21.1 MySQL

MySQL

This material uses MySQL as a reference relational database system. Although the material is relevant to all relational databases, SQL syntax and many activities are based on MySQL.

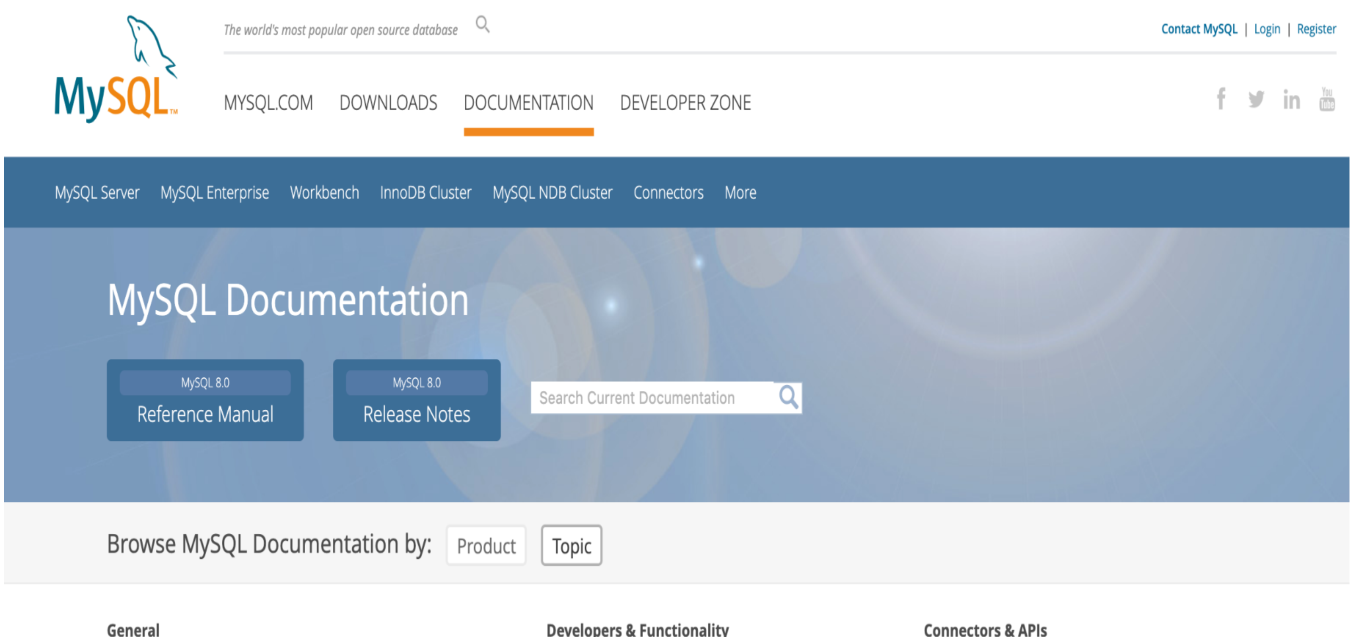
MySQL is a leading relational database system sponsored by Oracle. MySQL is relatively easy to install and use, yet has many advanced capabilities. MySQL runs on all major operating systems, including Linux, Unix, Mac OS, and Windows. For these reasons, MySQL is one of the most popular database systems.

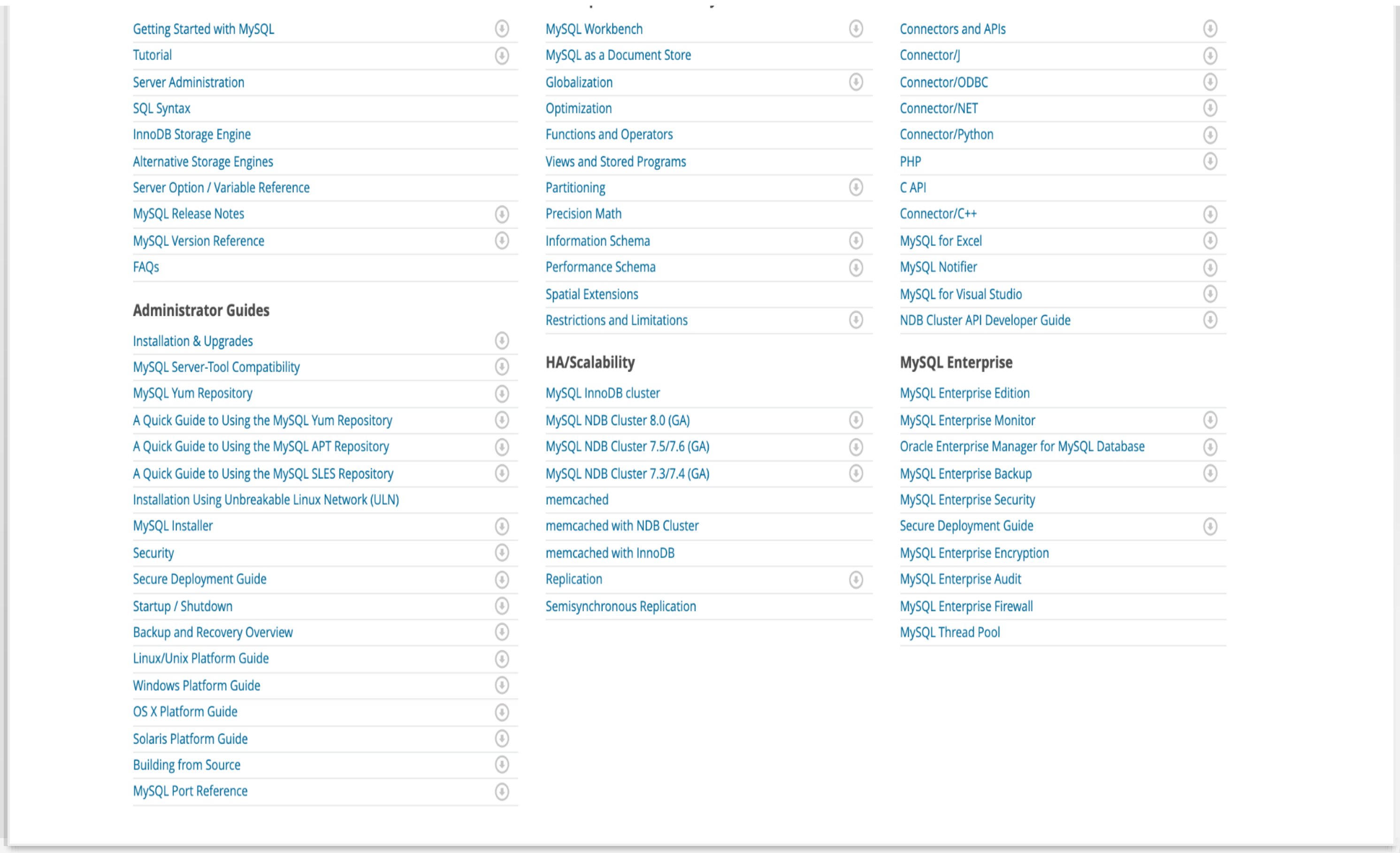
MySQL is available in two editions:

- ***MySQL Community***, commonly called ***MySQL Server***, is a free edition. MySQL Server includes a complete set of database services and tools, and is suitable for non-commercial applications such as education.
- ***MySQL Enterprise*** is a paid edition for managing commercial databases. MySQL Enterprise includes MySQL Server and additional administrative applications.

This book is based on MySQL Server release 8.0. Forthcoming versions of this book will be upgraded to MySQL Server 8.1, released in July 2023. Complete documentation for MySQL Server 8.0 is available online.

Figure 21.1.1: MySQL documentation (dev.mysql.com/doc).




Getting Started with MySQL
Tutorial
Server Administration
SQL Syntax
InnoDB Storage Engine
Alternative Storage Engines
Server Option / Variable Reference
MySQL Release Notes
MySQL Version Reference
FAQs

Administrator Guides

Installation & Upgrades
MySQL Server-Tool Compatibility
MySQL Yum Repository
A Quick Guide to Using the MySQL Yum Repository
A Quick Guide to Using the MySQL APT Repository
A Quick Guide to Using the MySQL SLES Repository
Installation Using Unbreakable Linux Network (ULN)
MySQL Installer
Security
Secure Deployment Guide
Startup / Shutdown
Backup and Recovery Overview
Linux/Unix Platform Guide
Windows Platform Guide
OS X Platform Guide
Solaris Platform Guide
Building from Source
MySQL Port Reference

MySQL Workbench
MySQL as a Document Store
Globalization
Optimization
Functions and Operators
Views and Stored Programs
Partitioning
Precision Math
Information Schema
Performance Schema
Spatial Extensions
Restrictions and Limitations

HA/Scalability

MySQL InnoDB cluster
MySQL NDB Cluster 8.0 (GA)
MySQL NDB Cluster 7.5/7.6 (GA)
MySQL NDB Cluster 7.3/7.4 (GA)
memcached
memcached with NDB Cluster
memcached with InnoDB
Replication
Semisynchronous Replication

Connectors and APIs
Connector/J
Connector/ODBC
Connector/NET
Connector/Python
PHP
C API
Connector/C++
MySQL for Excel
MySQL Notifier
MySQL for Visual Studio
NDB Cluster API Developer Guide

MySQL Enterprise

MySQL Enterprise Edition
MySQL Enterprise Monitor
Oracle Enterprise Manager for MySQL Database
MySQL Enterprise Backup
MySQL Enterprise Security
Secure Deployment Guide
MySQL Enterprise Encryption
MySQL Enterprise Audit
MySQL Enterprise Firewall
MySQL Thread Pool

Instructions for downloading and installing MySQL Server 8.0 on Windows or Mac OS are available from the 'Exploring further' links below.

When installing MySQL Server, the user must enter a password for the ***root account***, the administrative account that has full control of MySQL. Other database user accounts may optionally be created. After installation, MySQL Server runs as a service in the background. MySQL Server automatically starts and stops when the operating system starts and stops.

PARTICIPATION ACTIVITY | 21.1.1: MySQL.

1) Refer to the website db-engines.com. What is the overall MySQL ranking, compared to all database systems?

- ○ 1
- ○ 2
- ○ 5

2) What account can create other user accounts?

- ○ MySQL account

○ Root account

○ All accounts can create user accounts

3) Content in this material only applies to the MySQL database system.

○ True

○ False

MySQL Command-Line Client

The ***MySQL Command-Line Client*** is a text interface included in the MySQL Server download. The Command-Line Client allows developers to connect to the database server, perform administrative functions, and execute SQL statements.

To run the Command-Line Client, a user must first open a Command Prompt on Windows or a Terminal on a Mac:

- Windows: Click the Start button in the Taskbar, type "cmd", then click Command Prompt.
- Mac: Click on the Terminal application, usually found in the Applications > Utilities folder.

When MySQL Command-Line Client is started with the root account, the user is prompted to enter the root account password. Then Command-Line Client attempts to connect to the database server running on the local machine.

PARTICIPATION ACTIVITY | 21.1.2: Using the MySQL Command-Line Client.

```
$ mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3
...
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> USE world;
Database changed.
mysql> SELECT * FROM city LIMIT 10;
+----+----------------+-------------+---------------+------------+
| ID | Name           | CountryCode | District      | Population |
+----+----------------+-------------+---------------+------------+
|  1 | Kabul          | AFG         | Kabol         |    1780000 |
|  2 | Qandahar       | AFG         | Qandahar      |     237500 |
|  3 | Herat          | AFG         | Herat         |     186800 |
|  4 | Mazar-e-Sharif | AFG         | Balkh         |     127800 |
|  5 | Amsterdam      | NLD         | Noord-Holland |     731200 |
|  6 | Rotterdam      | NLD         | Zuid-Holland  |     593321 |
|  7 | Haag           | NLD         | Zuid-Holland  |     440900 |
|  8 | Utrecht        | NLD         | Utrecht       |     234323 |
|  9 | Eindhoven      | NLD         | Noord-Brabant |     201843 |
| 10 | Tilburg        | NLD         | Noord-Brabant |     193238 |
+----+----------------+-------------+---------------+------------+
10 rows in set (0.00 sec)

mysql>
```

Animation content:

Static figure:
A console appears with database commands and responses.

Step 1: From a command-line prompt, the user starts the MySQL Command-Line Client. The -u option names the account, and -p indicates a password must be entered. The following line appears on the console:
$mysql -u root -p

Step 2: The user enters the root password established during installation. A second line appears on the console:
Enter password: ********

Step 3: After a successful login, the user is presented a startup screen and a mysql prompt. Additional lines appear on the console:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3
...
Type help; or \h for help. Type \c to clear the current input statement.
mysql>

Step 4: Commands are entered one by one on the command line. The SELECT statement returns the first 10 rows from the city table. The following command appears after the mysql> prompt:
USE world;

Two more lines appears on the console:
Database changed.
mysql> SELECT * FROM city LIMIT 10;

A table appears on the console. The table has columns ID, Name, CountryCode, District, and Population. The table has ten rows containing data for ten cities.

Two more lines appear on the console:
10 rows in set (0.00 sec)
mysql>

Animation captions:

1. From a command-line prompt, the user starts the MySQL Command-Line Client. The -u option names the account, and -p indicates a password must be entered.
2. The user enters the root password established during installation.
3. After a successful login, the user is presented a startup screen and a mysql prompt.
4. Commands are entered one by one on the command line. The SELECT statement returns the first 10 rows from the city table.

The animation above shows the user typing SQL commands that use the 'world' database, a database that is usually installed with MySQL. The world database contains three tables: city, country, and countrylanguage. Users can practice entering SQL statements that work with and manipulate the world database. Some installations do not include the world database, so users must download and install the world database from MySQL.com separately.

MySQL Server returns an ***error code*** and description when an SQL statement is syntactically incorrect or the database cannot execute the statement.

Figure 21.1.2: MySQL error codes.

```
mysql> SELECT FROM city;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to
your MySQL Server version for the right syntax to use near 'FROM city' at line 1

mysql> INSERT INTO city VALUES (123, 'Amsterdam', 'NLD', 'Noord-Holland', 731200);
ERROR 1062 (23000): Duplicate entry '123' for key 'PRIMARY'
```

PARTICIPATION ACTIVITY | 21.1.3: Introduction to MySQL.

1) The root account password is set when installing MySQL.

- ○ True
- ○ False

2) The database server must be manually started each time the user runs the MySQL Command-Line Client.

- True
- False

3) The MySQL Command-Line Client provides a graphical interface for interacting with the database server.

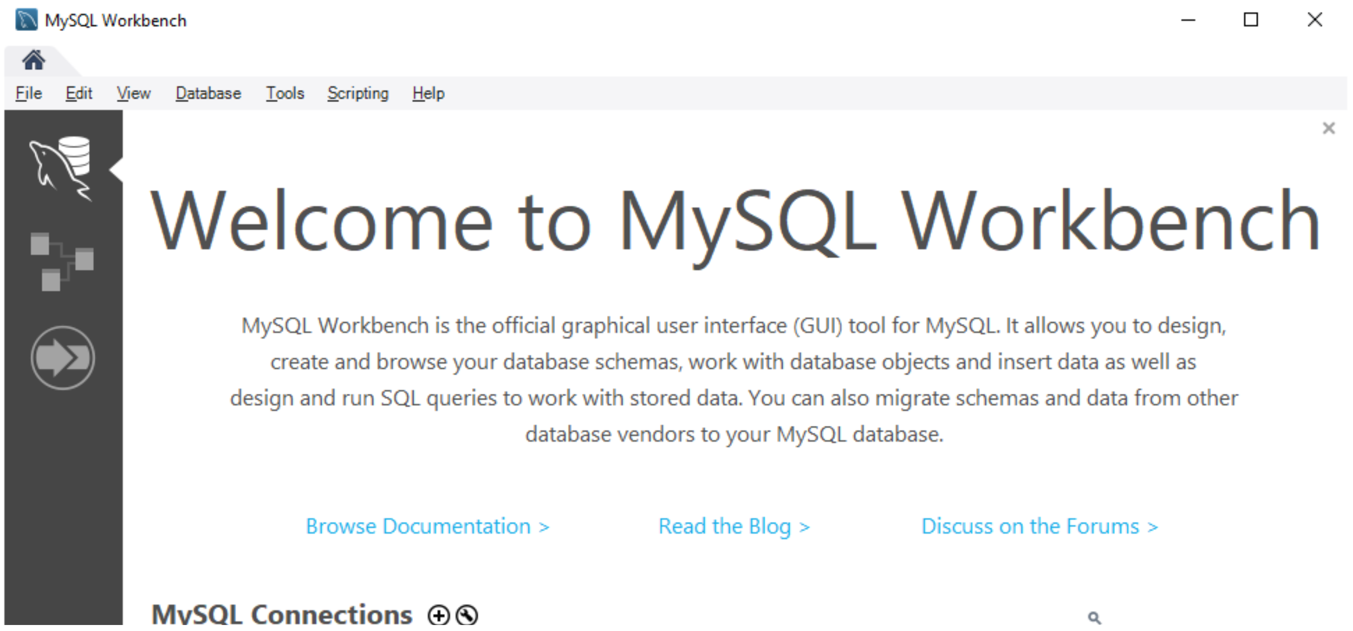
- True
- False

MySQL Workbench

Some developers prefer to interact with MySQL Server via a graphical user interface. ***MySQL Workbench*** is installed with MySQL Server and allows developers to execute SQL commands using an editor. When MySQL Workbench is started, the user can connect to MySQL Server running on the local machine or on the network.

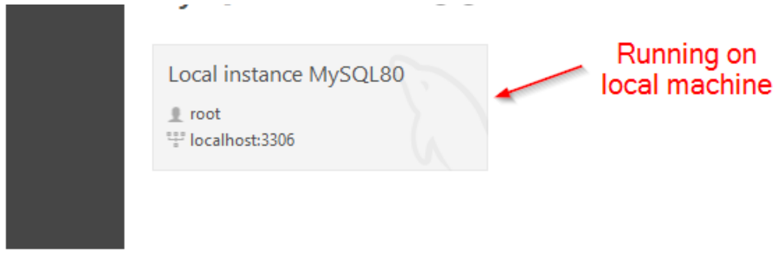
The figure below shows the MySQL Workbench home screen on Windows. The Mac version has some minor differences. Clicking on the box labeled *Local Instance MySQL80* connects to MySQL Server running on the same computer as MySQL Workbench.

Figure 21.1.3: MySQL Workbench home screen.



