

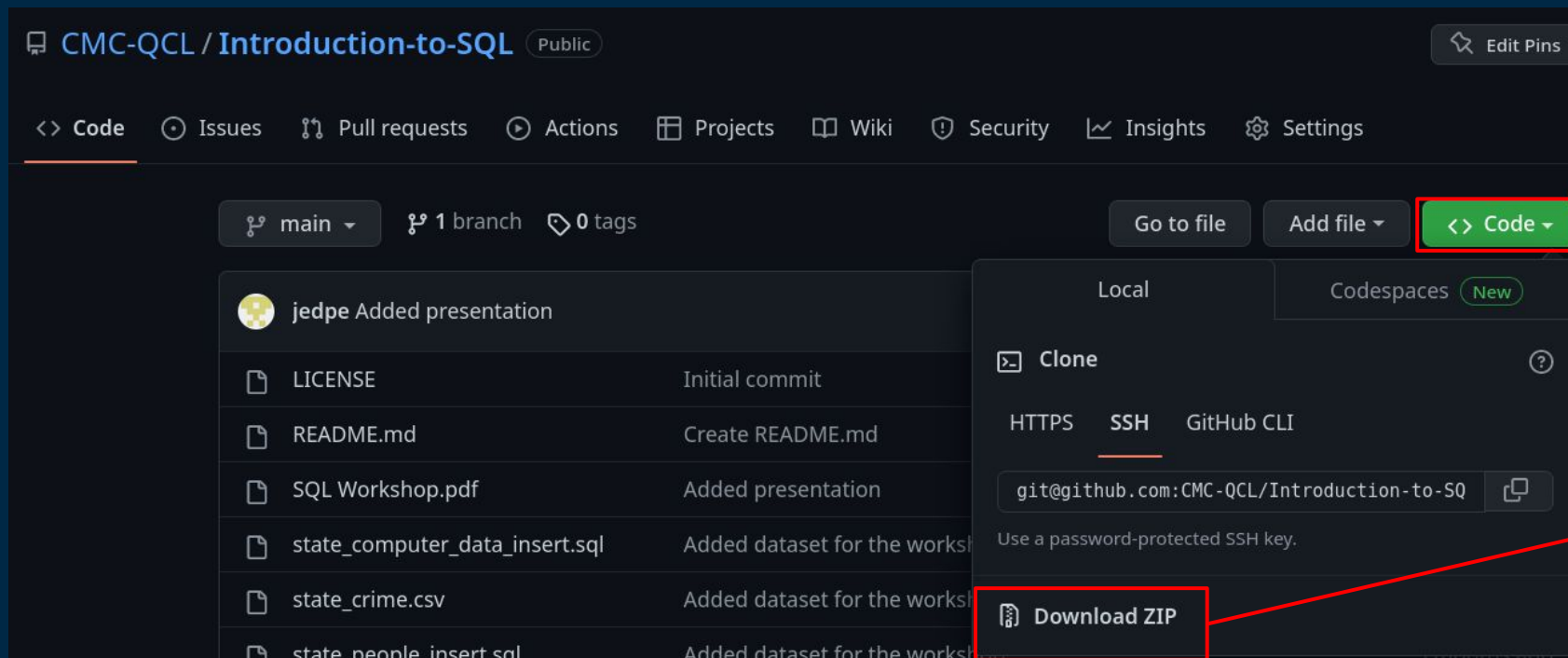


# Introduction to SQL

Jorge Peña  
QCL Graduate Fellow

# Before we start

- ❑ Download the materials for this workshop
  - ❑ <https://github.com/CMC-QCL/Introduction-to-SQL>



# What is SQL?

- SQL stands for Structured Query Language
- DBeaver is a Database Administration Software
  - Connect to Databases
  - Add data to your databases and tables
  - Retrieve data



# Interface

SQL Editor  
controls

Choose your connection  
and database

SQL Editor

Create a  
new  
connection

Databases  
and tables

Project view

Results  
view

The screenshot displays the DBeaver 22.2.3 interface. The top menu bar includes File, Edit, Navigate, Search, SQL Editor, Database, Window, and Help. The toolbar contains various icons for file operations and database actions. The main window is divided into several panels:

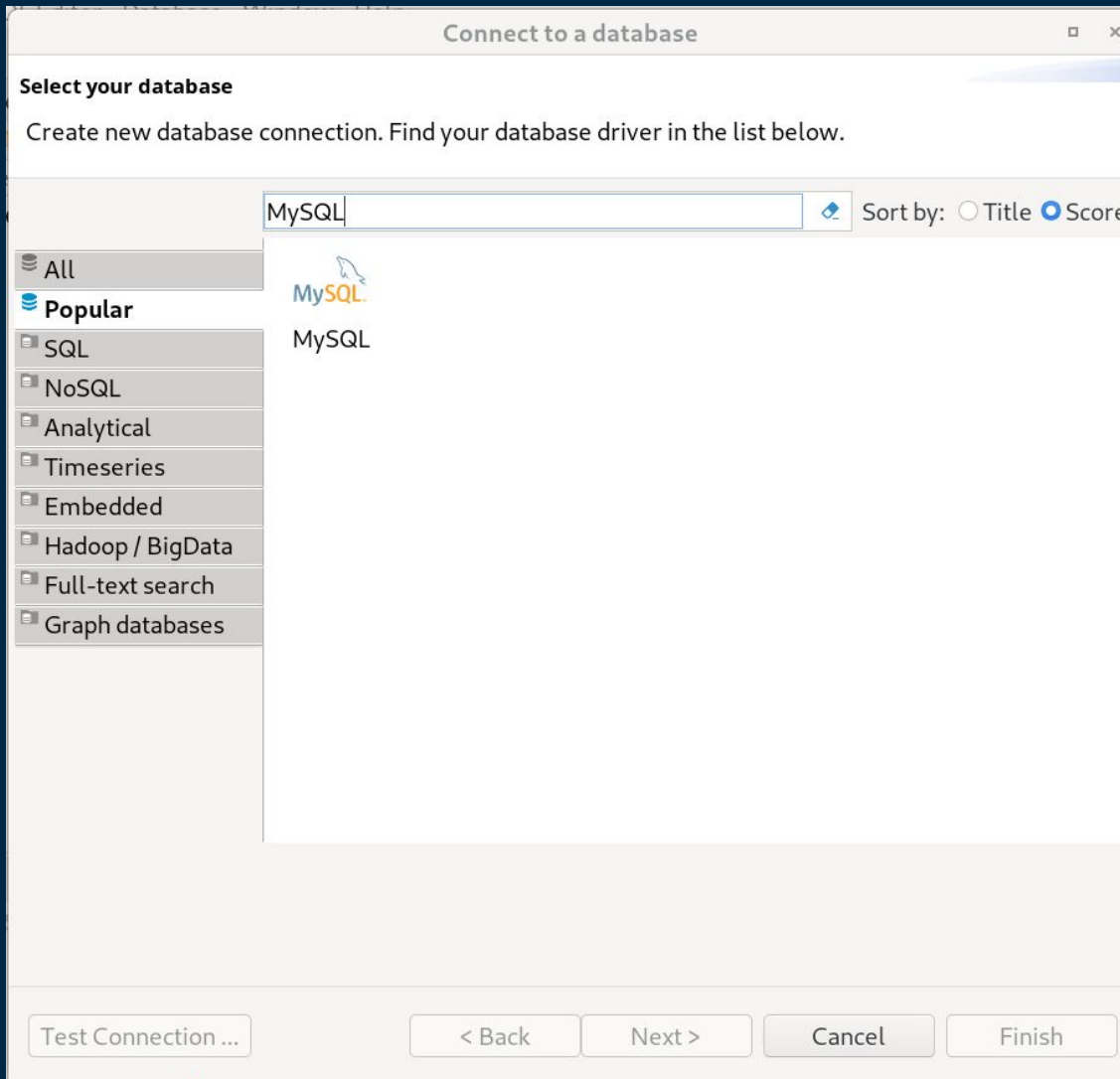
- Database Navigator:** Located on the left, it shows a tree view of the database structure. The 'localhost - 134.173.191.241:3306' connection is selected, showing a list of databases including 'DBeaver Sample Database (SQLite)', 'JorgeDB', 'MyDB', and 'QCLWorkshop'. Under 'QCLWorkshop', there are tables like 'state\_computer\_data', 'state\_crime', 'state\_people', and 'state\_workforce', as well as Views, Indexes, Procedures, Triggers, Events, and 'cmcdb'.
- SQL Editor:** The central panel shows a script named 'Script-2' with the query: `SELECT * FROM state_crime;`
- Results View:** The bottom panel displays the results of the query in a table format. The table has columns: State, Crime\_Year, Population, Rates\_Property\_Theft, Rates\_Violent\_Robbery, Totals\_Property\_Theft, and Totals\_Violent\_R. The data is filtered to show 156 rows.

Red arrows point from the text labels to the corresponding interface elements: 'Create a new connection' points to the '+' icon in the Database Navigator; 'Databases and tables' points to the tree view; 'Project view' points to the 'Project - General' tab; 'SQL Editor controls' points to the toolbar; 'Choose your connection and database' points to the connection dropdown; 'SQL Editor' points to the script editor area; and 'Results view' points to the results table.

	State	Crime_Year	Population	Rates_Property_Theft	Rates_Violent_Robbery	Totals_Property_Theft	Totals_Violent_R
1	Alabama	2,019	4,903,185	532	80	26,079	
2	Alaska	2,019	731,545	487	113	3,563	
3	Arizona	2,019	7,278,717	394	88	28,699	
4	Arkansas	2,019	3,017,804	600	52	18,095	
5	California	2,019	39,512,223	386	132	152,555	
6	Colorado	2,019	5,758,736	348	64	20,064	
7	Connecticut	2,019	3,565,287	181	54	6,441	
8	Delaware	2,019	973,764	305	81	2,968	
9	District of Columbia	2,019	705,749	261	384	1,843	

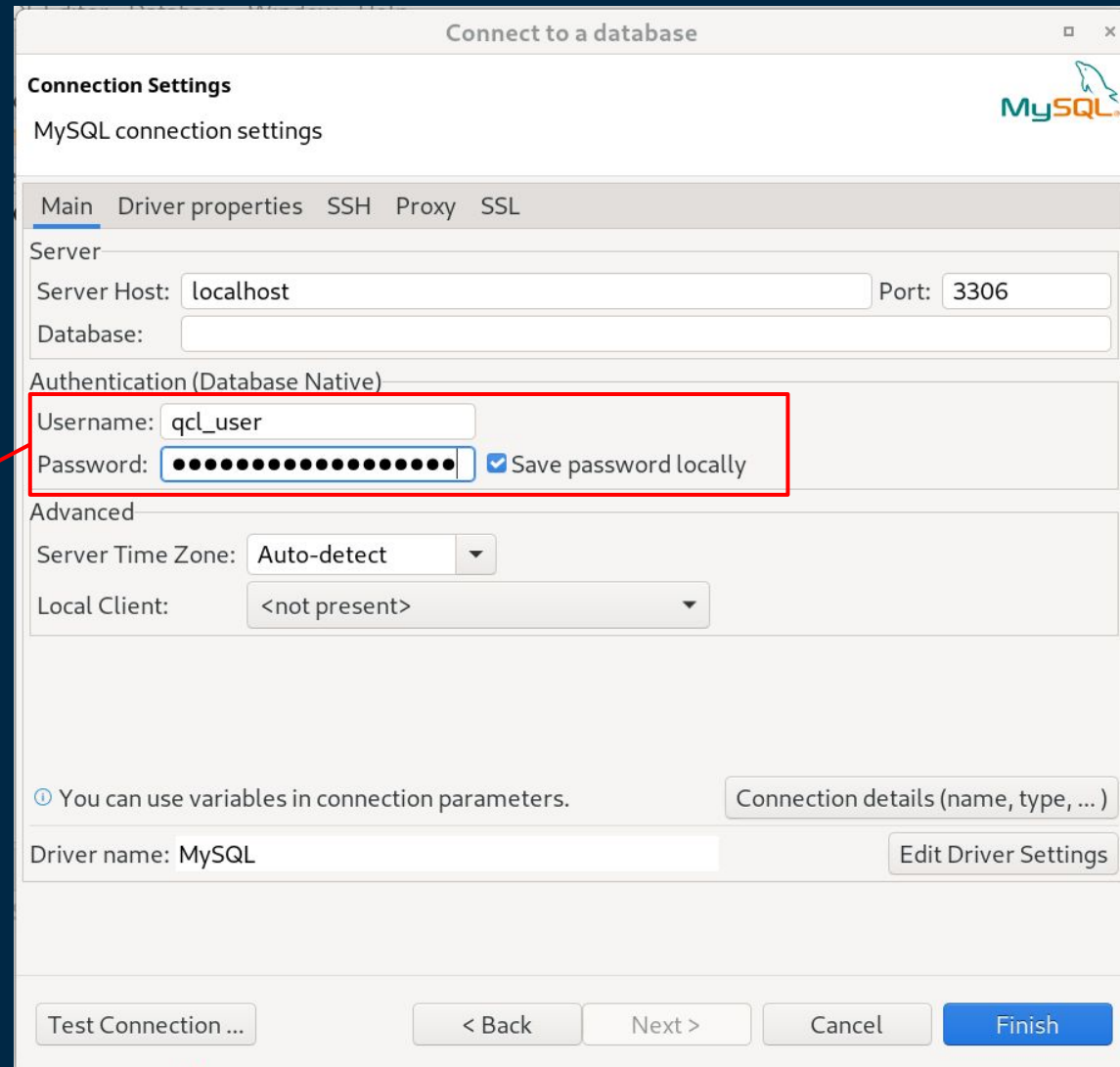
# Create a Connection

- Create a new connection using MySQL



# Database Settings

Enter the Username and Password for the database



Connect to a database

Connection Settings  
MySQL connection settings

Main Driver properties SSH Proxy SSL

Server

Server Host: localhost Port: 3306

Database:

Authentication (Database Native)

Username: qcl\_user

Password: [masked] ☒ Save password locally

Advanced

Server Time Zone: Auto-detect

Local Client: <not present>

① You can use variables in connection parameters. Connection details (name, type, ...)

Driver name: MySQL Edit Driver Settings

Test Connection ... < Back Next > Cancel Finish

# SSH Tunnel

Select the  
SSH tab

Enter the IP  
**134.173.191.241**

Select "Use SSH  
Tunnel"

Enter the port  
**5033**

Enter the Username  
and Password you  
receive through email

Select "Password"  
for authentication  
method

Finally test your  
configuration

The screenshot shows the 'Connect to a database' dialog box with the 'SSH' tab selected. The 'Use SSH Tunnel' checkbox is checked. The 'Host/IP' field contains '134.173.191.241' and the 'Port' field contains '5033'. The 'User Name' field contains 'your\_user' and the 'Authentication Method' is set to 'Password'. The 'Password' field is masked with dots, and the 'Save Password/Passphrase' checkbox is checked. At the bottom, there is a 'Test tunnel configuration' button and a 'Test Connection ...' button. The 'Finish' button is highlighted in blue.

