

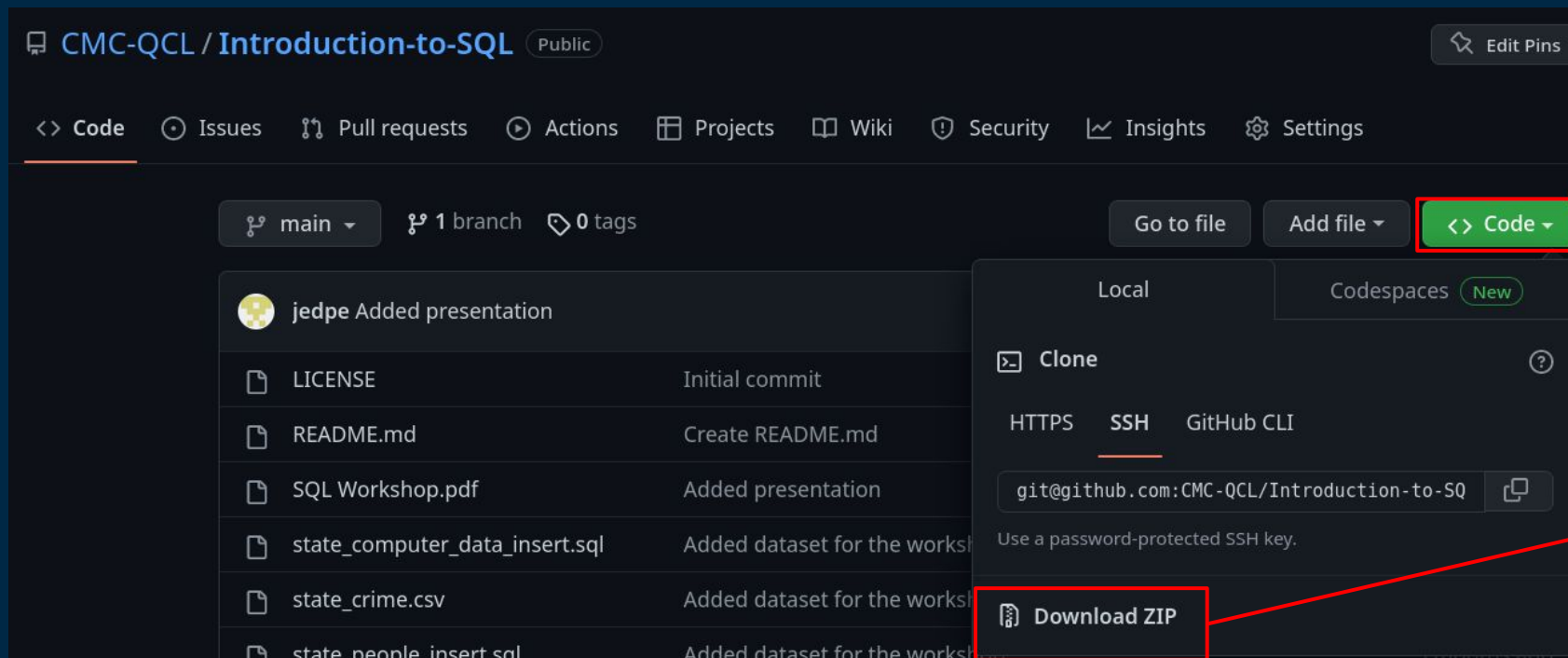


# Introduction to SQL

Jorge Peña  
QCL Graduate Fellow

# Before we start

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



Click on the "Code" button

Download it as a ZIP file

# 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:

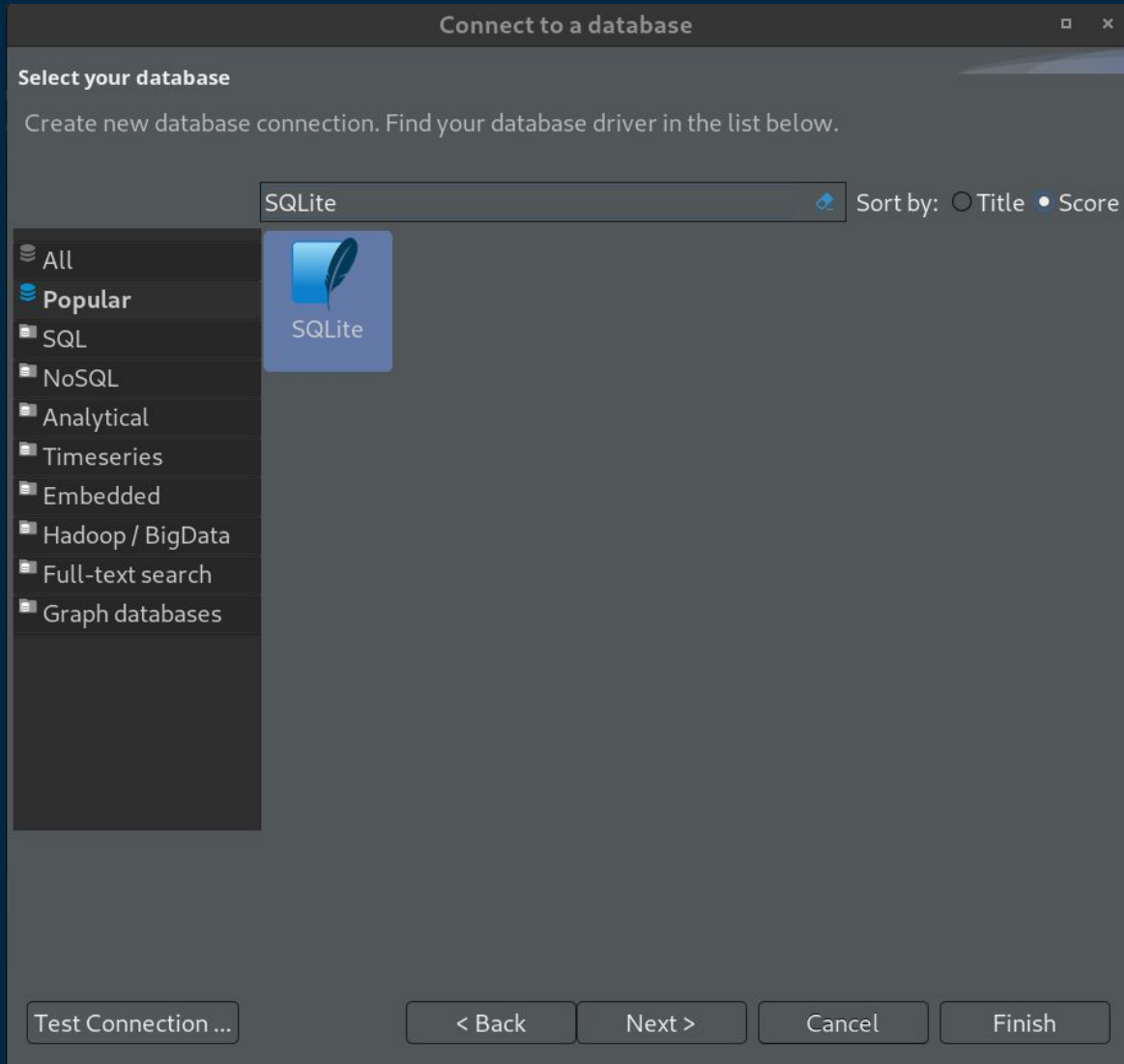
- Database Navigator:** Located on the left, it shows a tree view of the database structure. The 'Databases' folder is expanded, showing 'JorgeDB', 'MyDB', and 'QCLWorkshop'. Under 'QCLWorkshop', the 'Tables' folder is expanded, listing 'state\_computer\_data', 'state\_crime', 'state\_people', and 'state\_workforce'. The 'state\_crime' table is selected.
- SQL Editor:** The central panel shows the SQL query: `SELECT * FROM state_crime;`
- Results View:** The bottom panel displays the results of the query in a table format. The table has 9 columns: State, Crime\_Year, Population, Rates\_Property\_Theft, Rates\_Violent\_Robbery, Totals\_Property\_Theft, Totals\_Violent\_Robbery, and two unnamed columns. The data is sorted by State.

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

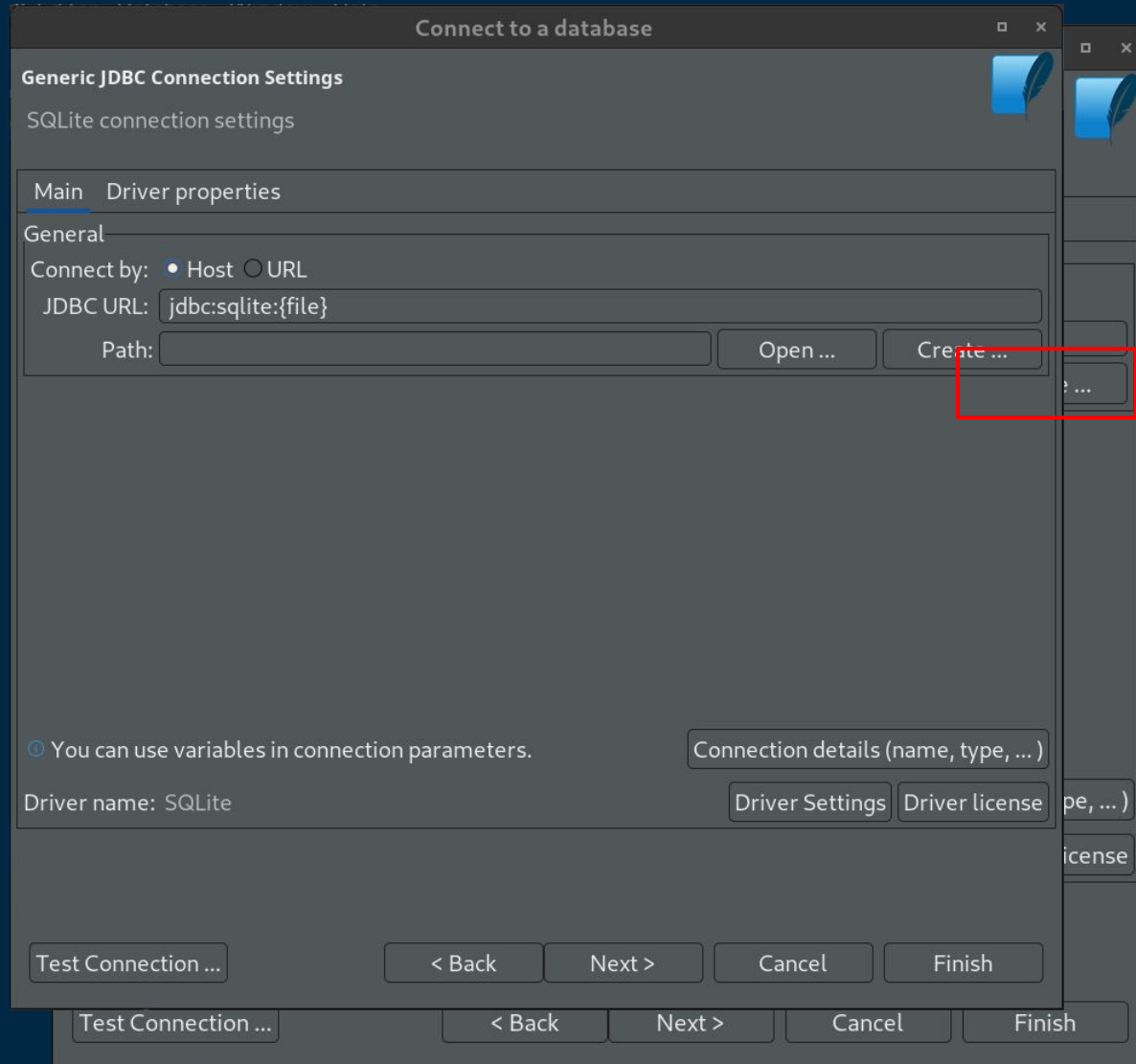
	State	Crime_Year	Population	Rates_Property_Theft	Rates_Violent_Robbery	Totals_Property_Theft	Totals_Violent_Robbery
1	Alabama	2,019	4,903,185	532	80	26,079	
2	Alaska	2,019	731,545	487	113	3,563	
3	Arizona	2,019	7,278,717	394	88	28,699	
4	Arkansas	2,019	3,017,804	600	52	18,095	
5	California	2,019	39,512,223	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 SQLite



