNT(e.emp_no) AS "# of employees"
2 FROM employees AS e
3 INNER JOIN dept_emp AS de ON de.emp_no=e.emp_no
4 INNER JOIN departments AS d ON de.dept_no=d.dept_no
5 GROUP BY d.dept_name
6 ORDER BY COUNT(e.emp_no) DESC
7
8
```

Data Output Explain Messages Notifications

|  | dept_name | # of employees |
|--|-----------|----------------|
|--|-----------|----------------|

character varying (40) bigint

|   |                    |       |
|---|--------------------|-------|
| 1 | Development        | 85707 |
| 2 | Production         | 73485 |
| 3 | Sales              | 52245 |
| 4 | Customer Service   | 23580 |
| 5 | Research           | 21126 |
| 6 | Marketing          | 20211 |
| 7 | Quality Management | 20117 |
| 8 | Human Resources    | 17786 |
| 9 | Finance            | 17346 |

# QUESTION

***WHAT IF WE WANT TO COMBINE  
THE RESULTS OF MULTIPLE  
GROUPINGS?***

# UNION



```
SELECT col1, SUM(col2)
FROM table
GROUP BY col1
```

UNION

```
SELECT SUM(col2)
FROM table
```

# UNION ALL



```
SELECT col1, SUM(col2)
FROM table
GROUP BY col1
```

UNION ALL

```
SELECT SUM(col2)
FROM table
```

# **THE DIFFERENCE**

***UNION ALL DOES NOT REMOVE  
DUPLICATE RECORDS***

```
1 SELECT NULL AS "prod_id", sum(ol.quantity)
2 FROM orderlines AS ol
3
4 UNION
5
6 SELECT prod_id AS "prod_id" , sum(ol.quantity)
7 FROM orderlines AS ol
8 GROUP BY prod_id
9 ORDER BY prod_id DESC;
10
```

|   | prod_id | sum    |
|---|---------|--------|
| 1 | <NULL>  | 120719 |
| 2 | 10000   | 9      |
| 3 | 9999    | 13     |

```
1| SELECT NULL AS "prod_id", sum(ol.quantity)
2| FROM orderlines AS ol
3|
4| UNION ALL
5|
6| SELECT prod_id AS "prod_id" , sum(ol.quantity)
7| FROM orderlines AS ol
8| GROUP BY prod_id
9| ORDER BY prod_id DESC;
10|
```

|   | prod_id | sum    |
|---|---------|--------|
| 1 | <NULL>  | 120719 |
| 2 | 10000   | 9      |
| 3 | 9999    | 13     |
| 4 | 9998    | 3      |
| 5 | 9997    | 16     |



```
1 -- SELECT NULL as "prod_id", sum(ol.quantity)
2 -- FROM orderlines As ol
3 --
4 -- union
5 --
6 SELECT prod_id AS "prod_id" , sum(ol.quantity)
7 FROM orderlines AS ol
8 GROUP BY prod_id
9 ORDER BY prod_id DESC;
```

10

|   | prod_id | sum    |
|---|---------|--------|
| 1 | <NULL>  | 120719 |
| 2 | 10000   | 9      |
| 3 | 9999    | 13     |
| 4 | 9998    | 3      |
| 5 | 9997    | 16     |



```
3 --
4 -- union
5 --
6 SELECT prod_id AS "prod_id" , sum(ol.quantity)
7 FROM orderlines AS ol
8 GROUP BY |
9 GROUPING SETS ()
10 (prod_id)
11
12)
13 ORDER BY prod_id DESC;
14
```

|   | prod_id | sum    |
|---|---------|--------|
| 1 | <NULL>  | 120719 |
| 2 | 10000   | 9      |
| 3 | 9999    | 13     |
| 4 | 9998    | 3      |
| 5 | 9997    | 16     |

```
6 SELECT prod_id AS "prod_id" , sum(ol.quantity)
7 FROM orderlines AS ol
8 GROUP BY
9 GROUPING SETS (
10 (),
11 (prod_id),
12 (orderlineid)
13)
```

|      | prod_id | sum   |
|------|---------|-------|
| 9971 | 7227    | 10    |
| 9972 | 790     | 12    |
| 9973 | 2850    | 10    |
| 9974 | 5642    | 22    |
| 9975 | <NULL>  | 537   |
| 9976 | <NULL>  | 2708  |
| 9977 | <NULL>  | 8032  |
| 9978 | <NULL>  | 23915 |
| 9979 | <NULL>  | 13544 |
| 9980 | <NULL>  | 16191 |

Number of records: 9983 Number of fields: 2 Query time: 62 millisecond(s)  Read-Only

# WHAT ARE WE MISSING?

***HOW DO WE APPLY FUNCTIONS***

***AGAINST A SET OF ROWS RELATED TO***

***THE CURRENT ROW?***

# SAY WHAT?

***WHAT IF YOU WANTED TO KNOW  
THE AVERAGE SALARY PER  
DEPARTMENT?***



# EASY PEASY