Connect to a database

MySQL connection settings

Main Driver properties **SSH** Proxy SSL

☒ Use SSH Tunnel

Profile:

Settings

Host/IP: 134.173.191.241 Port: 5033

User Name: your\_user

Authentication Method: Password

Password:  ☒ Save Password/Passphrase

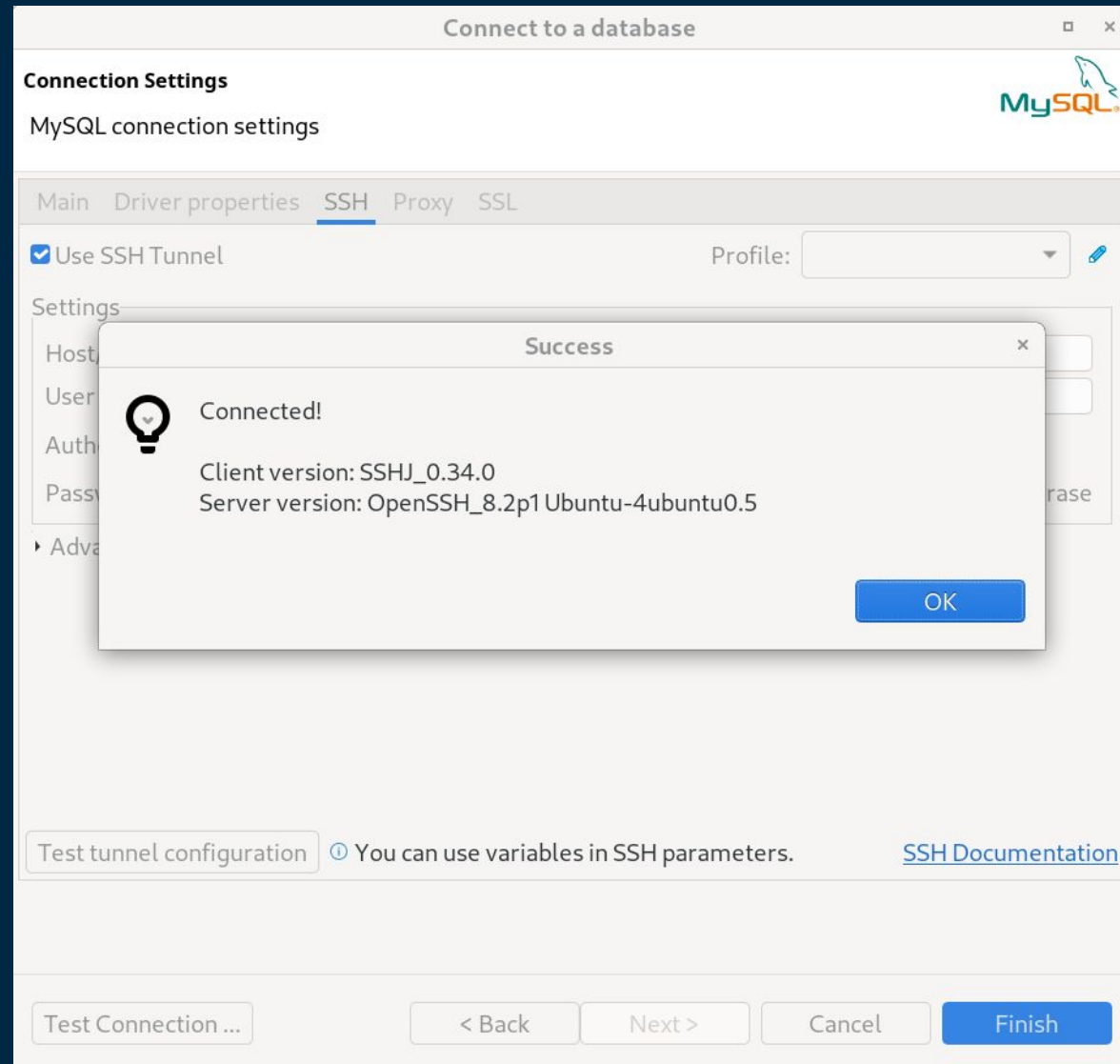
Advanced settings

Test tunnel configuration ⓘ You can use variables in SSH parameters. [SSH Documentation](#)

Test Connection ... < Back Next > Cancel Finish

# Successful SSH

- Make sure to approve the push notification on DUO
- You will see a success message if the SSH connection works

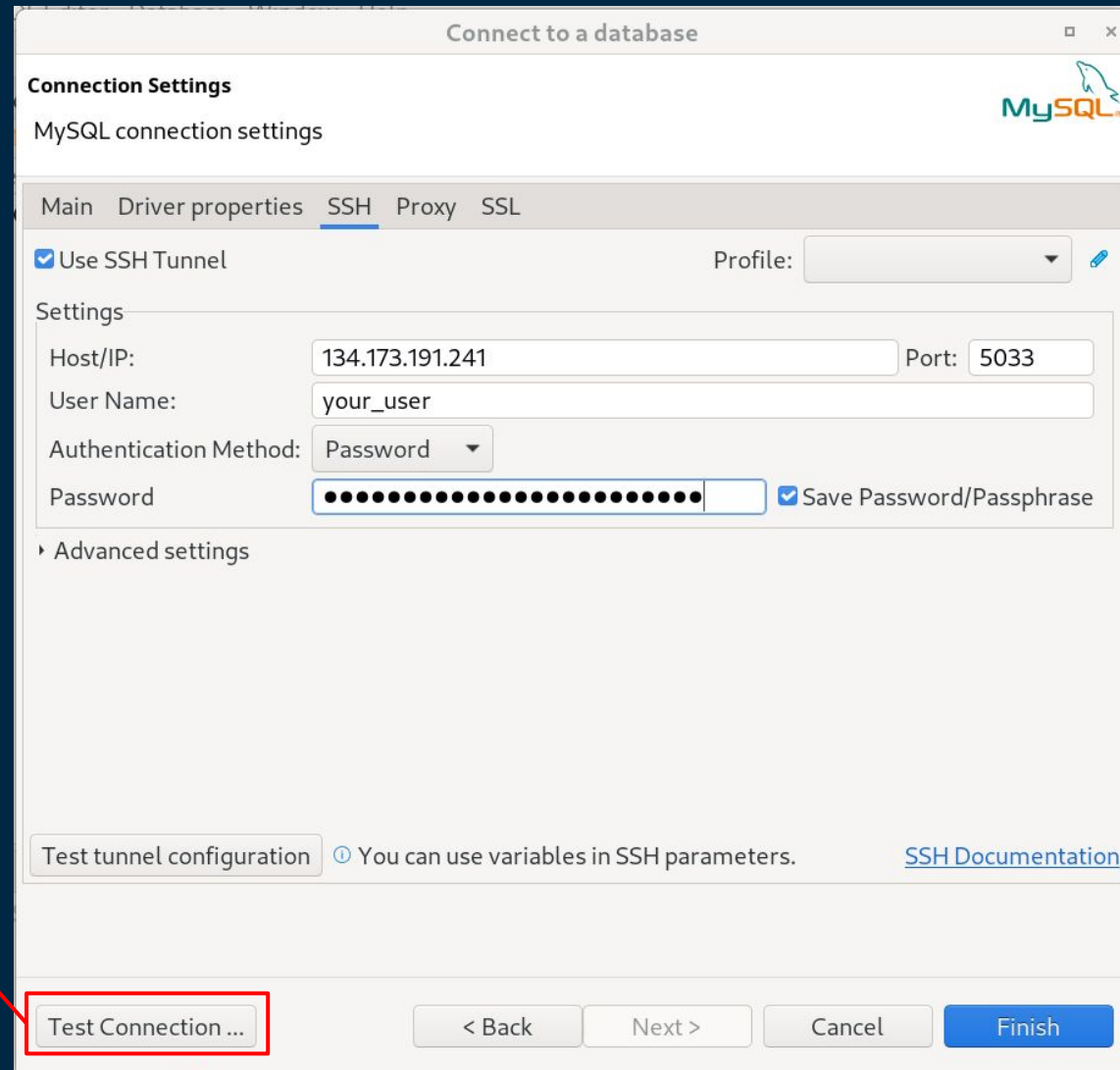




# Test the connection

- Now you need to test the database connection

Test your  
connection



Connect to a database

Connection Settings  
MySQL connection settings

Main Driver properties **SSH** Proxy SSL

☒ Use SSH Tunnel Profile: ▼

Settings

Host/IP: 134.173.191.241 Port: 5033

User Name: your\_user

Authentication Method: Password ▼

Password: ●●●●●●●●●●●●●●●● ☒ Save Password/Passphrase

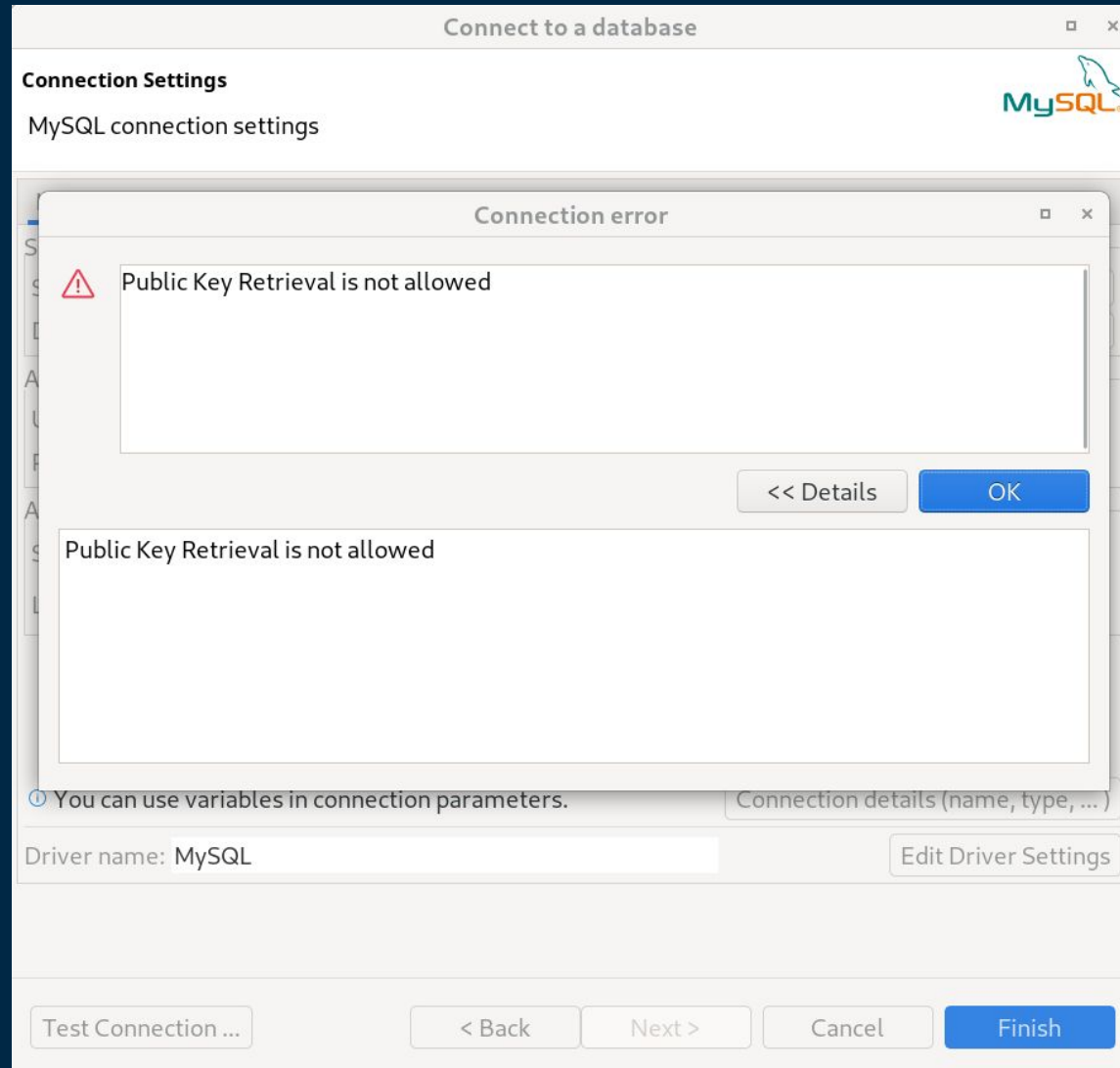
Advanced settings

Test tunnel configuration ⓘ You can use variables in SSH parameters. [SSH Documentation](#)

Test Connection ... < Back Next > Cancel Finish

# Connection error

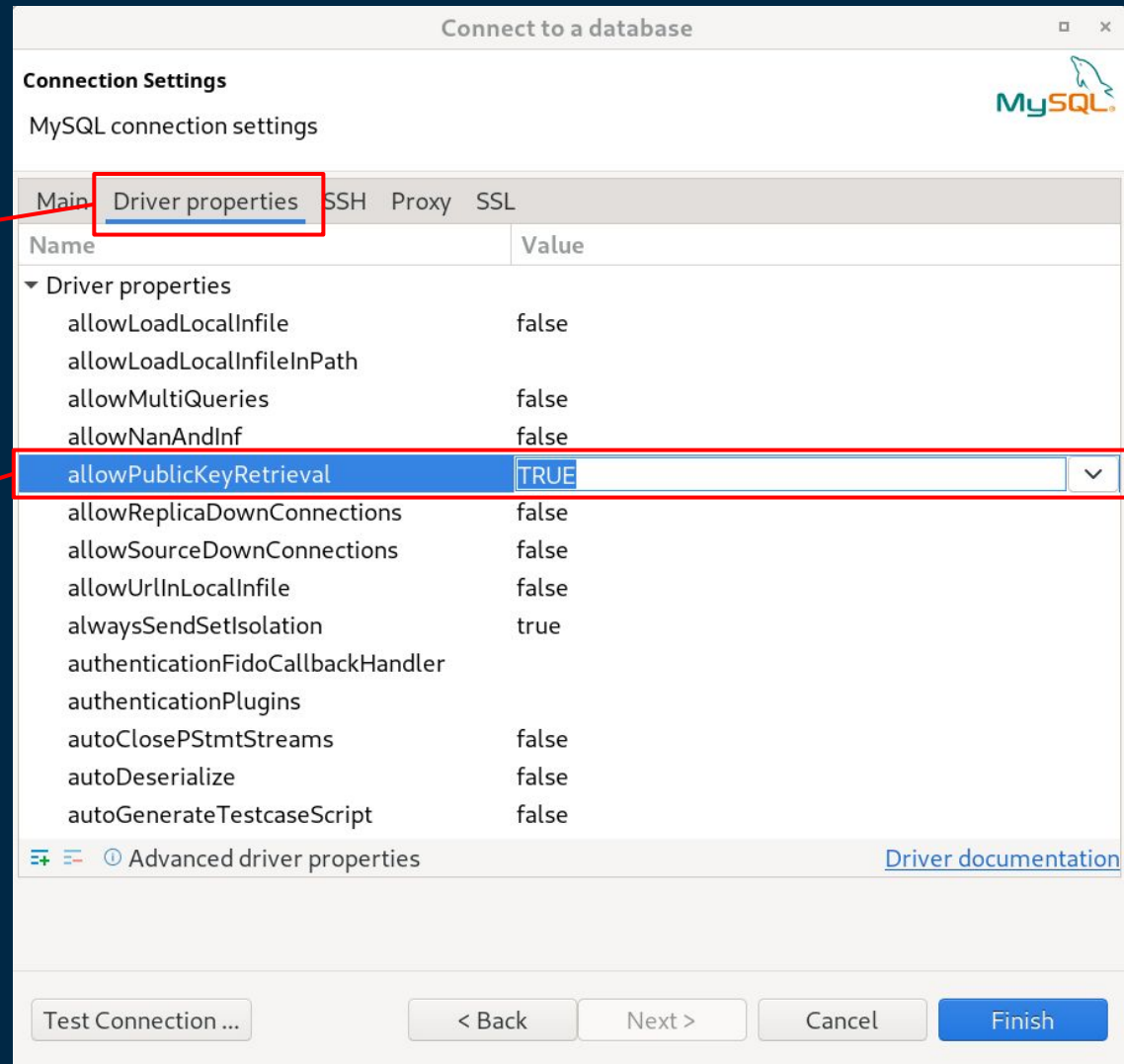
- If you get a connection error like this one follow the steps on the following slides