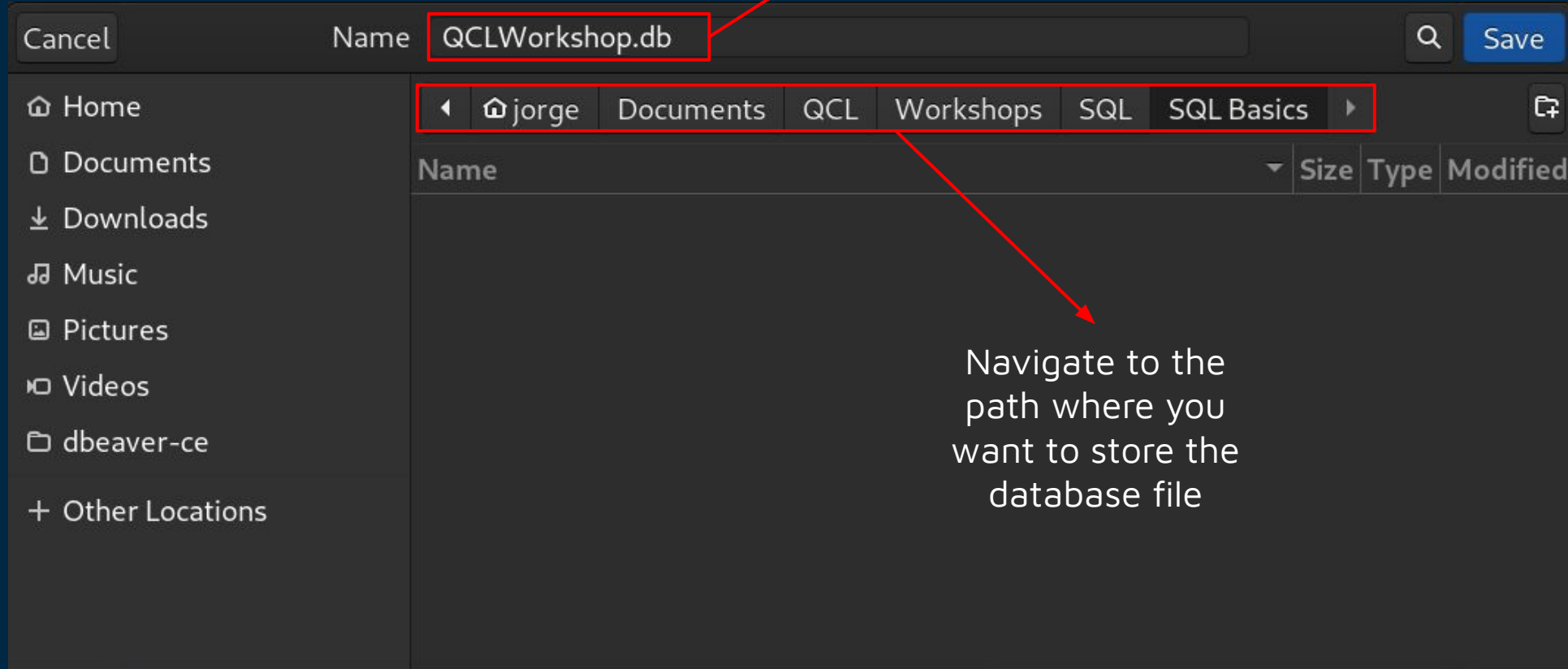
# Create a Database



Click on "Create". This will open a file dialog window to select where you want to store your database file

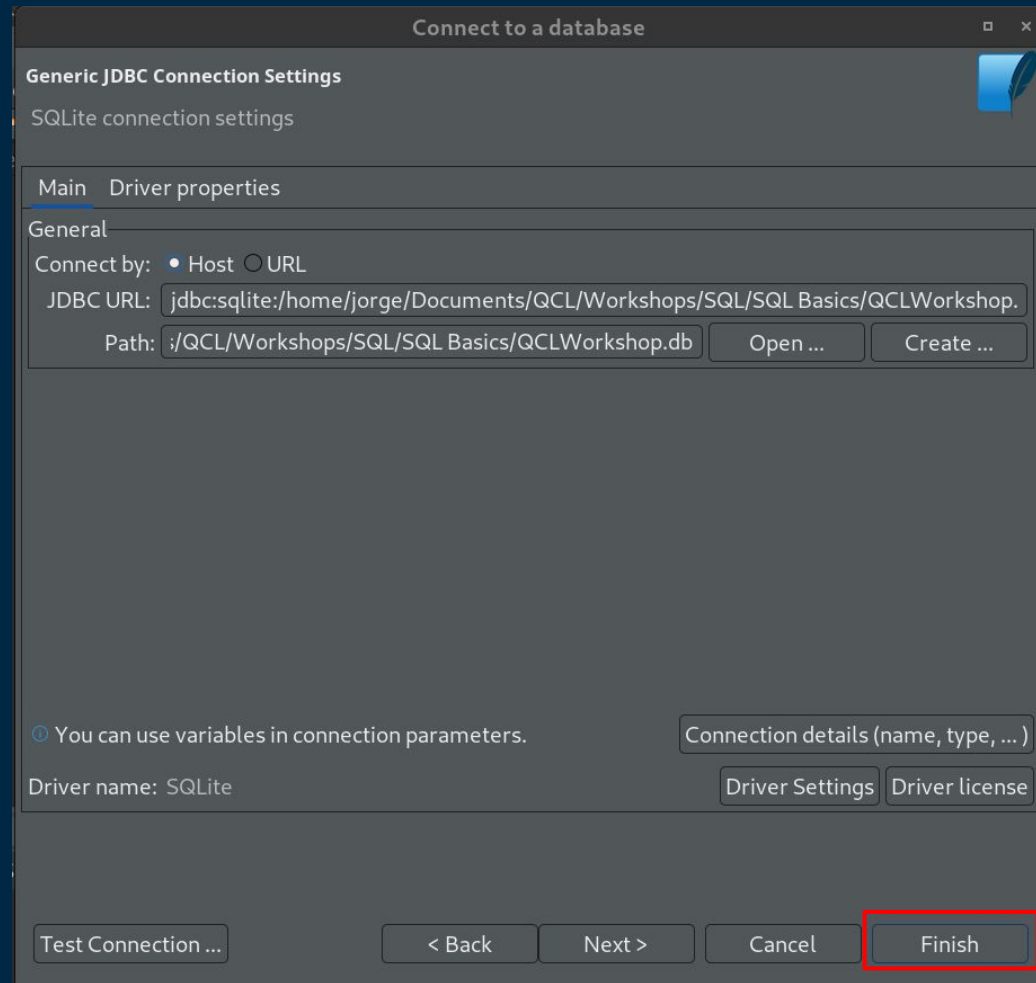
# Select Path

Give a name to your  
database file



Navigate to the  
path where you  
want to store the  
database file

# Finish the Database Creation



The screenshot shows a 'Connect to a database' dialog box with the following details:

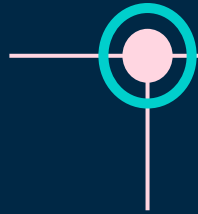
- Title Bar:** Connect to a database
- Section:** Generic JDBC Connection Settings
- Sub-section:** SQLite connection settings
- Tabs:** Main (selected), Driver properties
- General Section:**
  - Connect by:** ☒ Host ☐ URL
  - JDBC URL:** jdbc:sqlite:/home/jorge/Documents/QCL/Workshops/SQL/SQL Basics/QCLWorkshop.
  - Path:** /QCL/Workshops/SQL/SQL Basics/QCLWorkshop.db
  - Buttons:** Open ... (disabled), Create ... (disabled)
- Footer:**
  - Test Connection ...** (disabled)
  - < Back** (disabled)
  - Next >** (disabled)
  - Cancel** (disabled)
  - Finish** (highlighted with a red box)

Finally, click on "Finish"



# Agenda

## Databases and SQL



- Relational Databases
- SQL Overview

## Tables and Data



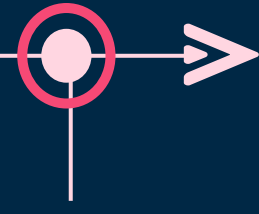
- Databases and Tables
- Today's data

## Writing Queries



- Retrieving Data
- Sorting and Filtering

## Subqueries and Joins



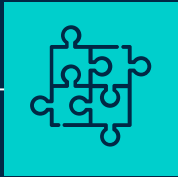
- Subqueries and Aliases
- Joining Tables



# 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

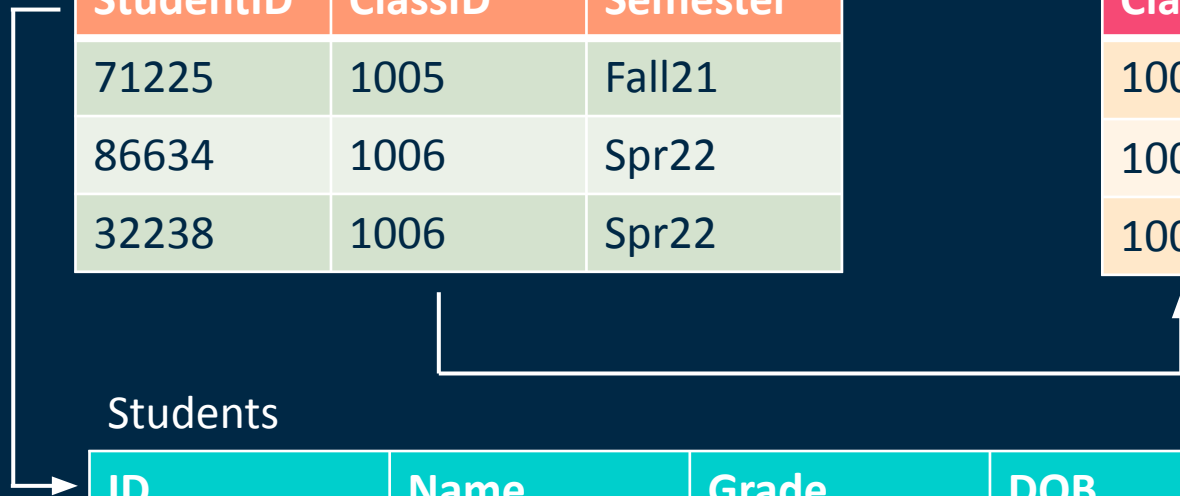
StudentID	ClassID	Semester
71225	1005	Fall21
86634	1006	Spr22
32238	1006	Spr22