```
SELECT d.dept_name, ROUND(AVG(salary))
FROM salaries
join dept_emp as de USING(emp_no)
JOIN departments as d USING(dept_no)
GROUP by dept_no, dept_name •
```

# EASY PEASY

***ADD THE AVERAGE TO EVERY SALARY SO WE  
COULD VISUALLY SEE HOW MUCH  
EACH EMPLOYEE IS FROM THE AVERAGE.***



Not Window What...

# WHAT NOW?

# **WINDOW FUNCTIONS!**



# WHAT NOW?

**WINDOW FUNCTIONS CREATE A NEW COLUMN**

**BASED ON FUNCTIONS PERFORMED ON A SUBSET**

**OR “WINDOW” OF THE DATA**

# WHAT NOW?

```
window_function(arg1, arg2,..) OVER (
 [PARTITION BY partition_expression]
 [ORDER BY sort_expression [ASC | DESC] [NULLS {FIRST | LAST }]
```



- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (7)
- > departments
- > dept\_emp
  - > Columns (4)
    - emp\_no
    - dept\_no
    - from\_date
    - to\_date
  - > Constraints
  - > Indexes
  - > RLS Policies
  - > Rules
  - > Triggers
- > dept\_manager
- > employees
- > salaries
- > timezones
- > titles
- > Trigger Functions
- > Types
- > Views
- > Subscriptions

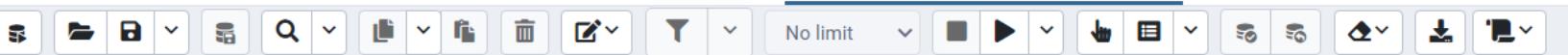


```
1 SELECT MAX(salary) FROM salaries
2
3
```

|   | max     |   |
|---|---------|---|
|   | integer | 🔒 |
| 1 | 158220  |   |



- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (7)
- > departments
- > dept\_emp
  - > Columns (4)
    - emp\_no
    - dept\_no
    - from\_date
    - to\_date
  - > Constraints
  - > Indexes
  - > RLS Policies
  - > Rules
  - > Triggers
- > dept\_manager
- > employees
- > salaries
- > timezones
- > titles
- > Trigger Functions
- > Types
- > Views
- > Subscriptions



```
1 SELECT
2 * ,
3 MAX(salary) OVER()
4 FROM salaries
5
6
```

|    | emp_no<br>[PK] integer | salary<br>integer | from_date<br>[PK] date | to_date<br>date | max<br>integer |
|----|------------------------|-------------------|------------------------|-----------------|----------------|
| 1  | 10001                  | 60117             | 1986-06-26             | 1987-06-26      | 158220         |
| 2  | 10001                  | 62102             | 1987-06-26             | 1988-06-25      | 158220         |
| 3  | 10001                  | 66074             | 1988-06-25             | 1989-06-25      | 158220         |
| 4  | 10001                  | 66596             | 1989-06-25             | 1990-06-25      | 158220         |
| 5  | 10001                  | 66961             | 1990-06-25             | 1991-06-25      | 158220         |
| 6  | 10001                  | 71046             | 1991-06-25             | 1992-06-24      | 158220         |
| 7  | 10001                  | 74333             | 1992-06-24             | 1993-06-24      | 158220         |
| 8  | 10001                  | 75286             | 1993-06-24             | 1994-06-24      | 158220         |
| 9  | 10001                  | 75994             | 1994-06-24             | 1995-06-24      | 158220         |
| 10 | 10001                  | 76884             | 1995-06-24             | 1996-06-23      | 158220         |

✓ Successfully run. Total query runtime: 4 secs 681 msec. 2844047 rows affected.

# Window of data

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (7)
- > departments
- > dept\_emp
  - > Columns (4)
    - emp\_no
    - dept\_no
    - from\_date
    - to\_date
  - > Constraints
  - > Indexes
  - > RLS Policies
  - > Rules
  - > Triggers
- > dept\_manager
- > employees
- > salaries
- > timezones
- > titles
- > Trigger Functions
- > Types
- > Views
- > Subscriptions

Dashboard Properties SQL Statistics Dependencies Dependents data1/postgres@PostgreSQL 14 \*

Query Editor Query History