# Driver properties

Click on the Driver properties tab

Change the value of **allowPublicKeyRetrieval** to TRUE



Connect to a database

Connection Settings

MySQL connection settings

Main **Driver properties** SSH Proxy SSL

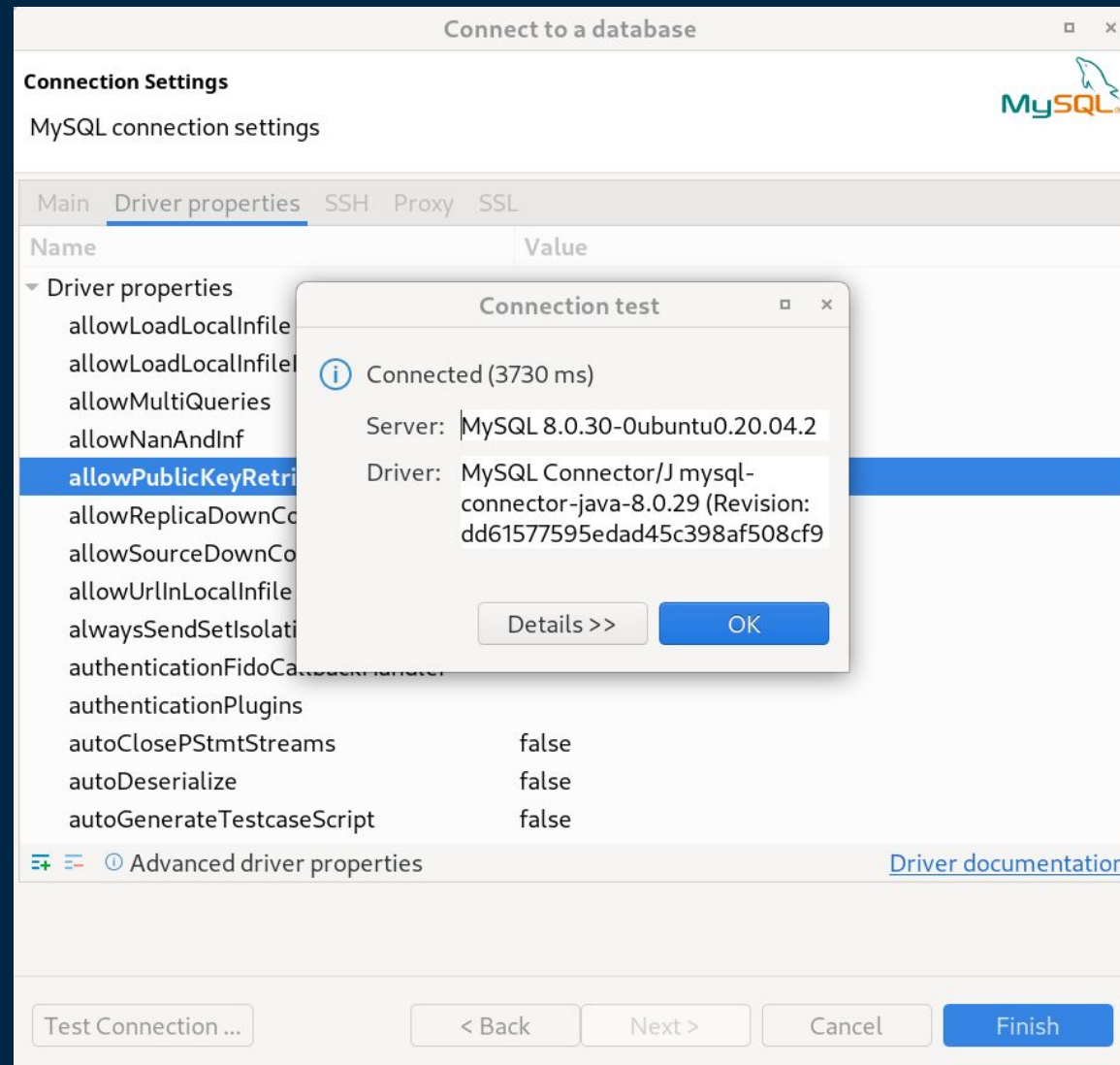
Name	Value
▼ Driver properties	
allowLoadLocalInfile	false
allowLoadLocalInfileInPath	
allowMultiQueries	false
allowNanAndInf	false
<b>allowPublicKeyRetrieval</b>	<b>TRUE</b>
allowReplicaDownConnections	false
allowSourceDownConnections	false
allowUrlInLocalInfile	false
alwaysSendSetIsolation	true
authenticationFidoCallbackHandler	
authenticationPlugins	
autoClosePStmtStreams	false
autoDeserialize	false
autoGenerateTestcaseScript	false

Advanced driver properties [Driver documentation](#)

Test Connection ... < Back Next > Cancel Finish

# Successful connection

- Test your connection once again and you should see the success



# Agenda

## Databases and SQL



- Relational Databases
- SQL Overview

## Tables and Data



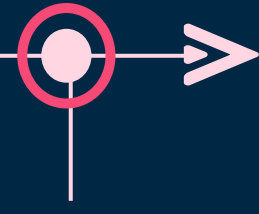
- Databases and Tables
- Today's data

## Writing Queries



- Retrieving Data
- Sorting and Filtering

## Subqueries and Joins



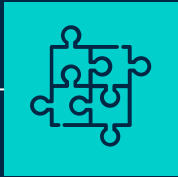
- Subqueries and Aliases
- Joining Tables



# Databases and SQL



# Database Advantages



01

## BETTER DATA INTEGRATION

Improves data  
handling and reduces  
redundancy



02

## STORAGE IS MORE SECURE

Provides better  
privacy and security  
policies



03

## FASTER DATA ACCESS

Produce quick  
answers to data  
queries

# Relational Databases

Students\_Registration

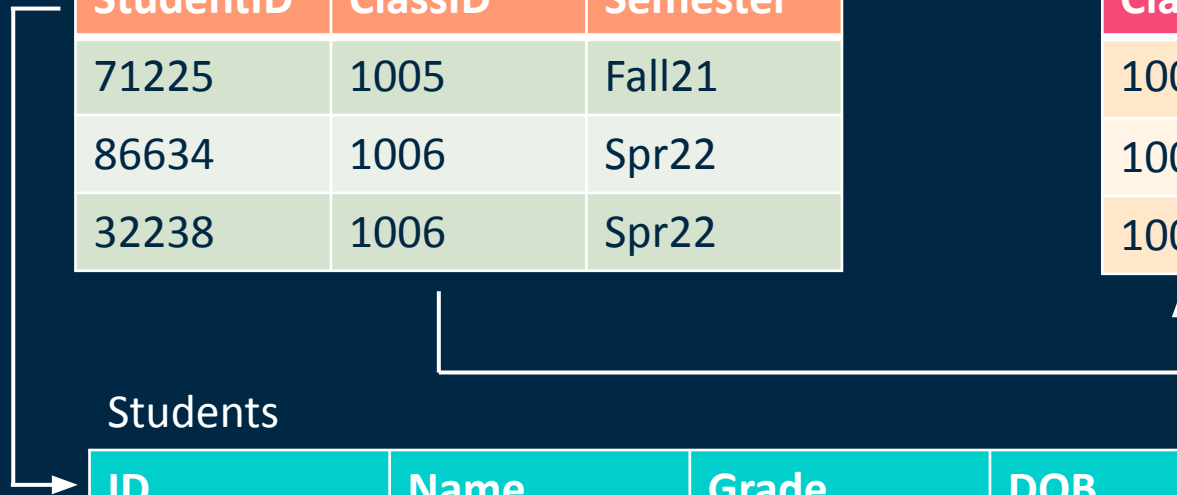
StudentID	ClassID	Semester
71225	1005	Fall21
86634	1006	Spr22
32238	1006	Spr22

School\_Courses

ClassID	Title	ClassNum
1005	Intro to Art History	500
1006	Intro to SQL	501
1009	Intro to Databases	300

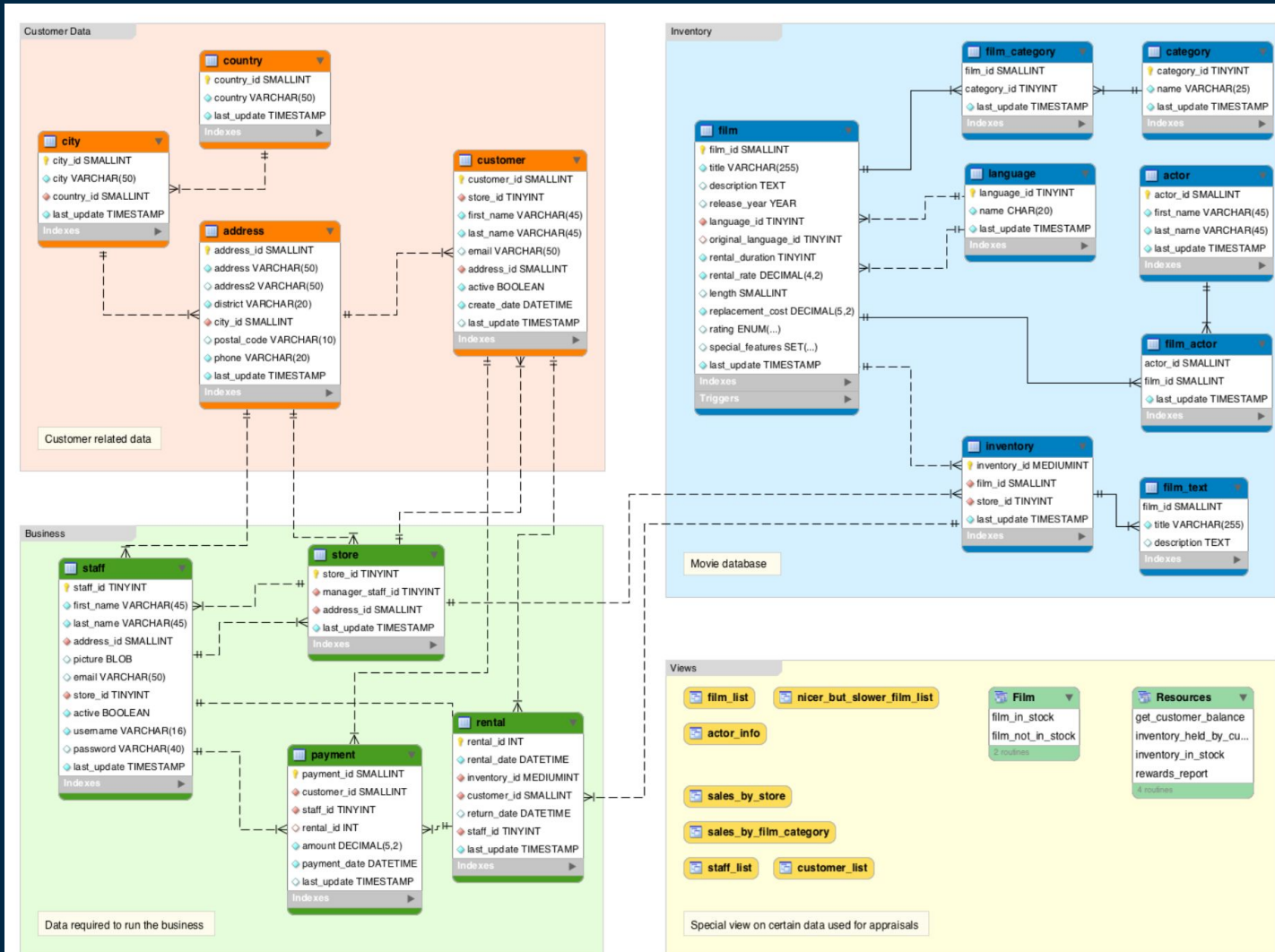
Students

ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0





# Database Schema

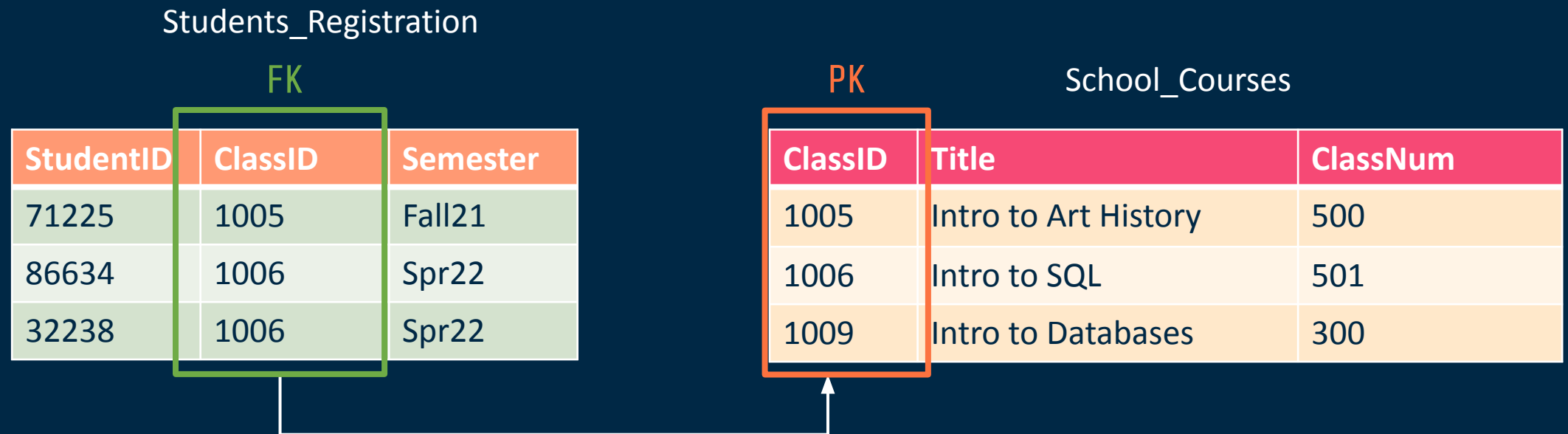


A database schema diagram of the Sakila Sample Database.