School\_Courses

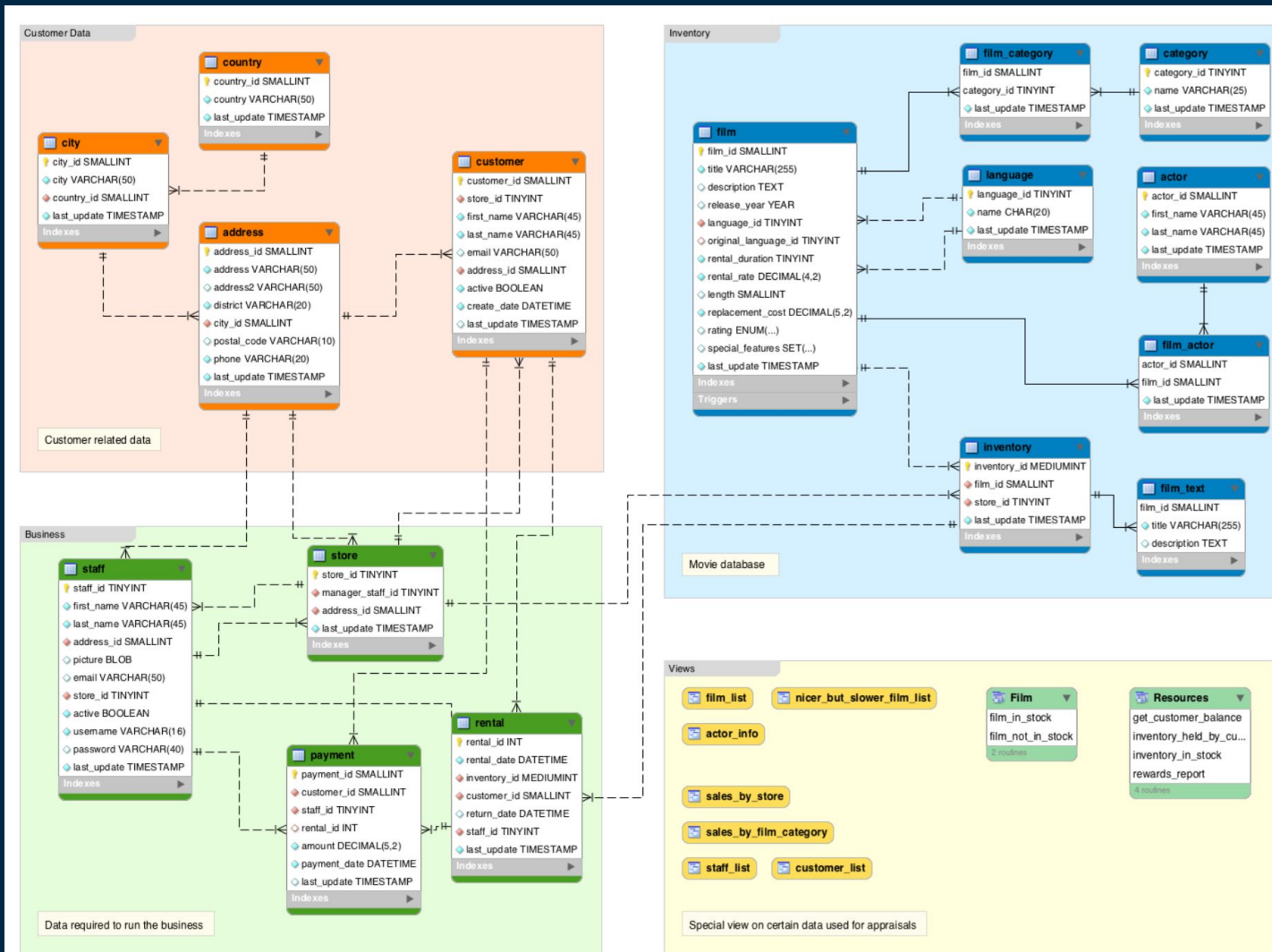
ClassID	Title	ClassNum
1005	Intro to Art History	500
1006	Intro to SQL	501
1009	Intro to Databases	300

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

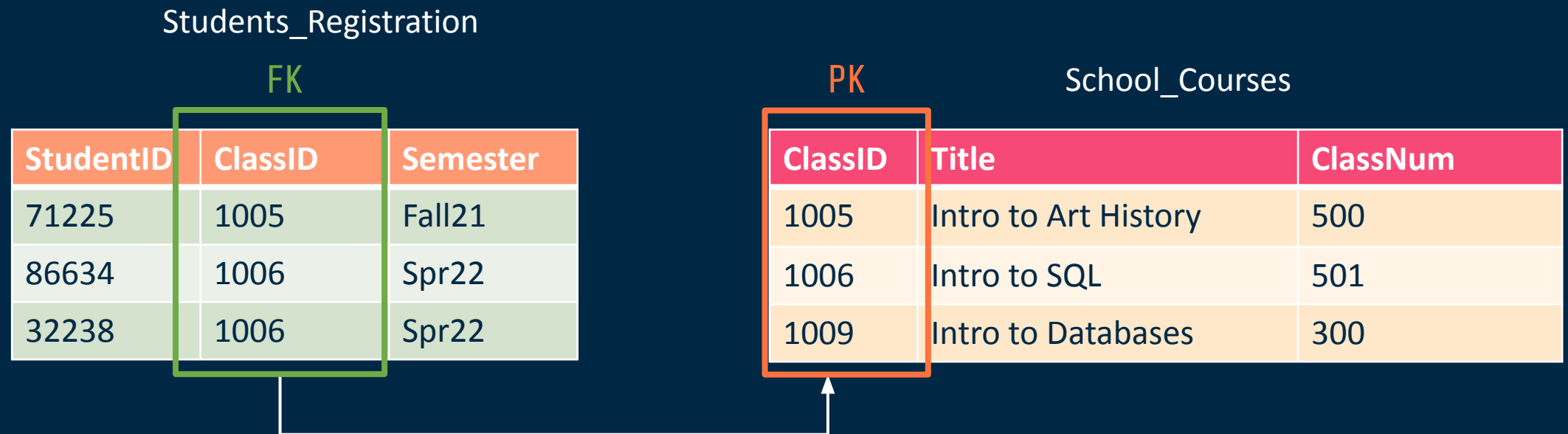


A database schema diagram of the Sakila Sample Database.

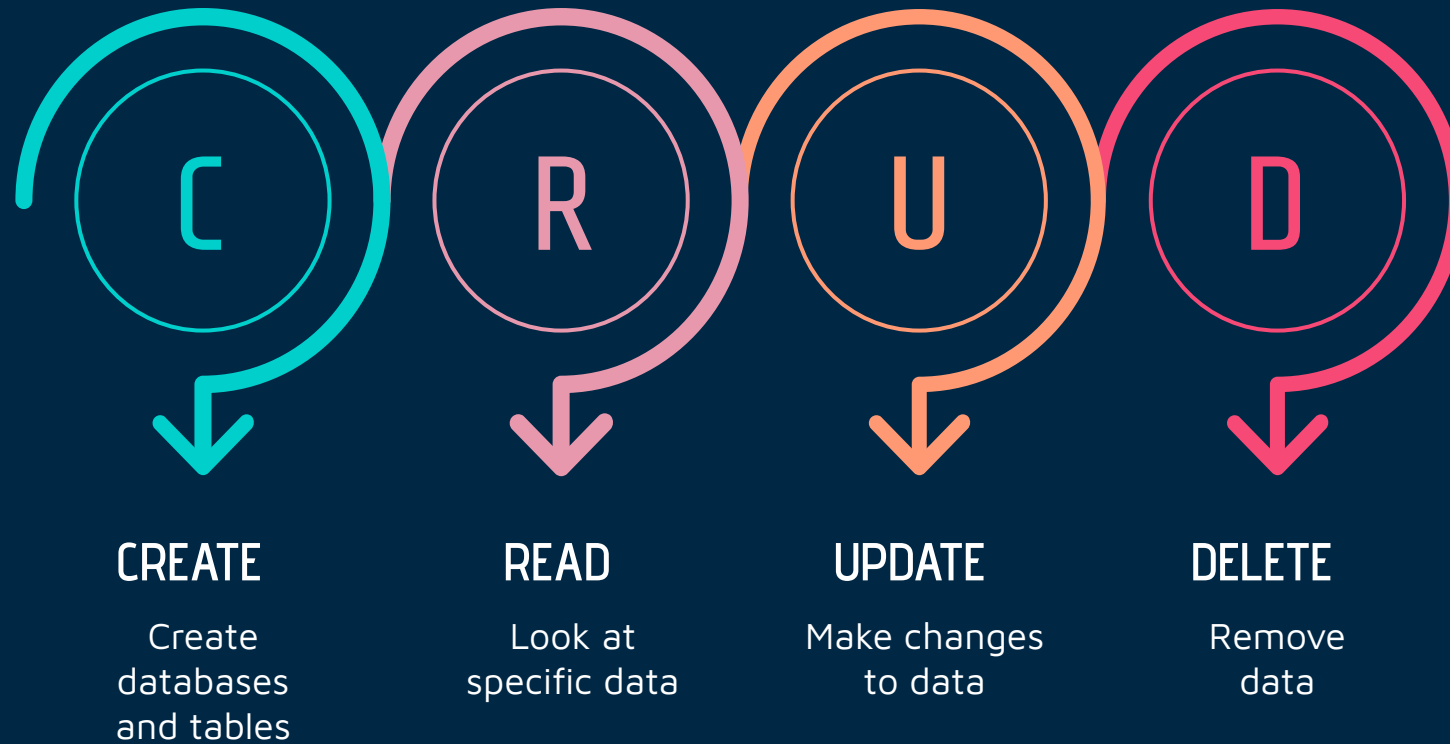
# Primary and Foreign Keys

*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

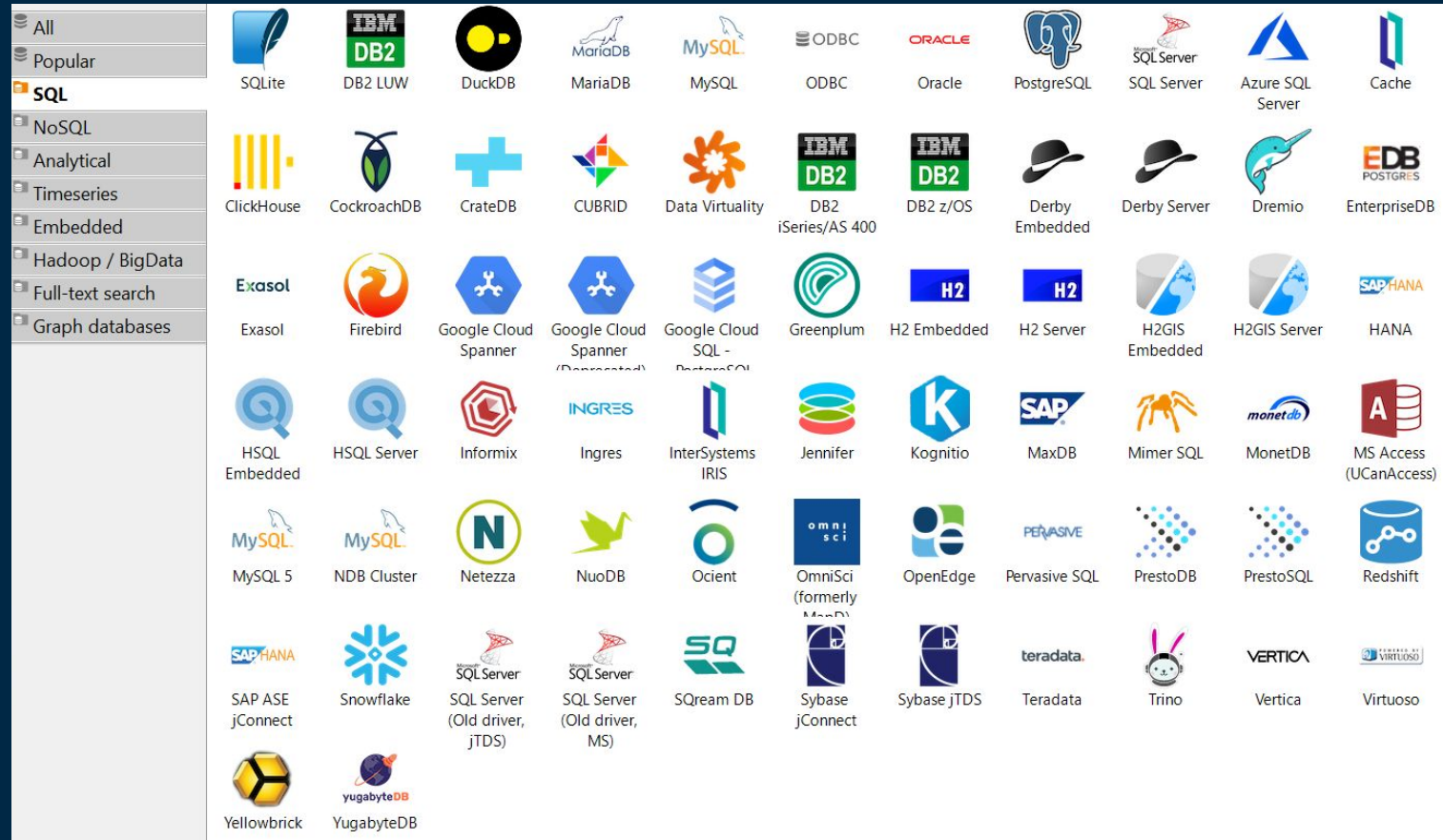


# Vocabulary

- ▶ 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 Flavors



# 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

- INTEGER is a signed whole number

# Text

Students

Text fields

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

- TEXT can contain letters, numbers and special characters. Uses a pre-defined size limit.

NOTE: Other SQL flavors use VARCHAR(size), a variable length string of maximum size length.

# Real

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

Real field

- REAL is used for numbers with a fractional part

# 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 and datetime values
- DATE with format YYYY-MM-DD
- DATETIME with a combination with format YYYY-MM-DD hh:mm:ss

NOTE: SQLite does not have a storage class for dates and/or times. These can be stored as text.