```
1 SELECT
2 * ,
3 MAX(salary) OVER()
4 FROM salaries
5 WHERE salary < 70000
6
7
```

Data Output Explain Messages Notifications

|    | emp_no<br>[PK] integer | salary<br>integer | from_date<br>[PK] date | to_date<br>date | max<br>integer |
|----|------------------------|-------------------|------------------------|-----------------|----------------|
| 1  | 10001                  | 60117             | 1986-06-26             | 1987-06-26      | 69999          |
| 2  | 10001                  | 62102             | 1987-06-26             | 1988-06-25      | 69999          |
| 3  | 10001                  | 66074             | 1988-06-25             | 1989-06-25      | 69999          |
| 4  | 10001                  | 66596             | 1989-06-25             | 1990-06-25      | 69999          |
| 5  | 10001                  | 66961             | 1990-06-25             | 1991-06-25      | 69999          |
| 6  | 10002                  | 65828             | 1996-08-03             | 1997-08-03      | 69999          |
| 7  | 10002                  | 65909             | 1997-08-03             | 1998-08-03      | 69999          |
| 8  | 10002                  | 67534             | 1998-08-03             | 1999-08-03      | 69999          |
| 9  | 10002                  | 69366             | 1999-08-03             | 2000-08-02      | 69999          |
| 10 | 10003                  | 40006             | 1995-12-03             | 1996-12-02      | 69999          |

3:53 PM 7/6/2023

# SYNTAX

## PARTITION BY:

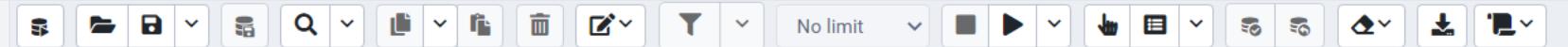
**DIVIDE ROWS INTO GROUPS TO APPLY THE**

**FUNCTION AGAINST (OPTIONAL)**

```
AVG(s.salary) OVER () as "average global salary"
```



- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (7)
- > departments
- > dept\_emp
  - > Columns (4)
    - emp\_no
    - dept\_no
    - from\_date
    - to\_date
  - > Constraints
  - > Indexes
  - > RLS Policies
  - > Rules
  - > Triggers
- > dept\_manager
- > employees
- > salaries
- > timezones
- > titles
- > Trigger Functions
- > Types
- > Views
- > Subscriptions



```
1 SELECT
2 * ,
3 AVG(salary) OVER()
4 FROM salaries
5
6
7
```

|    | emp_no<br>[PK] integer | salary<br>integer | from_date<br>[PK] date | to_date<br>date | avg<br>numeric     |
|----|------------------------|-------------------|------------------------|-----------------|--------------------|
| 1  | 10001                  | 60117             | 1986-06-26             | 1987-06-26      | 63810.744836143706 |
| 2  | 10001                  | 62102             | 1987-06-26             | 1988-06-25      | 63810.744836143706 |
| 3  | 10001                  | 66074             | 1988-06-25             | 1989-06-25      | 63810.744836143706 |
| 4  | 10001                  | 66596             | 1989-06-25             | 1990-06-25      | 63810.744836143706 |
| 5  | 10001                  | 66961             | 1990-06-25             | 1991-06-25      | 63810.744836143706 |
| 6  | 10001                  | 71046             | 1991-06-25             | 1992-06-24      | 63810.744836143706 |
| 7  | 10001                  | 74333             | 1992-06-24             | 1993-06-24      | 63810.744836143706 |
| 8  | 10001                  | 75286             | 1993-06-24             | 1994-06-24      | 63810.744836143706 |
| 9  | 10001                  | 75994             | 1994-06-24             | 1995-06-24      | 63810.744836143706 |
| 10 | 10001                  | 76884             | 1995-06-24             | 1996-06-23      | 63810.744836143706 |