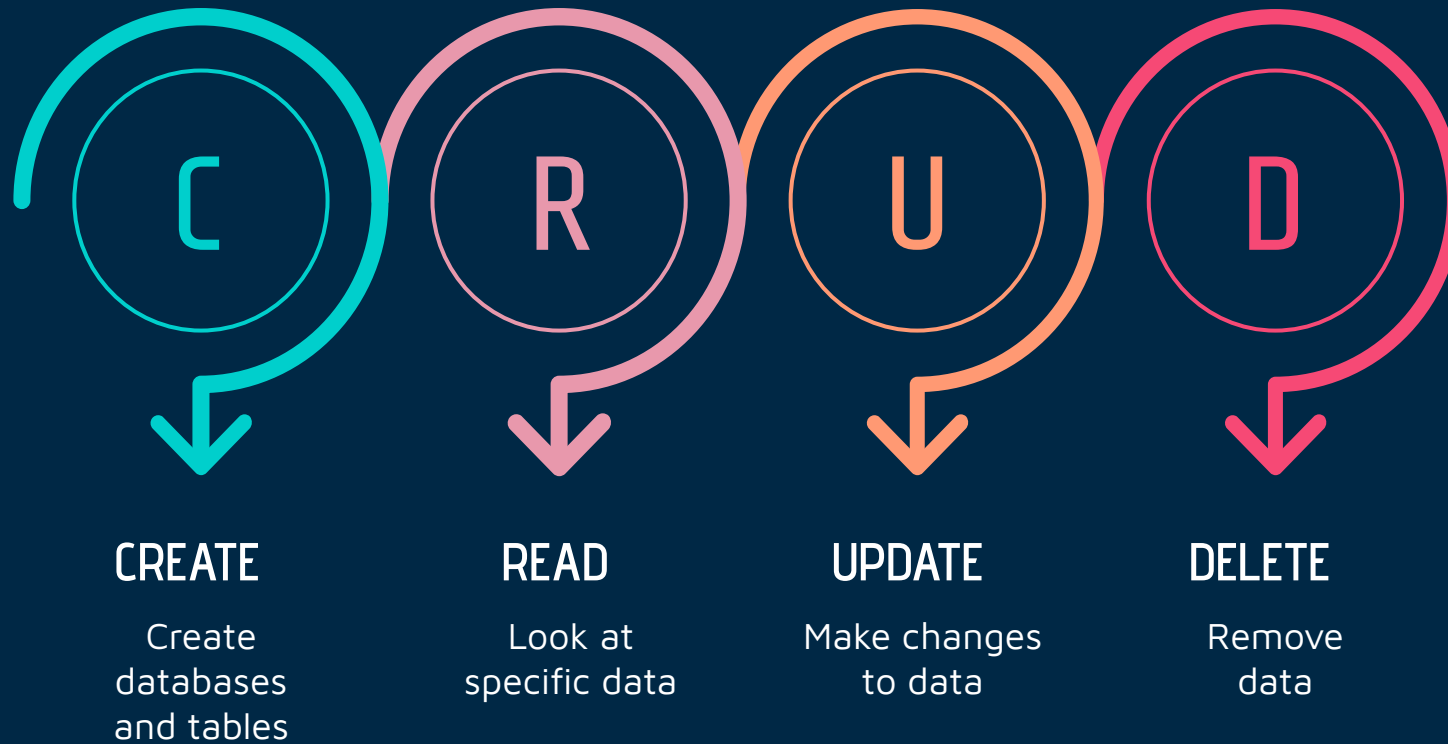
# Primary and Foreign Keys

*Keys* are used to create relationships between tables

- *Primary Keys* (PK) are columns that uniquely identified a row in a table
- *Foreign Keys* (FK) are columns that correspond to the primary key in another table



# SQL Overview



# Vocabulary

- ▶ **Data Definition Language (DDL):**
  - ▶ CREATE, DROP, ALTER, TRUNCATE
- ▶ **Data Manipulation Language (DML):**
  - ▶ INSERT, UPDATE, DELETE
- ▶ **Data Query Language (DQL):**
  - ▶ SELECT, JOIN
- ▶ **Data Control Language (DCL) or Transaction Control Language (TCL):**
  - ▶ GRANT, REVOKE

# SQL Data Types

Students

ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0

- Each column stores only one type of data
- Data types determine the available functions
- Different types of data take up different space

# Integers

Students

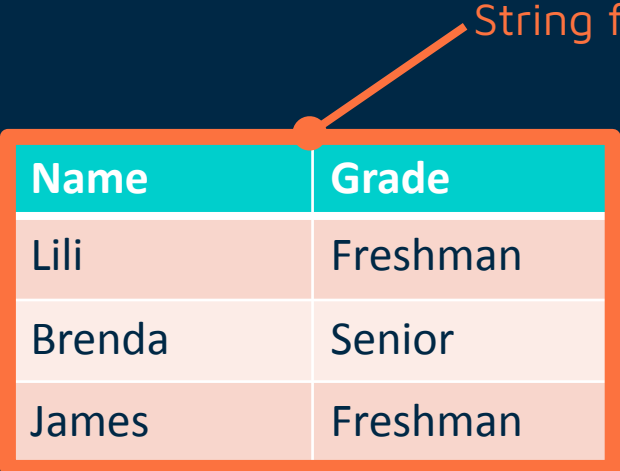
ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0

Integer field

- INT / INTEGER is a signed whole number

# Strings

Students



The diagram shows a table with five columns: ID, Name, Grade, DOB, and GPA. The 'Name' and 'Grade' columns are highlighted with an orange border. An orange arrow points from the text 'String fields' to the 'Name' column header.

ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0

- Can contain letters, numbers and special characters
- `VARCHAR(size)` is a variable length string of maximum size length (up to 65,535)

# Date and Time

Date field

Students

ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0

- Represent temporal values of date, time and datetime values
- DATE with format YYYY-MM-DD
- TIME with format hh:mm:ss
- DATETIME with a combination with format YYYY-MM-DD hh:mm:ss
- MySQL can store years with the YEAR data type with a range of 1901 to 2155

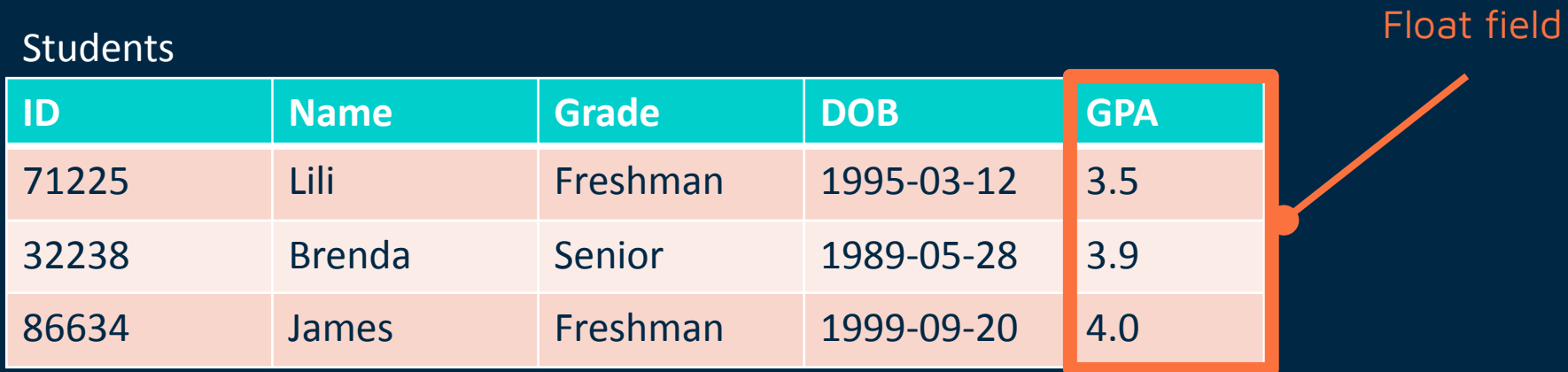


# Floats

Students

ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0

Float field



- Numbers with a fractional part
- In MySQL NUMERIC and DECIMAL are equivalent
- DECIMAL(M, d) has M digits with d of them being digits after decimal point
  - e.g. DECIMAL(5,2) stores any number between -999.99 and 999.99

# Data types summary

Students

ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0

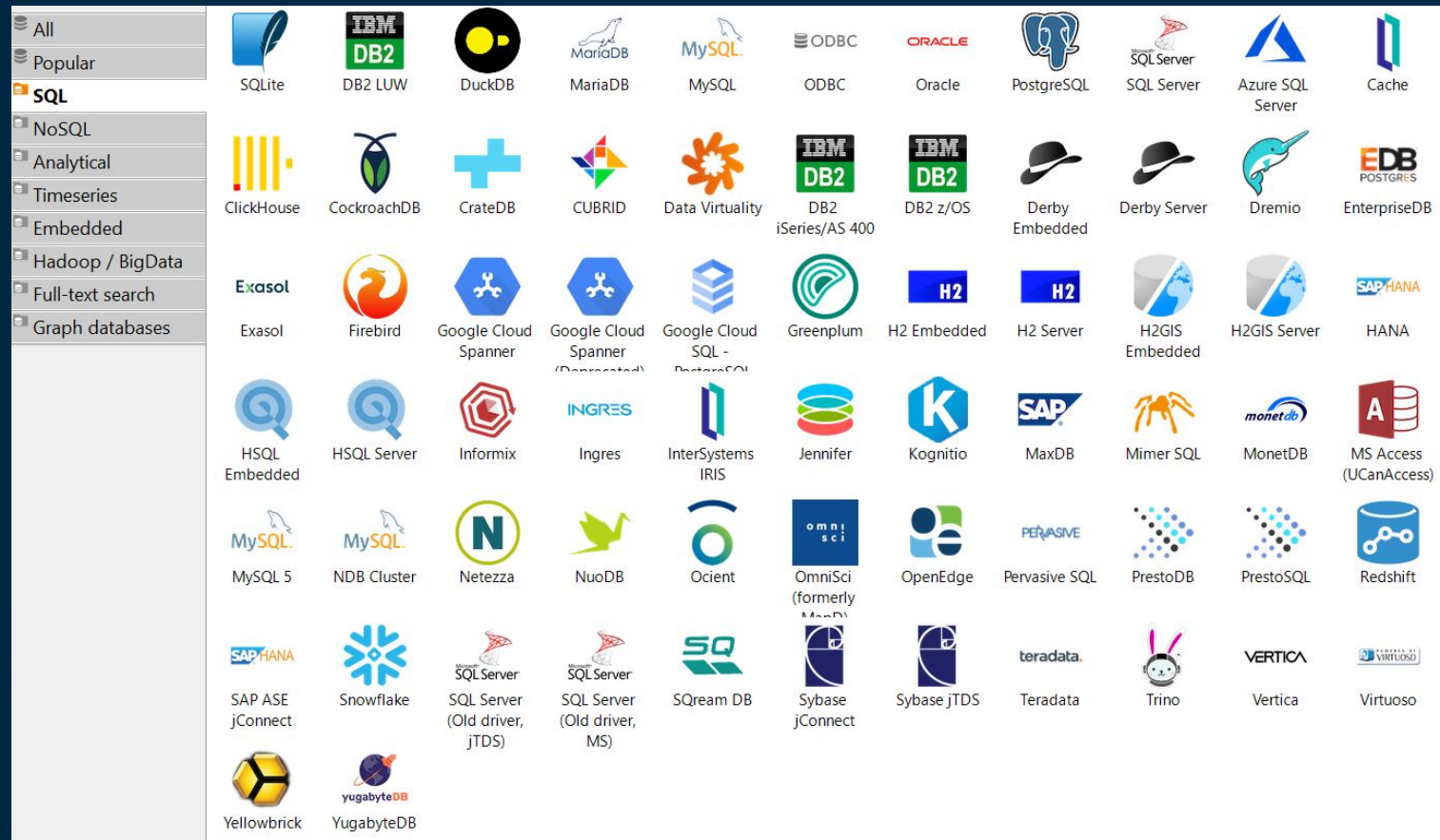
String field

Date fields

Integer field

Float field

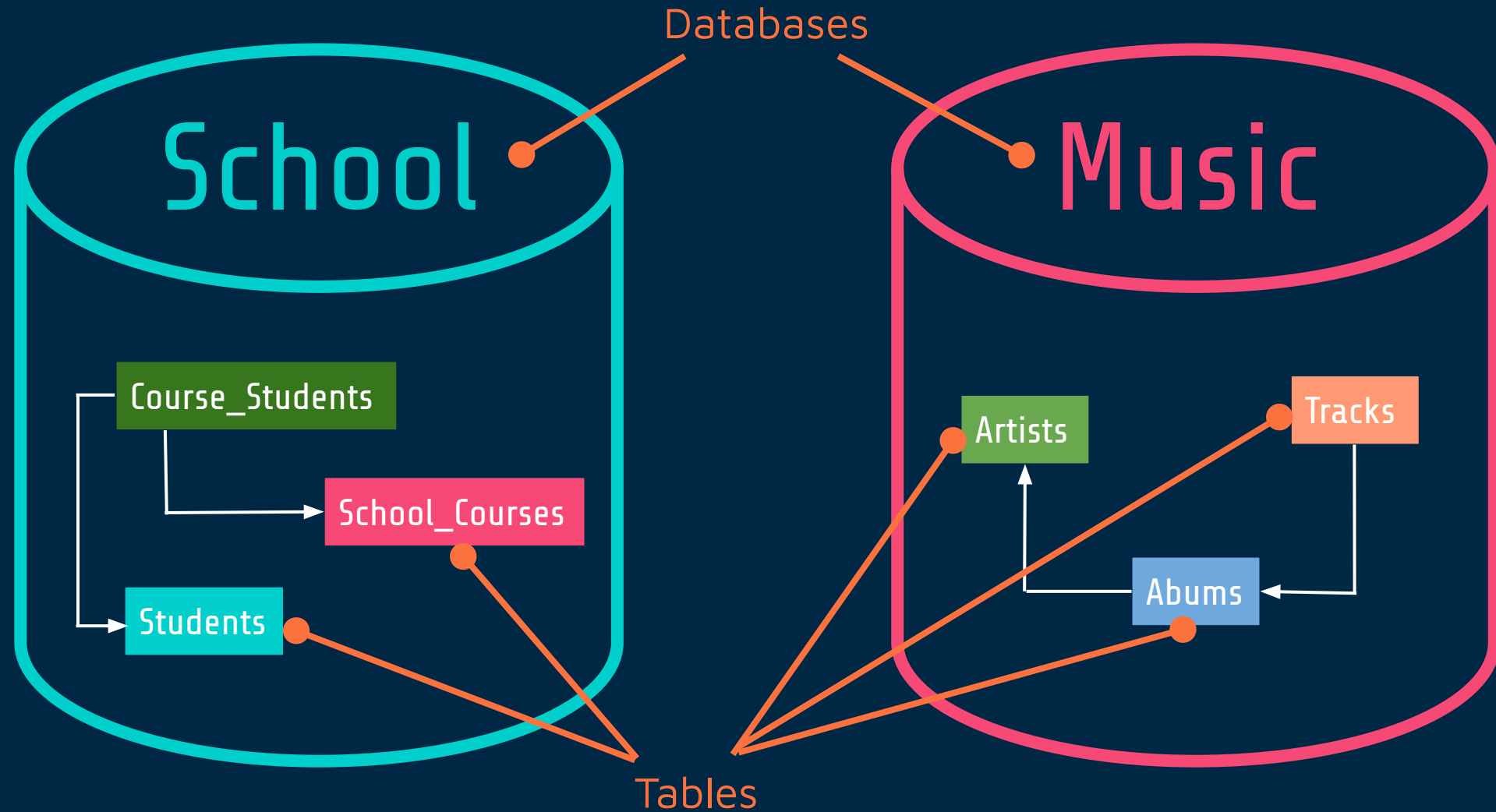
# SQL Flavors



# Tables and Data



# Databases and Tables



# Records and Fields

Students

ID	Name	Grade	DOB	GPA
71225	Lili	Freshman	1995-03-12	3.5
32238	Brenda	Senior	1989-05-28	3.9
86634	James	Freshman	1999-09-20	4.0

Row = Tuple or Record

Holds information for one observation

Column = Attribute or Field

Holds specific information about all observations

# Today's data

- ▶ Modified datasets for workshop (4 files total)
- ▶ **State Crime CSV File**
  - ▶ **state\_crime.csv**
  - ▶ information on the crime rates and totals for states across the United States for a wide range of years
  - ▶ reports go from 1960 to 2019 (only used 2010, 2014 and 2019)
  - ▶ [https://corgis-edu.github.io/corgis/csv/state\\_crime/](https://corgis-edu.github.io/corgis/csv/state_crime/)
- ▶ **State Demographics CSV and SQL Files**
  - ▶ **state\_computer\_data.sql, state\_workforce.csv, state\_people.sql**
  - ▶ summarized information obtained about states in the United States from 2015 through 2019 through the United States Census Bureau
  - ▶ just the summarized data as of 2019
  - ▶ [https://corgis-edu.github.io/corgis/csv/state\\_demographics/](https://corgis-edu.github.io/corgis/csv/state_demographics/)