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

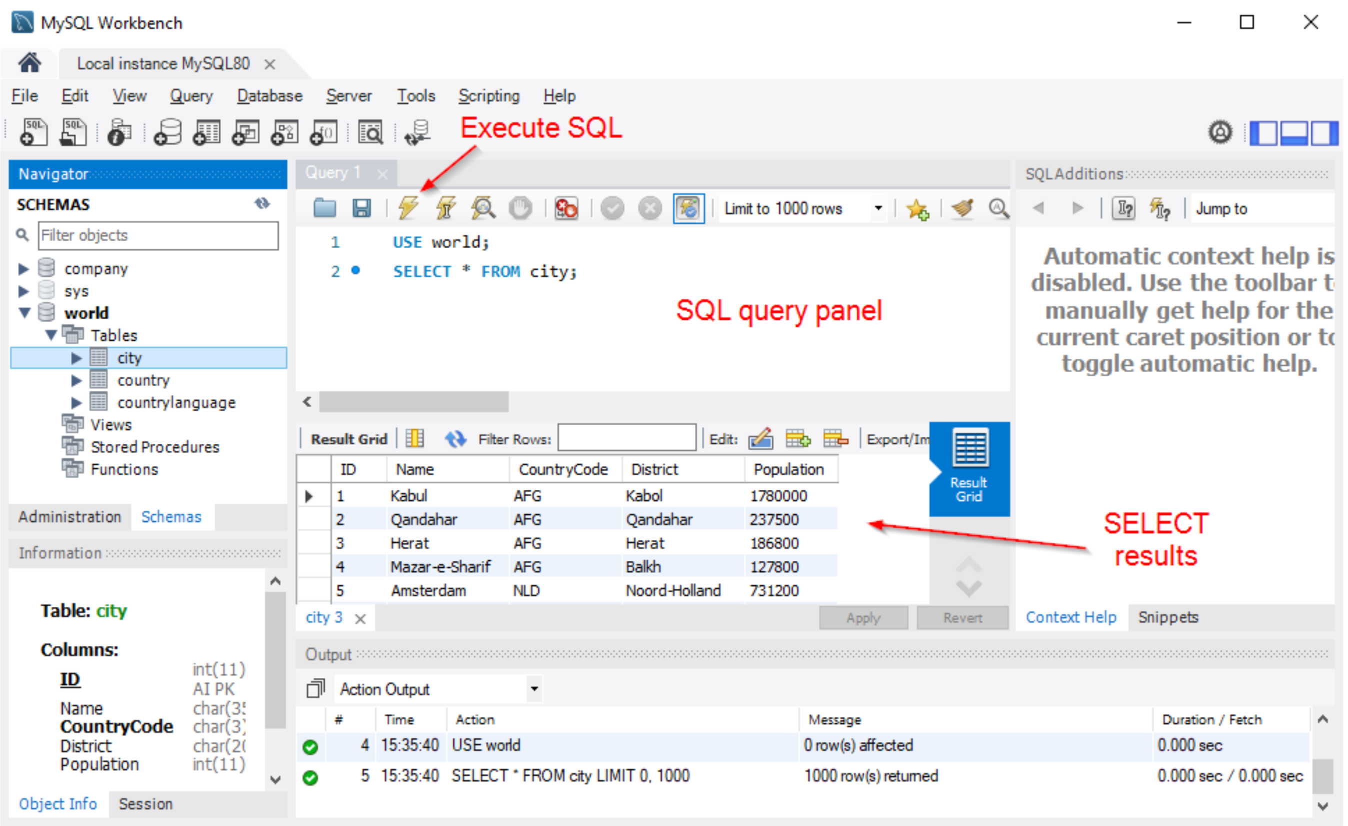

Local instance MySQL80
root
localhost:3306

After connecting to MySQL server, Workbench shows the *Navigator* sidebar on the left with two tabs:

1. The *Administration* tab shows various administrative options, like checking the server's status, importing/exporting data, and starting/stopping the MySQL server.
2. The *Schemas* tab shows a list of available databases. A database can be expanded to show the database's tables.

The figure below shows the world database's three tables: city, country, and countrylanguage. The query panel is where the user enters SQL statements. Pressing the lightning bolt icon executes the SQL statements and shows the results below the query panel. A summary of the executed statements is shown at the bottom of the window.

Figure 21.1.4: MySQL Workbench executing a SELECT statement.



```
USE world;
SELECT * FROM city;
```

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.

Information

Table: city

Columns:

ID	int(11) AI PK
Name	char(3
CountryCode	char(3
District	char(2
Population	int(11)

#	Time	Action	Message	Duration / Fetch
4	15:35:40	USE world	0 row(s) affected	0.000 sec
5	15:35:40	SELECT * FROM city LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec

PARTICIPATION ACTIVITY 21.1.4: MySQL Workbench.

1) MySQL Workbench and MySQL Command-Line Client both allow the user to type SQL statements.

- ○ True
- ○ False

2) The SQL statements in the SQL query panel are not executed until the lightning bolt is clicked.

- ○ True
- ○ False

3) The MySQL Workbench screenshot above shows the columns that make up the City table.

- ○ True
- ○ False

Exploring further:

- MySQL download
- MySQL installation video (Windows)
- MySQL installation video (Mac)
- MySQL documentation - home page
- MySQL documentation - installation
- MySQL documentation - Workbench
- MySQL documentation - problems and common errors
- MySQL documentation - Sakila database installation

21.2 MySQL architecture

Layers

Architecture describes the components of a computer system and the relationships between components. This section describes MySQL architecture. Other relational databases have similar components, but component details and relationships vary greatly.

MySQL components are organized in four layers:

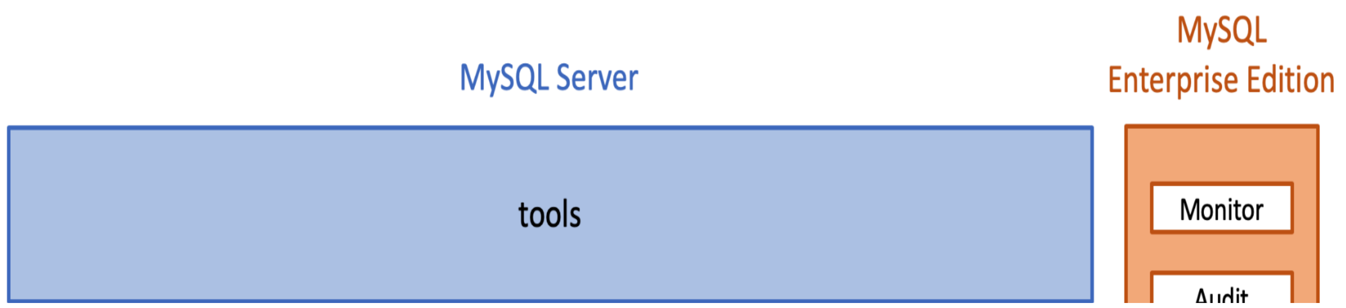
- ***Tools*** interact directly with database users and administrators, and send queries to the query processor.
- The ***query processor*** manages connections from multiple users and compiles queries into low-level instructions for the storage engine.
- The ***storage engine***, also called a ***storage manager***, executes instructions, manages indexes, and interacts with the file system. Some storage engines support database transactions, described elsewhere in this material.
- The ***file system*** accesses data on storage media. The file system contains both system and user data, such as log files, tables, and indexes.

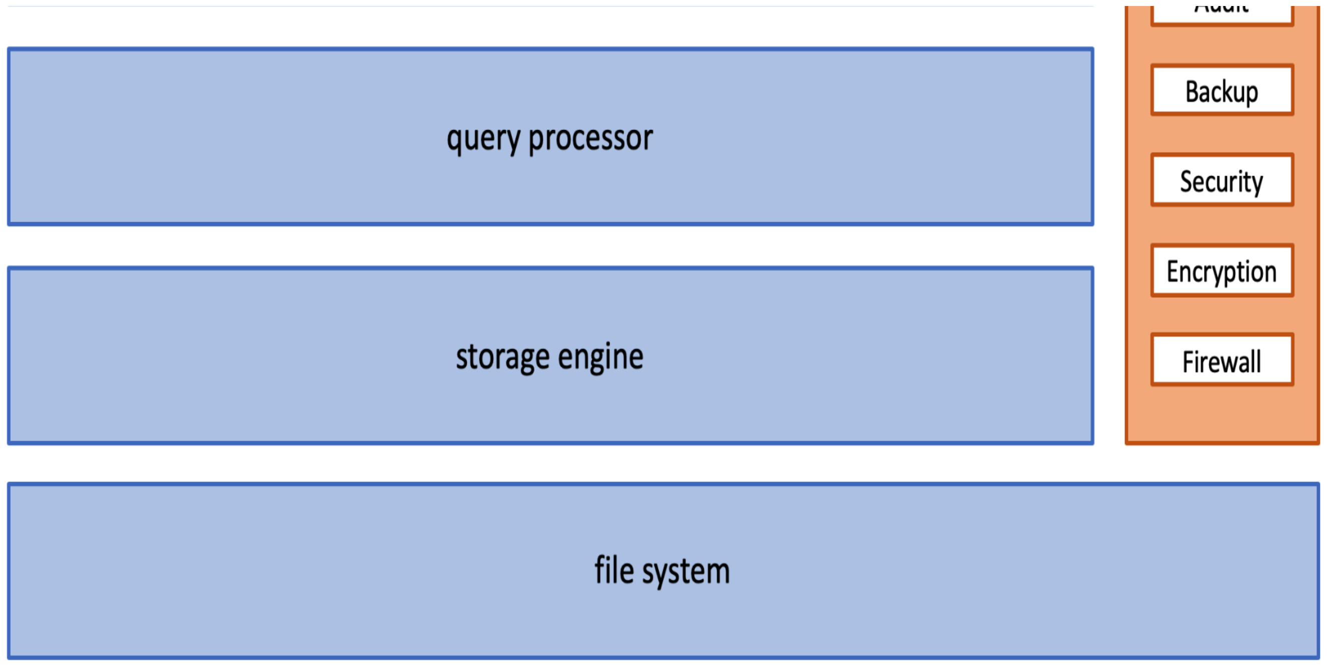
MySQL is available in a free version, called ***MySQL Server***, and a paid version, called ***MySQL Enterprise Edition***. The Enterprise Edition includes MySQL Server and components for high-end commercial installations, such as:

- ***Monitor*** collects and displays information on CPU, memory, and index utilization, as well as queries and results. Database administrators use Enterprise Monitor to manage and tune large databases with many users.
- ***Audit*** keeps track of all database changes. For each change, Audit tracks the time of change and who made the change. Audit supports government and business audit requirements for sensitive databases such as financial, medical, and defense.

Additional Enterprise Edition components provide advanced support for backup, security, encryption, and firewall. Enterprise Edition components are intended for database administrators, not users.

Figure 21.2.1: MySQL layers.





PARTICIPATION ACTIVITY | 21.2.1: MySQL layers.

Match the layer with the description.

If unable to drag and drop, refresh the page.

file system | storage engine | query processor | tools

	Manages database connections and compiles SQL queries.
	Contains MySQL Workbench.
	Does not directly interact with the query processor.
	Executes instructions compiled from SQL queries.

Reset

Tools

The tools layer includes Connectors and APIs, Workbench, and utility programs.

Connectors and APIs are groups of application programming interfaces, linking applications to the query processor layer. Connectors are newer and developed by Oracle, which sponsors MySQL. APIs are older and, with the exception of the C API, developed by other organizations. Most programmers use Connectors, but system programmers may write specialized utilities in C with the C API.

Workbench is a desktop application to manage and use databases. Workbench is designed for both database administrators and users.

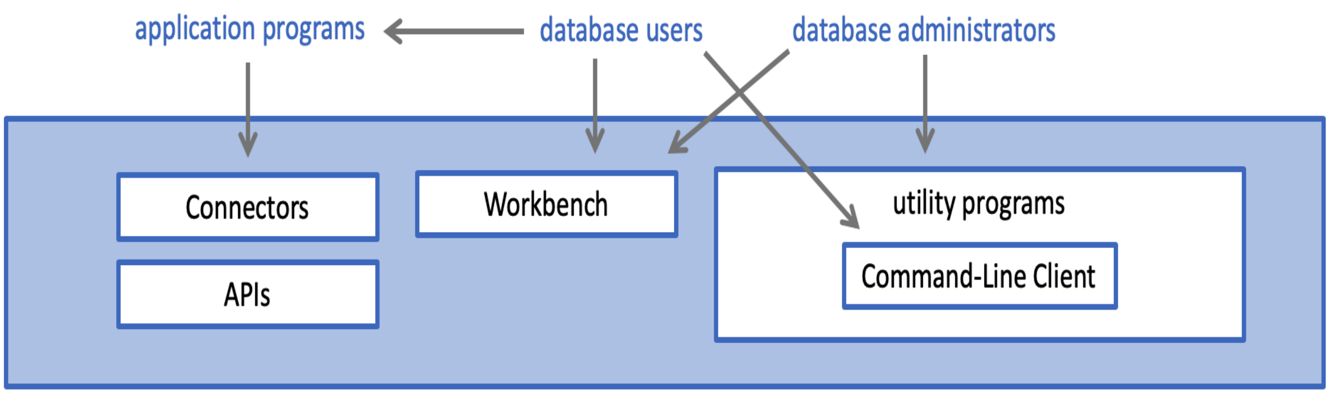
Utility programs include approximately 30 tools, grouped in five categories: installation, client, administrative, developer, and miscellaneous tools. Most utility programs are intended for database administrators or programmers. Example functions include:

- Upgrade existing databases to a new MySQL release
- Backup databases
- Import data to databases
- Inspect log files
- Administer database servers

The Command-Line Client is a particularly important utility program, commonly used by both database administrators and users. The Command-Line Client displays the `mysql>` prompt and processes individual SQL queries interactively.

Connectors, Workbench, and the Command-Line Client are described elsewhere in this material.

Figure 21.2.2: Tools.



PARTICIPATION ACTIVITY 21.2.2: Tools.

1) An application program is part of the tools layer.

tools layer.

- ○ True
- ○ False

2) The Workbench is intended primarily for database users.

- ○ True
- ○ False

3) A program that helps database administrators configure MySQL is considered a utility program.

- ○ True
- ○ False

4) Many Connector tools are built on top of API tools.

- ○ True
- ○ False

Query processor

The query processor layer has two main functions: manage connections and compile queries.

A ***connection*** is a link between tools and the query processor. Each connection specifies a database name, server address, logon name, and password. The connection manager creates connections and manages communications between tools and the query parser.

Query compilation generates a query execution plan. An ***execution plan*** is a detailed, low-level sequence of steps that specify exactly how to process a query.

PARTICIPATION ACTIVITY | 21.2.3: Execution plan that joins two tables.

Query

```
SELECT FullName, DeptName
FROM Employee, Department
WHERE Employee.DeptCode = Department.Code
AND Employee.State = 'IL';
```

Employee

Department

ID	FullName	State	DeptCode
2538	Lisa Ellison	IL	44
5384	Sam Snead	MO	82
6381	Maria Rodriguez	IL	82
8820	Jiho Chan	IL	12
9053	Salma Abbas	IL	44

Code	DeptName	ManagerID
44	Engineering	2538
82	Sales	6381
12	Marketing	6381
70	Accounting	NULL
99	Technical Support	NULL

Step 2 result

FullName	DeptCode
Jiho Chan	12
Lisa Ellison	44
Salma Abbas	44
Maria Rodriguez	82

Step 4 result

Code	DeptName
12	Marketing
44	Engineering
82	Sales

Query result

FullName	DeptName
Jiho Chan	Marketing
Lisa Ellison	Engineering
Salma Abbas	Engineering
Maria Rodriguez	Sales

Animation content:

Static figure:
An SQL statement and five tables appear. Two of the tables are named with step numbers of the query execution plan, not to be confused with animation step numbers.

Begin SQL code:

```
SELECT FullName, DeptName
FROM Employee, Department
WHERE Employee.DeptCode = Department.Code
AND Employee.State = 'IL';
```

End SQL code.

The Employee table has columns ID, FullName, State, and DeptCode. Employee has five rows:
2538, Lisa Ellison, IL, 44
5384, Sam Snead, MO, 82
6381, Maria Rodriguez, IL, 82
8820, Jiho Chan, IL, 12
9053, Salma Abbas, IL, 44

The Department table has columns Code, DeptName, and ManagerID. Department has five rows:

44, Engineering, 2538
82, Sales, 6381
12, Marketing, 6381
70, Accounting, NULL
99, Technical Support, NULL

The Step 2 result table has columns FullName and DeptCode, with four rows:
Jiho Chan, 12
Lisa Ellison, 44
Salma Abbas, 44
Maria Rodiriguez, 82

The Step 4 result table has columns Code and DeptName, with three rows:
12, Marketing
44, Engineering
82, Sales

The Query result table has columns FullName and DeptName, with four rows:
Jiho Chan, Marketing
Lisa Ellison, Engineering
Salma Abbas, Engineering
Maria Rodiriguez, Sales

Step 1: The query selects employee name and department name for employees that work in Illinois. The SQL statement appears. Tables Employee and Department appear.

Step 2: Query step 1 - The plan retrieves Illinois employees using an index on State. The Step 2 result table appears. The table includes only employees with State = 'IL'.

Step 3: Query step 2 - The plan sorts selected employees by DeptCode. Rows in the Step 2 result table are sorted by DeptCode.

Step 4: Query step 3 - The plan retrieves matching departments using a table scan. The Step 4 result table appears. The table includes only rows for which DeptCode matches a value in the Step 2 result table

Step 5: Query step 4 - The plan sorts the selected departments by Code. Rows in the Step 4 result table are sorted by DeptCode.

Step 6: Query step 5 - The plan merges the two result tables using the join technique called 'sort-merge'. An arrow points to rows of the Step 2 result table. Another arrow points to rows of the Step 4 result table. The arrows move through table rows one-by-one, matching rows with the same DeptCode. As arrows move through the tables, the FullName and DeptName from matching rows appear in the Query result table.

Animation captions:

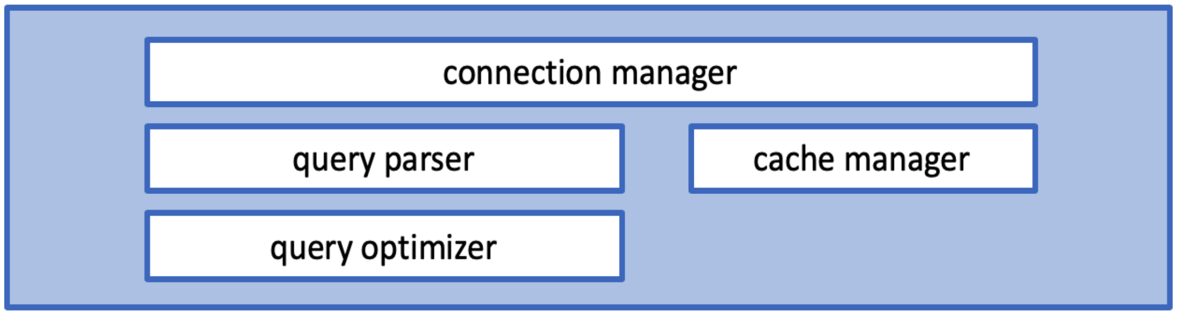
1. The query selects employee name and department name for employees that work in Illinois.
2. Step 1: The plan retrieves Illinois employees using an index on State.
3. Step 2: The plan sorts selected employees by DeptCode.
4. Step 3: The plan retrieves matching departments using a table scan.
5. Step 4: The plan sorts the selected departments by Code.
6. Step 5: The plan merges the two result tables using the join technique called 'sort-merge'.

The query processor generates an execution plan in two steps:

1. The ***query parser*** checks each query for syntax errors and converts valid queries to an internal representation.
2. The ***query optimizer*** reads the internal representation, generates alternative execution plans, estimates execution times, and selects the fastest plan. Estimates are based on heuristics and statistics about data, like the number of rows in each table and the number of values in each column. These statistics are maintained in the data dictionary, described below.

For optimal performance, the query processor layer has a ***cache manager*** that stores reusable information in main memory. Ex: The cache manager retains execution plans for queries that are submitted multiple times. If data used in repeated queries does not change, the cache manager may also save query results.

Figure 21.2.3: Query processor.



PARTICIPATION ACTIVITY 21.2.4: Query processor.

1) Which component detects a missing semicolon at the end of an SQL statement?

- ○ Connection manager
- ○ Query parser
- ○ Query optimizer

2) Which component detects an incorrect database server address?

- ○ Connection manager
- ○ Query optimizer
- ○ Cache manager

3) Which component determines that a query was recently executed?

- ○ Connection manager
- ○ Query optimizer
- ○ Cache manager

Storage engine

The storage engine layer has two main functions: transaction management and data access.