```
1 SELECT
2 * ,
3 d.dept_name,
4 AVG(salary) OVER()
5
6 FROM salaries
7 JOIN dept_emp AS de USING(emp_no)
8 JOIN departments as d USING(dept_no)
9
10
```

|     | dept_no<br>character (4) | emp_no<br>integer | salary<br>integer | from_date<br>date | to_date<br>date | from_date<br>date | to_date<br>date | dept_name<br>character varying (40) | dept_name<br>character varying (40) | avg<br>numeric  |
|-----|--------------------------|-------------------|-------------------|-------------------|-----------------|-------------------|-----------------|-------------------------------------|-------------------------------------|-----------------|
| 524 | d009                     | 10817             | 54602             | 1990-12-26        | 1991-12-26      | 2000-01-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 63805.400520353 |
| 525 | d007                     | 10817             | 54602             | 1990-12-26        | 1991-12-26      | 1990-12-26        | 2000-01-24      | Sales                               | Sales                               | 63805.400520353 |
| 526 | d009                     | 10817             | 54735             | 1991-12-26        | 1992-12-25      | 2000-01-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 63805.400520353 |
| 527 | d007                     | 10817             | 54735             | 1991-12-26        | 1992-12-25      | 1990-12-26        | 2000-01-24      | Sales                               | Sales                               | 63805.400520353 |
| 528 | d009                     | 10817             | 55089             | 1992-12-25        | 1993-12-25      | 2000-01-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 63805.400520353 |
| 529 | d007                     | 10817             | 55089             | 1992-12-25        | 1993-12-25      | 1990-12-26        | 2000-01-24      | Sales                               | Sales                               | 63805.400520353 |
| 530 | d009                     | 10817             | 59517             | 1993-12-25        | 1994-12-25      | 2000-01-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 63805.400520353 |
| 531 | d007                     | 10817             | 59517             | 1993-12-25        | 1994-12-25      | 1990-12-26        | 2000-01-24      | Sales                               | Sales                               | 63805.400520353 |
| 532 | d009                     | 10817             | 61803             | 1994-12-25        | 1995-12-25      | 2000-01-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 63805.400520353 |
| 533 | d007                     | 10817             | 61803             | 1994-12-25        | 1995-12-25      | 1990-12-26        | 2000-01-24      | Sales                               | Sales                               | 63805.400520353 |



- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- Tables (7)

- > departments
- > dept\_emp
  - > Columns (4)
    - emp\_no
    - dept\_no
    - from\_date
    - to\_date

- > Constraints
- > Indexes
- > RLS Policies
- > Rules
- > Triggers
- > dept\_manager
- > employees
- > salaries
- > timezones
- > titles

- > Trigger Functions
- > Types
- > Views
- > Subscriptions

