

# Final Project: Advanced SQL Techniques



Estimated time needed: **60** minutes

## Scenario

You have to analyse the following datasets for the city of Chicago, as available on the Chicago City data portal.

- Socioeconomic indicators in Chicago
- Chicago public schools
- Chicago crime data

Based on the information available in the different tables, you have to run specific queries using Advanced SQL techniques that generate the required result sets.

The lab will be followed by a graded quiz that will have questions on all problems in this lab. Hence, remember to take screenshots of your SQL queries and their outputs for reference.

## Objectives

After completing this lab, you will be able to:

1. Use joins to query data from multiple tables
2. Create and query views
3. Write and run stored procedures
4. Use transactions

## Software Used in this Lab

In this lab, you will use [MySQL](#). MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.



To complete this lab you will utilize MySQL relational database service available as part of IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

## Database Used in this Lab

**Mysql\_learners** database has been used in this lab.

Here you will be creating and inserting data into the below mentioned 3 tables

1. `chicago_public_schools`
2. `chicago_socioeconomic_data`
3. `chicago_crime`

Here you will be using 3 dump files for this purpose.

[chicago\\_public\\_schools](#)

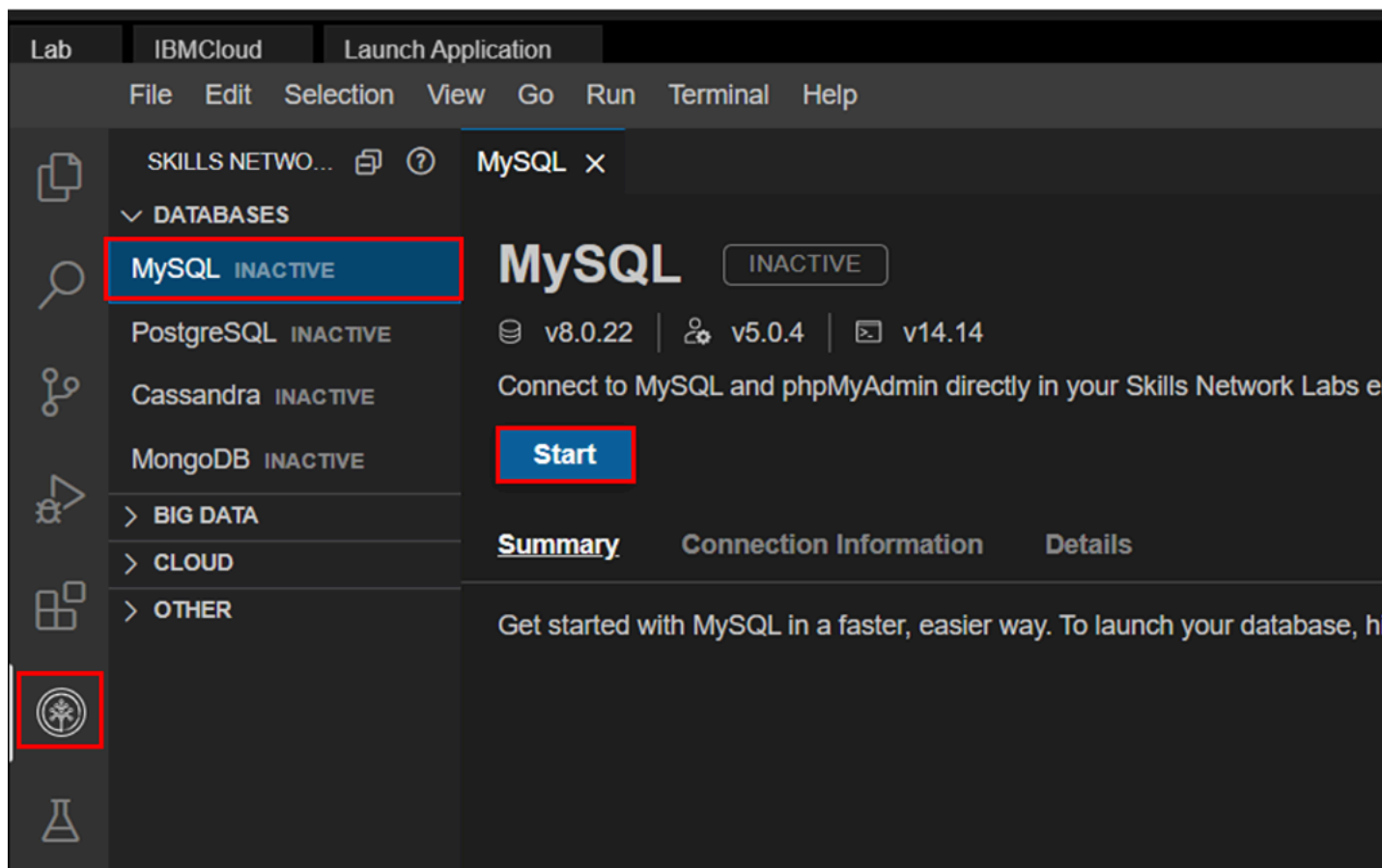
[chicago\\_crime](#)