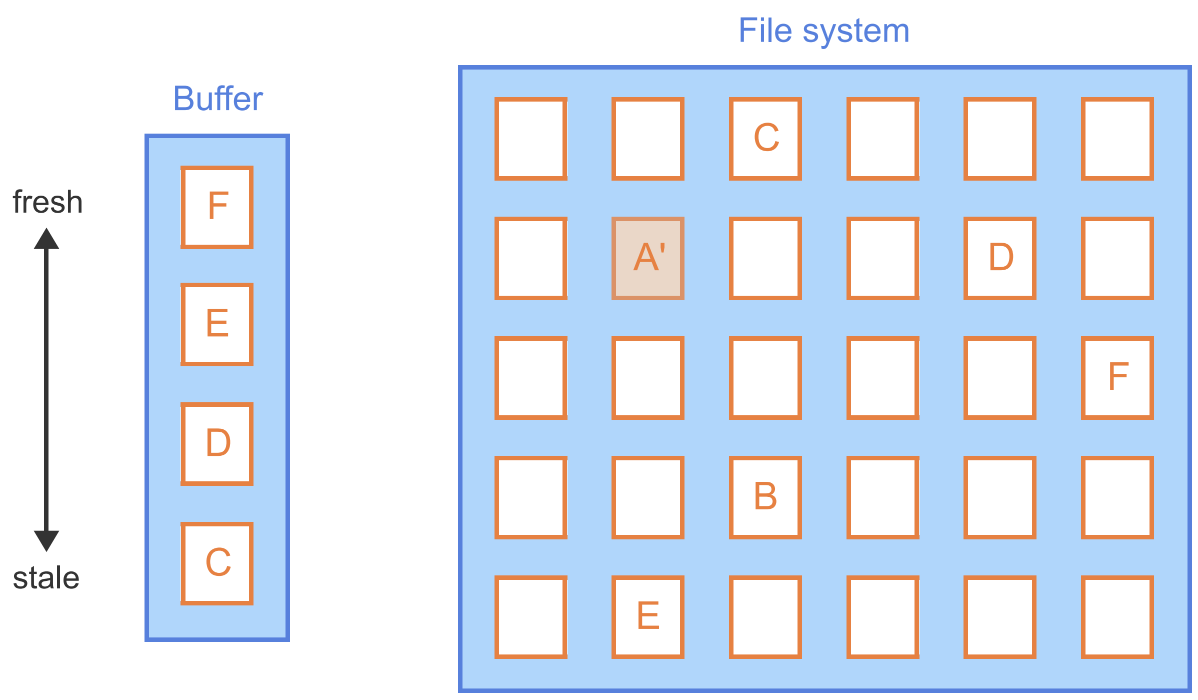
Transaction management includes the concurrency system, recovery system, and lock manager. These components ensure all transactions are atomic, consistent, isolated, and durable, as explained elsewhere in this material.

The data access component communicates with the file system and translates table, column, and index reads into block addresses.

To reduce data access time, the ***buffer manager*** retains data blocks from the file system for possible reuse. The data blocks are retained in an area of main memory called the ***buffer***. Ex: If queries frequently access department data, the buffer manager may retain some or all blocks of the Department table. The buffer manager is similar to the cache manager of the query processor layer.

The buffer manager has a fixed amount of memory. As the database processes queries and reads blocks, an algorithm determines which blocks to retain and which to discard. The InnoDB buffer manager uses a ***least recently used*** or ***LRU*** algorithm. The LRU algorithm tracks the time each block was last accessed and, when space is needed, discards 'stale' blocks. If data in a block has been updated, discarded blocks are first saved on disk.

PARTICIPATION ACTIVITY | 21.2.5: Least recently used algorithm.



Animation content:

Step 1: Data in the file system is organized into blocks. A file system containing data blocks.

Step 2: The database reads blocks from the file system and stores blocks in the buffer. A buffer next to the file system with an arrow indicating blocks enter at the fresh end and exit at the stale end. The buffer can hold four blocks. A duplicate of block A from the file system is added to the fresh end of the buffer, indicating block A has been read.

Step 3: As the database reads new blocks, prior blocks become 'stale'. Block B is read in the file system. A duplicate of block B is added to the fresh end of the buffer, pushing block A in the buffer toward the stale end.

Step 4: Data block A becomes 'fresh' when read or updated. Block A in the buffer is updated. Updated block A switches positions with block B in the buffer. Updated block A is now fresher than block B.

Step 5: Eventually, the buffer fills up. Block C is read in the file system, adding a duplicate to the fresh end of the buffer. Block B and updated block A in the buffer are pushed toward the stale end. Block D is read next in the file system, adding a duplicate to the fresh end of the buffer. The buffer is now full. From fresh to stale, the buffer holds block D, block C, updated block A, and block B.

Step 6: To make space for block E, stale block B is deleted from the buffer. Block E is read in the file system. The duplicate of block E is stored in the fresh end of the filled up buffer, forcing block B out of the buffer on the stale end where Block B disappears.

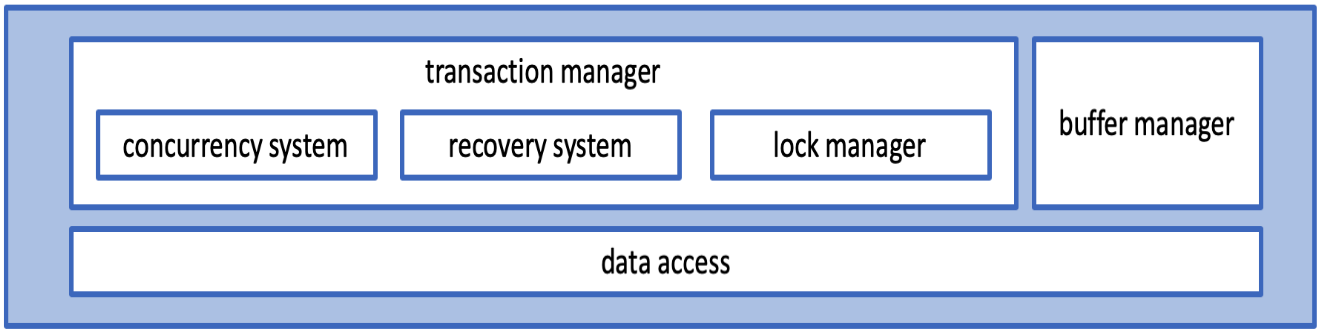
Step 7: Block A has been updated and must be saved to the file system before deletion from the buffer. Updated block A is removed from the buffer and saved in the file system, replacing the original block A. Block F is read in the file system. The read block F is stored at the fresh end of the buffer. The buffer is full again.

Animation captions:

1. Data in the file system is organized into blocks.
2. The database reads blocks from the file system and stores blocks in the buffer.
3. As the database reads new blocks, prior blocks become 'stale'.
4. Data block A becomes 'fresh' when read or updated.
5. Eventually, the buffer fills up.
6. To make space for block E, stale block B is deleted from the buffer.
7. Block A has been updated and must be saved to the file system before deletion from the buffer.

MySQL supports nine storage engines, including InnoDB, MyISAM, CSV, and MEMORY. Each storage engine is optimized for a specific application, such as transaction management or analytics. The database administrator can assign a different storage engine to each table in a database. InnoDB is the default and most commonly used storage engine. InnoDB supports transactions, but many other storage engines do not.

Figure 21.2.4: Storage engine.



PARTICIPATION ACTIVITY | 21.2.6: Storage engine.

1) What block does InnoDB discard when more space is needed in the buffer?

- ○ The block that was loaded to the buffer first.
- ○ The block that was loaded to the buffer last.
- ○ The block that has not been accessed for the longest time.

2) After the above animation completes, block C is updated and block G is read from the file system. What happens to the buffer?

- ○ Block F is discarded and replaced with block G.
- ○ Block C is saved to the file system and discarded. Block G is added to the top.
- ○ Block C moves to the top, block D is discarded, and block G moves to the top.

3) Which is *not* a MySQL storage engine?

- ○ MEMORY
- ○ MongoDB
- ○ CSV

4) Which storage engine is commonly used for transaction management?

- ○ InnoDB
- ○ ARCHIVE
- ○ MEMORY

5) Oracle Database has multiple storage engines.

- ○ True
- ○ False

File system

The file system layer consists of data stored on storage media and organized in files. The file system contains three types of data for each database: user data, log files, and a data dictionary.

User data includes tables and indexes. Specific storage structures for tables and indexes are described elsewhere in this material.

Log files contain a detailed, sequential record of each change applied to a database. The recovery system uses log files to restore data in the event of a transaction, system, or storage media failure.

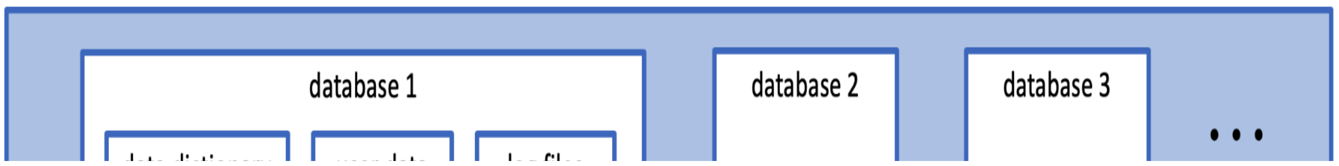
A ***catalog***, also known as a ***data dictionary***, is a directory of tables, columns, keys, indexes, and other objects in a relational database. All relational databases contain a catalog. Query processors and storage managers use catalog information when queries are processed and executed.

MySQL uses the term 'data dictionary'. The MySQL data dictionary contains roughly 30 tables, including:

- `tables` describes all tables
- `table_stats` contains table statistics, such as the number of rows in each table
- `columns` describes all columns
- `foreign_keys` describes all foreign keys
- `indexes` describes all indexes
- `routines` describes all stored procedures and stored functions
- `triggers` describes all triggers

Data dictionary tables cannot be accessed directly with SELECT, INSERT, UPDATE, and DELETE queries. However, the table contents can be accessed indirectly. The SHOW query is compiled as a SELECT query against dictionary tables. Ex: SHOW COLUMNS generates a SELECT query against the `columns` table. CREATE generates an INSERT, ALTER generates an UPDATE, and DROP generates a DELETE against dictionary tables.

Figure 21.2.5: File system.



PARTICIPATION ACTIVITY | 21.2.7: File system.

1) The ________ contains one row for each database object.

Check Show answer

2) The ________ table provides various table statistics and is used by the query optimizer to generate efficient execution plans.

Check Show answer

3) ________ support database recovery in the event of a system failure.

Check Show answer

Exploring further:

- MySQL utility programs
- MySQL data dictionary
- InnoDB storage engine

21.3 MySQL Workbench: Import and export

Import a database

MySQL Workbench can import an entire database from an SQL file. The SQL file normally contains SQL statements to create a database, create tables for the database, and insert the data into the tables.

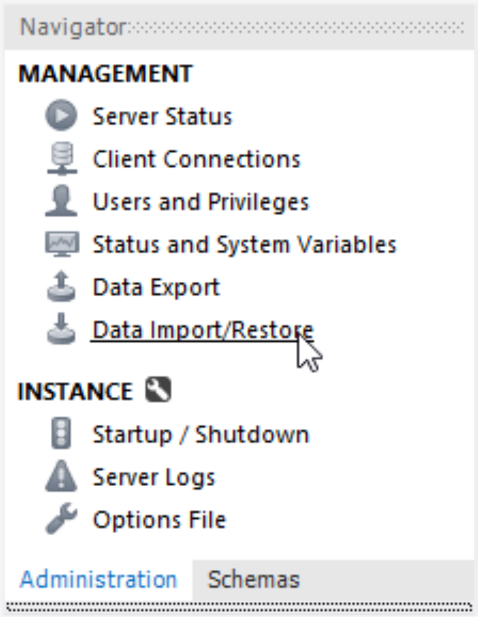
The figure below shows the contents of an SQL file called company.sql. The SQL statements create a database called company with Employee and Department tables. Five employees and four departments are inserted into the tables.

Figure 21.3.1: company.sql for importing.

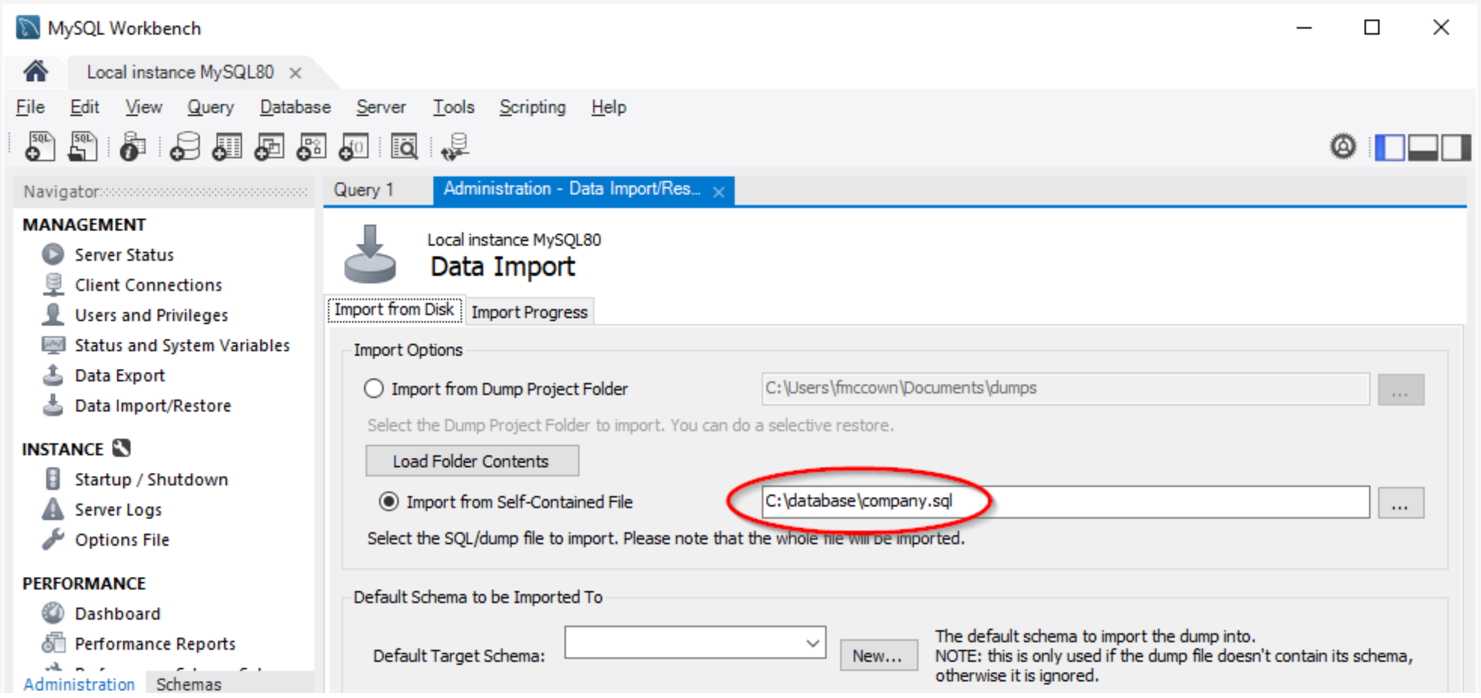
```
CREATE DATABASE IF NOT EXISTS company;

USE company;

DROP TABLE IF EXISTS Employee;
DROP TABLE IF EXISTS Department;

CREATE TABLE Employee (
   ID SMALLINT UNSIGNED,
   Name VARCHAR(60),
   Salary DECIMAL(7,2) NOT NULL,
   PRIMARY KEY(ID)
);

CREATE TABLE Department (
   Code    TINYINT UNSIGNED
AUTO_INCREMENT,
   Name    VARCHAR(20) NOT NULL,
   Manager SMALLINT UNSIGNED,
   PRIMARY KEY (Code)
);

INSERT INTO Employee VALUES
   (2538, 'Lisa Ellison', 45000),
   (5384, 'Sam Snead', 32000),
   (6381, 'Maria Rodriguez', 95000),
   (7343, 'Gary Smith', 24500),
   (8392, 'Anna Watson', 41000);

INSERT INTO Department VALUES
   (44, 'Engineering', 2538),
   (82, 'Sales', 6381),
   (12, 'Marketing', 6381),
   (99, 'Technical support', NULL);
```

The following steps show how to import a database from an SQL file. The screenshots below are from Windows. Screens and buttons vary slightly on a Mac.

1. Start MySQL Workbench and connect to MySQL server.
2. Click the **Administration** tab in the Navigator sidebar.
3. In the Management section, click **Data Import/Restore**.



4. Click **Import from Self-Contained File**.
5. Click **...** button.
6. Navigate to and select the SQL file to import.
7. Click **Start Import** button.



Local instance MySQL80

Data Import

Import Options

( ) Import from Dump Project Folder C:\Users\fmccown\Documents\dumps ...

Select the Dump Project Folder to import. You can do a selective restore.

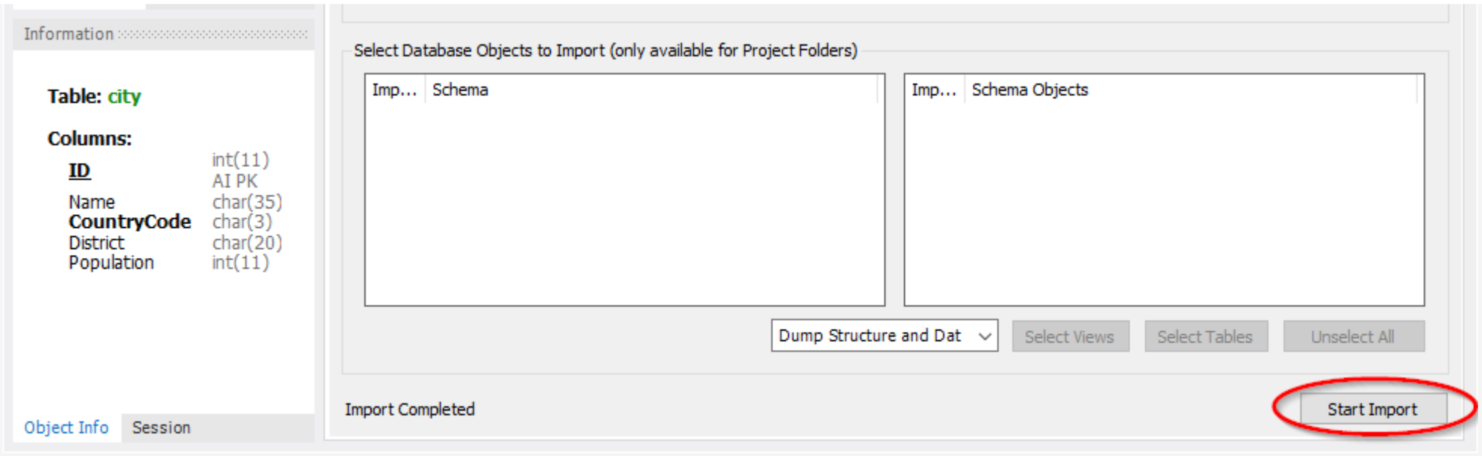
(•) Import from Self-Contained File C:\database\company.sql ...

Select the SQL/dump file to import. Please note that the whole file will be imported.