# 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

Annotations:

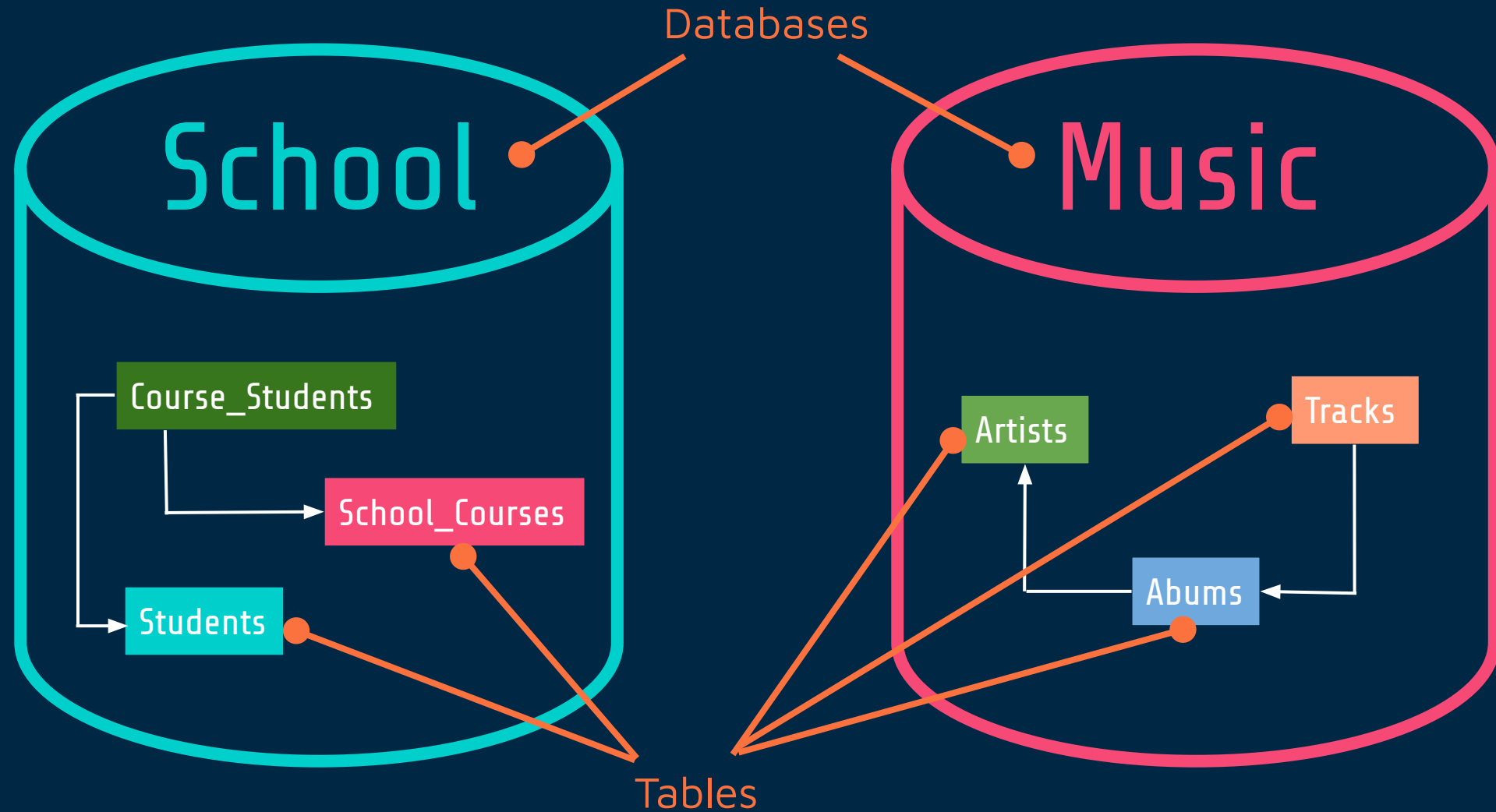
- Integer field: points to the ID column.
- Text field: points to the Name column.
- Date field: points to the DOB column.
- Real field: points to the GPA column.

# 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

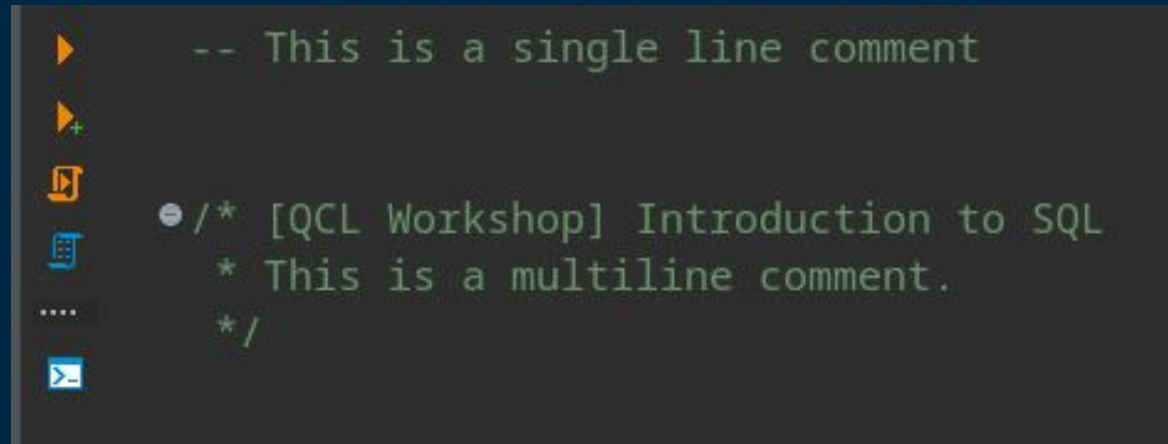
- ▶ Modified datasets for workshop (4 files total)
- ▶ **State Crime CSV File**
  - ▶ **state\_crime.csv**
  - ▶ information on the crime rates and totals for states across the United States for a wide range of years
  - ▶ reports go from 1960 to 2019 (only used 2010, 2014 and 2019)
  - ▶ [https://corgis-edu.github.io/corgis/csv/state\\_crime/](https://corgis-edu.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 Table

```
-- Create a table
CREATE TABLE state_computer_data (
  State TEXT,
  Persons_per_household REAL,
  Households_with_computer REAL,
  Households_with_internet REAL
);

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

# Comments



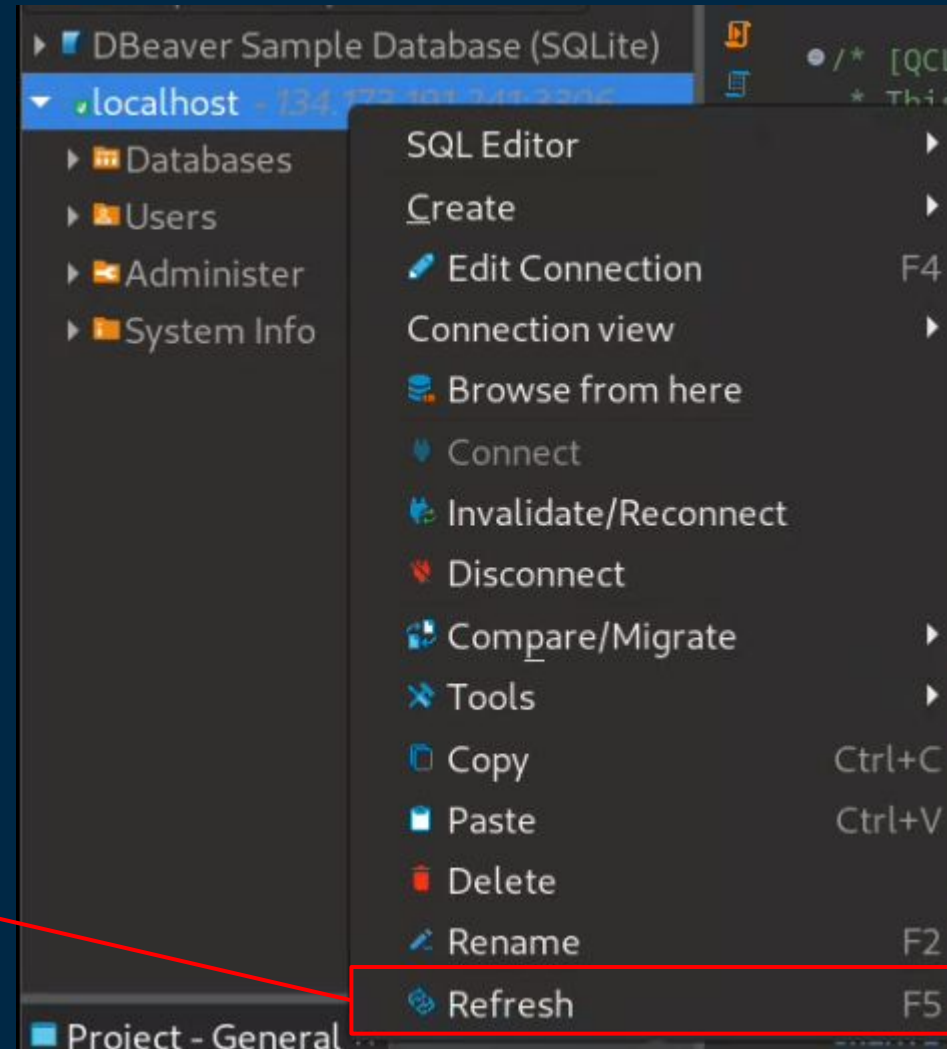
```
-- This is a single line comment

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

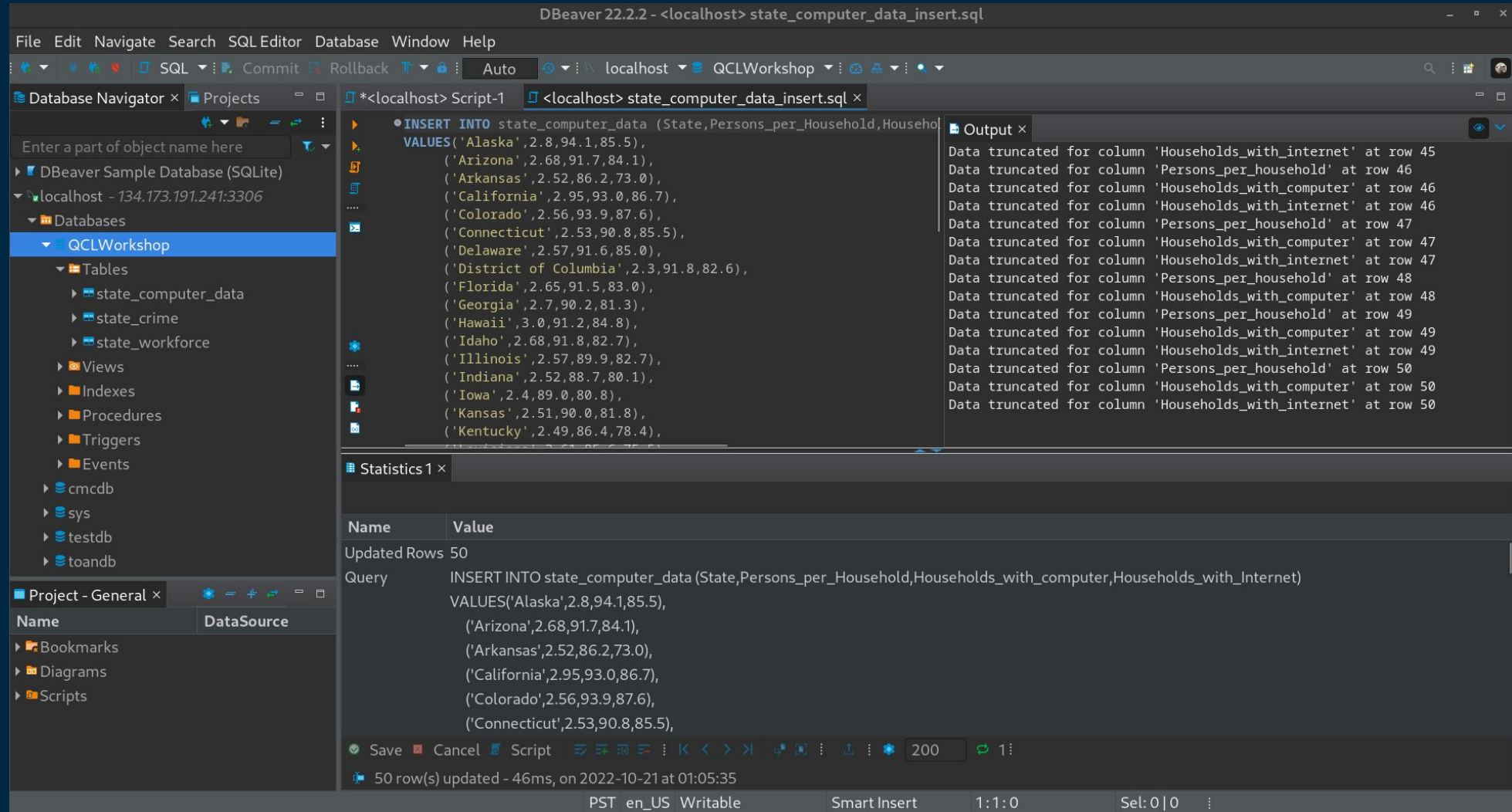
# Refresh

- Right click on your connection and click on refresh

Click on Refresh



# Insert multiple values



DBeaver 22.2.2 - <localhost> state\_computer\_data\_insert.sql

File Edit Navigate Search SQL Editor Database Window Help

SQL Commit Rollback Auto 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\_computer\_data