# Create a Database

```
-- Create a database
CREATE DATABASE QCLWorkshop;

-- Calling the database
USE QCLWorkshop;
```

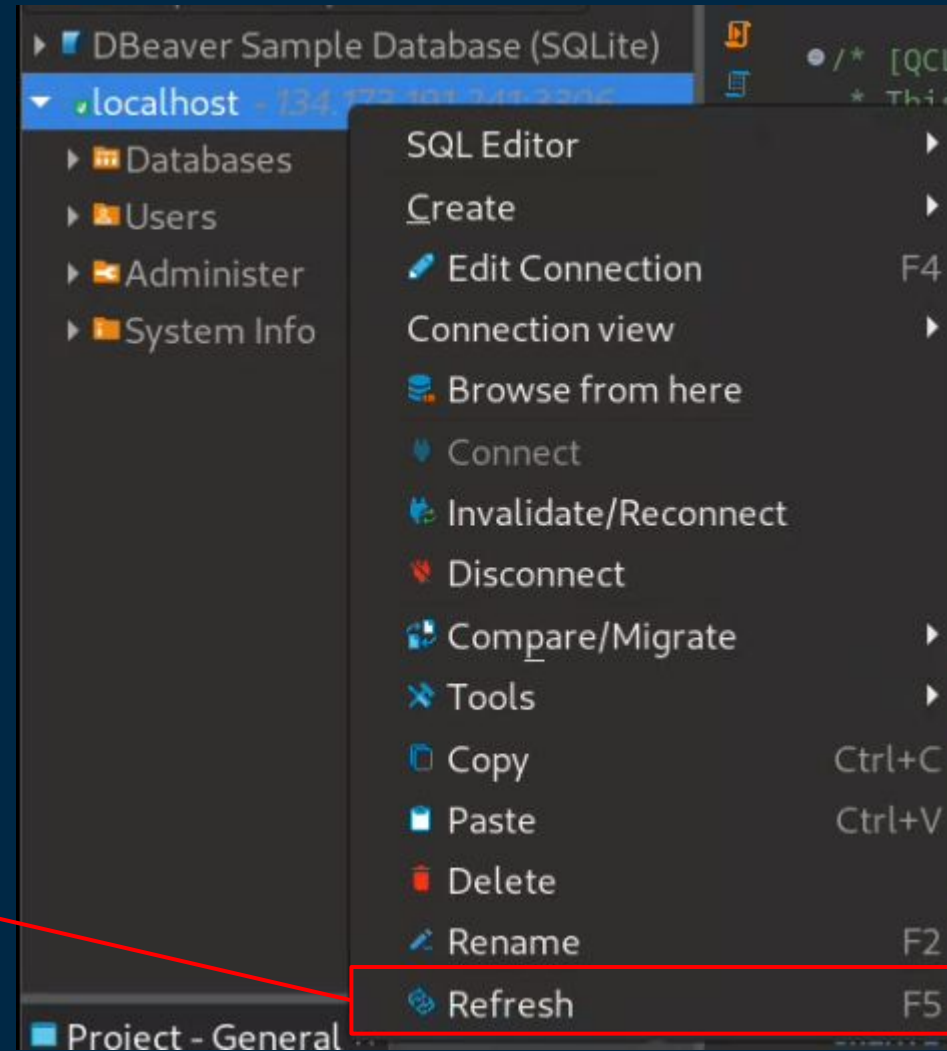
Statistics 1 ×	
Name	Value
Updated Rows	1
Query	-- Create a database CREATE DATABASE QCLWorkshop
Finish time	Thu Oct 20 13:48:51 PDT 2022



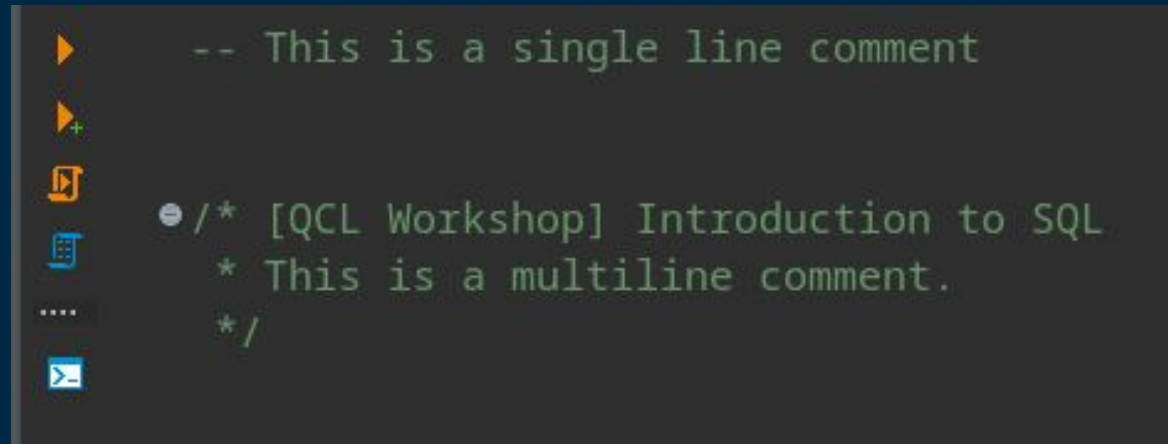
# Refresh

- Right click on your connection and click on refresh

Click on Refresh



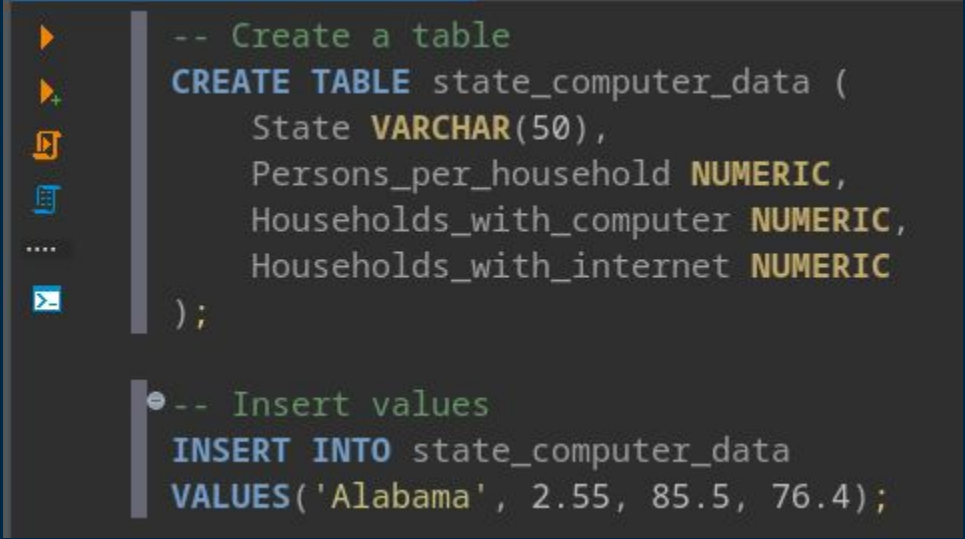
# Comments



```
-- This is a single line comment

/* [QCL Workshop] Introduction to SQL
   * This is a multiline comment.
   */
```

# Create a Table



```
-- Create a table
CREATE TABLE state_computer_data (
  State VARCHAR(50),
  Persons_per_household NUMERIC,
  Households_with_computer NUMERIC,
  Households_with_internet NUMERIC
);

-- Insert values
INSERT INTO state_computer_data
VALUES('Alabama', 2.55, 85.5, 76.4);
```

- After creating the table click on Refresh once more

## An abstract graphic on a dark blue background. It features several geometric shapes: a vertical white line on the left, a small pink rectangle below it, a white square in the lower-left, a small orange square in the center, a white square in the upper-right, and a large orange rectangle in the bottom-right corner.

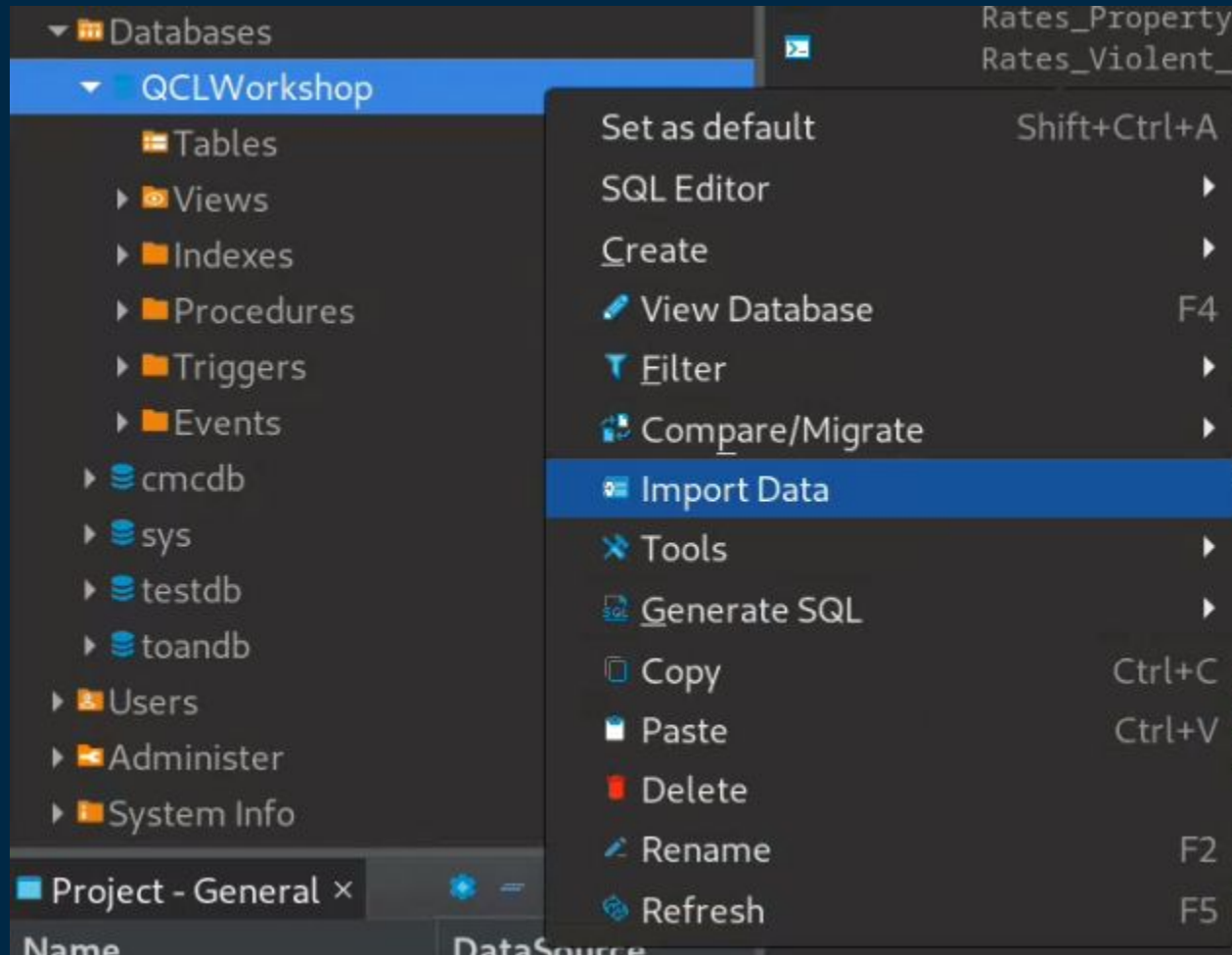




# Hands-on

- ▶ Create a table named `state_people` and add the attributes
  1. State as VARCHAR with 20 characters
  2. Employment\_Firms\_Total as INT
  3. Age\_Percent\_Under\_18\_Years as NUMERIC
  4. Age\_Percent\_65\_and\_Older as NUMERIC
- ▶ Insert data into the new table with the file `state_people_insert.sql`

# Import your data





# Input file

**Data Transfer**

**Input file(s)**  
Configure input files or directories

- ✓ Import source
- ✓ **Input file(s)**
- Tables mapping
- Data load settings
- Confirm

**Input files:**

Source	Target
state_crime.csv	?

**Importer settings:**

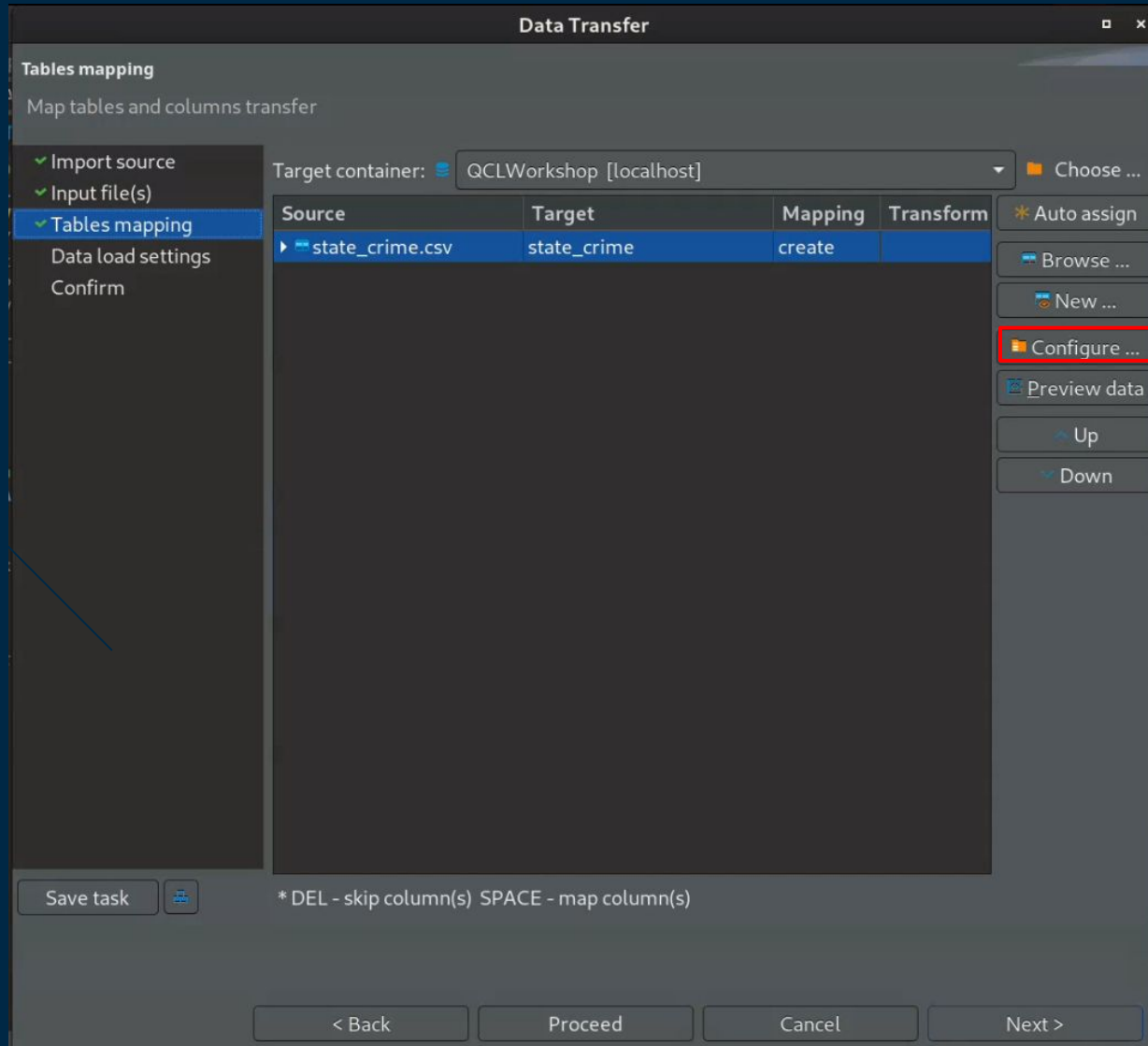
Name	Value
▼ Properties	
Extension	csv,tsv,txt
Encoding	utf-8
Column delimiter	,
Header position	top
Quote char	"
Escape char	\
NULL value mark	
Set empty strings to NULL	[ ]
Date/time format	yyyy-MM-dd[ HH:mm:ss[.SSS]]
Trim whitespaces	[ ]
Timezone ID	

Save task

< Back Proceed Cancel Next >



# Tables mapping



Configure the table mappings

# Configure mappings

Data Transfer

Tables mapping

Configure metadata structure

Columns mapping Table properties Target DDL

Source container: state\_crime.csv

Source entity: state\_crime.csv

Target container: localhost

Target entity: state\_crime

Source Column	Source Type	Target Column	Target Type	Mapping	Transform
State	VARCHAR(50)	State	VARCHAR(255)	new	
Crime_Year	INTEGER	Crime_Year	INTEGER	new	
Population	INTEGER	Population	INTEGER	new	
Rates_Property	REAL	Rates_Property	NUMERIC	new	
Rates_Violent_	REAL	Rates_Violent_	NUMERIC	new	
Totals_Property	INTEGER	Totals_Property	INTEGER	new	
Totals_Violent_	INTEGER	Totals_Violent_	INTEGER	new	

Cancel OK

Save task

\* DEL - skip column(s) SPACE - map column(s)

< Back Proceed Cancel Next >

Choose ...