Default Schema to be Imported To

Default Target Schema:

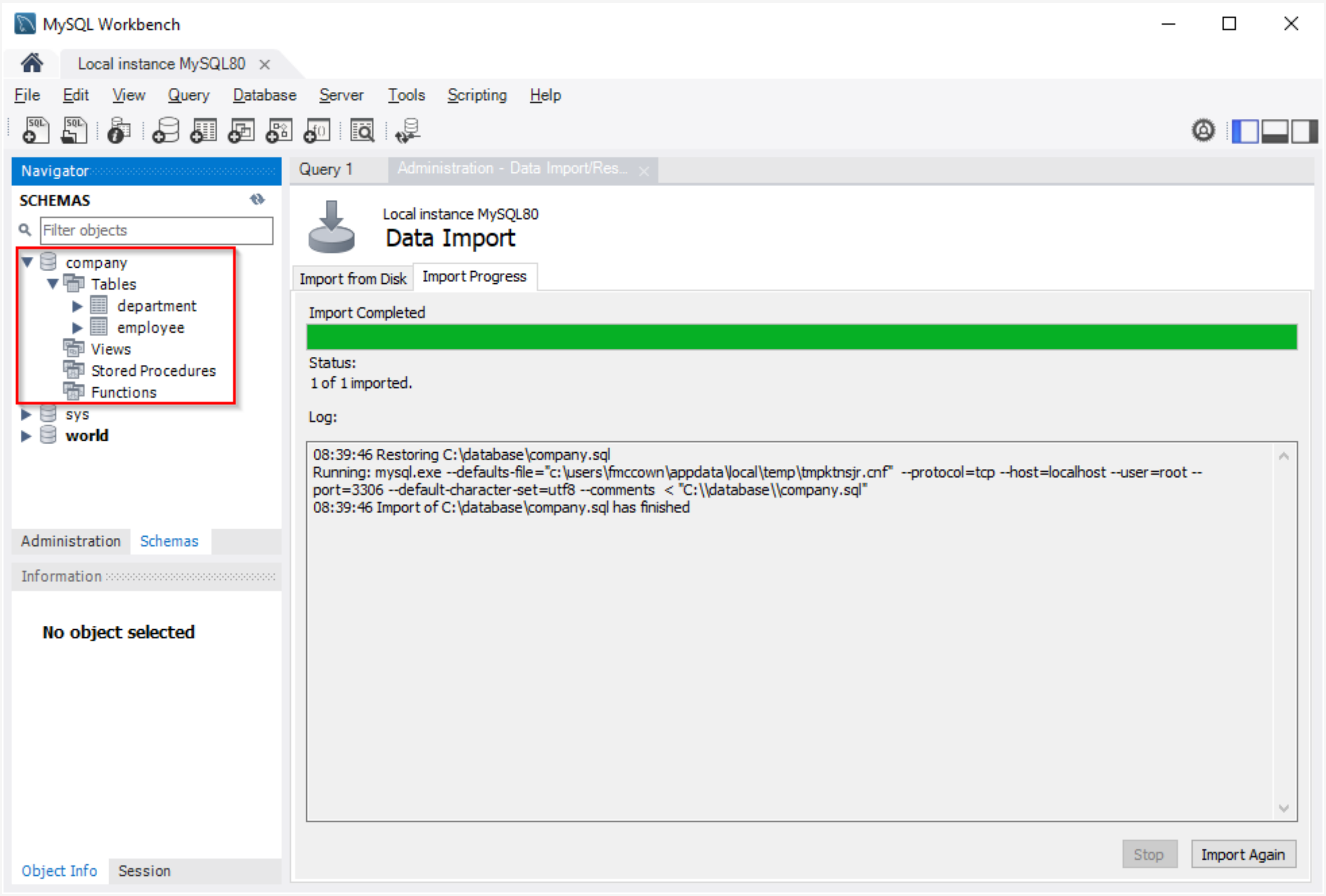
The default schema to import the dump into.
NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored.



8. Wait until notification that import has completed.

To verify the import worked:

1. Click the **Schemas** tab in the Navigator sidebar.
2. Click the refresh button in the upper-right corner of the Navigator sidebar to refresh the list of Schemas.
3. Verify the new database appears in the Schemas list.



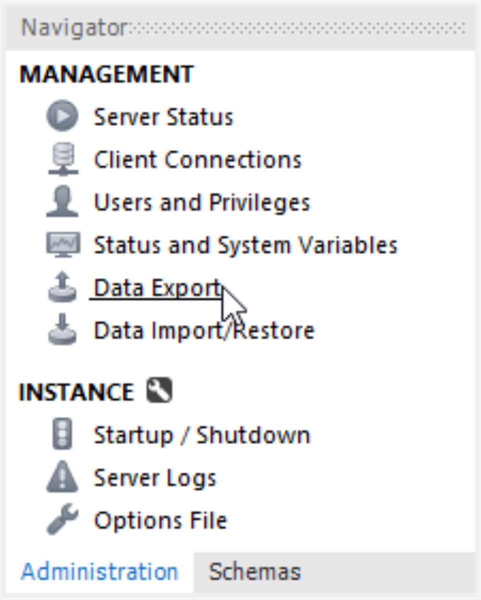
```
08:39:46 Restoring C:\database\company.sql
Running: mysql.exe --defaults-file="c:\users\fmccown\appdata\local\temp\tmpktnsjr.cnf"  --protocol=tcp --host=localhost --user=root --
port=3306 --default-character-set=utf8 --comments  < "C:\\database\\company.sql"
08:39:46 Import of C:\database\company.sql has finished
```

Export a database

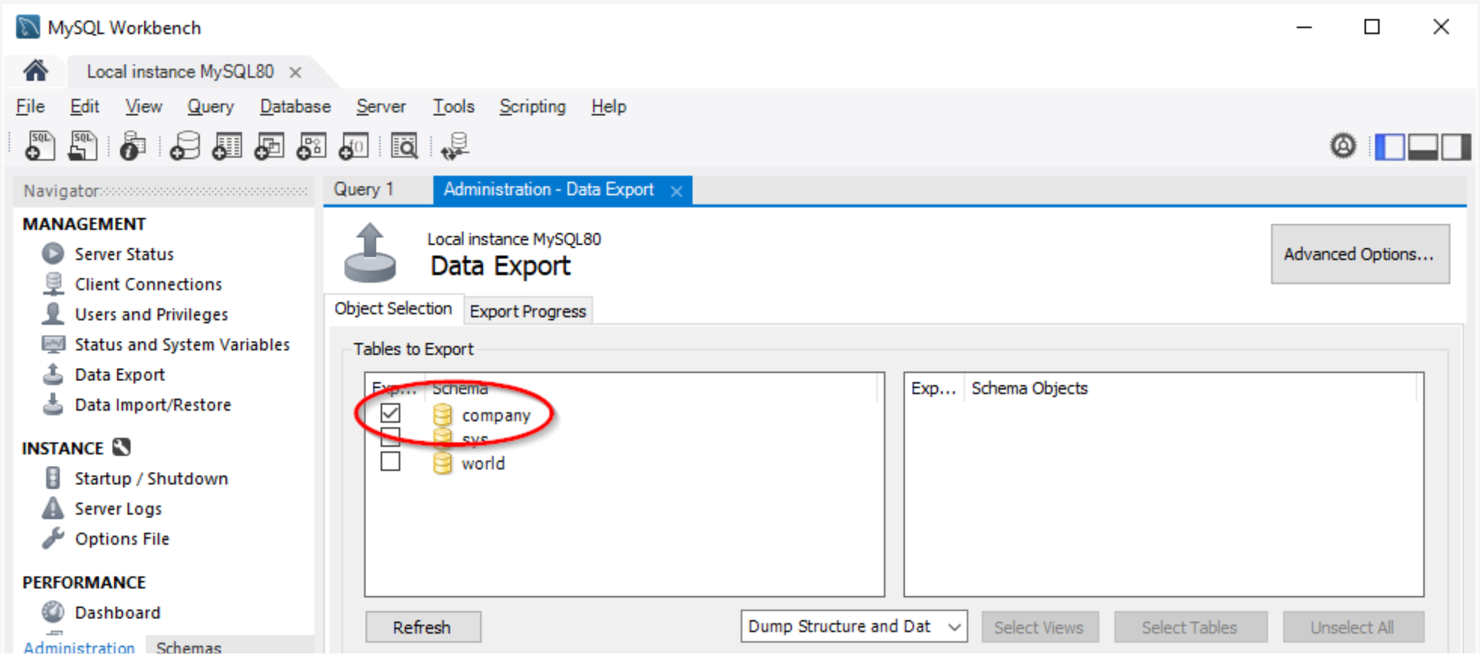
MySQL Workbench can export an entire database to an SQL text file.

The following steps export a database to a file. The screenshots below are from Windows. Screens and buttons vary slightly on a Mac.

1. Start MySQL Workbench and connect to MySQL server.
2. Click the **Administration** tab in the Navigator sidebar.
3. In the Management section, click **Data Export**.



4. Check the checkbox next to the database to be exported.
5. Select **Export to Self-Contained File** and enter the file path.
6. Click **Start Export** button.



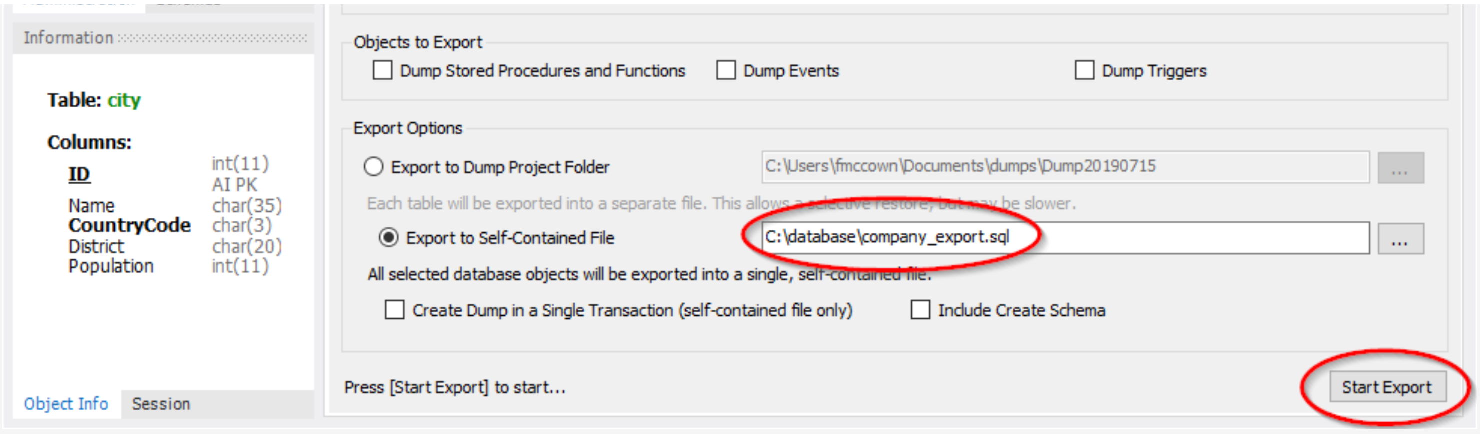


Table: city

Columns:

ID	int(11) AI PK
Name	char(35)
CountryCode	char(3)
District	char(20)
Population	int(11)

- [ ] Dump Stored Procedures and Functions
- [ ] Dump Events
- [ ] Dump Triggers

( ) Export to Dump Project Folder: C:\Users\fmccown\Documents\dumps\Dump20190715

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

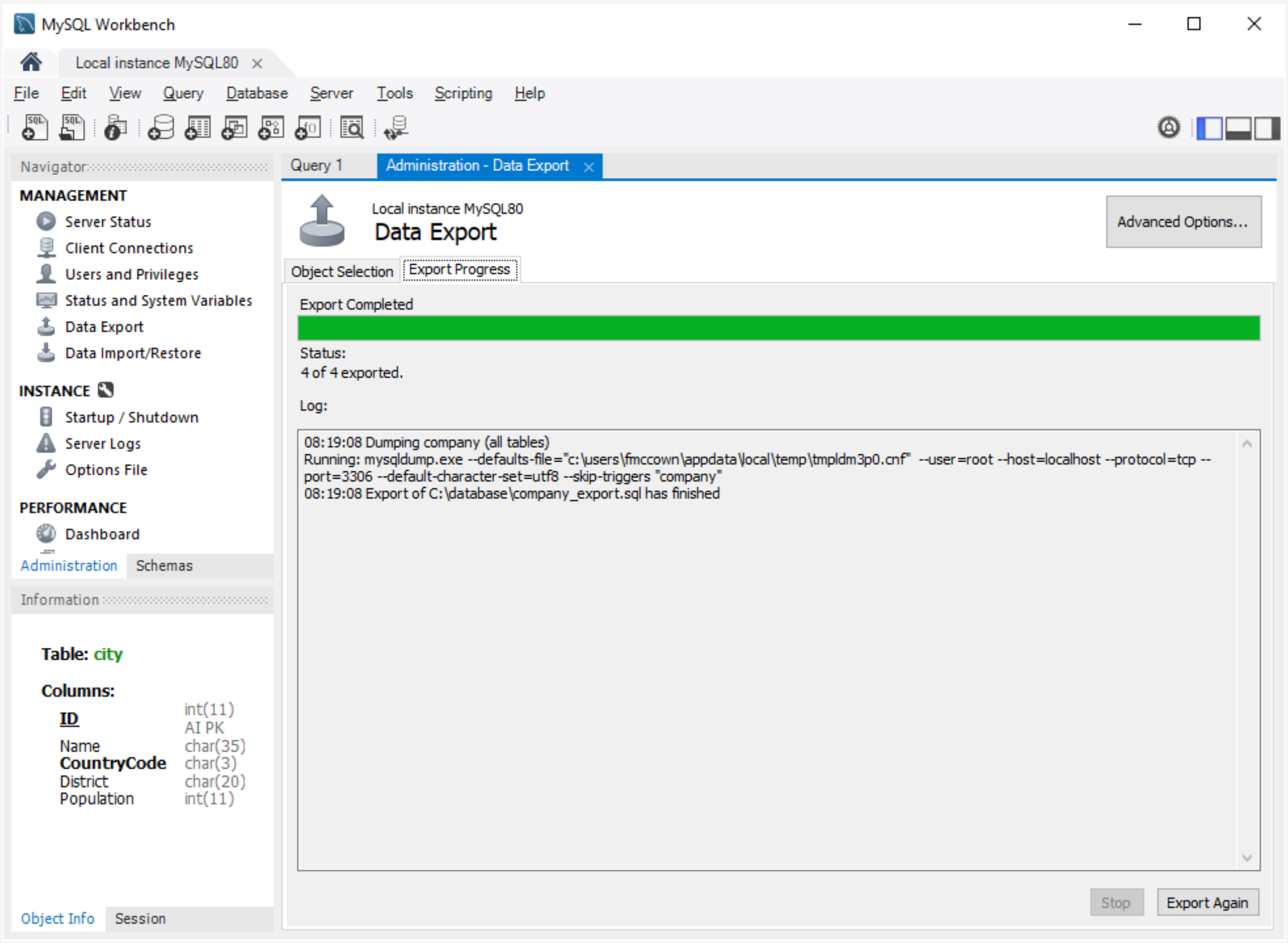
(●) Export to Self-Contained File: C:\database\company_export.sql

All selected database objects will be exported into a single, self-contained file.

- [ ] Create Dump in a Single Transaction (self-contained file only)
- [ ] Include Create Schema

Press [Start Export] to start...

7. Wait until notification that export has completed.



Export Completed

Status:
4 of 4 exported.

Log:

```
08:19:08 Dumping company (all tables)
Running: mysqldump.exe --defaults-file="c:\users\fmccown\appdata\local\temp\tmpldm3p0.cnf"  --user=root --host=localhost --protocol=tcp --
port=3306 --default-character-set=utf8 --skip-triggers "company"
08:19:08 Export of C:\database\company_export.sql has finished
```

Table: city

Columns:

ID	int(11) AI PK
Name	char(35)
CountryCode	char(3)
District	char(20)
Population	int(11)

Verify the export worked by locating the SQL file produced by the export.

Exported files will use backticks around all table and column names. The ***backtick*** (`) delimits literals that represent identifiers, which allows spaces and reserved words to be used as identifiers. Ex: `` CREATE TABLE `Employee` ``

Importing table data

Data may be imported into a specific table from a CSV (Comma-Separated Values) file or JSON file.

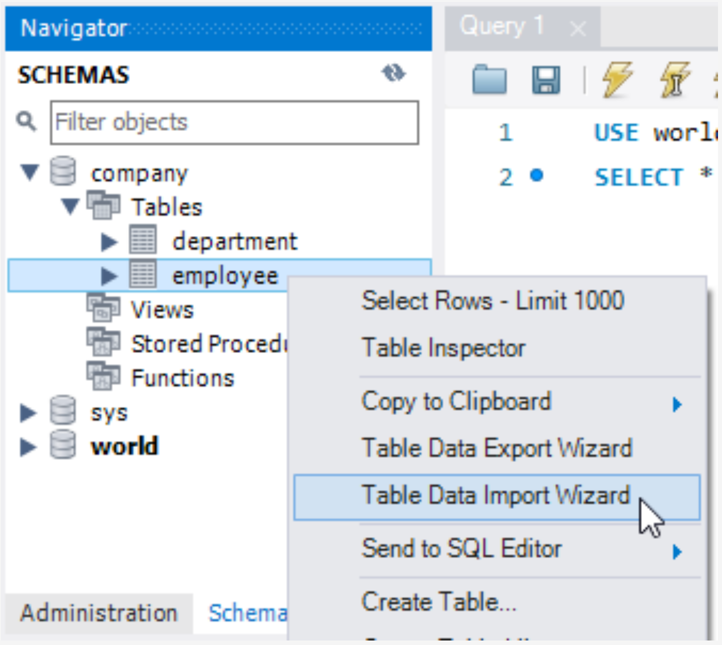
The figure below shows employee data in a CSV file named employee_table.csv. The first line lists the column names, and all other lines contain the ID, name, and salary for each employee. The CSV file uses commas to separate each data field, but MySQL supports other field separators like semicolons, colons, vertical bars, and tabs.

Figure 21.3.2: employee_table.csv.

```
"ID","Name","Salary"
2538,"Lisa Ellison",45000
5384,"Sam Snead",32000
6381,"Maria
Rodriguez",95000
7343,"Gary Smith",24500
8392,"Anna Watson",41000
```

The following steps import a CSV or JSON file. The screenshots below are from Windows. Screens and buttons vary slightly on a Mac.

1. Click the **Schemas** tab in the Navigator sidebar, and expand the desired database's table list.
2. Right-click the table name that will receive the imported data.
3. Choose **Table Data Import Wizard** from the context menu.