state\_crime

state\_workforce

Views

Indexes

Procedures

Triggers

Events

cmcdb

sys

testdb

toandb

Project - General

Name DataSource

Bookmarks

Diagrams

Scripts

Script-1

<localhost> state\_computer\_data\_insert.sql

```
INSERT INTO state_computer_data (State,Persons_per_Household,Households_with_internet,Households_with_computer)
VALUES ('Alaska',2.8,94.1,85.5),
       ('Arizona',2.68,91.7,84.1),
       ('Arkansas',2.52,86.2,73.0),
       ('California',2.95,93.0,86.7),
       ('Colorado',2.56,93.9,87.6),
       ('Connecticut',2.53,90.8,85.5),
       ('Delaware',2.57,91.6,85.0),
       ('District of Columbia',2.3,91.8,82.6),
       ('Florida',2.65,91.5,83.0),
       ('Georgia',2.7,90.2,81.3),
       ('Hawaii',3.0,91.2,84.8),
       ('Idaho',2.68,91.8,82.7),
       ('Illinois',2.57,89.9,82.7),
       ('Indiana',2.52,88.7,80.1),
       ('Iowa',2.4,89.0,80.8),
       ('Kansas',2.51,90.0,81.8),
       ('Kentucky',2.49,86.4,78.4),
       ('Louisiana',2.51,86.4,78.4),
       ('Maine',2.51,86.4,78.4),
       ('Maryland',2.51,86.4,78.4),
       ('Massachusetts',2.51,86.4,78.4),
       ('Michigan',2.51,86.4,78.4),
       ('Minnesota',2.51,86.4,78.4),
       ('Mississippi',2.51,86.4,78.4),
       ('Missouri',2.51,86.4,78.4),
       ('Montana',2.51,86.4,78.4),
       ('Nebraska',2.51,86.4,78.4),
       ('Nevada',2.51,86.4,78.4),
       ('New Hampshire',2.51,86.4,78.4),
       ('New Jersey',2.51,86.4,78.4),
       ('New Mexico',2.51,86.4,78.4),
       ('New York',2.51,86.4,78.4),
       ('North Carolina',2.51,86.4,78.4),
       ('North Dakota',2.51,86.4,78.4),
       ('Ohio',2.51,86.4,78.4),
       ('Oklahoma',2.51,86.4,78.4),
       ('Oregon',2.51,86.4,78.4),
       ('Pennsylvania',2.51,86.4,78.4),
       ('Rhode Island',2.51,86.4,78.4),
       ('South Carolina',2.51,86.4,78.4),
       ('South Dakota',2.51,86.4,78.4),
       ('Tennessee',2.51,86.4,78.4),
       ('Texas',2.51,86.4,78.4),
       ('Utah',2.51,86.4,78.4),
       ('Vermont',2.51,86.4,78.4),
       ('Virginia',2.51,86.4,78.4),
       ('Washington',2.51,86.4,78.4),
       ('West Virginia',2.51,86.4,78.4),
       ('Wisconsin',2.51,86.4,78.4),
       ('Wyoming',2.51,86.4,78.4)
```

Output

Data truncated for column 'Households\_with\_internet' at row 45

Data truncated for column 'Persons\_per\_household' at row 46

Data truncated for column 'Households\_with\_computer' at row 46

Data truncated for column 'Households\_with\_internet' at row 46

Data truncated for column 'Persons\_per\_household' at row 47

Data truncated for column 'Households\_with\_computer' at row 47

Data truncated for column 'Households\_with\_internet' at row 47

Data truncated for column 'Persons\_per\_household' at row 48

Data truncated for column 'Households\_with\_computer' at row 48

Data truncated for column 'Persons\_per\_household' at row 49

Data truncated for column 'Households\_with\_computer' at row 49

Data truncated for column 'Households\_with\_internet' at row 49

Data truncated for column 'Persons\_per\_household' at row 50

Data truncated for column 'Households\_with\_computer' at row 50

Data truncated for column 'Households\_with\_internet' at row 50

Statistics 1

Name	Value
Updated Rows	50

Query

```
INSERT INTO state_computer_data (State,Persons_per_Household,Households_with_computer,Households_with_Internet)
VALUES('Alaska',2.8,94.1,85.5),
      ('Arizona',2.68,91.7,84.1),
      ('Arkansas',2.52,86.2,73.0),
      ('California',2.95,93.0,86.7),
      ('Colorado',2.56,93.9,87.6),
      ('Connecticut',2.53,90.8,85.5),
      ('Delaware',2.57,91.6,85.0),
      ('District of Columbia',2.3,91.8,82.6),
      ('Florida',2.65,91.5,83.0),
      ('Georgia',2.7,90.2,81.3),
      ('Hawaii',3.0,91.2,84.8),
      ('Idaho',2.68,91.8,82.7),
      ('Illinois',2.57,89.9,82.7),
      ('Indiana',2.52,88.7,80.1),
      ('Iowa',2.4,89.0,80.8),
      ('Kansas',2.51,90.0,81.8),
      ('Kentucky',2.49,86.4,78.4),
      ('Louisiana',2.51,86.4,78.4),
      ('Maine',2.51,86.4,78.4),
      ('Maryland',2.51,86.4,78.4),
      ('Massachusetts',2.51,86.4,78.4),
      ('Michigan',2.51,86.4,78.4),
      ('Minnesota',2.51,86.4,78.4),
      ('Mississippi',2.51,86.4,78.4),
      ('Missouri',2.51,86.4,78.4),
      ('Montana',2.51,86.4,78.4),
      ('Nebraska',2.51,86.4,78.4),
      ('Nevada',2.51,86.4,78.4),
      ('New Hampshire',2.51,86.4,78.4),
      ('New Jersey',2.51,86.4,78.4),
      ('New Mexico',2.51,86.4,78.4),
      ('New York',2.51,86.4,78.4),
      ('North Carolina',2.51,86.4,78.4),
      ('North Dakota',2.51,86.4,78.4),
      ('Ohio',2.51,86.4,78.4),
      ('Oklahoma',2.51,86.4,78.4),
      ('Oregon',2.51,86.4,78.4),
      ('Pennsylvania',2.51,86.4,78.4),
      ('Rhode Island',2.51,86.4,78.4),
      ('South Carolina',2.51,86.4,78.4),
      ('South Dakota',2.51,86.4,78.4),
      ('Tennessee',2.51,86.4,78.4),
      ('Texas',2.51,86.4,78.4),
      ('Utah',2.51,86.4,78.4),
      ('Vermont',2.51,86.4,78.4),
      ('Virginia',2.51,86.4,78.4),
      ('Washington',2.51,86.4,78.4),
      ('West Virginia',2.51,86.4,78.4),
      ('Wisconsin',2.51,86.4,78.4),
      ('Wyoming',2.51,86.4,78.4)
```