```
1 SELECT
2 * ,
3 d.dept_name,
4 AVG(salary) OVER(
5 PARTITION BY d.dept_name
6)
7
8 FROM salaries
9 JOIN dept_emp AS de USING(emp_no)
10 JOIN departments as d USING(dept_no)
11
12
```

|       | dept_no<br>character (4) | emp_no<br>integer | salary<br>integer | from_date<br>date | to_date<br>date | from_date<br>date | to_date<br>date | dept_name<br>character varying (40) | dept_name<br>character varying (40) | avg<br>numeric  |
|-------|--------------------------|-------------------|-------------------|-------------------|-----------------|-------------------|-----------------|-------------------------------------|-------------------------------------|-----------------|
| 30992 | d009                     | 52403             | 48439             | 1999-05-04        | 2000-05-03      | 1997-12-23        | 2002-03-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 30993 | d009                     | 52403             | 50454             | 2000-05-03        | 2001-05-03      | 1997-12-23        | 2002-03-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 30994 | d009                     | 52403             | 52671             | 2001-05-03        | 2002-03-01      | 1997-12-23        | 2002-03-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 30995 | d009                     | 52407             | 52517             | 1995-07-24        | 1996-07-23      | 1995-07-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 30996 | d009                     | 52407             | 54761             | 1996-07-23        | 1997-07-23      | 1995-07-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 30997 | d009                     | 52407             | 55571             | 1997-07-23        | 1998-07-23      | 1995-07-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 30998 | d009                     | 52407             | 57937             | 1998-07-23        | 1999-07-23      | 1995-07-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 30999 | d009                     | 52407             | 58121             | 1999-07-23        | 2000-07-22      | 1995-07-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 58770.366479762 |
| 31000 | d009                     | 52407             | 62246             | 2000-07-22        | 2001-07-22      | 1995-07-24        | 9999-01-01      | Customer Service                    | Customer Service                    | 58770.366479762 |

# **Creating Databases and Tables**

- We've focused on querying and reading data from existing databases and tables.
- Let's now shift our focus to creating our own databases and tables.

- Data Types
- Primary and Foreign Keys
- Constraints
- CREATE
- INSERT
- UPDATE
- DELETE, ALTER, DROP

- Boolean
  - True or False
- Character
  - char, varchar, and text
- Numeric
  - integer and floating-point number
- Temporal
  - date, time, timestamp, and interval

- **UUID**
  - Universally Unique Identifiers
- **Array**
  - Stores an array of strings, numbers, etc.
- **JSON**
- **Hstore key-value pair**
- **Special types such as network address and geometric data.**

- When creating databases and tables, you should carefully consider which data types should be used for the data to be stored.
- Review the documentation to see limitations of data types:
- **[postgresql.org/docs/current/datatype.htm](http://postgresql.org/docs/current/datatype.htm)**

| Name                          | Storage Size | Description                     | Range                                                                                    |
|-------------------------------|--------------|---------------------------------|------------------------------------------------------------------------------------------|
| <code>smallint</code>         | 2 bytes      | small-range integer             | -32768 to +32767                                                                         |
| <code>integer</code>          | 4 bytes      | typical choice for integer      | -2147483648 to +2147483647                                                               |
| <code>bigint</code>           | 8 bytes      | large-range integer             | -9223372036854775808 to +9223372036854775807                                             |
| <code>decimal</code>          | variable     | user-specified precision, exact | up to 131072 digits before the decimal point; up to 16383 digits after the decimal point |
| <code>numeric</code>          | variable     | user-specified precision, exact | up to 131072 digits before the decimal point; up to 16383 digits after the decimal point |
| <code>real</code>             | 4 bytes      | variable-precision, inexact     | 6 decimal digits precision                                                               |
| <code>double precision</code> | 8 bytes      | variable-precision, inexact     | 15 decimal digits precision                                                              |
| <code>smallserial</code>      | 2 bytes      | small autoincrementing integer  | 1 to 32767                                                                               |
| <code>serial</code>           | 4 bytes      | autoincrementing integer        | 1 to 2147483647                                                                          |
| <code>bigserial</code>        | 8 bytes      | large autoincrementing integer  | 1 to 9223372036854775807                                                                 |

- Based on the limitations, you may think it makes sense to store it as a **BIGINT** data type, but we should really be thinking what is best for the situation.
- Why bother with numerics at all?
- We don't perform arithmetic with numbers, so it probably makes more sense as a **VARCHAR** data type instead.

- In fact, searching for best practice online, you will discover it's usually recommended to store as a text based data type due to a variety of issues
  - No arithmetic performed
  - Leading zeros could cause issues, 7 and 07 treated same numerically, but are not the same phone number

- When creating a database and table, take your time to plan for long term storage
- Remember you can always remove historical information you've decided you aren't using, but you can't go back in time to add in information!

# **Primary and Foreign Keys**

- A primary key is a column or a group of columns used to identify a row uniquely in a table.
- For example, in our dvdrental database we saw customers had a unique, non-null customer\_id column as their primary key.

- Primary keys are also important since they allow us to easily discern what columns should be used for joining tables together.

## ● Example of Primary Key

Query Editor    Query History

1    **SELECT \* FROM customer**

Data Output    Explain    Messages    Notifications

|   | customer_id<br>[PK] integer | store_id<br>smallint | first_name<br>character varying (45) | last_name<br>character varying (45) |
|---|-----------------------------|----------------------|--------------------------------------|-------------------------------------|
| 1 | 524                         | 1                    | Jared                                | Ely                                 |
| 2 | 1                           | 1                    | Mary                                 | Smith                               |
| 3 | 2                           | 1                    | Patricia                             | Johnson                             |
| 4 | 3                           | 1                    | Linda                                | Williams                            |