4. Click the **Browse...** button, select the CVS or JSON file to import, and click **Next**.

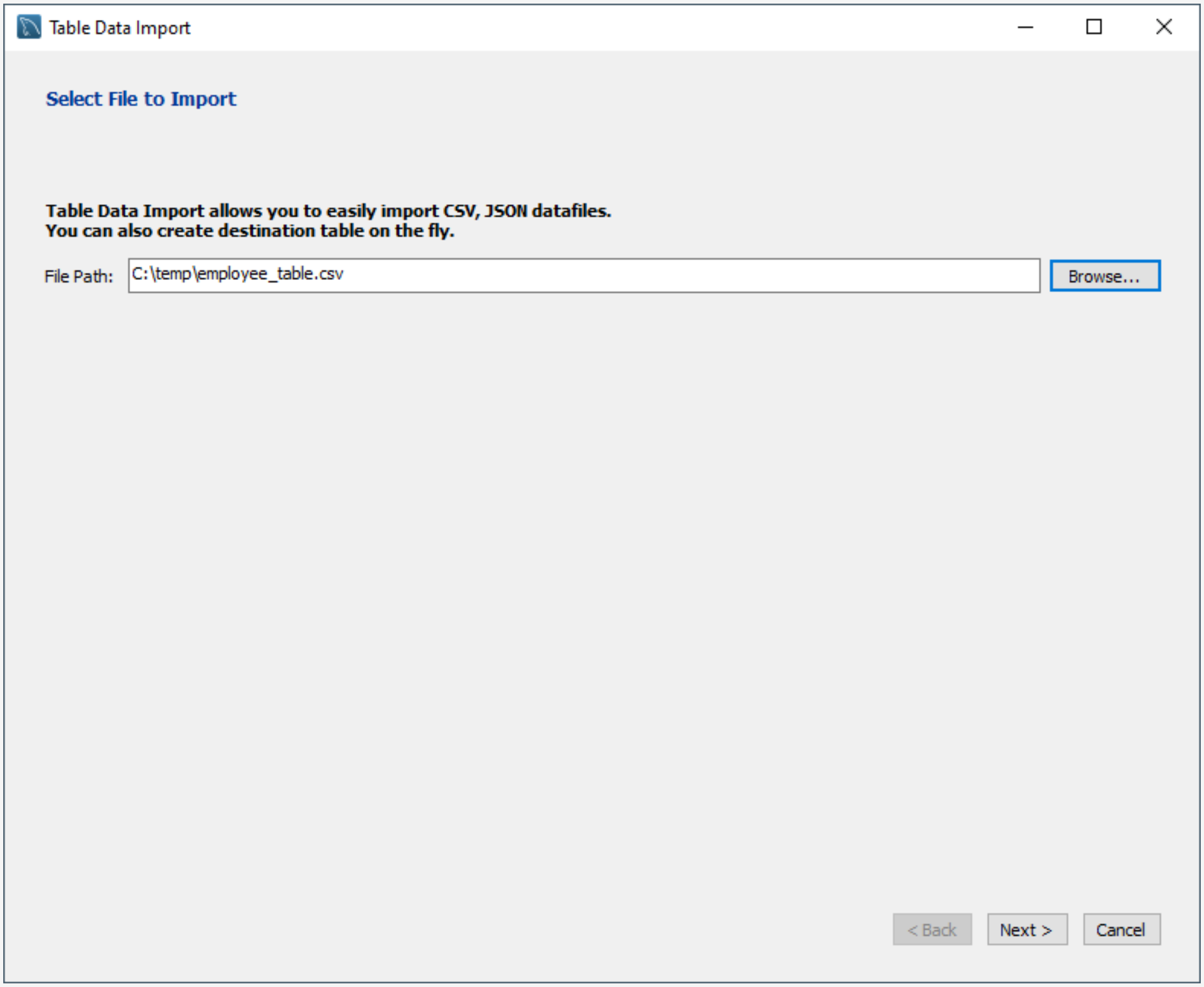


Table Data Import

Select File to Import

Table Data Import allows you to easily import CSV, JSON datafiles.
You can also create destination table on the fly.

File Path: C:\temp\employee_table.csv

5. Click **Next** to import the data into the current table.

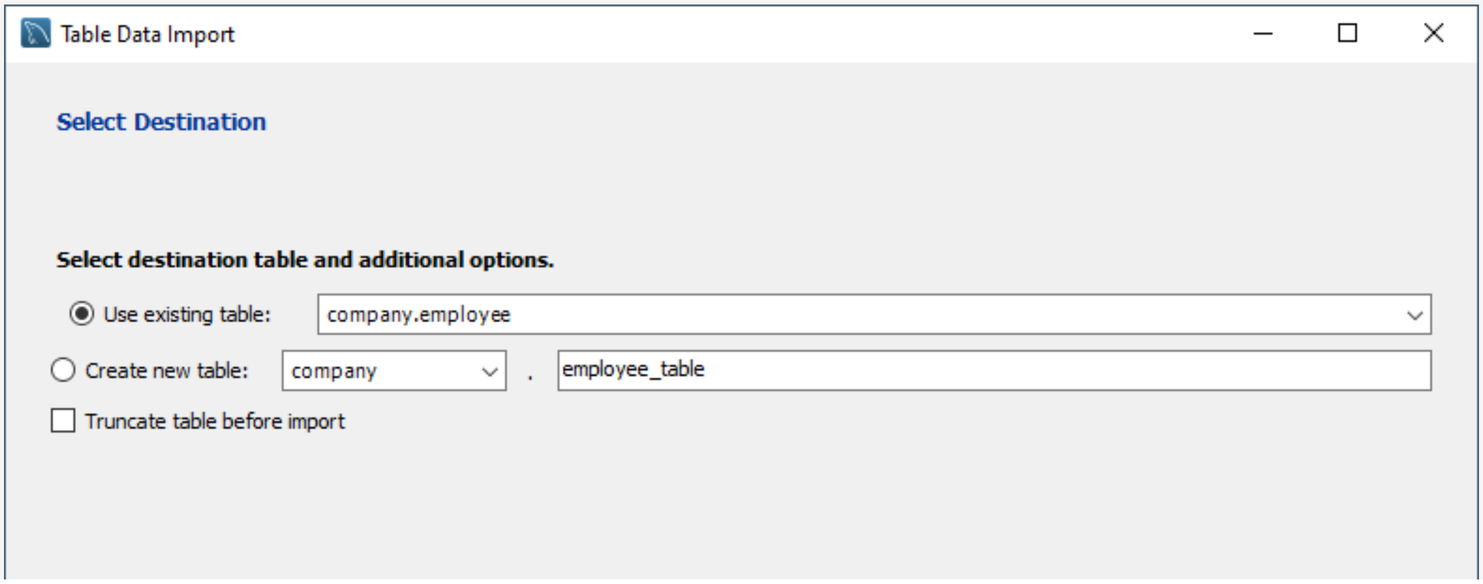


Table Data Import

Select Destination

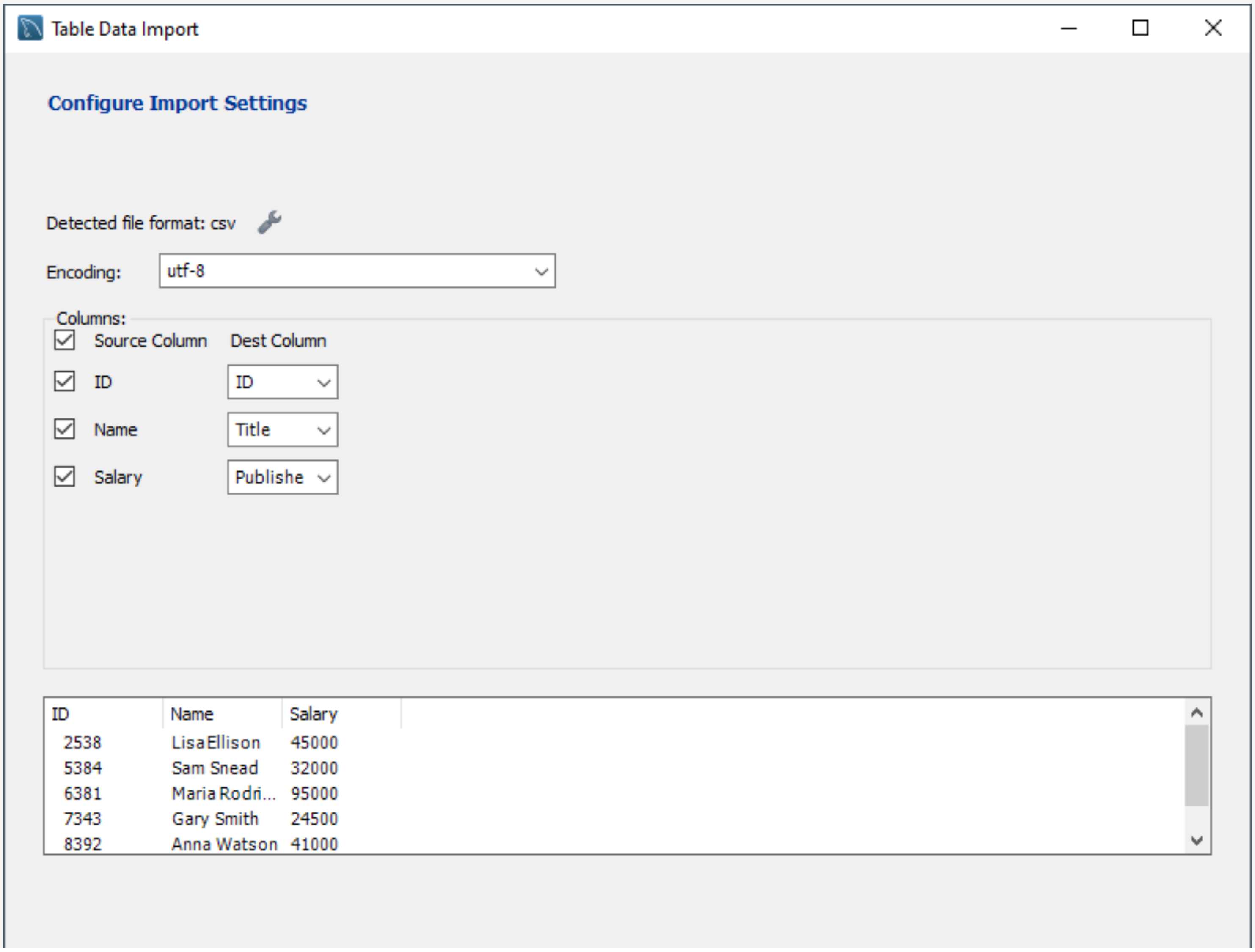
Select destination table and additional options.

(●) Use existing table: company.employee

( ) Create new table: company . employee_table

☐ Truncate table before import

6. The columns from the file are displayed and checked by default, and the data to import is shown underneath. Click **Next**.



Configure Import Settings

Detected file format: csv

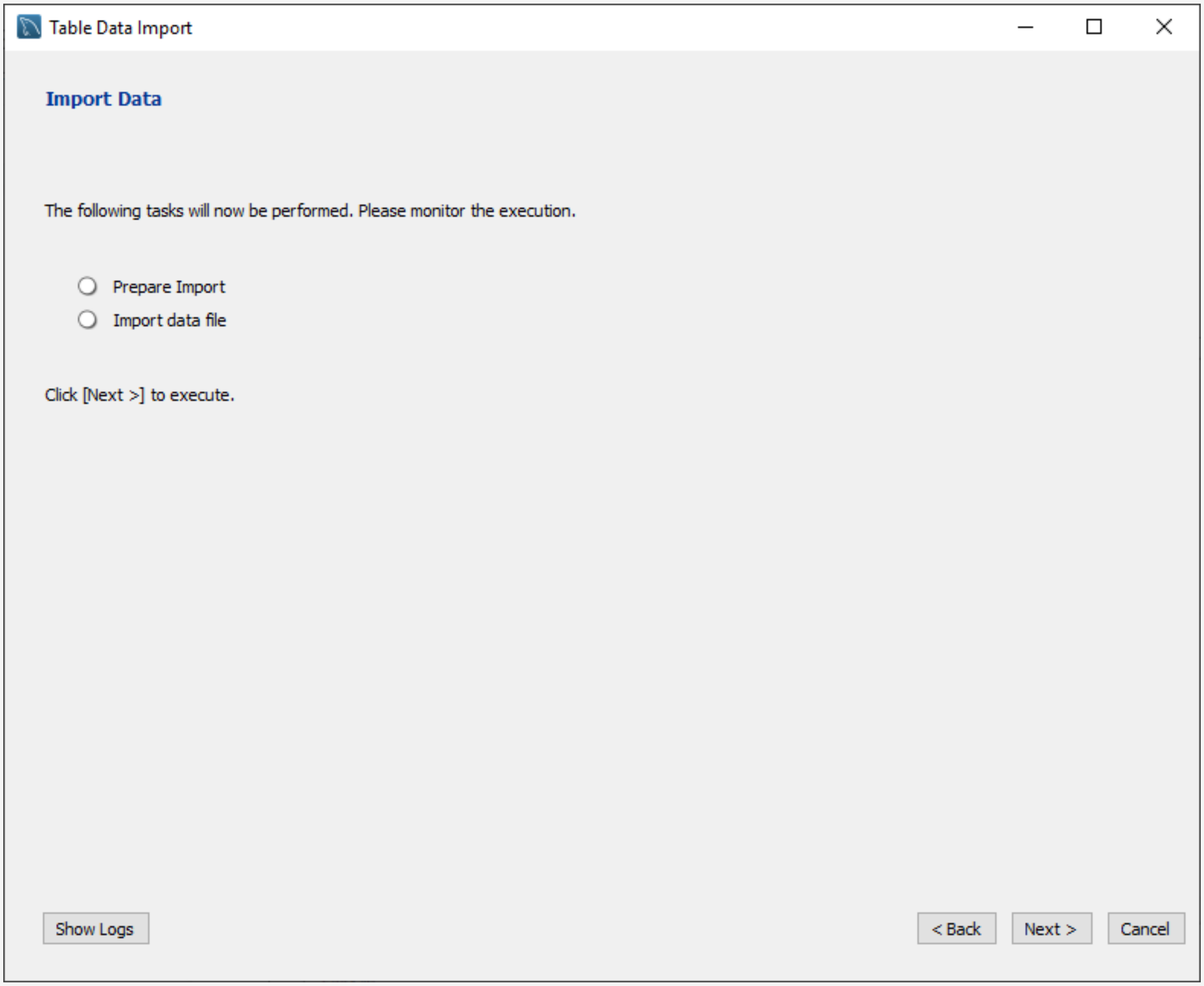
Encoding: utf-8

Columns:

	Source Column	Dest Column
✓	ID	ID
✓	Name	Title
✓	Salary	Publishe

ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	32000
6381	Maria Rodri...	95000
7343	Gary Smith	24500
8392	Anna Watson	41000

7. Click **Next** to begin the import.



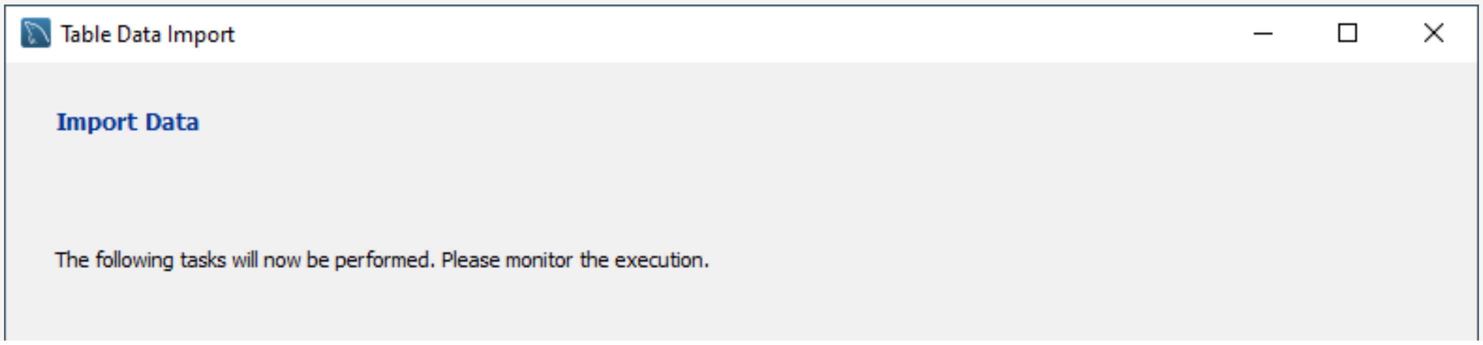
Import Data

The following tasks will now be performed. Please monitor the execution.

- Prepare Import
- Import data file

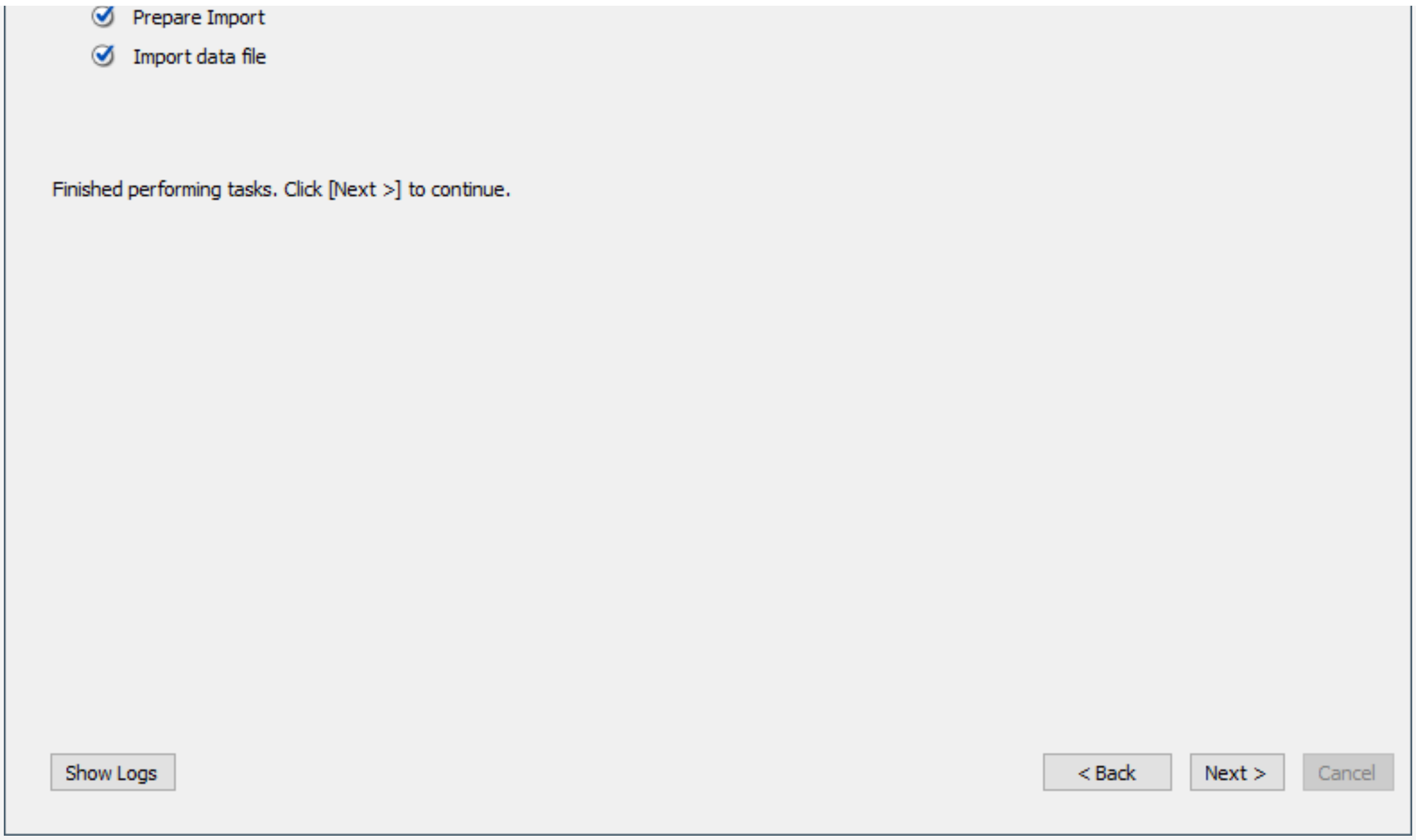
Click [Next >] to execute.