\* Auto assign

Browse ...

New ...

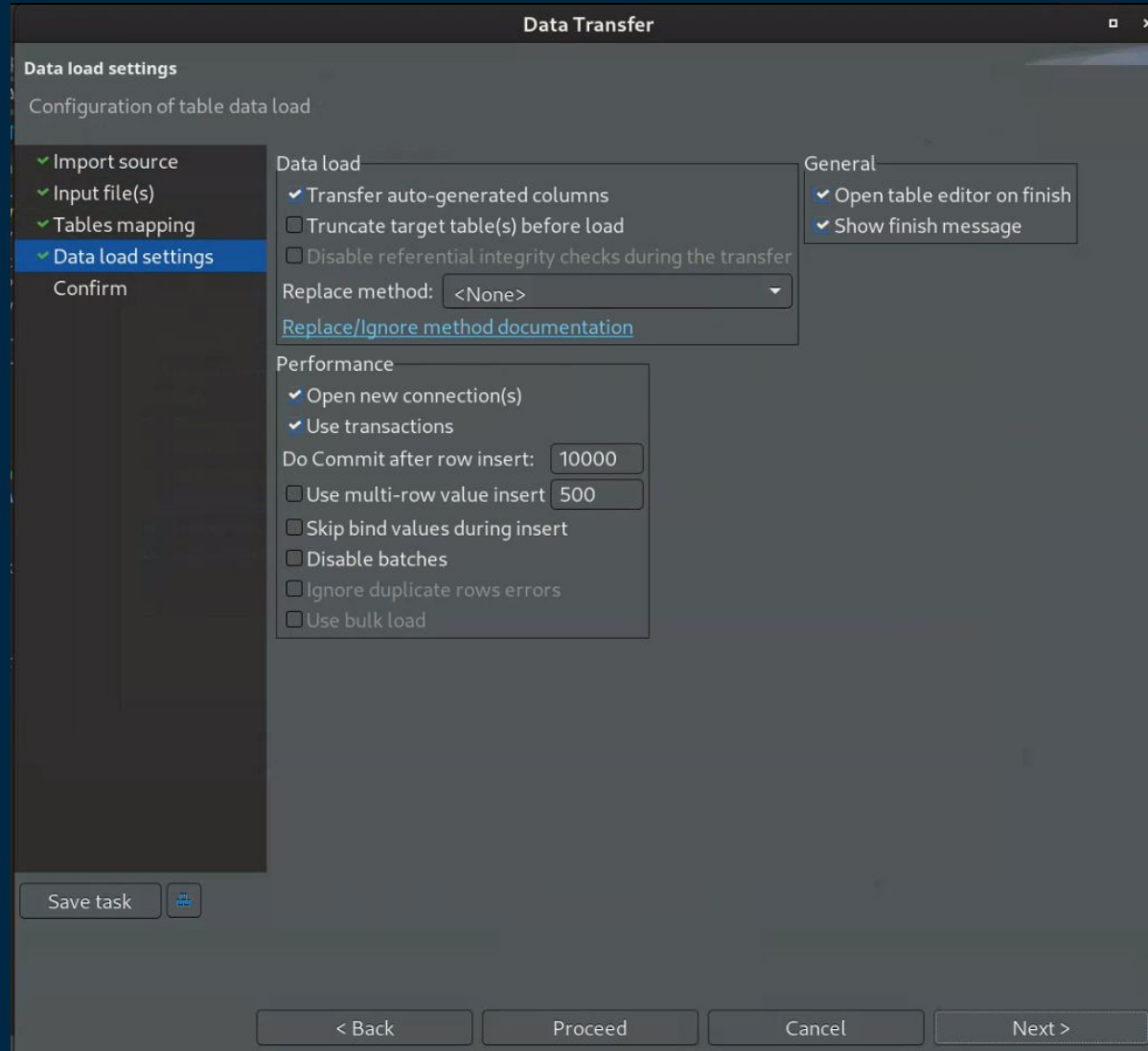
Configure ...

Preview data

Up

Down

# Data load settings



The image shows a 'Data Transfer' dialog box with a 'Data load settings' tab selected. The dialog is divided into three main sections: 'Data load', 'Performance', and 'General'. The 'Data load' section includes options for transferring auto-generated columns, truncating target tables, and disabling referential integrity checks. The 'Performance' section includes options for opening new connections, using transactions, and setting commit intervals. The 'General' section includes options for opening the table editor and showing finish messages. At the bottom, there are buttons for 'Save task', '< Back', 'Proceed', 'Cancel', and 'Next >'.

**Data Transfer**

**Data load settings**  
Configuration of table data load

**Data load**

- ☒ Transfer auto-generated columns
- ☐ Truncate target table(s) before load
- ☐ Disable referential integrity checks during the transfer
- Replace method:
- [Replace/Ignore method documentation](#)

**Performance**

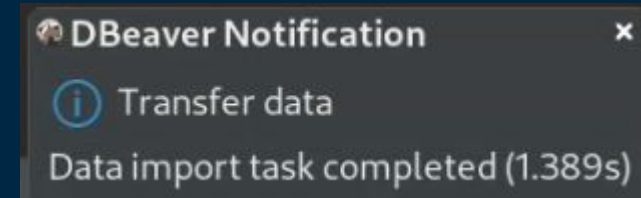
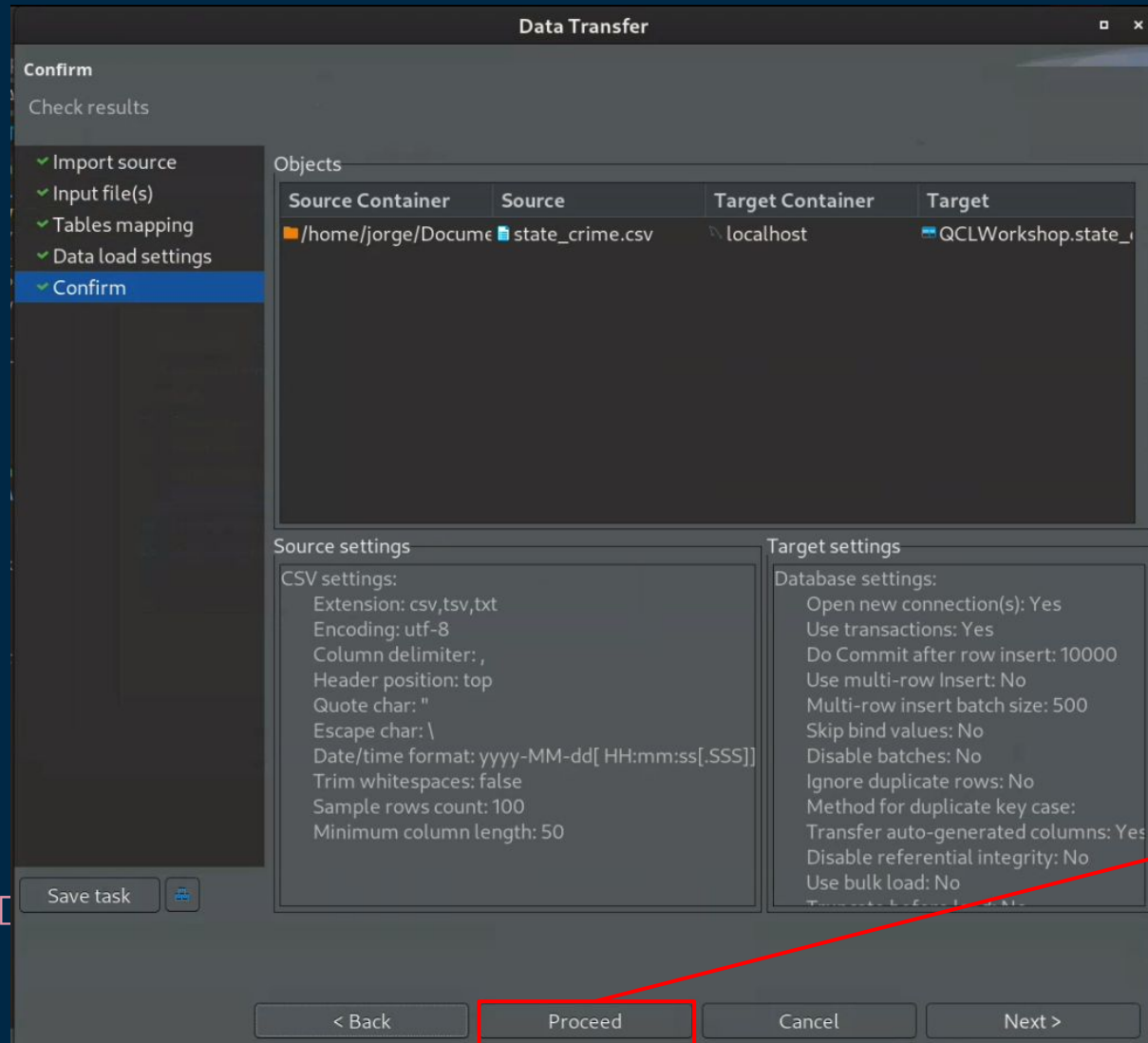
- ☒ Open new connection(s)
- ☒ Use transactions
- Do Commit after row insert:
- ☐ Use multi-row value insert
- ☐ Skip bind values during insert
- ☐ Disable batches
- ☐ Ignore duplicate rows errors
- ☐ Use bulk load

**General**

- ☒ Open table editor on finish
- ☒ Show finish message

**Confirm**

# Confirmation



Click on "Proceed"

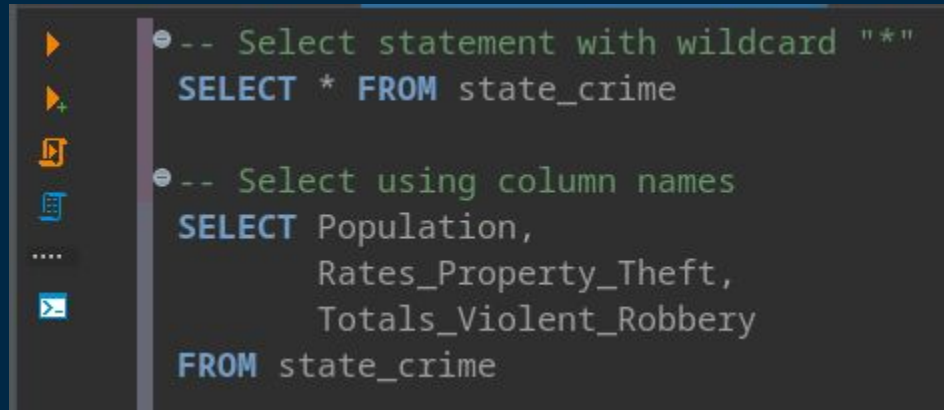


# Hands-on

- ▶ Import the file named `state_workforce`



# Retrieving Data



```
-- Select statement with wildcard "*"
SELECT * FROM state_crime

-- Select using column names
SELECT Population,
       Rates_Property_Theft,
       Totals_Violent_Robbery
FROM state_crime
```

- Wildcard selects all columns from the given table
- You can also specify the columns by name

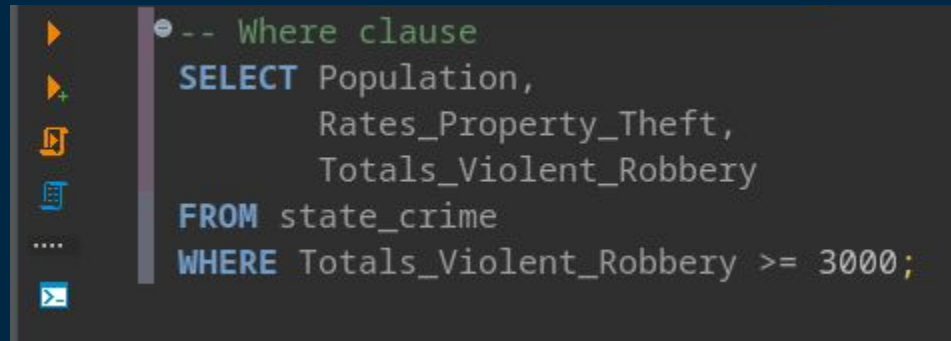


# Hands-on

- ▶ Retrieve the data from the `state_workforce` table
  1. How many rows and attributes does this table have?



# Filtering

A screenshot of a SQL code editor with a dark background. On the left, there is a vertical toolbar with icons for running queries, saving, and other functions. The main area contains SQL code with syntax highlighting: a comment '-- Where clause' in green, followed by 'SELECT' in blue, then the column names 'Population', 'Rates\_Property\_Theft', and 'Totals\_Violent\_Robbery' in white. Below these is 'FROM state\_crime' in blue, and finally 'WHERE Totals\_Violent\_Robbery >= 3000;' in white.

```
-- Where clause
SELECT Population,
       Rates_Property_Theft,
       Totals_Violent_Robbery
FROM state_crime
WHERE Totals_Violent_Robbery >= 3000;
```

- Used to filter records based on a condition
- Use **AND** | **OR** operators to use multiple conditions

# Filtering Operators

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal
NOT	Negates the boolean value
BETWEEN	Whether a value is within a range
LIKE	Search for a pattern

- These are only some of the operators
- You can find all the operators here:
  - <https://dev.mysql.com/doc/refman/8.0/en/non-typed-operators.html>

# Wildcards for LIKE

Wildcard	Description
%	Represents zero or more characters
_	Represents a single character

- Not all flavors of SQL support the same wildcards
- Take longer to run compared with using other operators

# Starts with

The screenshot shows the DBeaver 22.2.2 interface. The left sidebar displays the 'Database Navigator' with a tree view of databases and tables. The 'state\_crime' table is selected. The main editor shows a SQL script with three queries. The first query is highlighted in blue. The bottom panel shows the results of the first query in a table grid.