- Example of Primary Key

Query Editor    Query History

1    **SELECT \* FROM customer**

Data Output    Explain    Messages    Notifications

|   | customer_id<br>[PK] integer | store_id<br>smallint | first_name<br>character varying (45) | last_name<br>character varying (45) |
|---|-----------------------------|----------------------|--------------------------------------|-------------------------------------|
| 1 | 524                         | 1                    | Jared                                | Ely                                 |
| 2 | 1                           | 1                    | Mary                                 | Smith                               |
| 3 | 2                           | 1                    | Patricia                             | Johnson                             |
| 4 | 3                           | 1                    | Linda                                | Williams                            |

- A foreign key is a field or group of fields in a table that uniquely identifies a row in another table.
- A foreign key is defined in a table that references to the primary key of the other table.



- The table that contains the foreign key is called referencing table or child table.
- The table to which the foreign key references is called referenced table or parent table.
- A table can have multiple foreign keys depending on its relationships with other tables.



## ● Multiple Foreign Key References

Query Editor    Query History

1   **SELECT \* FROM payment**

Data Output    Explain    Messages    Notifications

|   | payment_id<br>[PK] integer | customer_id<br>smallint | staff_id<br>smallint | rental_id<br>integer | amount<br>numeric (5,2) |
|---|----------------------------|-------------------------|----------------------|----------------------|-------------------------|
| 1 | 17503                      | 341                     | 2                    | 1520                 | 7.99                    |
| 2 | 17504                      | 341                     | 1                    | 1778                 | 1.99                    |
| 3 | 17505                      | 341                     | 1                    | 1849                 | 7.99                    |
| 4 | 17506                      | 341                     | 2                    | 2829                 | 2.99                    |



- When creating tables and defining columns, we can use constraints to define columns as being a primary key, or attaching a foreign key relationship to another table.
- Let's quickly explore table properties in pgAdmin to see how to get information on primary and foreign keys!

- Constraints are the rules enforced on data columns on table.
  - These are used to prevent invalid data from being entered into the database.
  - This ensures the accuracy and reliability of the data in the database.
-

- Constraints can be divided into two main categories:
  - Column Constraints
    - Constrains the data in a column to adhere to certain conditions.
  - Table Constraints
    - applied to the entire table rather than to an individual column.

- The most common constraints used:
  - **NOT NULL Constraint**
    - Ensures that a column cannot have NULL value.
  - **UNIQUE Constraint**
    - Ensures that all values in a column are different.

- The most common constraints used:
  - **PRIMARY Key**
    - Uniquely identifies each row/record in a database table.
  - **FOREIGN Key**
    - Constrains data based on columns in other tables.

- The most common constraints used:
  - **CHECK** Constraint
    - Ensures that all values in a column satisfy certain conditions.

- The most common constraints used:
  - **EXCLUSION** Constraint
    - Ensures that if any two rows are compared on the specified column or expression using the specified operator, not all of these comparisons will return TRUE.

- Table Constraints
  - CHECK (condition)
    - to check a condition when inserting or updating data.
  - REFERENCES
    - to constrain the value stored in the column that must exist in a column in another table.

- Now that we understand data types, primary keys, foreign keys, and constraints we are ready to begin using SQL syntax to create tables!

- Let's now learn the syntax to create a table in SQL using the CREATE keyword and column syntax.

- Example Syntax
  - `CREATE TABLE table_name (`  
`column_name TYPE column_constraint,`  
`column_name TYPE column_constraint,`  
 );

- **SERIAL**
  - It will create a sequence object and set the next value generated by the sequence as the default value for the column.
  - This is perfect for a primary key, because it logs unique integer entries for you automatically upon insertion.

- **SERIAL**
  - If a row is later removed, the column with the SERIAL data type will not adjust, marking the fact that a row was removed from the sequence, for example
    - 1,2,3,5,6,7
      - You know row 4 was removed at some point

| Name                          | Storage Size | Description                     | Range                                                                                    |
|-------------------------------|--------------|---------------------------------|------------------------------------------------------------------------------------------|
| <code>smallint</code>         | 2 bytes      | small-range integer             | -32768 to +32767                                                                         |
| <code>integer</code>          | 4 bytes      | typical choice for integer      | -2147483648 to +2147483647                                                               |
| <code>bigint</code>           | 8 bytes      | large-range integer             | -9223372036854775808 to +9223372036854775807                                             |
| <code>decimal</code>          | variable     | user-specified precision, exact | up to 131072 digits before the decimal point; up to 16383 digits after the decimal point |
| <code>numeric</code>          | variable     | user-specified precision, exact | up to 131072 digits before the decimal point; up to 16383 digits after the decimal point |
| <code>real</code>             | 4 bytes      | variable-precision, inexact     | 6 decimal digits precision                                                               |
| <code>double precision</code> | 8 bytes      | variable-precision, inexact     | 15 decimal digits precision                                                              |
| <code>smallserial</code>      | 2 bytes      | small autoincrementing integer  | 1 to 32767                                                                               |
| <code>serial</code>           | 4 bytes      | autoincrementing integer        | 1 to 2147483647                                                                          |
| <code>bigserial</code>        | 8 bytes      | large autoincrementing integer  | 1 to 9223372036854775807                                                                 |

- Example Syntax
  - `CREATE TABLE players(  
 player_id SERIAL PRIMARY KEY,  
 age TYPE column_constraint,  
)`

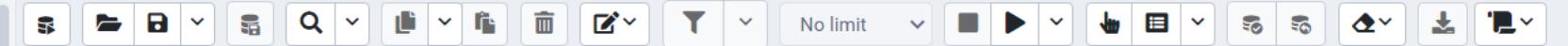
- Example Syntax
  - `CREATE TABLE players(  
 player_id SERIAL PRIMARY KEY,  
 age SMALLINT column_constraint  
)`

- Example Syntax
  - `CREATE TABLE players(  
 player_id SERIAL PRIMARY KEY,  
 age SMALLINT NOT NULL  
)`



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (8)
        - account
        - departments
        - dent.emp