8. Click **Next** again to confirm the import has finished.



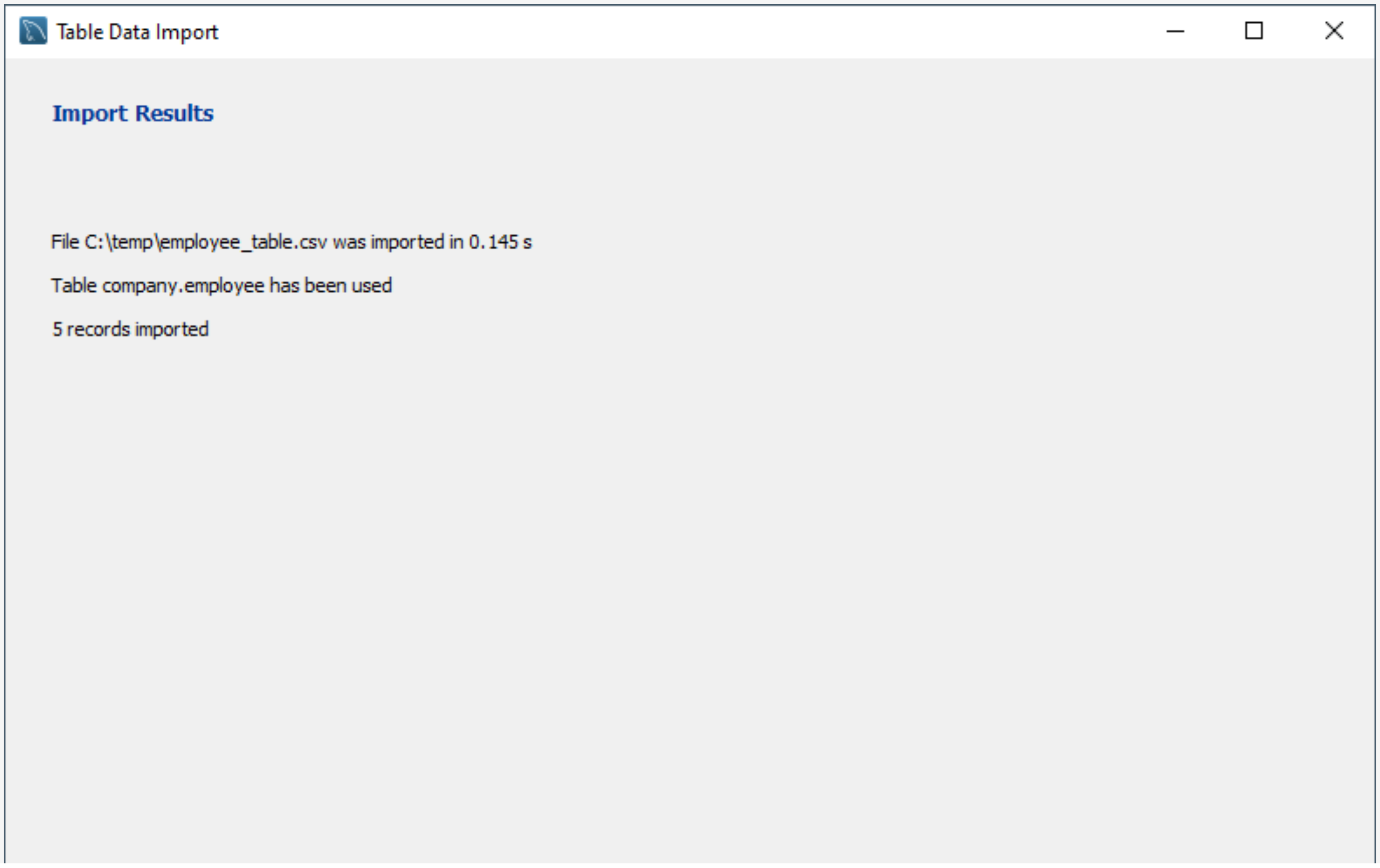
Import Data

The following tasks will now be performed. Please monitor the execution.



Finished performing tasks. Click [Next >] to continue.

9. View the import summary and press **Finish**.



Import Results

File C:\temp\employee_table.csv was imported in 0.145 s

Table company.employee has been used

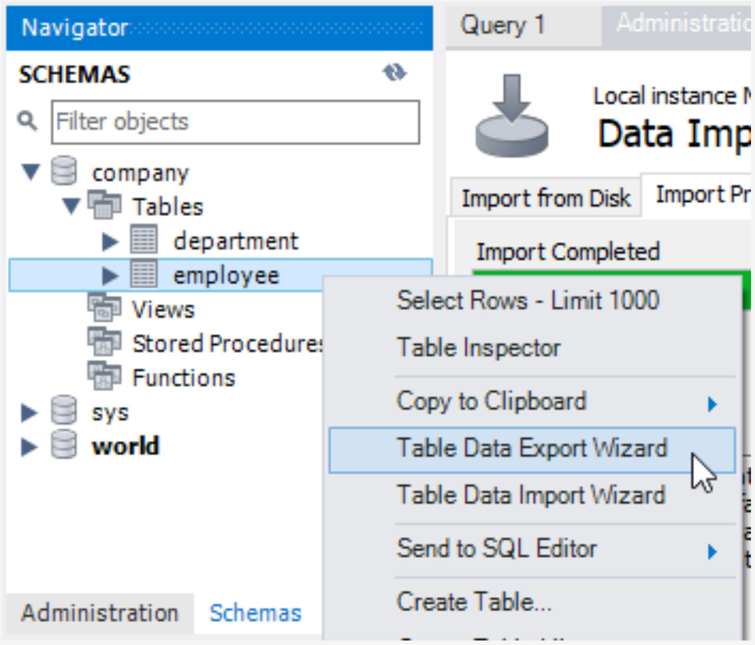
5 records imported

Note that if a row in the import data violates any column constraints, the row is not imported, and no error message is shown. Ex: If the row uses the same primary key value as an existing row in the table, the row is not imported.

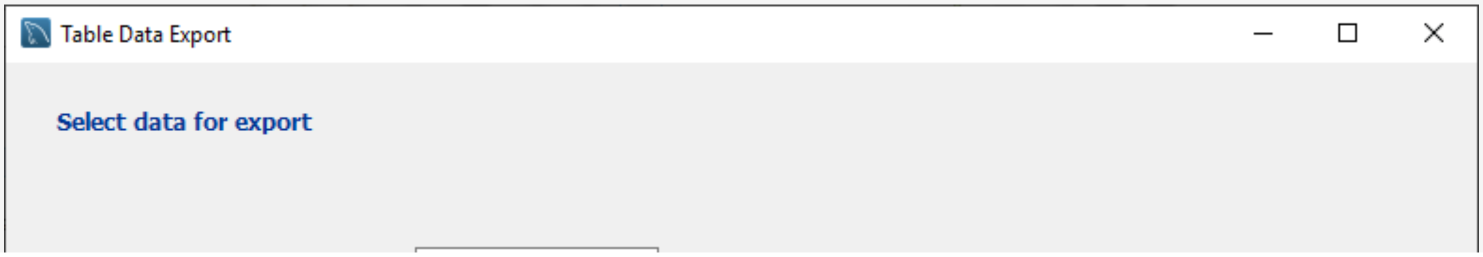
Export table data

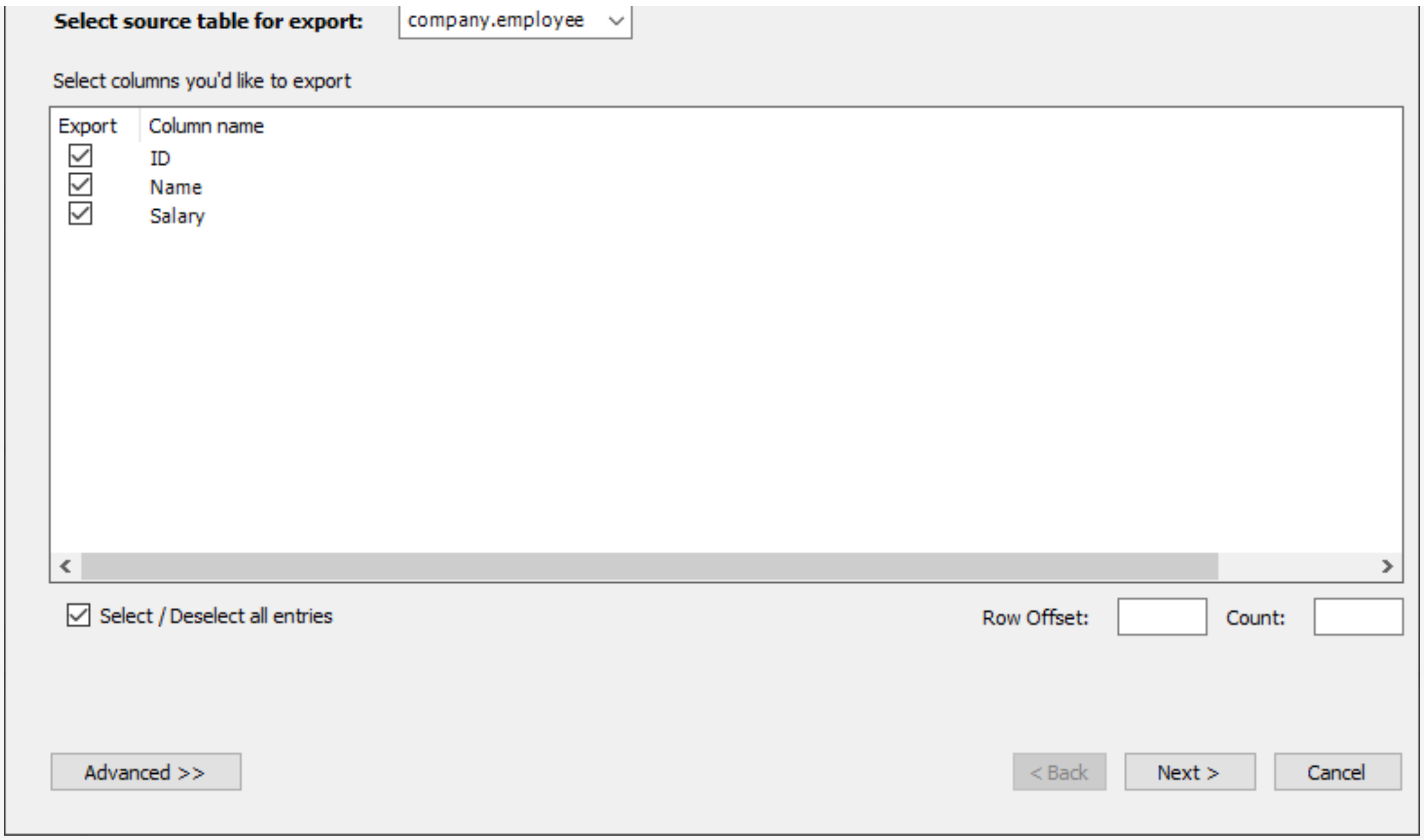
The following steps export a CSV or JSON file. The screenshots below are from Windows. Screens and buttons vary slightly on a Mac.

1. Click the **Schemas** tab in the Navigator sidebar, and expand the desired database's table list.
2. Right-click the table name that contains the data to export.
3. Choose **Table Data Export Wizard** from the context menu.



4. All the table's columns are selected by default. Click **Next**.




Select source table for export: company.employee

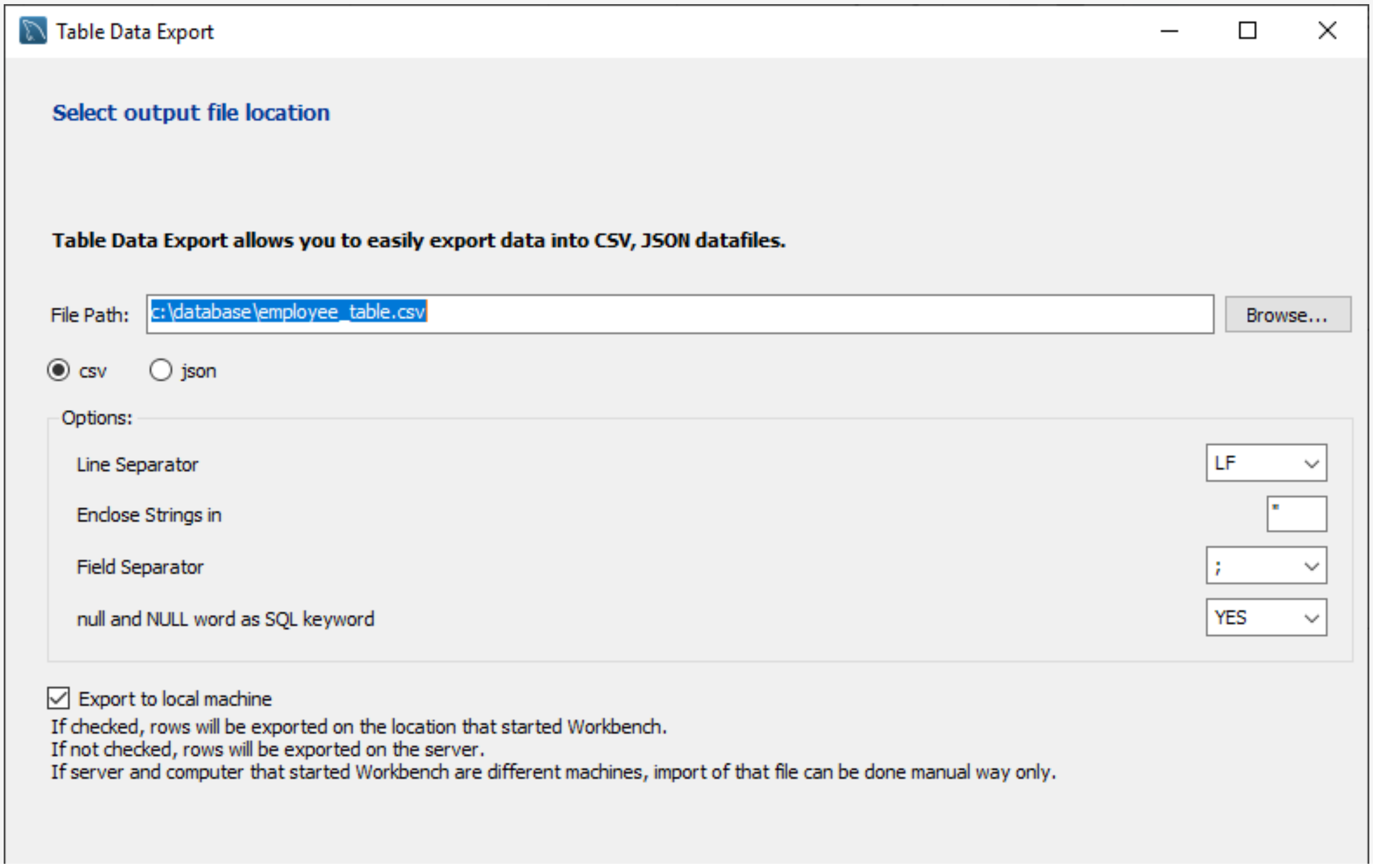
Select columns you'd like to export

Export	Column name
✓	ID
✓	Name
✓	Salary

☑ Select / Deselect all entries

Row Offset: Count:

5. Enter a file path to receive the exported data, choose **csv** or **json**, and click **Next**.



Select output file location

Table Data Export allows you to easily export data into CSV, JSON datafiles.

File Path: c:\database\employee_table.csv

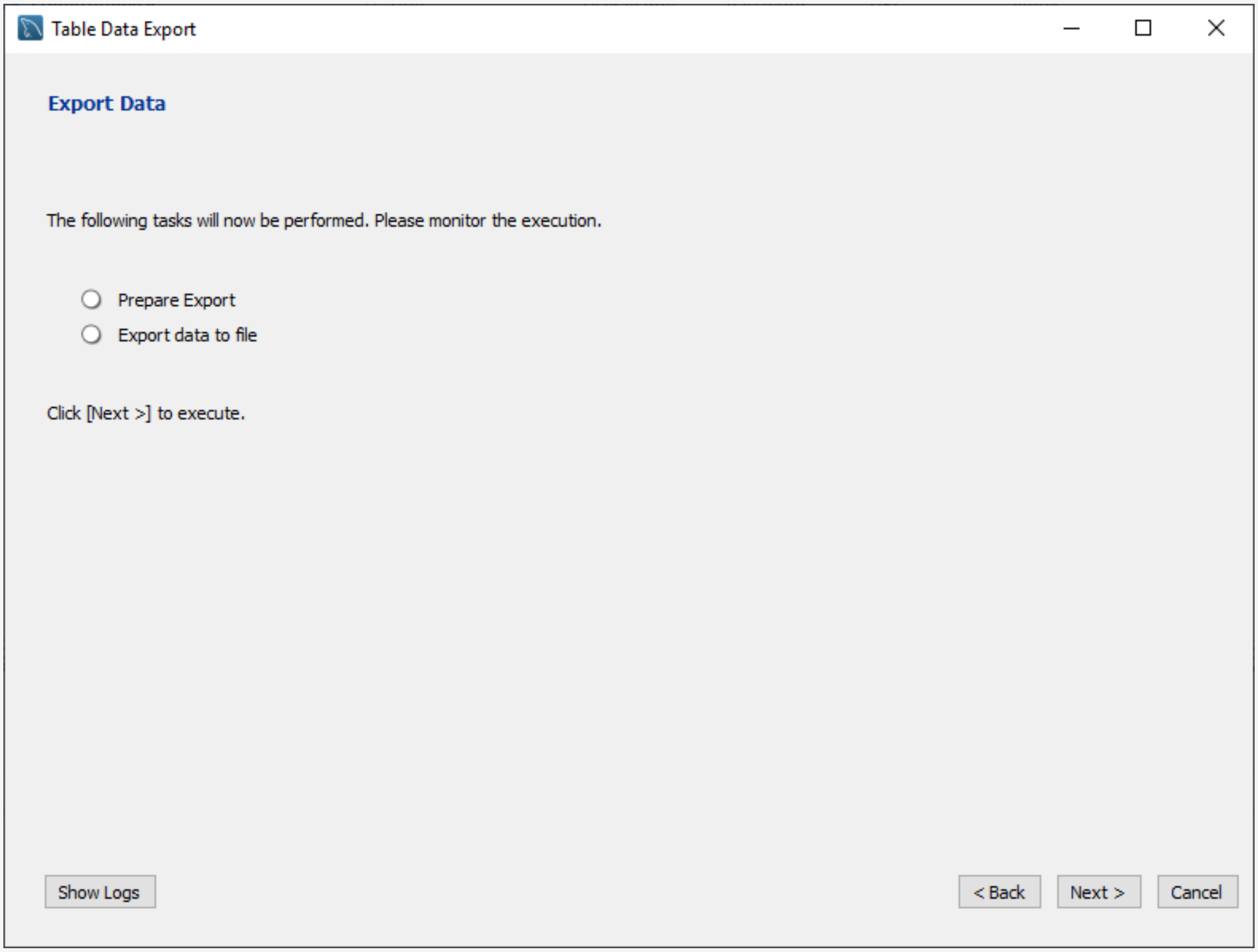
● csv ○ json

Options:

Line Separator	LF
Enclose Strings in	"
Field Separator	;
null and NULL word as SQL keyword	YES

☑ Export to local machine
If checked, rows will be exported on the location that started Workbench.
If not checked, rows will be exported on the server.
If server and computer that started Workbench are different machines, import of that file can be done manual way only.

6. Click **Next** to begin the export.



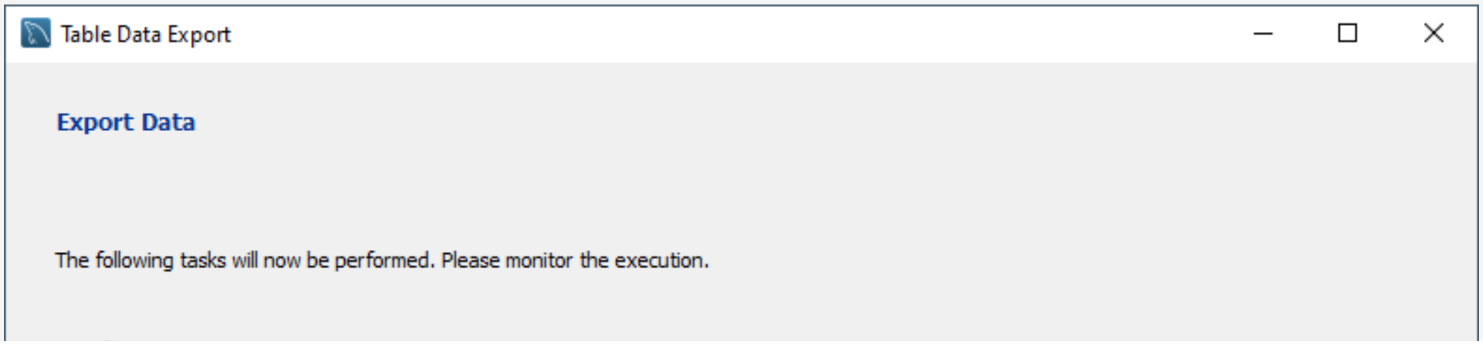
Export Data

The following tasks will now be performed. Please monitor the execution.