**SQL Script:**

```
/* Using "%" wildcard with LIKE operator
 * Grab anything that starts with "South" */
SELECT State,
       Population
FROM state_crime
WHERE State LIKE 'South %';

-- Grab anything that ends with "Carolina"
SELECT State,
       Population
FROM state_crime
WHERE State LIKE '% Carolina';

-- Grab anything that contains
SELECT State,
       Population
FROM state_crime
WHERE State LIKE '% of %';
```

**Results Table:**

State	Population
South Carolina	5,148,714
South Dakota	884,659
South Carolina	4,637,106
South Dakota	816,598
South Carolina	4,832,482
South Dakota	853,175

**Footer:** PST en\_US Writable Smart Insert 3:1 [76] Sel: 76 | 4

# Ends with

DBeaver 22.2.2 - <localhost> Script-1

File Edit Navigate Search SQL Editor Database Window Help

localhost QCLWorkshop

Database Navigator Projects

Enter a part of object name here

DBeaver Sample Database (SQLite)

localhost - 134.173.191.241:3306

Databases

QCLWorkshop

Tables

state\_crime

Views

Indexes

Procedures

Triggers

Events

cmcdb

sys

testdb

toandb

Users

Administer

Project - General

Name DataSource

Bookmarks

Diagrams

Scripts

```
/* Using "%" wildcard with LIKE operator
 * Grab anything that starts with "South" */
SELECT State,
       Population
FROM state_crime
WHERE State LIKE 'South %';

-- Grab anything that ends with "Carolina"
SELECT State,
       Population
FROM state_crime
WHERE State LIKE '% Carolina';

-- Grab anything that contains
SELECT State,
       Population
FROM state_crime
WHERE State LIKE '% of %';
```

state\_crime 1 x

SELECT State, Population FROM state\_crime

	State	Population
1	North Carolina	10,488,084
2	South Carolina	5,148,714
3	North Carolina	9,560,234
4	South Carolina	4,637,106
5	North Carolina	9,943,964
6	South Carolina	4,832,482

6 row(s) fetched - 21ms, on 2022-10-20 at 23:01:33

PST en\_US Writable Smart Insert 9:1 [79] Sel: 79 | 4

# Contains

The screenshot displays the DBeaver 22.2.2 interface. The top menu bar includes File, Edit, Navigate, Search, SQL Editor, Database, Window, and Help. The toolbar contains icons for SQL, Commit, Rollback, and Auto. The Database Navigator on the left shows a tree structure with 'DBeaver Sample Database (SQLite)' expanded, revealing 'localhost - 134.173.191.241:3306' and its contents: Databases (QCLWorkshop, Tables, state\_crime), Views, Indexes, Procedures, Triggers, Events, cmcdb, sys, testdb, toandb, Users, and Administer. The SQL Editor in the center contains three SQL queries: 1. A comment about using a wildcard with the LIKE operator, followed by a query to select State and Population from state\_crime where State starts with 'South'. 2. A comment about grabbing anything that ends with 'Carolina', followed by a query to select State and Population from state\_crime where State ends with 'Carolina'. 3. A comment about grabbing anything that contains, followed by a query to select State and Population from state\_crime where State contains 'of'. The bottom panel shows a data grid for 'state\_crime 1' with columns State and Population. The grid contains three rows of data for the District of Columbia. The status bar at the bottom indicates '3 row(s) fetched - 21ms, on 2022-10-20 at 23:01:51' and shows the current selection as 'Sel: 75 | 4'.

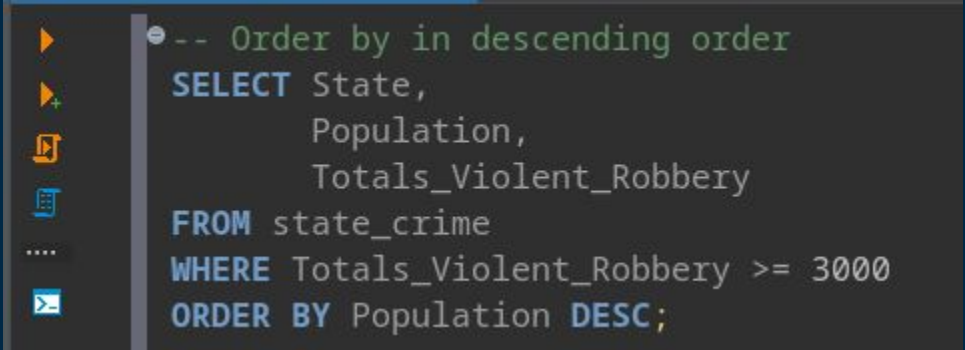
```
/* Using "%" wildcard with LIKE operator
 * Grab anything that starts with "South" */
SELECT State,
       Population
FROM state_crime
WHERE State LIKE 'South %';

-- Grab anything that ends with "Carolina"
SELECT State,
       Population
FROM state_crime
WHERE State LIKE '% Carolina';

-- Grab anything that contains
SELECT State,
       Population
FROM state_crime
WHERE State LIKE '% of %';
```

State	Population
District of Columbia	705,749
District of Columbia	604,912
District of Columbia	658,893

# Sorting

A screenshot of a SQL query editor interface. On the left is a vertical toolbar with icons for running a query, saving, and other functions. The main area contains a SQL query. The query starts with a comment '-- Order by in descending order'. It then uses a SELECT statement to retrieve 'State', 'Population', and 'Totals\_Violent\_Robbery' from a table named 'state\_crime'. A WHERE clause filters for 'Totals\_Violent\_Robbery >= 3000'. Finally, an ORDER BY clause sorts the results by 'Population' in descending order, indicated by the keyword 'DESC'.

```
-- Order by in descending order
SELECT State,
       Population,
       Totals_Violent_Robbery
FROM state_crime
WHERE Totals_Violent_Robbery >= 3000
ORDER BY Population DESC;
```

- Sorts the records by the given field in ascending order by default
- ASC | DESC for ascending or descending order



# Hands-on

- ▶ Find out how many people on average take longer than 20 mins to get to work
- ▶ Sort the results to find out what state has the longest on average time it takes to get to work
- ▶ Modify your query to show only the records of New York, New Jersey, New Hampshire and New Mexico

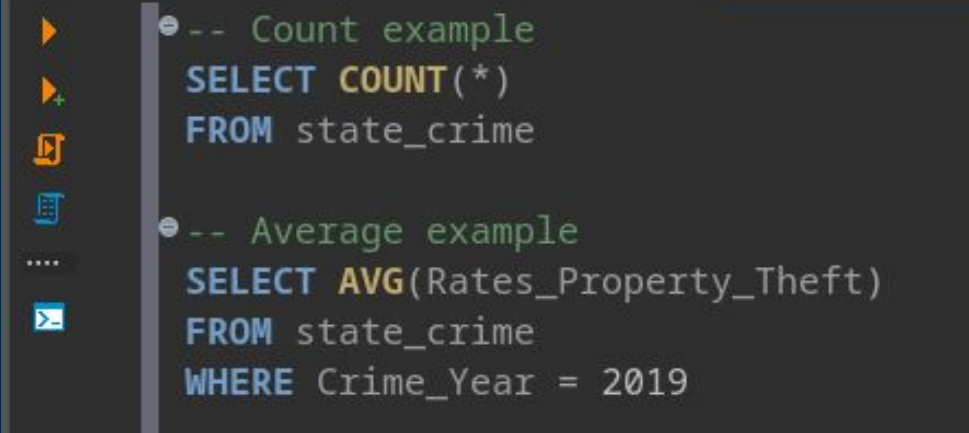


# Aggregate Functions

Function	Description	
COUNT()	Counts the number of records	Various data types
MAX()	Get maximum value	
MIN()	Get minimum value	
SUM()	Sums the field values	Numerical fields only
AVG()	Average of column values	

- Aggregate functions are used to summarize data

# Syntax examples

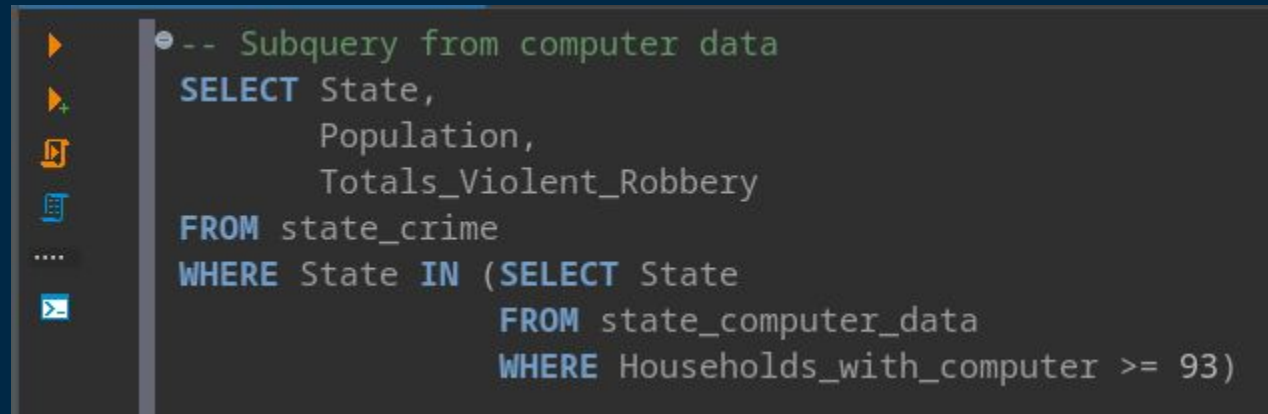


```
-- Count example
SELECT COUNT(*)
FROM state_crime

-- Average example
SELECT AVG(Rates_Property_Theft)
FROM state_crime
WHERE Crime_Year = 2019
```



# Subqueries

A screenshot of a SQL IDE window with a dark background. On the left is a vertical toolbar with icons for running queries, saving, and other database functions. The main area contains SQL code for a subquery. The code starts with a comment '-- Subquery from computer data', followed by a SELECT statement with columns State, Population, and Totals\_Violent\_Robbery from the table state\_crime. The WHERE clause uses an IN operator with a subquery in parentheses: (SELECT State FROM state\_computer\_data WHERE Households\_with\_computer >= 93).

```
-- Subquery from computer data
SELECT State,
       Population,
       Totals_Violent_Robbery
FROM state_crime
WHERE State IN (SELECT State
                FROM state_computer_data
                WHERE Households_with_computer >= 93)
```

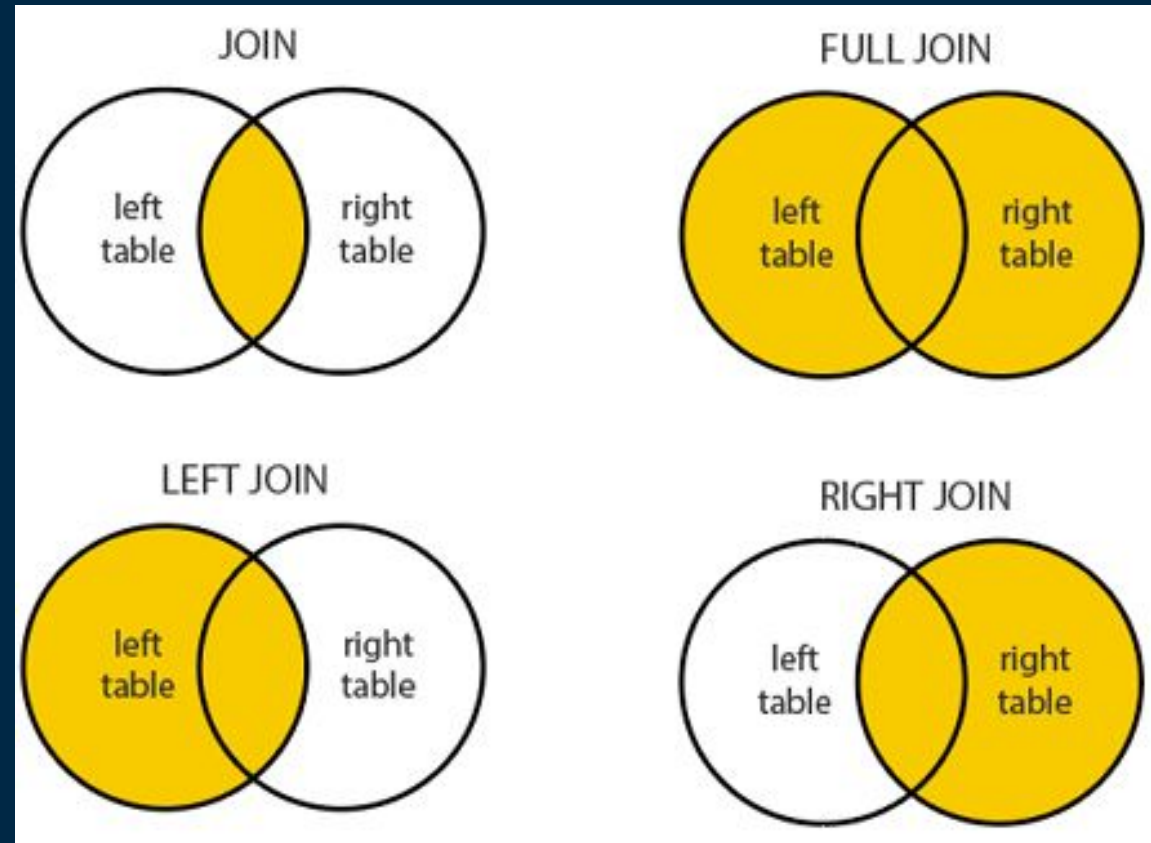
- Useful when you want to combine information from different tables
- SQL performs the innermost query first



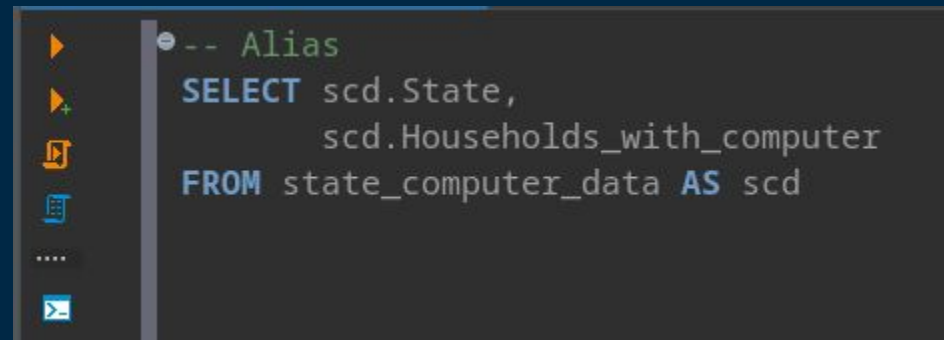
# Hands-on

- ▶ Find the maximum percentage of the people with education of high school or higher from states where either the rate of property theft is above 2,500 or the rate of violent robbery is above 30 during 2014

# SQL Joins



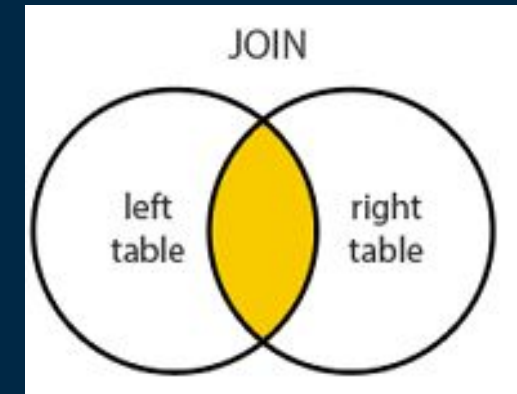
# Aliases



```
-- Alias
SELECT scd.State,
       scd.Households_with_computer
FROM state_computer_data AS scd
```