[chicago\\_socioeconomic\\_data](#)

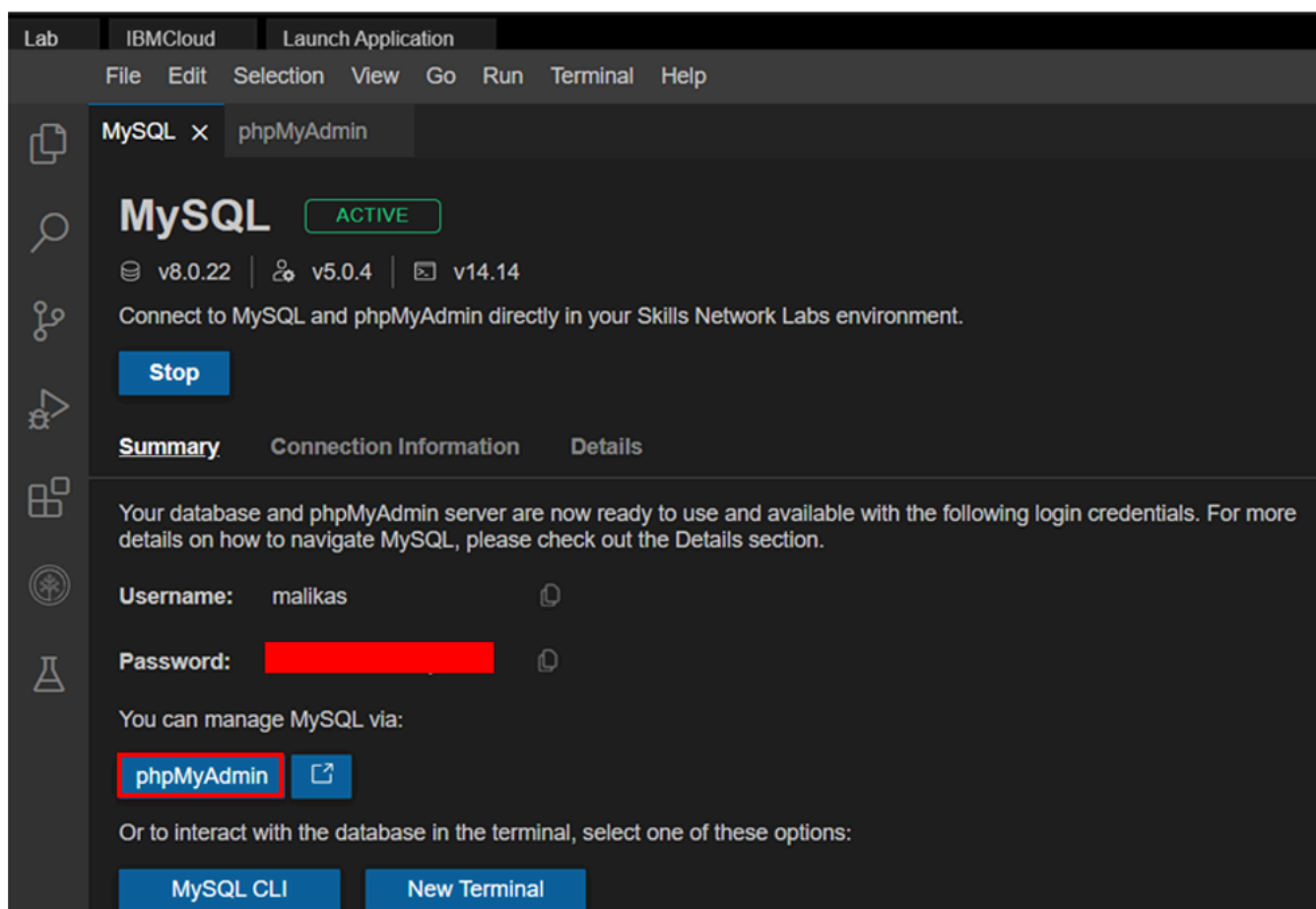
## Create the database

1. Click on **Skills Network Toolbox**. In **Database** section, click **MySQL**.

To start the MySQL click **Start**.