Save Cancel Script

50 row(s) updated - 46ms, on 2022-10-21 at 01:05:35

PST en\_US Writable Smart Insert 1:1:0 Sel: 0 | 0

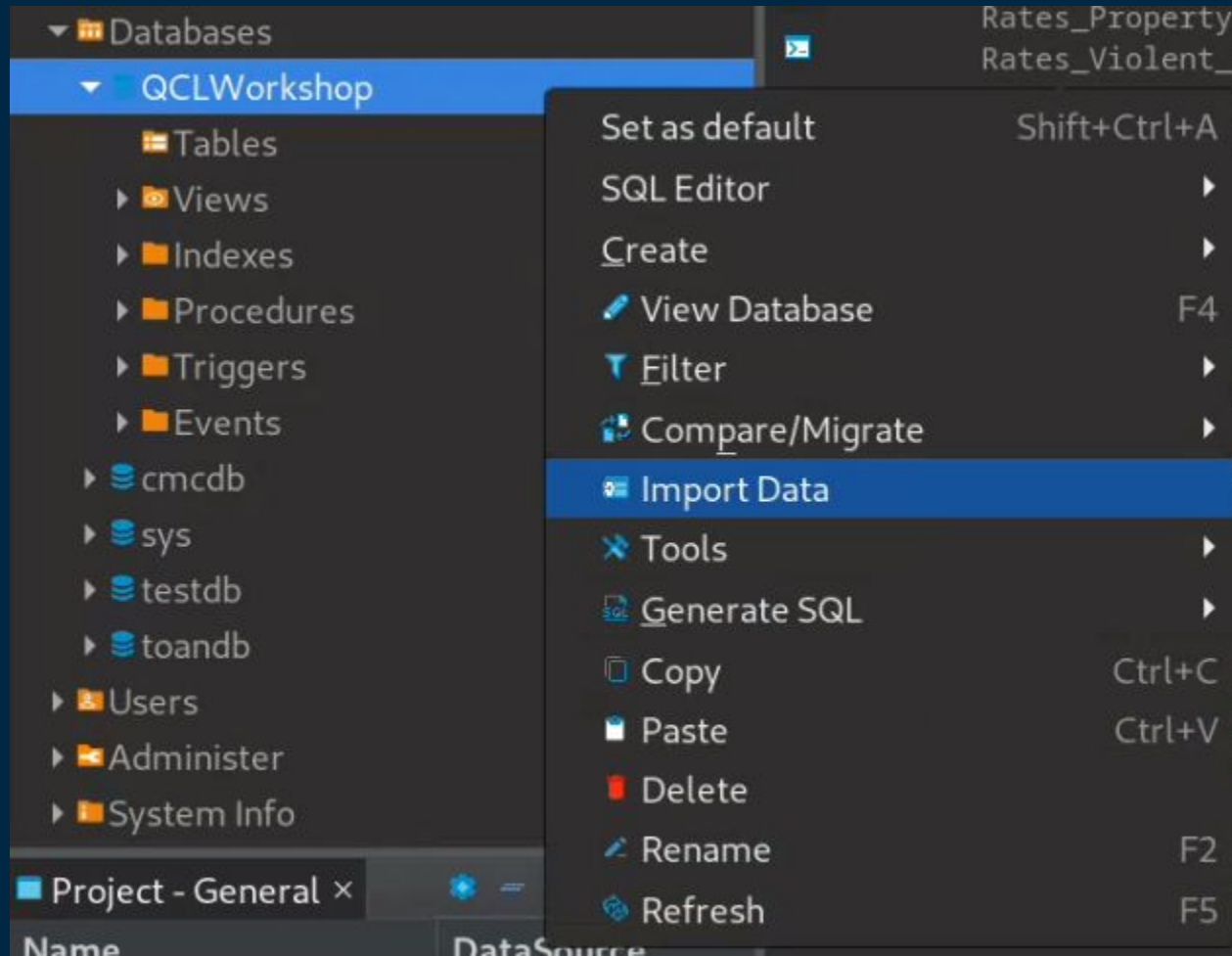


# Hands-on

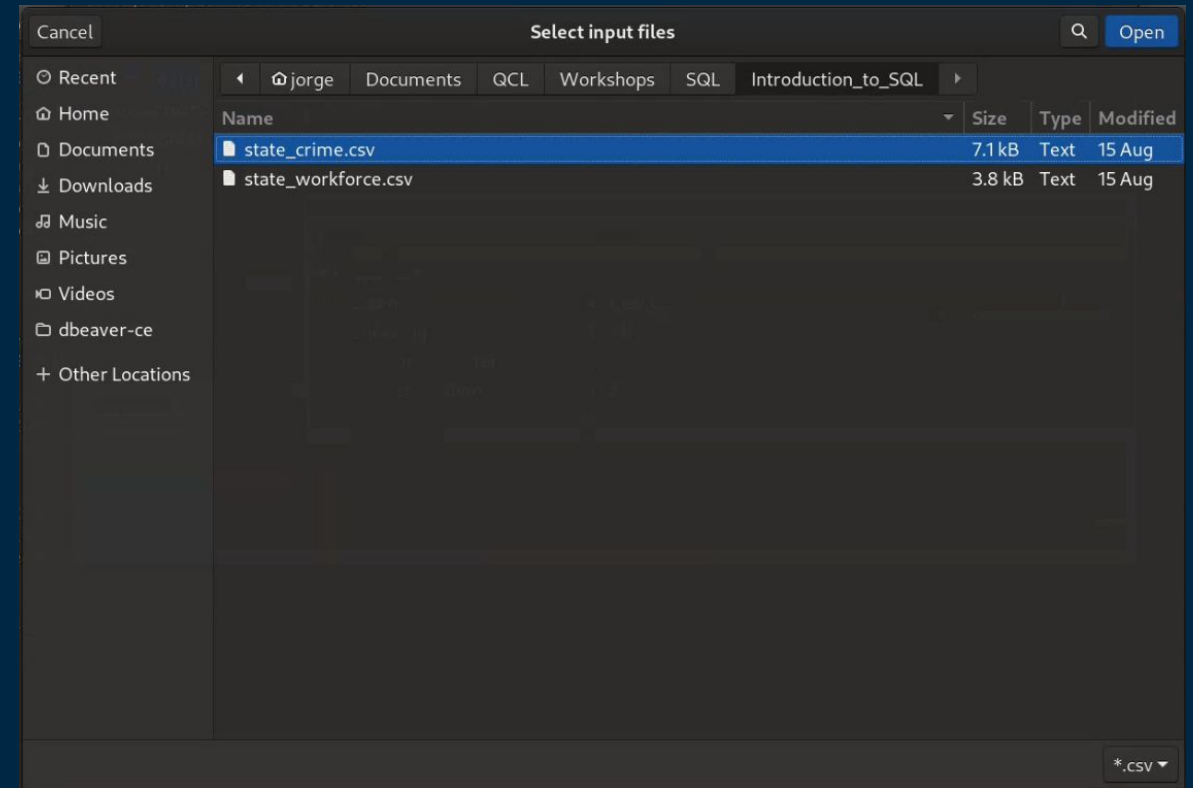
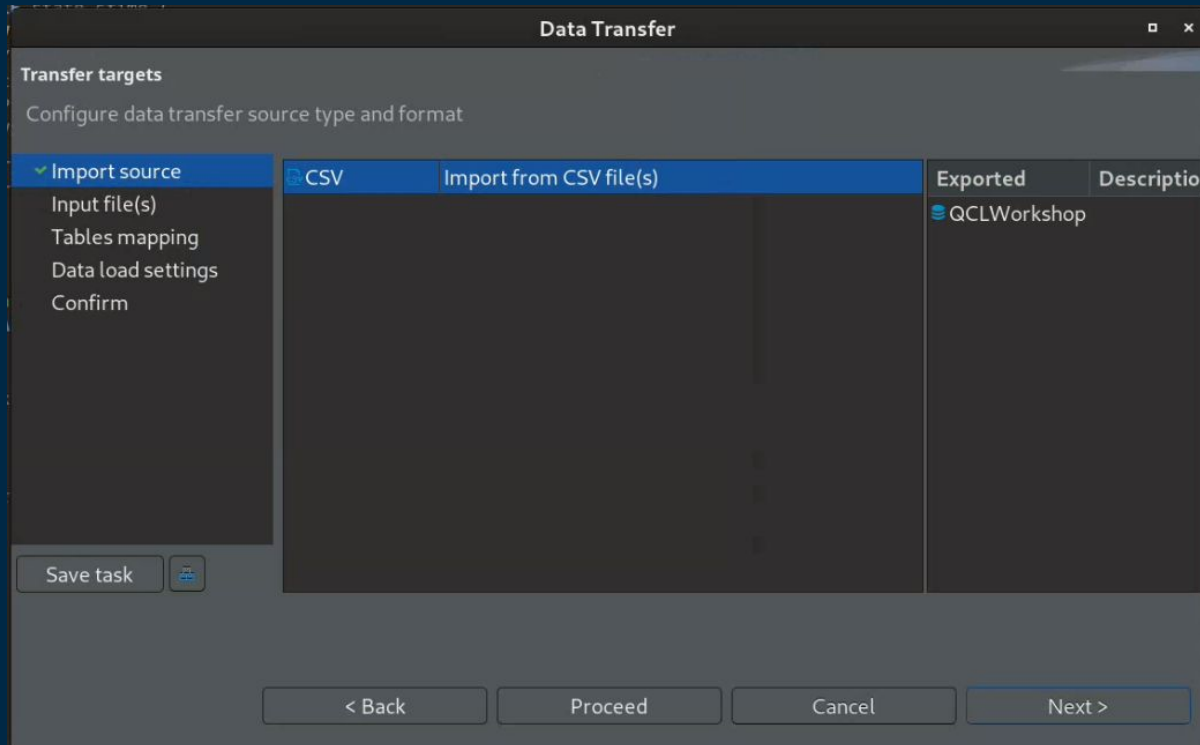
- ▶ Create a table named `state_people` and add the attributes
  1. State as TEXT
  2. Employment\_Firms\_Total as INTEGER
  3. Age\_Percent\_Under\_18\_Years as REAL
  4. Age\_Percent\_65\_and\_Older as REAL
- ▶ Insert data into the new table with the file `state_people_insert.sql`



# Import your data



# Import source



# 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	?

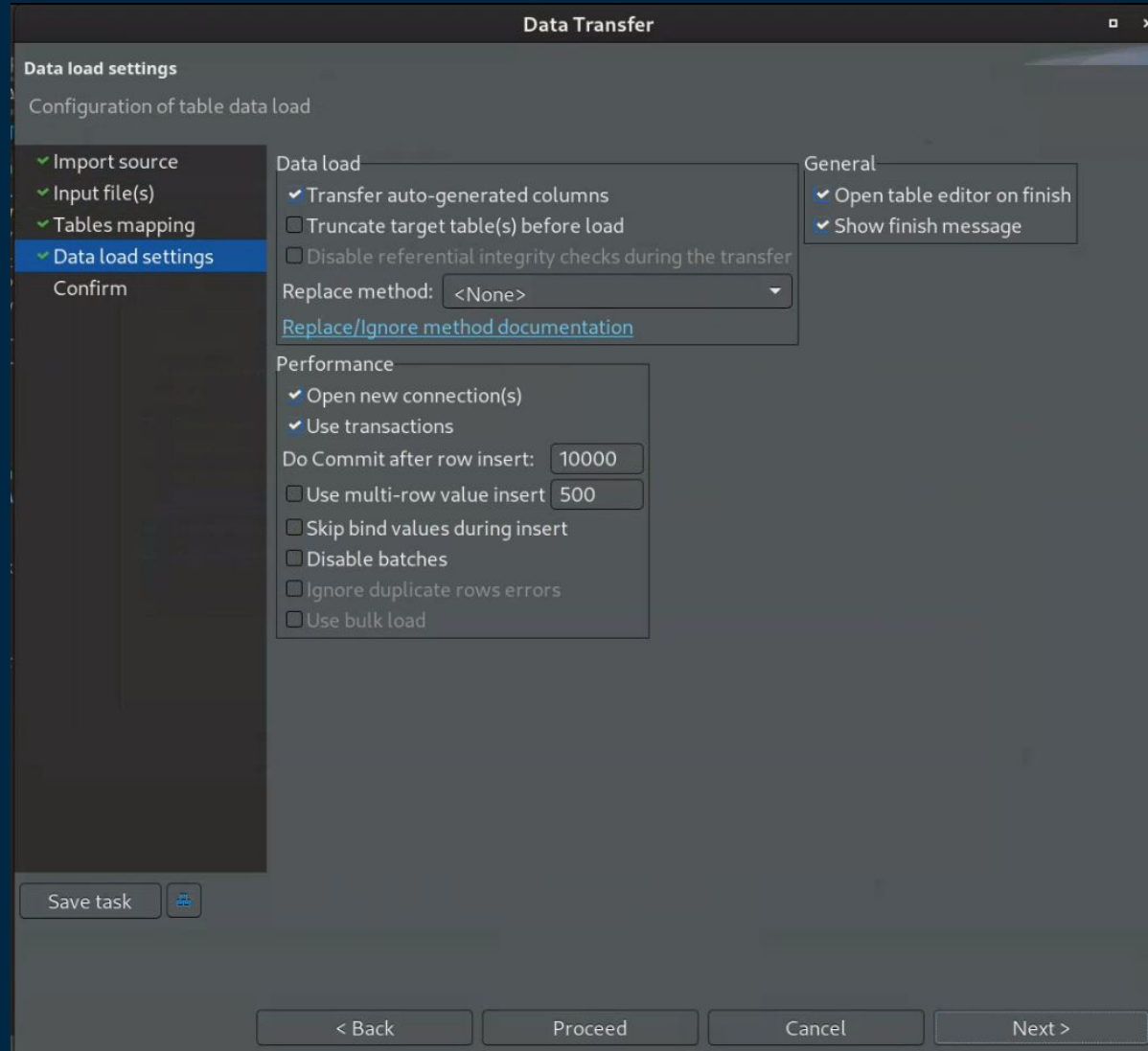
**Importer settings:**

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 >

# 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**

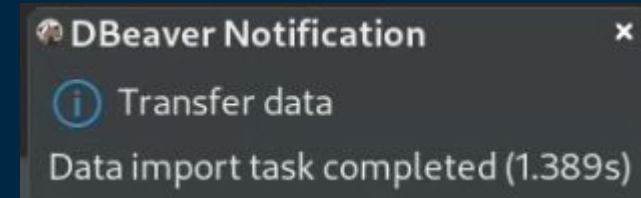
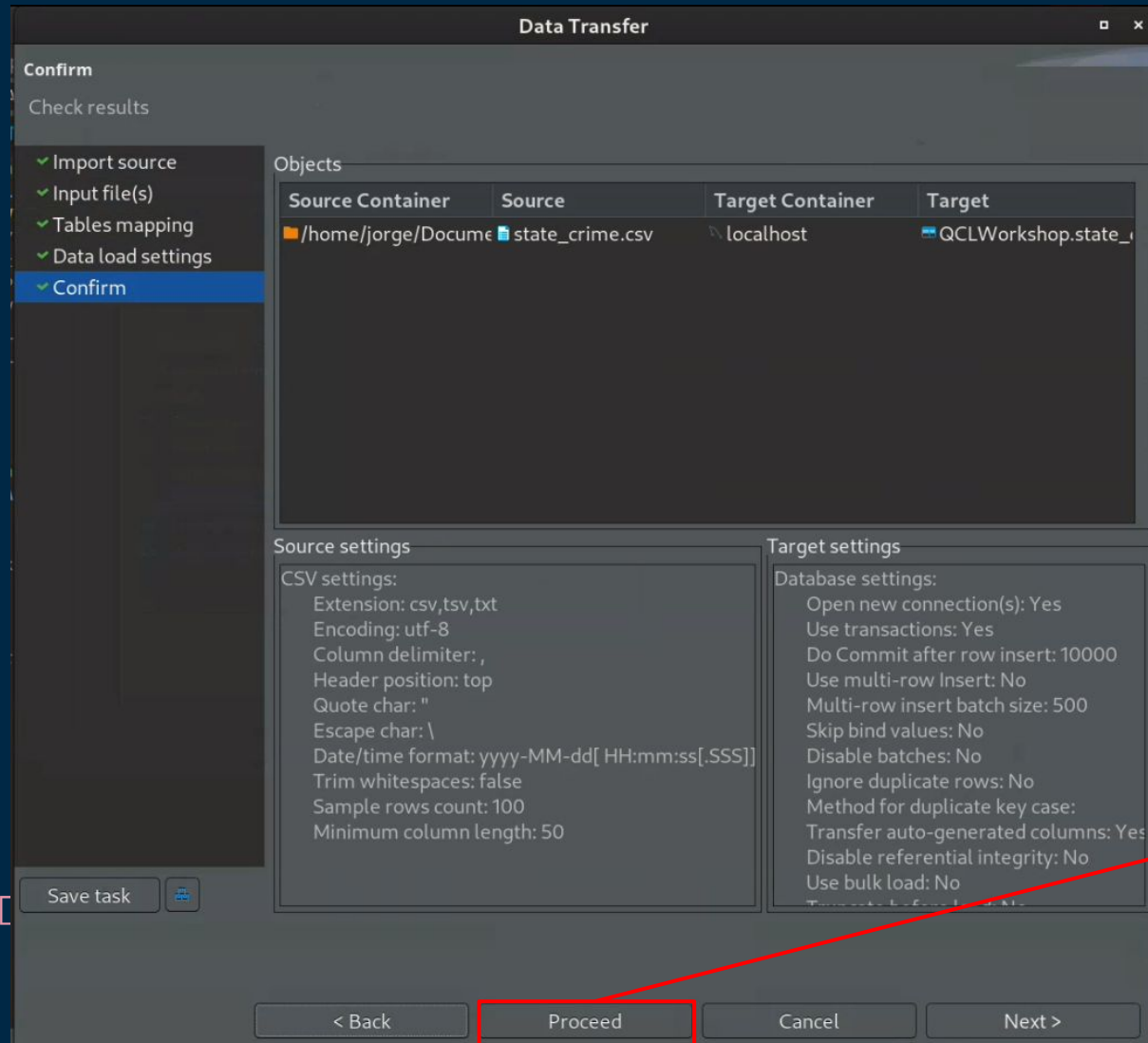
- ☒ Open new connection(s)
- ☒ Use transactions
- Do Commit after row insert:
- ☐ Use multi-row value insert
- ☐ Skip bind values during insert
- ☐ Disable batches
- ☐ Ignore duplicate rows errors
- ☐ Use bulk load