- Prepare Export
- Export data to file

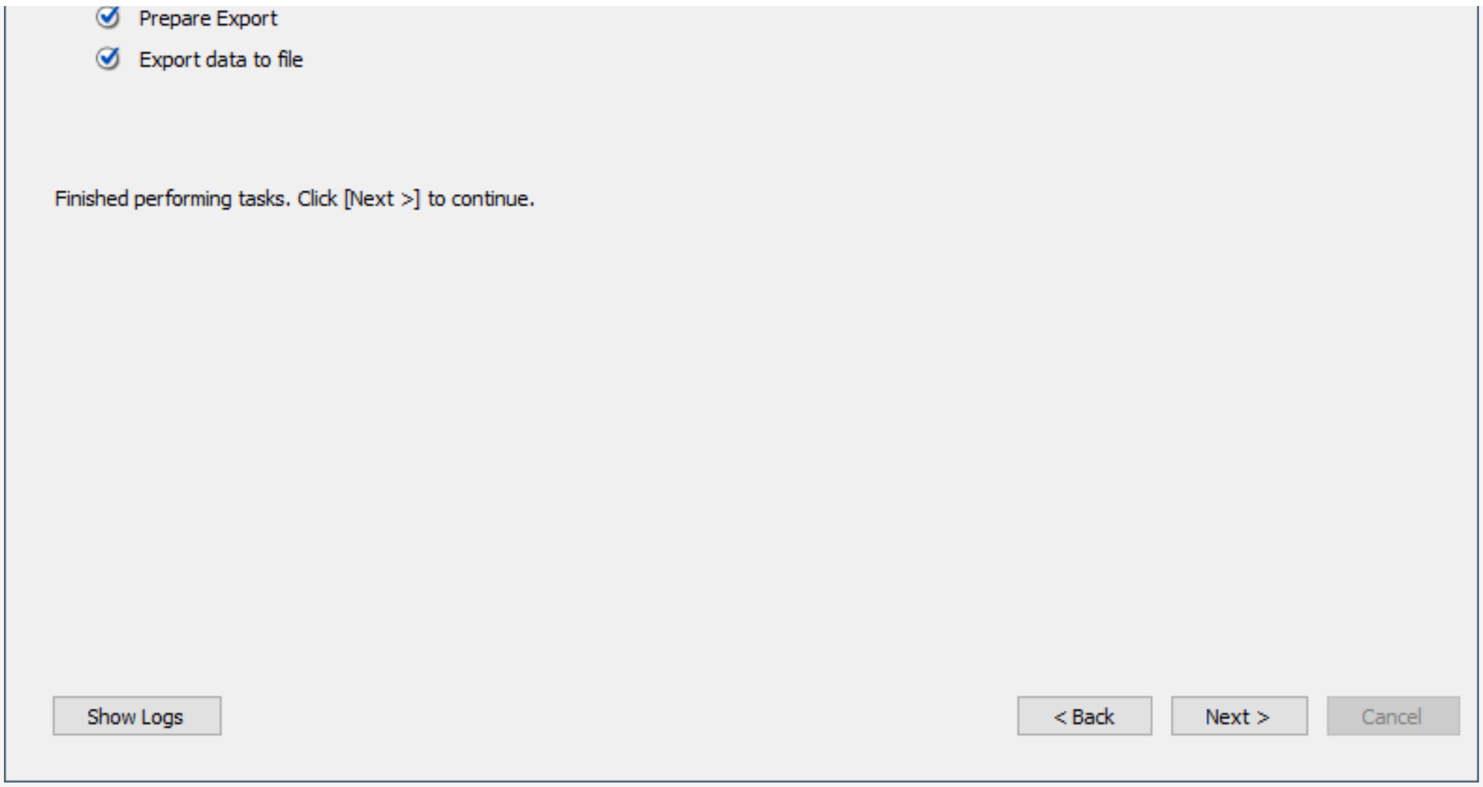
Click [Next >] to execute.

7. Click **Next** again to confirm the export has finished.



Export Data

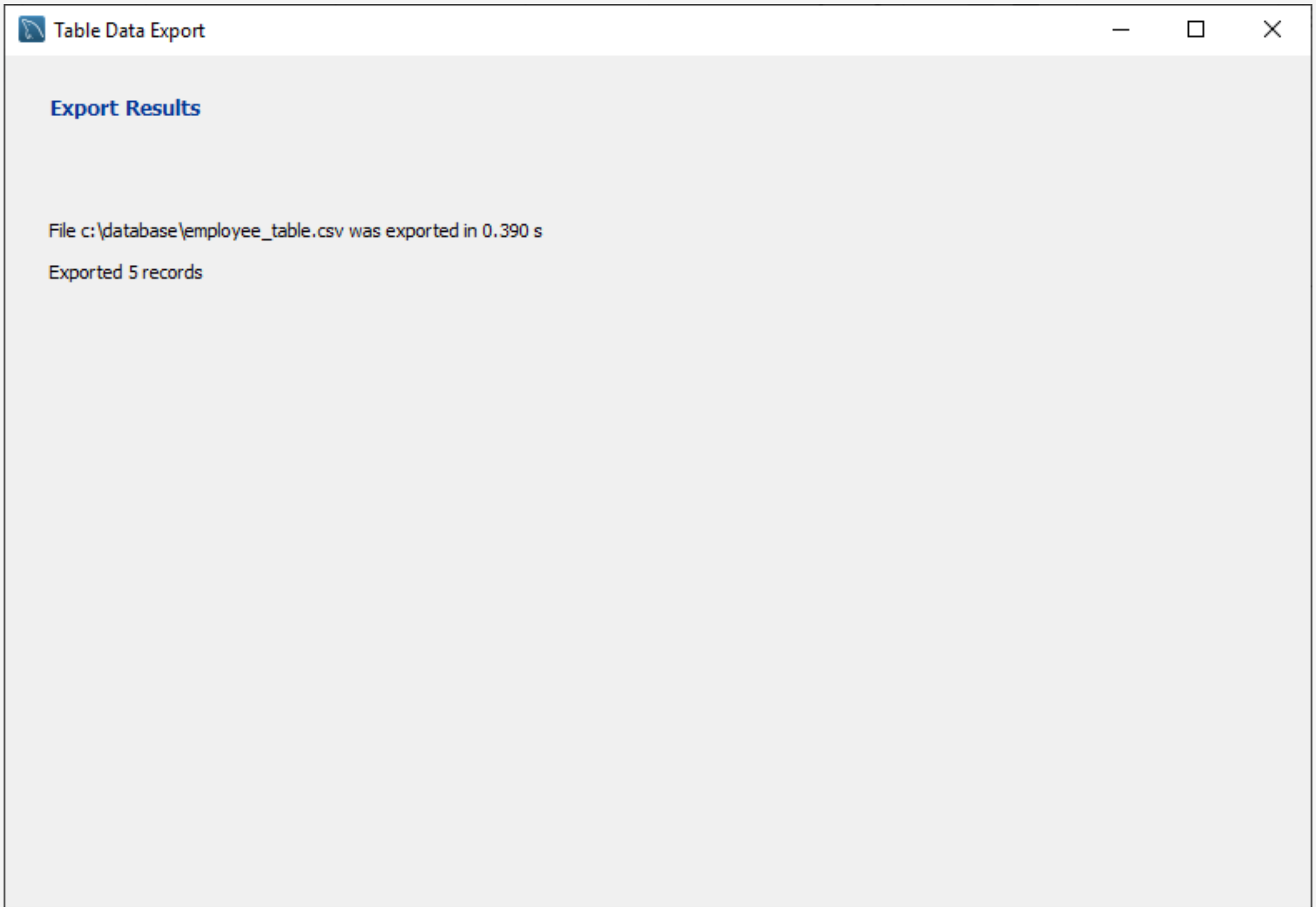
The following tasks will now be performed. Please monitor the execution.


Prepare Export

Export data to file

Finished performing tasks. Click [Next >] to continue.

8. View the export summary and click **Finish**.



Export Results

File c:\database\employee_table.csv was exported in 0.390 s

Exported 5 records

Verify the export worked by locating the export file and viewing the exported data in the file.

Exploring further:

- MySQL Data Import and Export from MySQL.com

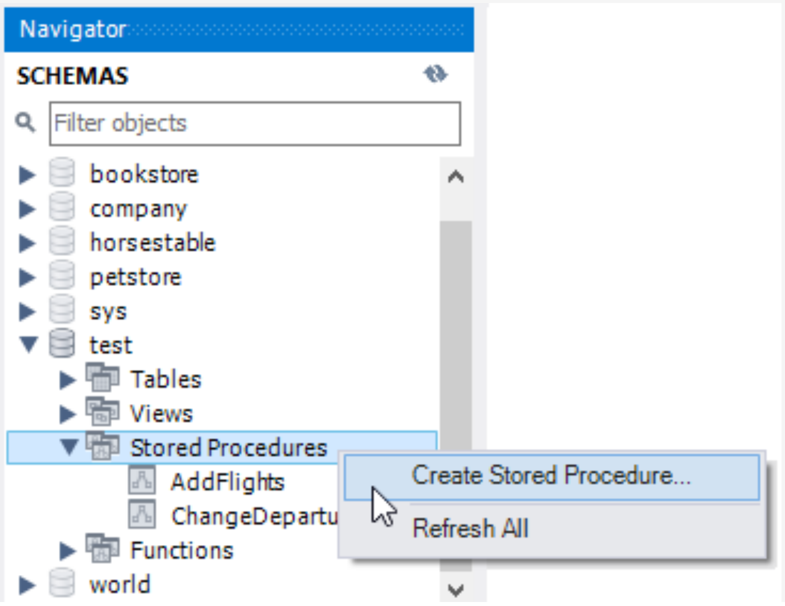
21.4 MySQL Workbench: Stored procedures and functions

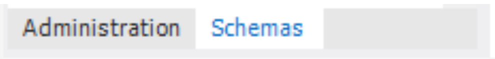
Creating a stored procedure

MySQL Workbench provides a code editor for creating stored procedures.

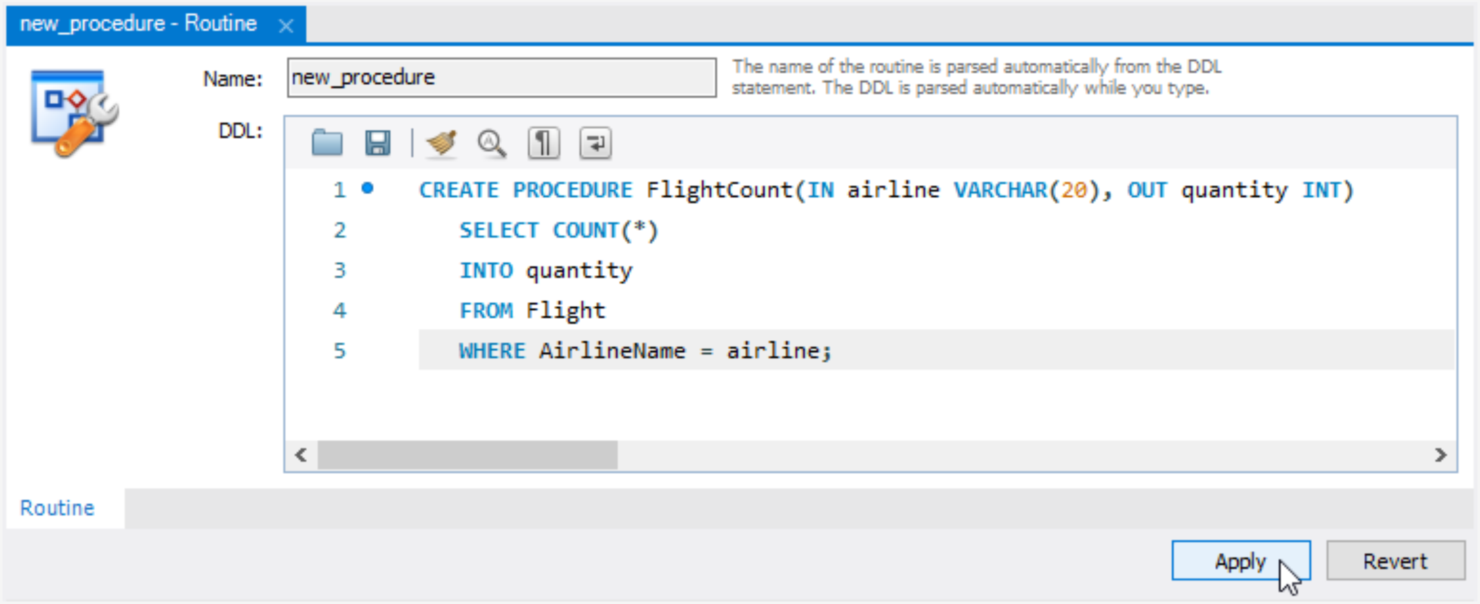
The following steps create a stored procedure . The screenshots below are from Windows. Screens and buttons vary slightly on a Mac.

1. Start MySQL Workbench and connect to MySQL server.
2. Click the **Schemas** tab in the Navigator sidebar.
3. Expand the desired database that will hold the stored procedure.
4. Right-click the database's **Stored Procedures** node.
5. Choose **Create Stored Procedure...** from the menu.





6. Enter the CREATE PROCEDURE code into the code window and click **Apply**.



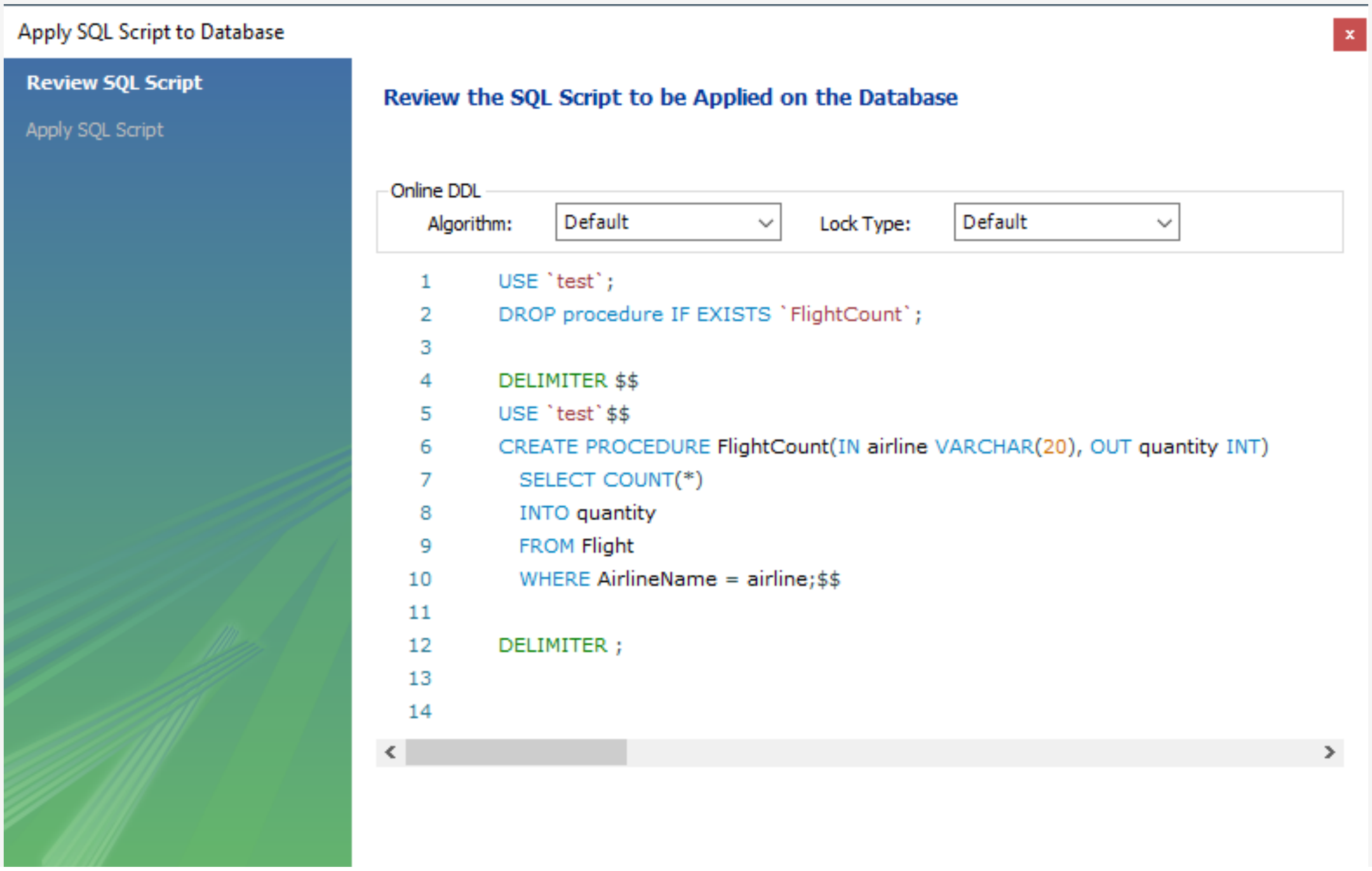
Name: new_procedure

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE PROCEDURE FlightCount(IN airline VARCHAR(20), OUT quantity INT)
   SELECT COUNT(*)
   INTO quantity
   FROM Flight
   WHERE AirlineName = airline;
```

7. The CREATE PROCEDURE code appears with some alteration in a dialog box. Click **Apply** to start the creation process.

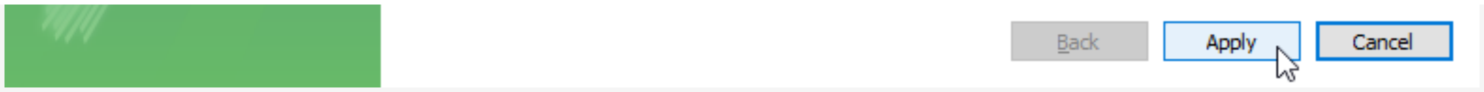


Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default   Lock Type: Default

```
USE `test`;
DROP procedure IF EXISTS `FlightCount`;

DELIMITER $$
USE `test`$$
CREATE PROCEDURE FlightCount(IN airline VARCHAR(20), OUT quantity INT)
  SELECT COUNT(*)
  INTO quantity
  FROM Flight
  WHERE AirlineName = airline;$$