2. Once **MySQL** has started, click on **phpMyAdmin** button to open **phpMyAdmin** in the same window.



3. You will see the phpMyAdmin GUI tool.

The screenshot displays the phpMyAdmin web interface. The browser's address bar shows the URL `sandipsahajo-8080.theiadocker-27.proxy.cognitive`. The interface is divided into a left sidebar and a main content area.

**Left Sidebar:**

- At the top is the **phpMyAdmin** logo and a row of icons for home, server status, help, document, settings, and export.
- Below the icons are two tabs: **Recent** and **Favorites**.
- A tree view shows the database structure:
  - New** (with a green plus icon)
  - information\_schema** (with a plus icon)
  - mysql** (with a plus icon)
  - performance\_schema** (with a plus icon)
  - sakila** (with a plus icon)
  - sys** (with a plus icon)

**Main Content Area:**

- The top navigation bar shows the selected server: **Server: mysql:3306**.
- Below the navigation bar are three tabs: **Databases** (selected), **SQL**, and **Status**.
- The **General settings** section is visible, showing:
  - Server connection collation:** `utf8r` (with a help icon).
  - A link for **More settings** (with a key icon).
- The **Appearance settings** section is also visible, showing:
  - Language:** `English` (with a help icon).
  - Theme:** `pmahomme` (with a dropdown arrow).

- 4. In the tree-view, click **New** to create a new empty database. Then enter **Mysql\_Learners** as the name of the database and click **Create**.

The encoding will be left as **utf8mb4\_0900\_ai\_ci**. UTF-8 is the most commonly used character encoding for content or data.

Proceed to Task B.

Databases

SQL

Status

User accounts

Export

Import

Settings

Binary log

Re

## Databases

Create database

Mysql\_learners

utf8mb4\_0900\_ai\_ci

Create

	Database	Collation	Master replication	Action
<input type="checkbox"/>	information_schema	utf8_general_ci	✓ Replicated	<a href="#">Check privileges</a>
<input type="checkbox"/>	mysql	utf8mb4_0900_ai_ci	✓ Replicated	<a href="#">Check privileges</a>
<input type="checkbox"/>	performance_schema	utf8mb4_0900_ai_ci	✓ Replicated	<a href="#">Check privileges</a>
<input type="checkbox"/>	sys	utf8mb4_0900_ai_ci	✓ Replicated	<a href="#">Check privileges</a>

Total: 4

⬆

☐ Check all

With selected:

[Drop](#)

⚠

Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.

•

[Enable statistics](#)

Load the dump files one by one into the database **Mysql\_learners** by clicking the **Import** tab and choose the file. Click on **Go** button.

Server: mysql:3306 » Database: Mysql\_learners

StructureSQLSearchQueryExportImportOperationsPrivilegesRoutines

## Importing into the database "Mysql\_learners"

### File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.  
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer:  chicago\_pu...\_schools.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

### Partial import:

☒ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large files)*

Skip this number of queries (for SQL) starting from the first one:

### Other options:

☒ Enable foreign key checks

### Format:

The screenshot displays the phpMyAdmin web interface. On the left, the 'Database: Mysql\_learners' is selected, showing a tree view with 'New', 'chicago\_public\_schools', 'performance\_schema', and 'sys'. The main panel shows the 'Structure' tab for the 'Mysql\_learners' database. It displays a list of tables: 'information\_schema', 'mysql', 'Mysql\_learners', 'New', 'chicago\_public\_schools', 'performance\_schema', and 'sys'. The 'chicago\_public\_schools' table is highlighted. The right panel shows the 'SQL' tab with a list of queries executed, all returning an empty result set (zero rows). The queries are:

- Import has been successfully finished, 22 queries executed. (chicago\_public\_schools.sql)
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0008 seconds.)
- phpMyAdmin SQL Dump -- version 5.0.4 -- https://www.phpmyadmin.net/ -- -- Host: Version: 7.4.15 SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO"
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
- START TRANSACTION
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
- SET time\_zone = "+00:00"
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
- /\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)
- /\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/

The tables are created and the data is loaded successfully. Repeat the same operation with the other 2 dump files to create and load the tables.

You will see a screen as below

Recent Favorites

There are no favorite tables.