**General**

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

**Confirm**

# Confirmation



Click on "Proceed"



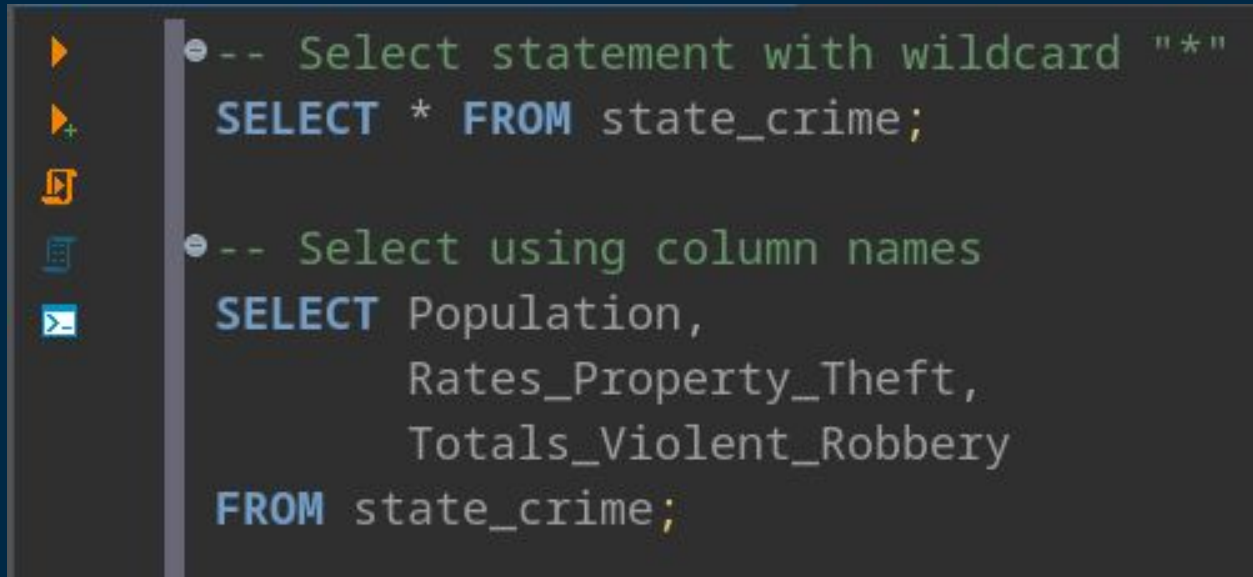
# Hands-on

- ▶ Import the file named `state_workforce`

# Writing Queries



# 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

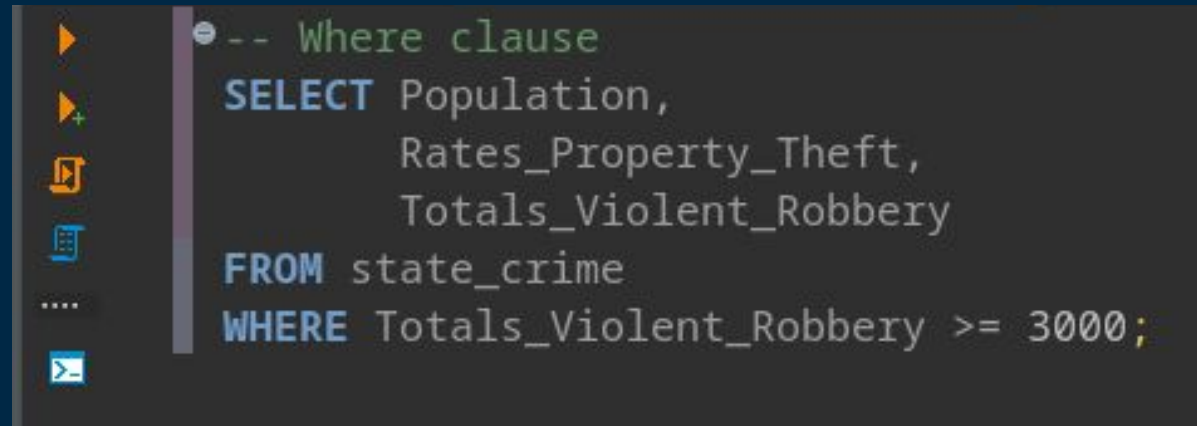




# Hands-on

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

# Filtering



```
-- 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 more information here:
  - [https://www.sqlite.org/lang\\_expr.html#operators\\_and\\_parse\\_affecting\\_attributes](https://www.sqlite.org/lang_expr.html#operators_and_parse_affecting_attributes)

# 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

# Ends with

DBeaver 22.2.2 - <localhost> Script-1

File Edit Navigate Search SQL Editor Database Window Help

SQL Commit Rollback Auto 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

Enter a SQL expression to filter results (use Ctrl+Space)

	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

Save Cancel Script

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

DBeaver 22.2.2 - <localhost> Script-1

File Edit Navigate Search SQL Editor Database Window Help

SQL Commit Rollback Auto 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

Enter a SQL expression to filter results (use Ctrl+Space)

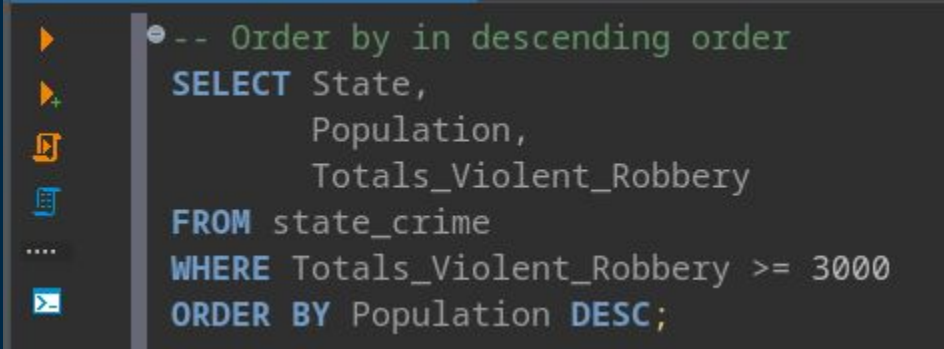
Grid	State	Population
1	District of Columbia	705,749
2	District of Columbia	604,912
3	District of Columbia	658,893

Save Cancel Script

3 row(s) fetched - 21ms, on 2022-10-20 at 23:01:51

PST en\_US Writable Smart Insert 15:1 [75] Sel: 75 | 4

# Sorting

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



# Subqueries

```
-- 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);
```

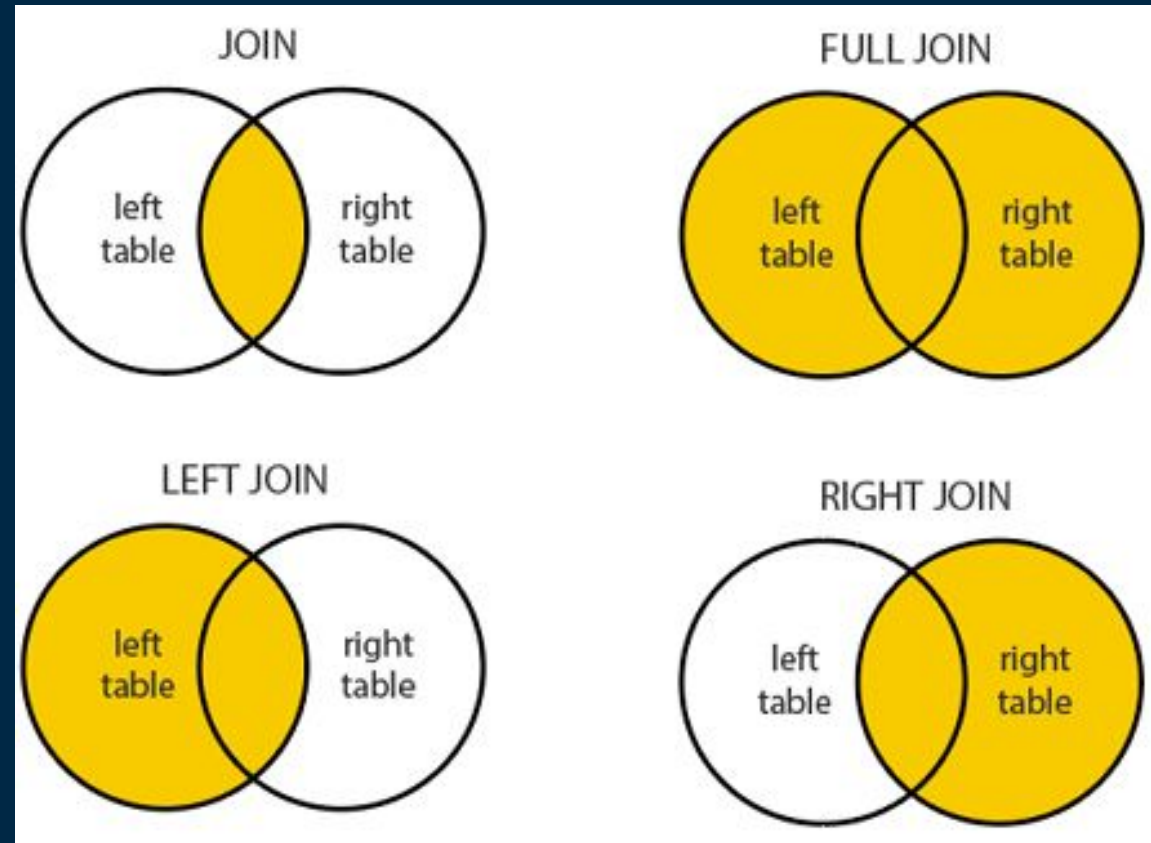
- Useful when you want to combine information from different tables
- Performs the innermost query first



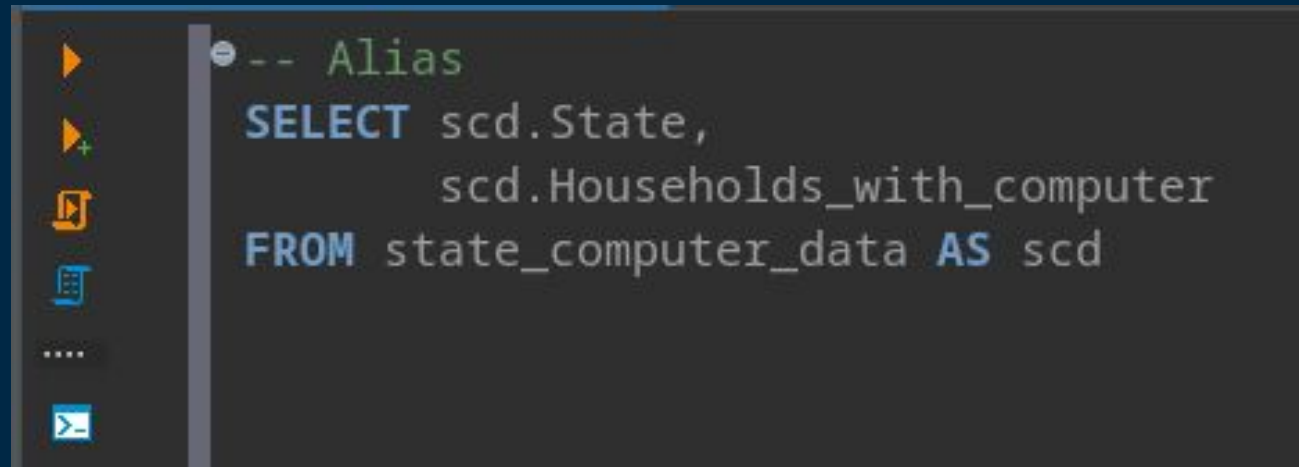
# Hands-on

- ▶ Find the maximum percentage of the people with education of high school or higher from states where the rate of property theft is above 2,500.

# SQL Joins



# Aliases

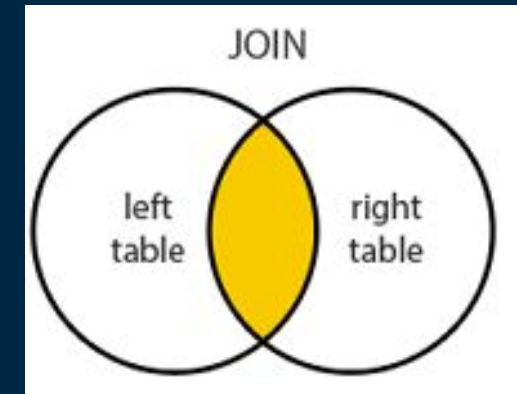