```
1 CREATE TABLE account1(
2
3 user_id SERIAL PRIMARY KEY,
4 username VARCHAR(50) UNIQUE NOT NULL,
5 password VARCHAR(50) NOT NULL,
6 email VARCHAR(250)UNIQUE NOT NULL,
7 created_on TIMESTAMP NOT NULL,
8 last_login TIMESTAMP
9
10)
11
```

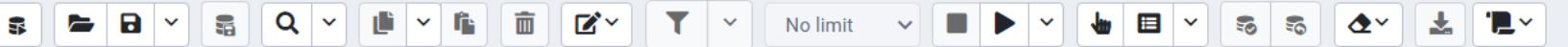
CREATE TABLE

Query returned successfully in 65 msec.



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
  - Tables (8)
    - account
    - departments
    - dent.emp



```
1 CREATE TABLE job(
2 job_id SERIAL PRIMARY KEY,
3 job_name VARCHAR(200) UNIQUE NOT NULL
4
5
6)
```

CREATE TABLE

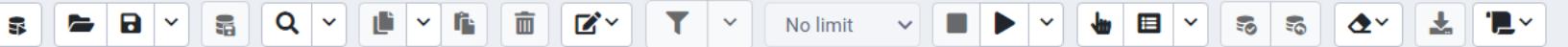
Query returned successfully in 62 msec.

✓ Query returned successfully in 62 msec.



Servers (2)

- PostgreSQL 14
- Databases (11)
  - advanced-SQL
  - data1
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
    - Schemas (1)
      - public
        - Aggregates
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Operators
        - Procedures
        - Sequences
      - Tables (8)
        - account
        - departments
        - dent.emp



```
1 CREATE TABLE account_job(
2 USER_id INTEGER REFERENCES account(user_id),
3 job_id INTEGER REFERENCES job(job_id),
4 hire_date TIMESTAMP
5
6)
```

CREATE TABLE

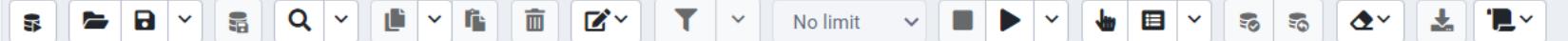
Query returned successfully in 60 msec.

Query returned successfully in 60 msec.



Schemas (1)

- public
  - Aggregates
  - Collations
  - Domains
  - FTS Configurations
  - FTS Dictionaries
  - FTS Parsers
  - FTS Templates
  - Foreign Tables
  - Functions
  - Materialized Views
  - Operators
  - Procedures
  - Sequences
- Tables (11)
  - account
  - account1
  - account\_job
    - Columns
    - Constraints (2)
      - account\_job\_job\_id\_fkey
      - account\_job\_user\_id\_fkey
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - departments
  - dept\_emp
  - dept\_manager
  - employees



```
1 CREATE TABLE account_job(
2 USER_id INTEGER REFERENCES account(user_id),
3 job_id INTEGER REFERENCES job(job_id),
4 hire_date TIMESTAMP
5
6)
```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 60 msec.



- **INSERT** allows you to add in rows to a table.
- **General Syntax**
  - **INSERT INTO** table (column1, column2, ...)  
**VALUES**  
(value1, value2, ...),  
(value1, value2, ...) ,...;

- Keep in mind, the inserted row values must match up for the table, including constraints.
- SERIAL columns do not need to be provided a value.
- Let's use INSERT in pgAdmin!

- Keep in mind, the inserted row values must match up for the table, including constraints.
- SERIAL columns do not need to be provided a value.
- Let's use INSERT in pgAdmin!



- ▼ Schemas (1)
  - ✓ public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > 1..3 Sequences
  - ▼ Tables (11)
    - > account
    - > account1
    - ✓ account\_job
      - > Columns
      - ▼ Constraints (2)
        - > account\_job\_job\_id\_fkey
        - > account\_job\_user\_id\_fkey
      - > Indexes
      - > RLS Policies
      - > Rules
      - > Triggers
    - > departments
    - > dept\_emp
    - > dept\_manager
    - > employees



No limit

Query Editor Query History

```
1 INSERT INTO account1(username,password,email,created_on)
2 VALUES
3 ('Ram','root','ram1@sanjivani.org.in',CURRENT_TIMESTAMP)
```

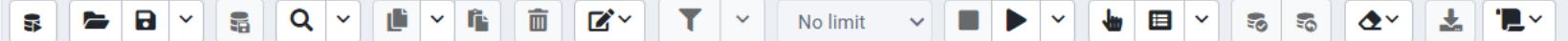
Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 58 msec.



- ▼ Schemas (1)
  - ✓ public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > 1..3 Sequences
  - ▼ Tables (11)
    - > account
    - > account1
    - ✓ account\_job
      - > Columns
      - ▼ Constraints (2)
        - > account\_job\_job\_id\_fkey
        - > account\_job\_user\_id\_fkey
      - > Indexes
      - > RLS Policies
      - > Rules
      - > Triggers
    - > departments
    - > dept\_emp
    - > dept\_manager
    - > employees