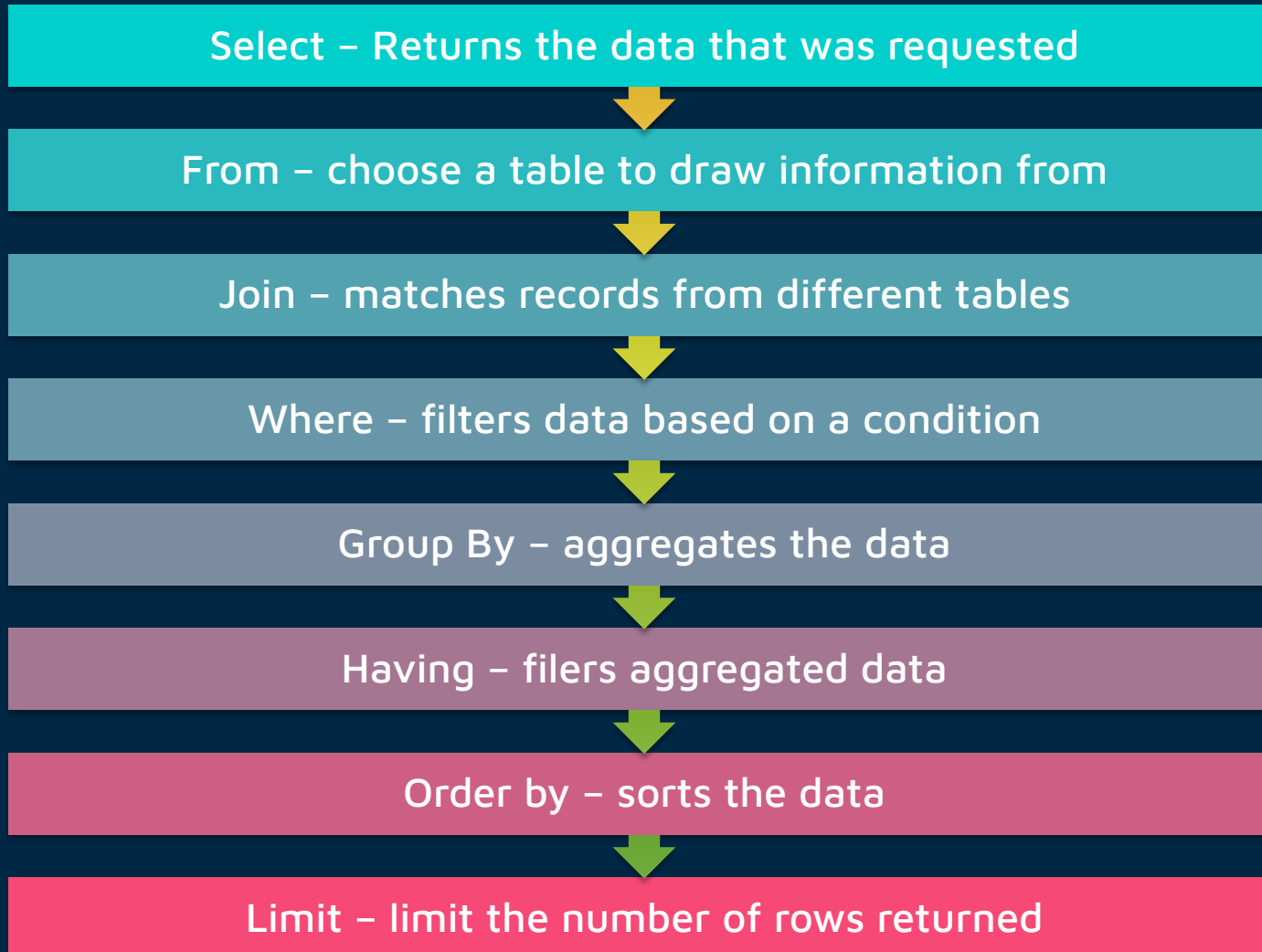
# (Inner) Join

```
-- Inner join
SELECT scd.State,
       scd.Households_with_computer,
       sc.Totals_Property_Theft,
       sc.Totals_Violent_Robbery
FROM state_computer_data AS scd
JOIN state_crime AS sc
ON scd.State = sc.State
```





# Operation order

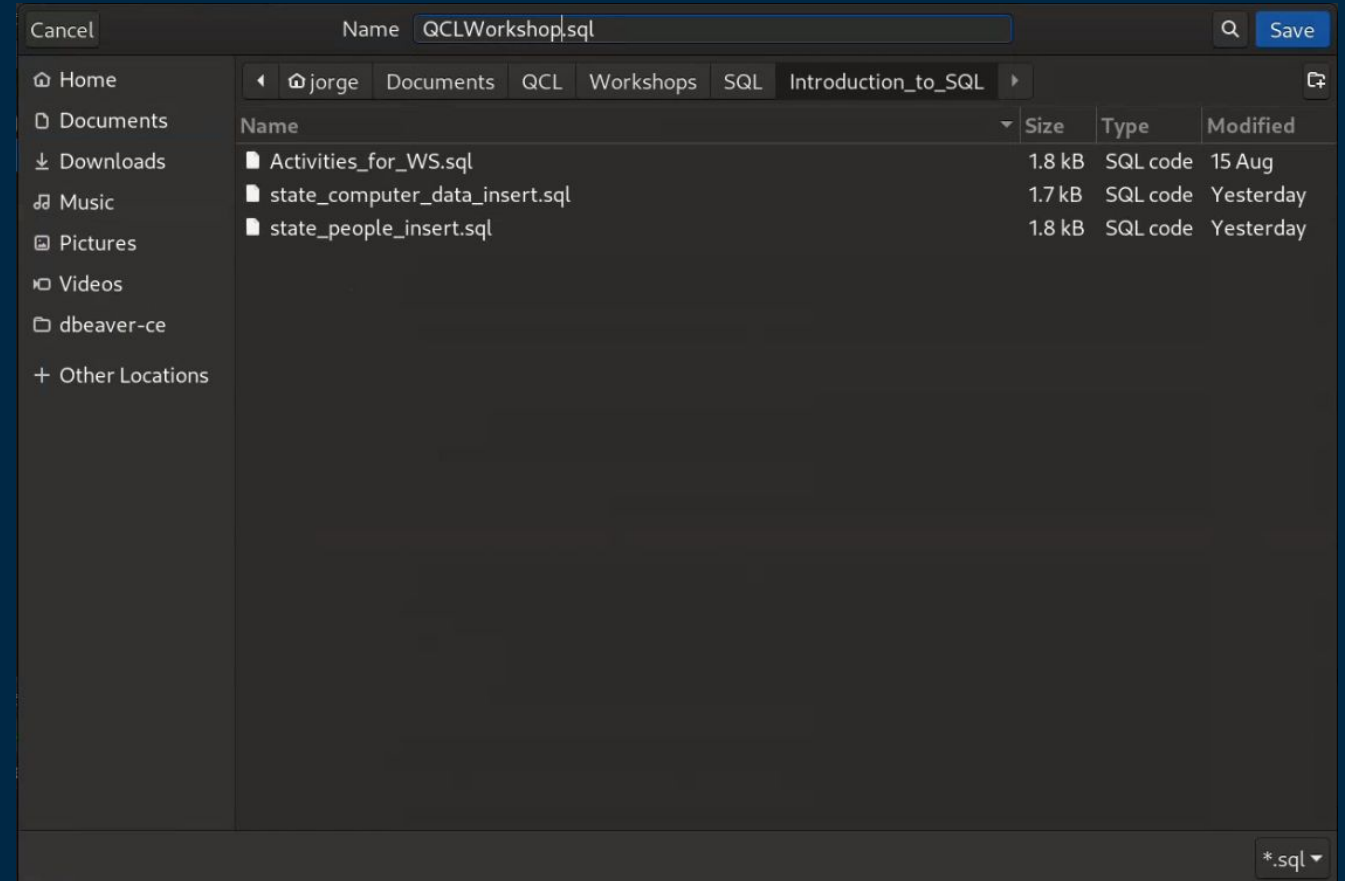
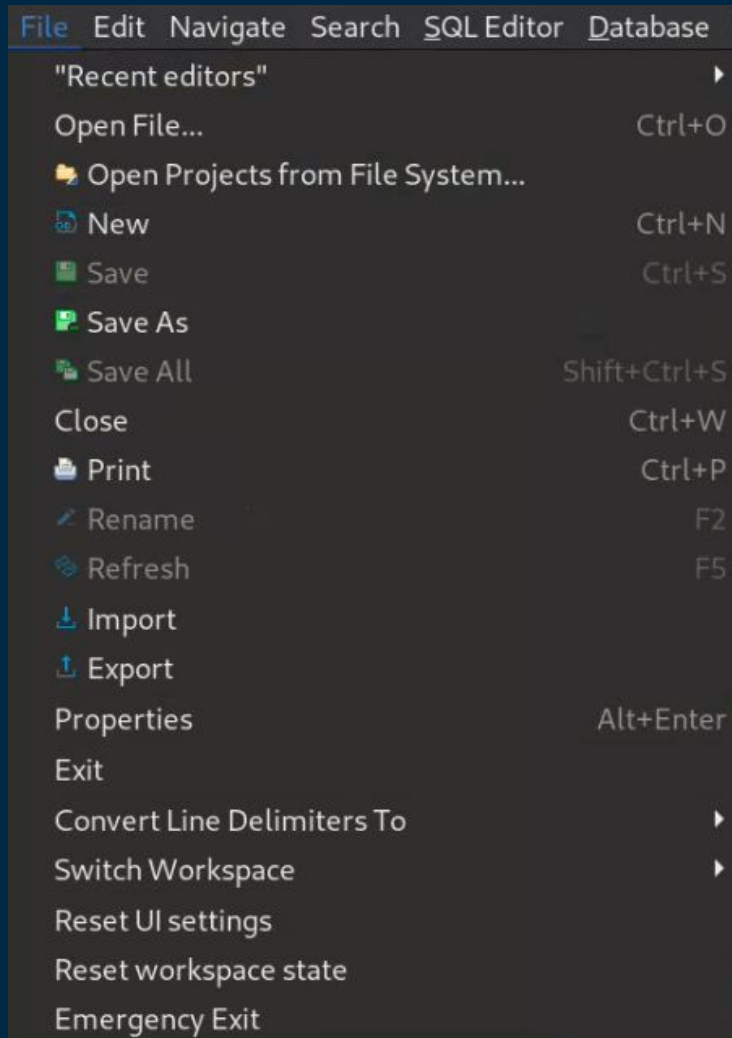




# Hands-on

- ▶ Create an inner join using aliases with tables `state_workforce` and `state_people`. Make sure to view attributes:
  1. State
  2. Mean\_Travel\_Time\_to\_Work
  3. Employment\_Firms\_Total

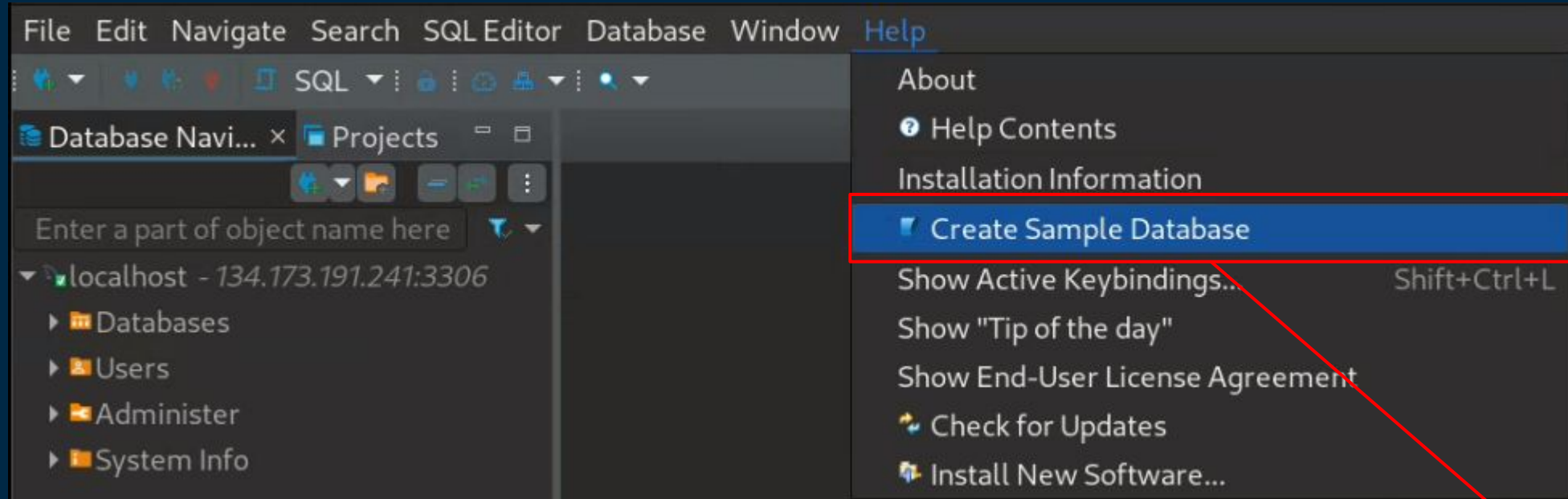
# Save your progress



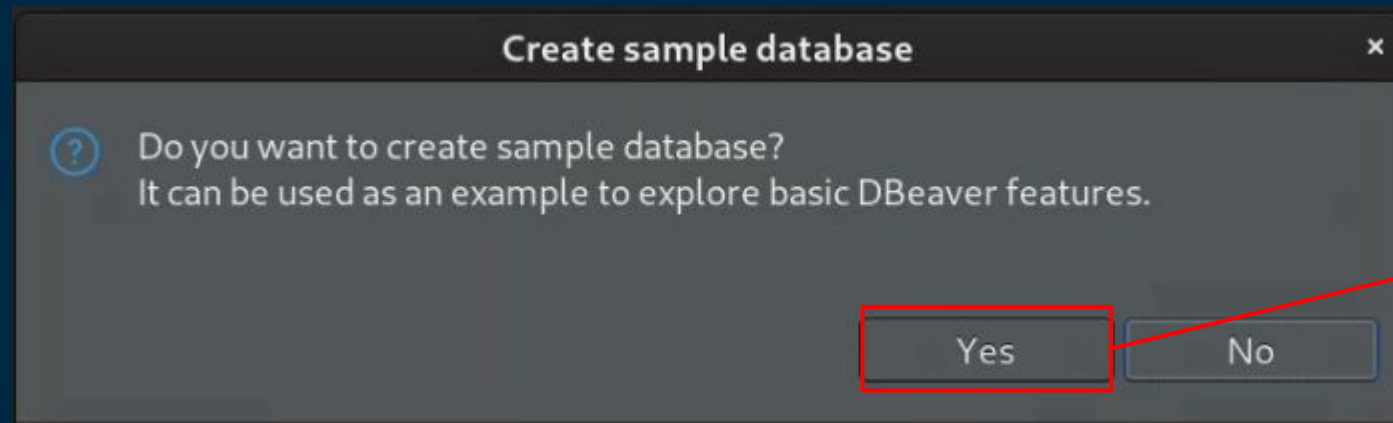
# Send your results

- Finish today's hands-on activities
- Digital badge:
  - Create a sample database
  - How many artist collaborations are there in the Artist table? (keyword is "Feat.")
  - Show a table with the Artist name and their Album's titles as the only columns
  - What are the top 3 Albums with the most tracks?
- Send your hands-on activities and digital badge activities to
  - [qcl@cmc.edu](mailto:qcl@cmc.edu)

# Create Sample Database



Create a sample database



Confirmation

# Resources

- Dbeaver Wiki - <https://github.com/dbeaver/dbeaver/wiki>
- W3schools - <https://www.w3schools.com/sql/>
- Codecademy - <https://www.codecademy.com/learn/learn-sql>

Best way to learn

- SQL Murder Mystery - <https://mystery.knightlab.com/>