New

information\_schema

mysql

Mysql\_learners

New

chicago\_crime

chicago\_public\_schools

chicago\_socioeconomic\_data

performance\_schema

sys

```

`PERCENT_OF_HOUSING_CROWDED`, `PERCENT_HOUSEHOLDS_BELOW_POVERTY`, `PERCENT_AGED_16_
`PERCENT_AGED_UNDER_18_OR_OVER_64`, `PER_CAPITA_INCOME`, `HARDSHIP_INDEX`) VALUES (
Ridge', '7.8', '17.2', '8.8', '20.8', '38.5', 23040, '46'), ('3', 'Uptown', '3.8',
'8.2', '13.4', '25.5', 37524, '17'), ('5', 'North Center', '0.3', '7.5', '5.2', '4.
60058, '5'), ('7', 'Lincoln Park', '0.8', '12.3', '5.1', '3.6', '21.5', 71551, '2')

```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

COMMIT

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */

```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

```

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */

```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */

```

Console

## Exercise 1: Using Joins

You have been asked to produce some reports about the communities and crimes in the Chicago area. You will need to use SQL join queries to access the data stored across multiple tables.

### Question 1

- Write and execute a SQL query to list the school names, community names and average attendance for communities with a hardship index of 98.

- Hint 1
- Hint 2

Take a screenshot showing the SQL query and its results.

### Question 2

- Write and execute a SQL query to list all crimes that took place at a school. Include case number, crime type and community name.

- Hint 1
- Hint 2
- Hint 3

Take a screenshot showing the SQL query and its results.

## Exercise 2: Creating a View

For privacy reasons, you have been asked to create a view that enables users to select just the school name and the icon fields from the CHICAGO\_PUBLIC\_SCHOOLS table. By providing a view, you can ensure that users cannot see the actual scores given to a school, just the icon associated with their score. You should define new names for the view columns to obscure the use of scores and icons in the original table.

### Question 1

- Write and execute a SQL statement to create a view showing the columns listed in the following table, with new column names as shown in the second column.

Column name in CHICAGO_PUBLIC_SCHOOLS	Column name in view
NAME_OF_SCHOOL	School_Name
Safety_Icon	Safety_Rating
Family_Involvement_Icon	Family_Rating
Environment_Icon	Environment_Rating
Instruction_Icon	Instruction_Rating
Leaders_Icon	Leaders_Rating
Teachers_Icon	Teachers_Rating

- Write and execute a SQL statement that returns all of the columns from the view.
- Write and execute a SQL statement that returns just the school name and leaders rating from the view.

Take a screenshot showing the last SQL query and its results.

## Exercise 3: Creating a Stored Procedure

The icon fields are calculated based on the value in the corresponding score field. You need to make sure that when a score field is updated, the icon field is updated too. To do this, you will write a stored procedure that receives the school id and a leaders score as input parameters, calculates the icon setting and updates the fields appropriately.

### Question 1

- Write the structure of a query to create or replace a stored procedure called UPDATE\_LEADERS\_SCORE that takes a in\_School\_ID parameter as an integer and a in\_Leader\_Score parameter as an integer.

Take a screenshot showing the SQL query.

### Question 2

- Inside your stored procedure, write a SQL statement to update the Leaders\_Score field in the CHICAGO\_PUBLIC\_SCHOOLS table for the school identified by in\_School\_ID to the value in the in\_Leader\_Score parameter.

Take a screenshot showing the SQL query.

### Question 3

- Inside your stored procedure, write a SQL IF statement to update the Leaders\_Icon field in the CHICAGO\_PUBLIC\_SCHOOLS table for the school identified by in\_School\_ID using the following information.

Score lower limit	Score upper limit	Icon
80	99	Very strong
60	79	Strong
40	59	Average
20	39	Weak
0	19	Very weak

► Hint 1

► Hint 2

Take a screenshot showing the SQL query.

### Question 4

- Run your code to create the stored procedure.

Take a screenshot showing the SQL query and its results.

- Write a query to call the stored procedure, passing a valid school ID and a leader score of 50, to check that the procedure works as expected.

## Exercise 4: Using Transactions

You realise that if someone calls your code with a score outside of the allowed range (0-99), then the score will be updated with the invalid data and the icon will remain at its previous value. There are various ways to avoid this problem, one of which is using a transaction.

### Question 1

- Update your stored procedure definition. Add a generic ELSE clause to the IF statement that rolls back the current work if the score did not fit any of the preceding categories.

► Hint 1



Take a screenshot showing the SQL query.

## Question 2

- Update your stored procedure definition again. Add a statement to commit the current unit of work at the end of the procedure.

► Hint 1

Take a screenshot showing the SQL query.

- Run your code to replace the stored procedure.
- Write and run one query to check that the updated stored procedure works as expected when you use a valid score of 38.
- Write and run another query to check that the updated stored procedure works as expected when you use an invalid score of 101.

## Conclusion

You can now write advanced SQL statements to query data from multiple tables, to obscure sensitive data from users, and to control how information is updated in your tables.

### Author(s)

[Lakshmi Holla](#)

[Malika Singla](#)

### Additional Contributor

[Abhishek Gagneja](#)

© IBM Corporation 2023. All rights reserved.