```
1 INSERT INTO account1(username,password,email,created_on)
2 VALUES
3 ('Ram1','root','ram1@sanjivani.org.in',CURRENT_TIMESTAMP)
```

ERROR: duplicate key value violates unique constraint "account1\_email\_key"  
DETAIL: Key (email)=(ram1@sanjivani.org.in) already exists.  
SQL state: 23505



- ▼ Schemas (1)
  - ▶ public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > Sequences
  - ▼ Tables (11)
    - > account
    - > account1
    - ▼ account\_job
      - > Columns
      - ▼ Constraints (2)
        - ▶ account\_job\_job\_id\_fkey
        - ▶ account\_job\_user\_id\_fkey
      - > Indexes
      - > RLS Policies
      - > Rules
      - > Triggers
    - > departments
    - > dept\_emp
    - > dept\_manager
    - > employees

1 SELECT \* FROM account1

No limit

Data Output Explain Messages Notifications

|   | user_id<br>[PK] integer | username<br>character varying (50) | password<br>character varying (50) | email<br>character varying (250) | created_on<br>timestamp without time zone | last_login<br>timestamp without time zone |
|---|-------------------------|------------------------------------|------------------------------------|----------------------------------|-------------------------------------------|-------------------------------------------|
| 1 | 1                       | ram                                | root                               | ram@sanjivani.org.in             | 2023-07-07 14:13:12.191293                | [null]                                    |
| 2 | 3                       | Ram                                | root                               | ram1@sanjivani.org.in            | 2023-07-07 14:14:12.891202                | [null]                                    |



- ▼ Schemas (1)
  - ✓ public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > Sequences
  - ▼ Tables (11)
    - > account
    - > account1
    - ▼ account\_job
      - > Columns
      - ▼ Constraints (2)
        - > account\_job\_job\_id\_fkey
        - > account\_job\_user\_id\_fkey
      - > Indexes
      - > RLS Policies
      - > Rules
      - > Triggers
    - > departments
    - > dept\_emp
    - > dept\_manager
    - > employees

```
1 INSERT INTO job(job_name)
2 VALUES('Data Scietist')
```

No limit



Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 53 msec.

✓ Query returned successfully in 53 msec.



- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- Tables (11)
  - > account
  - > account1
  - > account\_job
    - > Columns
    - > Constraints (2)
      - account\_job\_job\_id\_fkey
      - account\_job\_user\_id\_fkey
  - > Indexes
  - > RLS Policies
  - > Rules
  - > Triggers
  - > departments
  - > dept\_emp
  - > dept\_manager
  - > employees
  - job
    - > Columns (2)
      - job\_id
      - job\_name
    - > Constraints
    - > Indexes



1 SELECT \* FROM job

|   | job_id<br>[PK] integer | job_name<br>character varying (200) |
|---|------------------------|-------------------------------------|
| 1 | 1                      | Data Scietist                       |



- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- Tables (11)
  - > account
  - > account1
  - > account\_job
    - > Columns
    - > Constraints (2)
      - account\_job\_job\_id\_fkey
      - account\_job\_user\_id\_fkey
  - > Indexes
  - > RLS Policies
  - > Rules
  - > Triggers
  - > departments
  - > dept\_emp
  - > dept\_manager
  - > employees
  - > job
    - > Columns (2)
      - job\_id
      - job\_name
    - > Constraints
    - > Indexes



1 SELECT \* FROM job

|   | job_id<br>[PK] integer | job_name<br>character varying (200) |
|---|------------------------|-------------------------------------|
| 1 |                        | 1 Data Scietist                     |
| 2 |                        | 2 data analyst                      |



Functions

- > Materialized Views
- > Operators
- > Procedures
- > 1..3 Sequences
- > Tables (11)
  - > account
  - > account1
  - > account\_job
    - > Columns
    - > Constraints (2)
      - account\_job\_job\_id\_fkey
      - account\_job\_user\_id\_fkey
    - > Indexes
    - > RLS Policies
    - > Rules
    - > Triggers
  - > departments
  - > dept\_emp
  - > dept\_manager
  - > employees
  - > job
    - > Columns (2)
      - job\_id
      - job\_name
    - > Constraints (2)
      - ① job\_job\_name\_key
      - job\_pkey
    - > Indexes
    - > RLS Policies
    - > Rules

data1/postgres@PostgreSQL 14 \*

Query Editor Query History

```
1 INSERT INTO account_job(job_id,user_id,hire_date)
2 VALUES
3 (1,1,CURRENT_TIMESTAMP)
4
```

Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 79 msec.

- Example
  - `UPDATE account  
SET last_login = CURRENT_TIMESTAMP  
WHERE last_login IS NULL;`



Functions

- > Materialized Views
- > Operators
- > Procedures
- > 1..3 Sequences
- > Tables (11)
  - > account
  - > account1
  - > account\_job
    - > Columns
  - > Constraints (2)
    - account\_job\_job\_id\_fkey
    - account\_job\_user\_id\_fkey
  - > Indexes
  - > RLS Policies
  - > Rules
  - > Triggers
- > departments
- > dept\_emp
- > dept\_manager
- > employees
- > job
  - > Columns (2)
    - job\_id
    - job\_name
  - > Constraints (2)
    - ① job\_job\_name\_key
    - job\_pkey
  - > Indexes
  - > RLS Policies
  - > Rules

data1/postgres@PostgreSQL 14 \*

Query Editor Query History

```
1 UPDATE account1
2 SET last_login=CURRENT_TIMESTAMP
3
```

Data Output Explain Messages Notifications

UPDATE 2

Query returned successfully in 57 msec.