DELIMITER ;


```



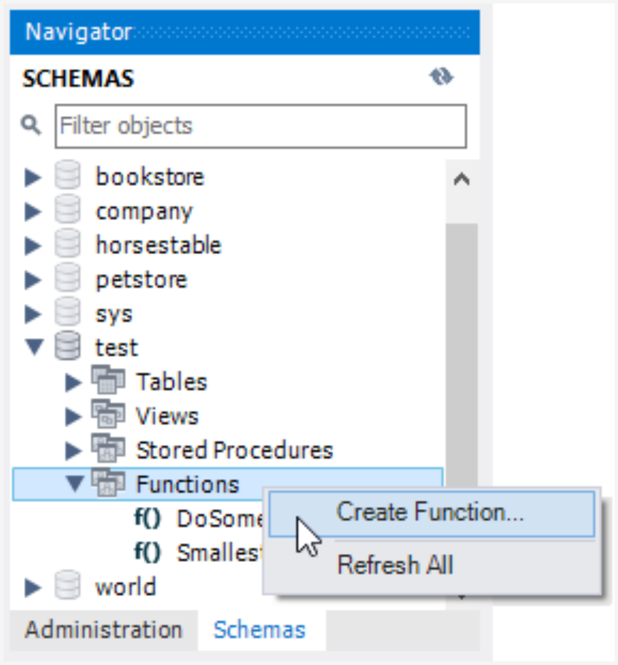
8. If the procedure is successfully created, click **Finish** in the dialog box.
9. If an error occurs, review the message log in the dialog box, click **Cancel**, and attempt to correct the problem.

Creating a function

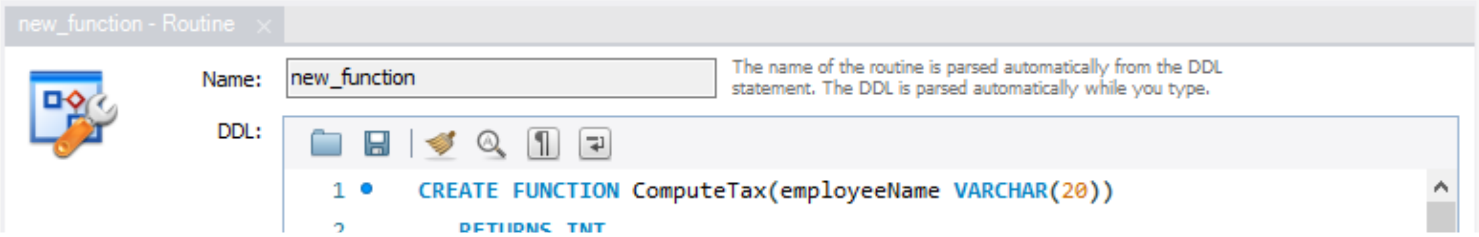
MySQL Workbench also provides a code editor for creating a function. The steps are similar to the steps for creating a stored procedure.

The screenshots below are from Windows. Screens and buttons vary slightly on a Mac.

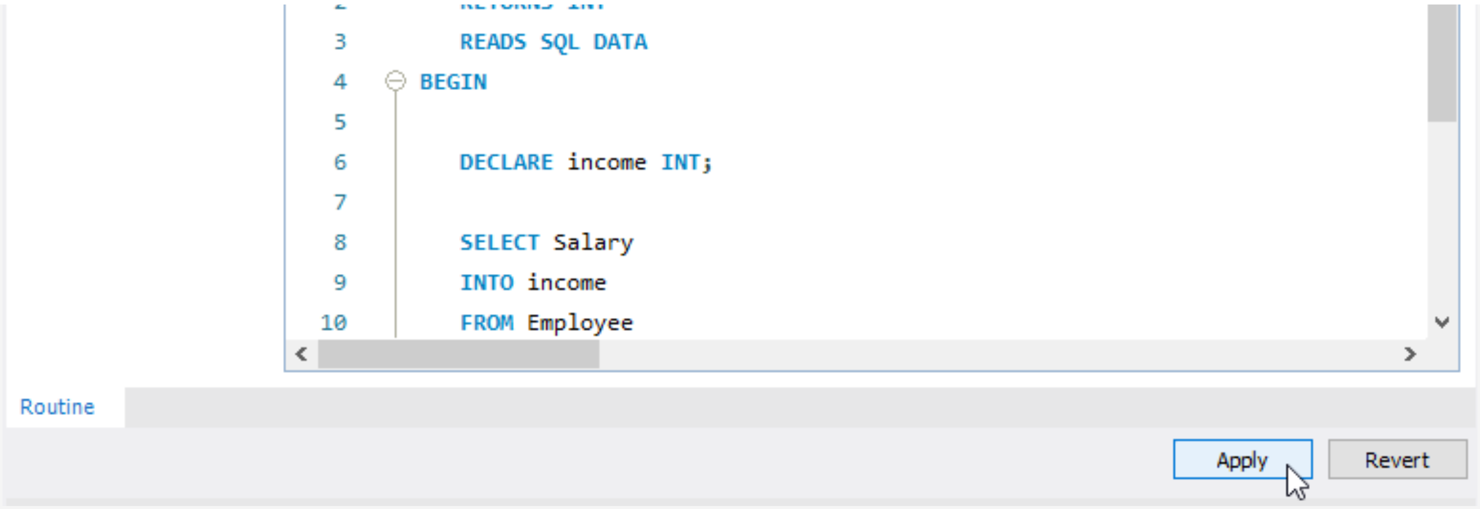
1. Start MySQL Workbench and connect to MySQL server.
2. Click the **Schemas** tab in the Navigator sidebar.
3. Expand the desired database that will hold the function.
4. Right-click the database's **Functions** node.
5. Choose **Create Function...** from the menu.



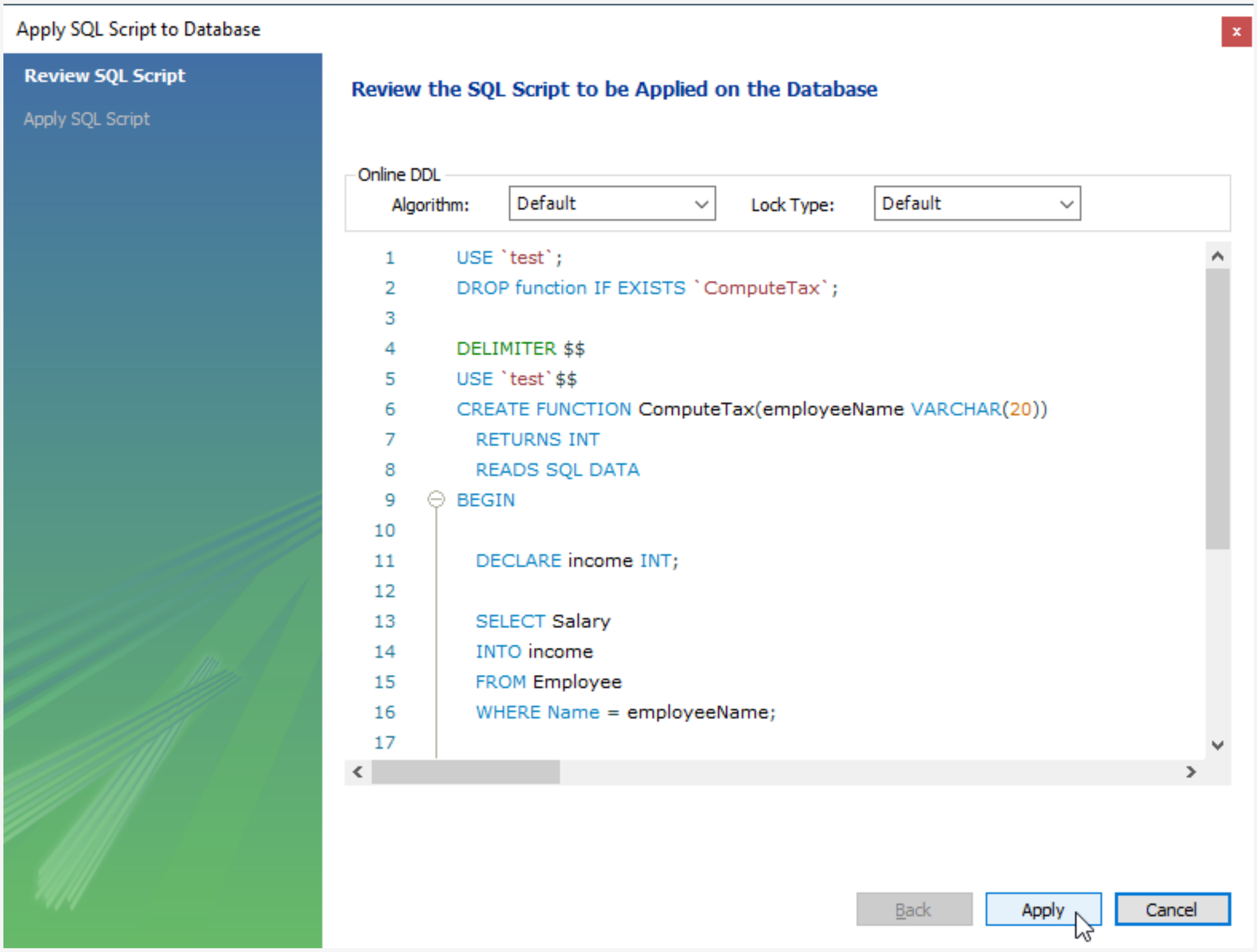
6. Enter the CREATE FUNCTION code into the code window and click **Apply**.


new_function - Routine
Name: new_function
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:

```
CREATE FUNCTION ComputeTax(employeeName VARCHAR(20))
    RETURNS INT
```


```
    READS SQL DATA
BEGIN

    DECLARE income INT;

    SELECT Salary
    INTO income
    FROM Employee
```

7. The CREATE FUNCTION code appears with some alteration in a dialog box. Click **Apply** to start the creation process.


Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Back
Apply
Cancel

Online DDL
Algorithm: Default
Lock Type: Default

```
USE `test`;
DROP function IF EXISTS `ComputeTax`;

DELIMITER $$
USE `test`$$
CREATE FUNCTION ComputeTax(employeeName VARCHAR(20))
  RETURNS INT
  READS SQL DATA
BEGIN

  DECLARE income INT;

  SELECT Salary
  INTO income
  FROM Employee
  WHERE Name = employeeName;

```

8. If the procedure is successfully created, click **Finish** in the dialog box.

9. If an error occurs, review the message log in the dialog box, click **Cancel**, and attempt to correct the problem.

21.5 LAB - MySQL Workbench review (Sakila)

Introduction

The purpose of this lab is to gain familiarity with MySQL Workbench. The lab also ensures the correct version of the Sakila sample database is installed on your computer, for use in other zyLabs.

This lab has three parts:

- Install the Sakila database.
- Run a simple query.
- Recreate a Sakila table in the zyLab environment.

Only the third part is graded.

Install the Sakila database

This lab requires access to MySQL Server via MySQL Workbench. Most students install and access MySQL Server and MySQL Workbench on their personal computer. Installation instructions are available at MySQL Server installation and MySQL Workbench installation.

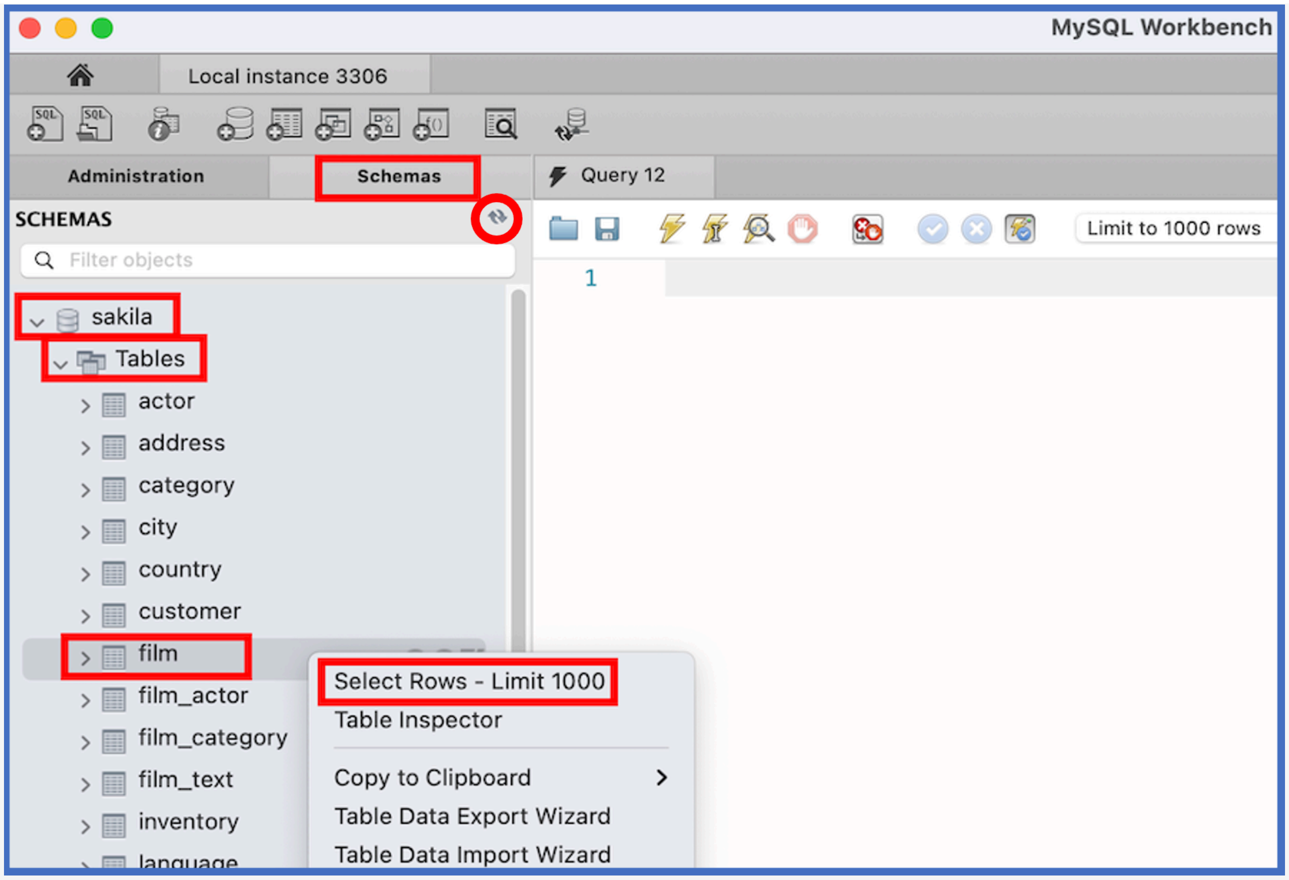
To create Sakila tables in MySQL, download the Sakila schema file, open MySQL Workbench, and click the following menu commands:

1. Click 'File' > 'Open SQL Script...' and open the Sakila schema file.
2. Click 'Query' > "Execute (All or Selection)'.

To load sample data to the Sakila tables, download the Sakila data file and repeat steps 1 and 2 with this file.

Run a simple query

Refer to the following MySQL Workbench screenshot, taken from a Mac computer. Workbench looks slightly different on Windows.



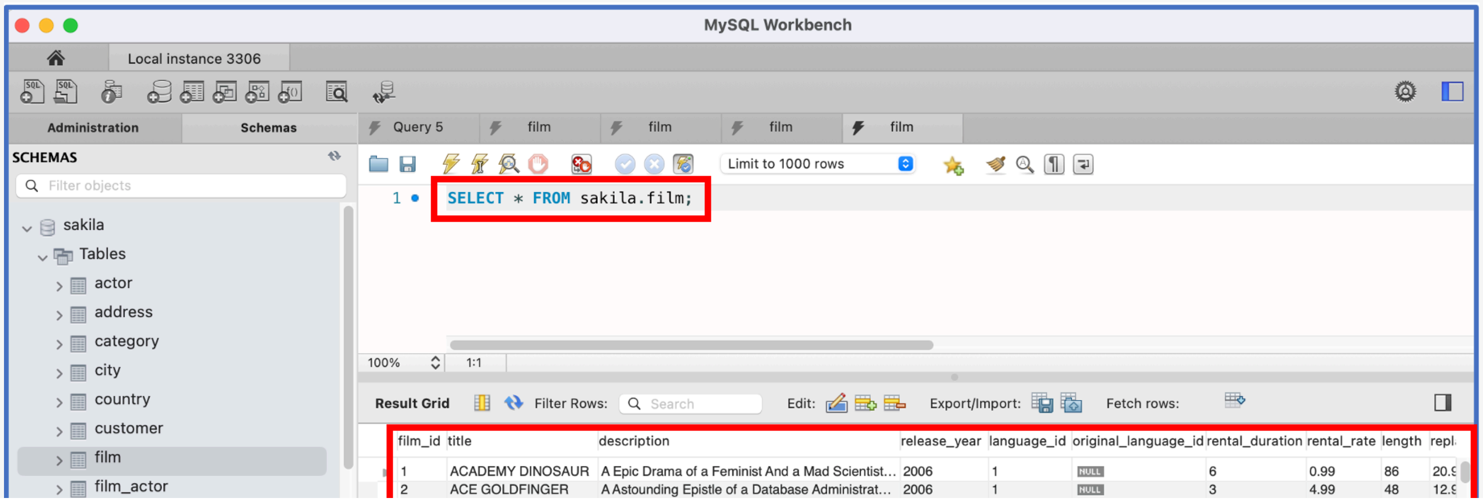
If 'sakila' does not appear under 'Schemas', click the refresh icon, in the red circle above. If 'sakila' still does not appear, repeat the installation process or request assistance.

Depending on Workbench configuration, a different Limit may appear after 'Select Rows'.

When 'sakila' appears under 'Schemas':

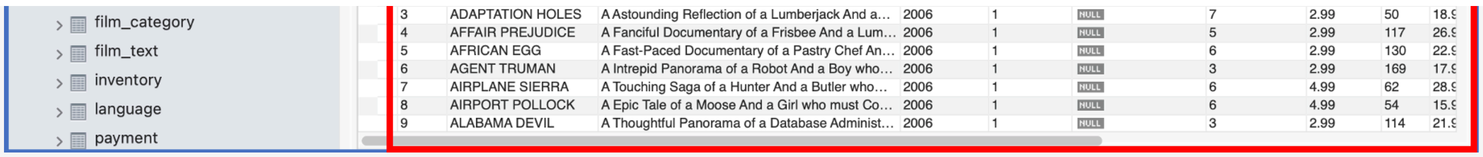
1. Click > to expand 'sakila'.
2. Click > to expand 'Tables'.
3. Right-click 'film'.
4. Click 'Select Rows - Limit 1000'.

MySQL Workbench executes `SELECT * FROM film;` and displays 1000 films:



```
SELECT * FROM sakila.film;
```

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	repl
1	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist…	2006	1	NULL	6	0.99	86	20.9
2	ACE GOLDFINGER	A Astounding Epistle of a Database Administrat…	2006	1	NULL	3	4.99	48	12.9



3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a...	2006	1	NULL	7	2.99	50	18.9
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lum...	2006	1	NULL	5	2.99	117	26.9
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef An...	2006	1	NULL	6	2.99	130	22.9
6	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who...	2006	1	NULL	3	2.99	169	17.9
7	AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who...	2006	1	NULL	6	4.99	62	28.9
8	AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl who must Co...	2006	1	NULL	6	4.99	54	15.9
9	ALABAMA DEVIL	A Thoughtful Panorama of a Database Administ...	2006	1	NULL	3	2.99	114	21.9

Recreate a Sakila table in the zyLab environment

To recreate the `actor` table in the zyLab environment:

1. Right-click 'actor'.
2. Select 'Copy to Clipboard' > 'Create Statement' to copy the `CREATE TABLE` statement to your clipboard.
3. Paste the `CREATE TABLE` statement into the zyLab Main.sql box.

The `CREATE TABLE` statement creates `actor` columns, keys, and indexes. No result is displayed in Develop mode. The tests in Submit mode verify that the zyLab and Sakila `actor` tables are identical.

544874.3500394.qx3zqy7

LAB ACTIVITY | 21.5.1: LAB - MySQL Workbench review (Sakila) 10 / 10

Main.sql **Load default template...**

```sql
-- Your CREATE TABLE statement goes here
CREATE TABLE actor (
actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
first_name VARCHAR(45) NOT NULL,
last_name VARCHAR(45) NOT NULL,
last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_T
PRIMARY KEY (actor_id),
KEY idx_actor_last_name (last_name)
);
```

Develop mode | Submit mode

Explore the database and run your program as often as

Develop mode Submit mode

you'd like, before submitting for grading. Click **Run program** and observe the program's output in the second box.

Run program

Main.sql
(Your program)

$\longrightarrow$ Output (shown below)

Program output displayed here

Coding trail of your work What is this?

```
4/24 W0,10 min:4
```