```
-- Alias
SELECT scd.State,
       scd.Households_with_computer
FROM state_computer_data AS scd
```

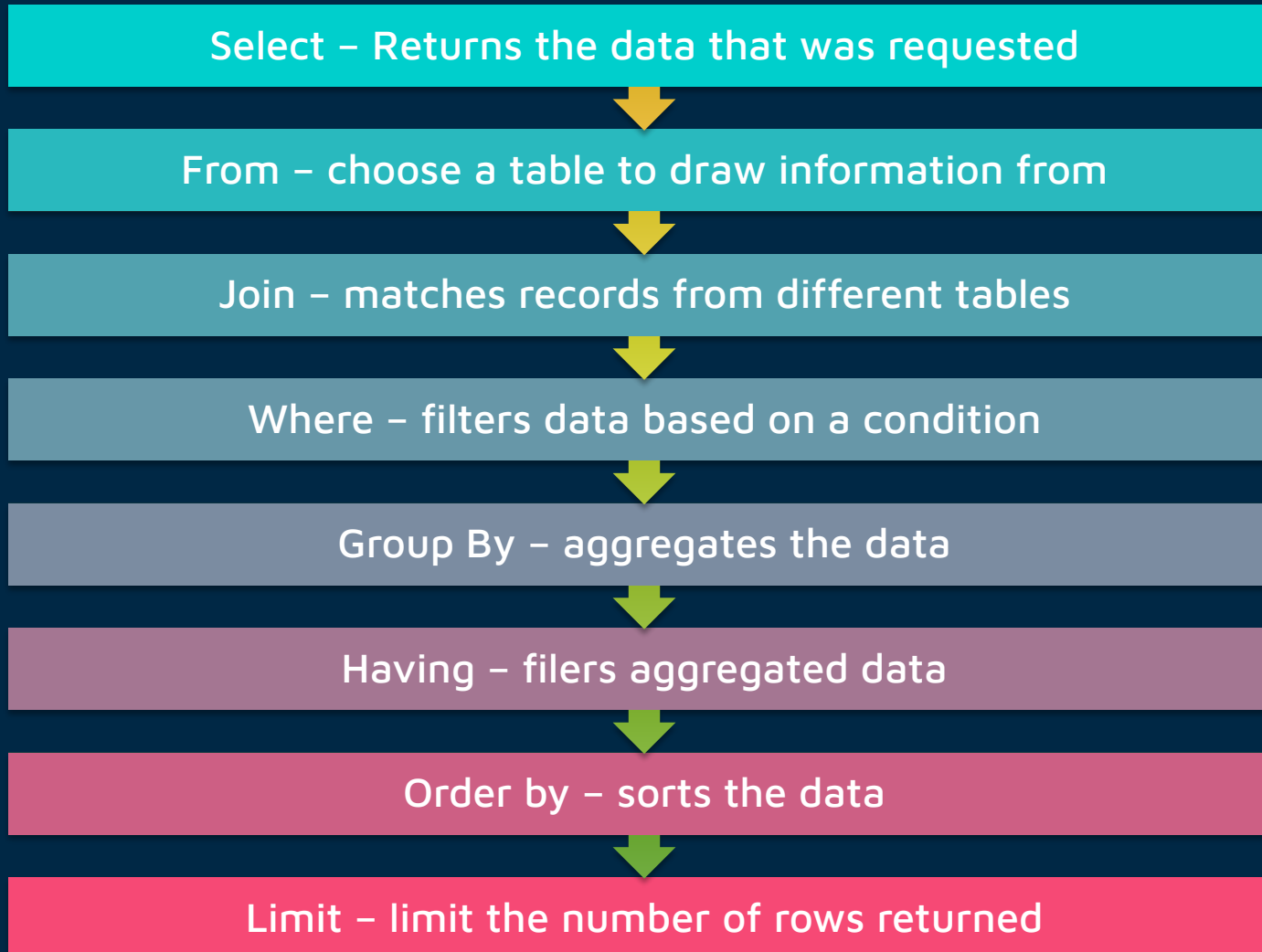


# (Inner) Join

```
-- 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

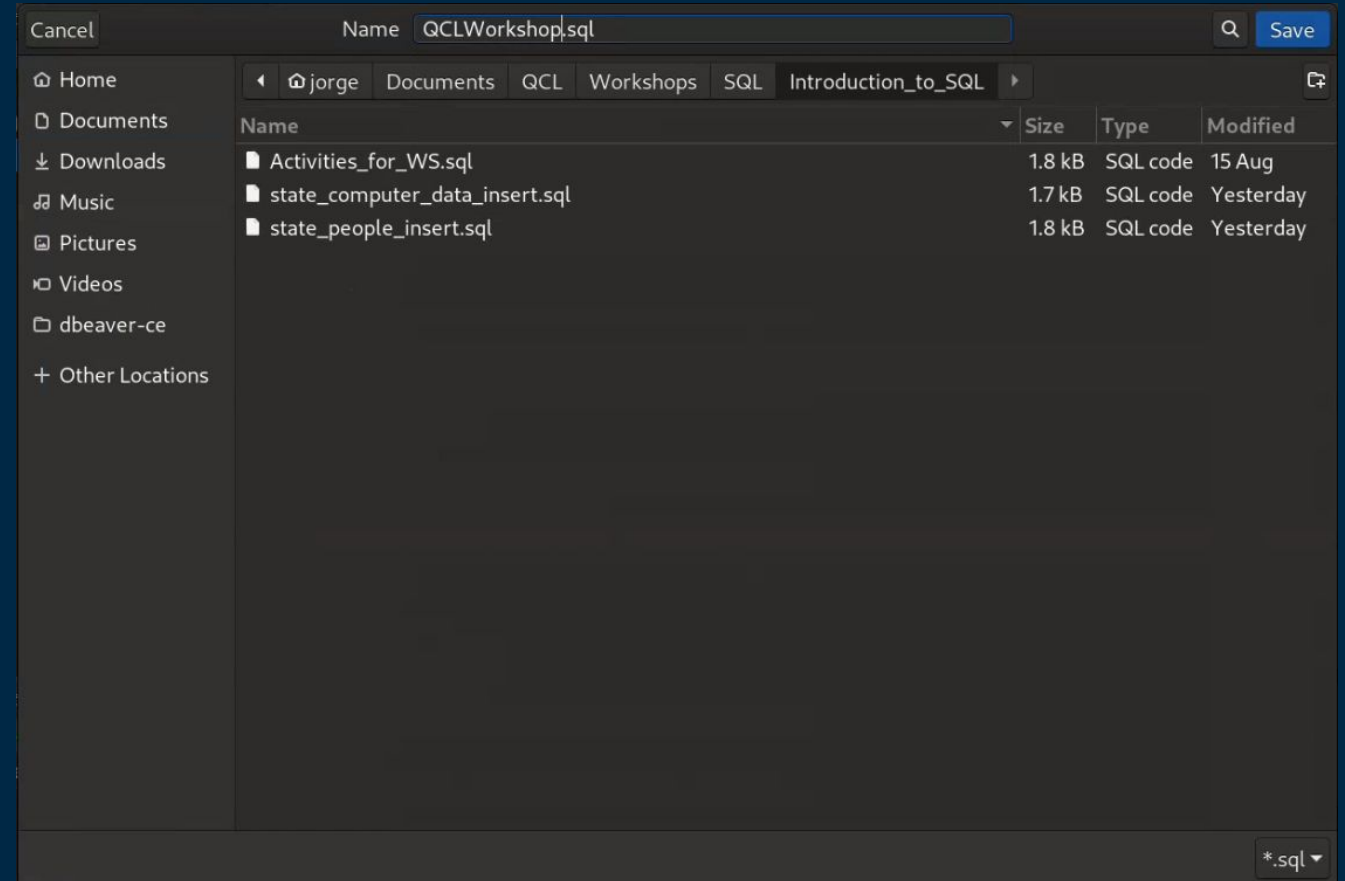
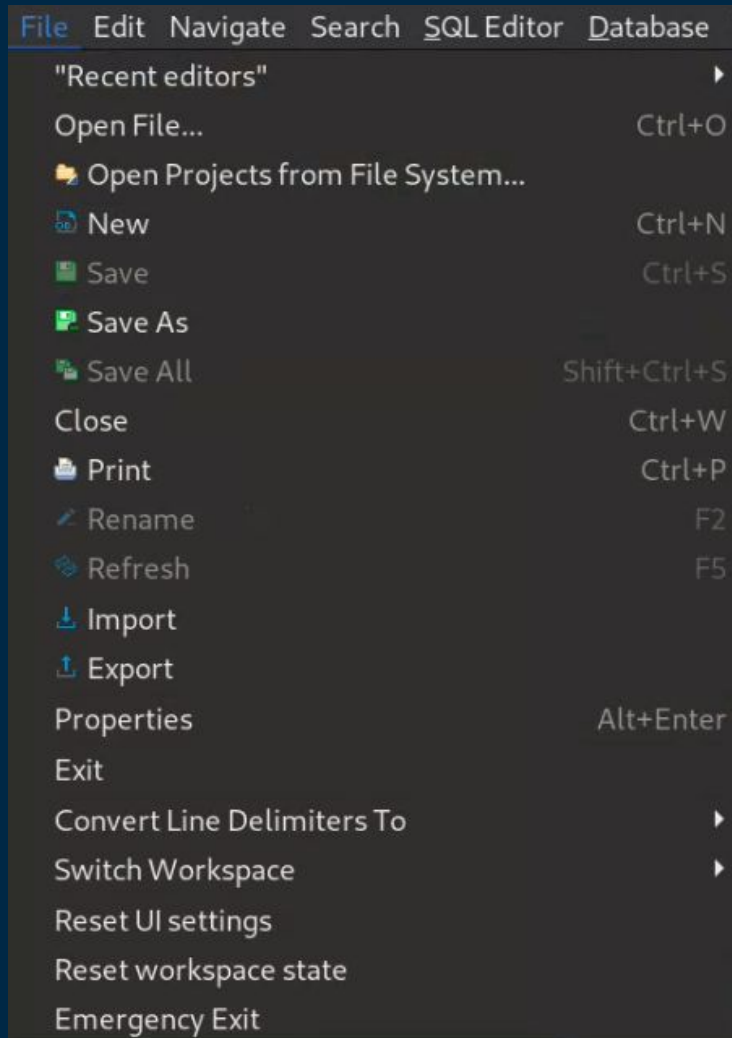




# Hands-on

- ▶ 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)

# 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/>