

# Hands-on Lab : MySQL User Management, Access Control, and Encryption

**Estimated time needed:** 25 minutes

In this lab, first you will learn how to manage MySQL user accounts and roles using phpMyAdmin graphical user interface (GUI) tool. Then you will learn how to control access to MySQL databases and their objects. Finally you will learn how to secure your data adding extra layer of security using data encryption.

## Objectives

After completing this lab, you will be able to use the phpMyAdmin to:

- Manage MySQL user accounts and roles
- Control access to MySQL databases and their objects
- Add last line of defense to secure data using encryption

## Software Used in this Lab

In this lab, you will use [MySQL](https://dev.mysql.com/doc/world-setup/en/). MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.



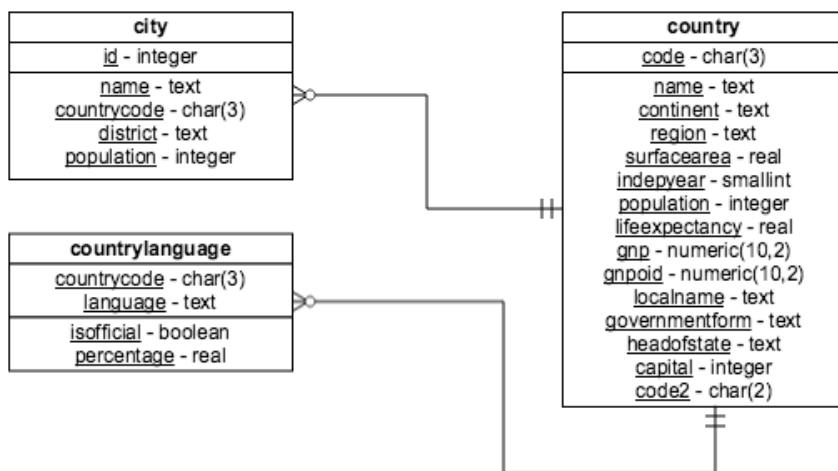
To complete this lab you will utilize the MySQL relational database service available as part of the IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

## Database Used in this Lab

The World database used in this lab comes from the following source: <https://dev.mysql.com/doc/world-setup/en/> under [CC BY 4.0 License](#) with [Copyright 2021 - Statistics Finland](#).

You will use a modified version of the database for the lab, so to follow the lab instructions successfully please use the database provided with the lab, rather than the database from the original source.

The following ERD diagram shows the schema of the World database:



The first row is the table name, the second is the primary key, and the remaining items are any additional attributes.

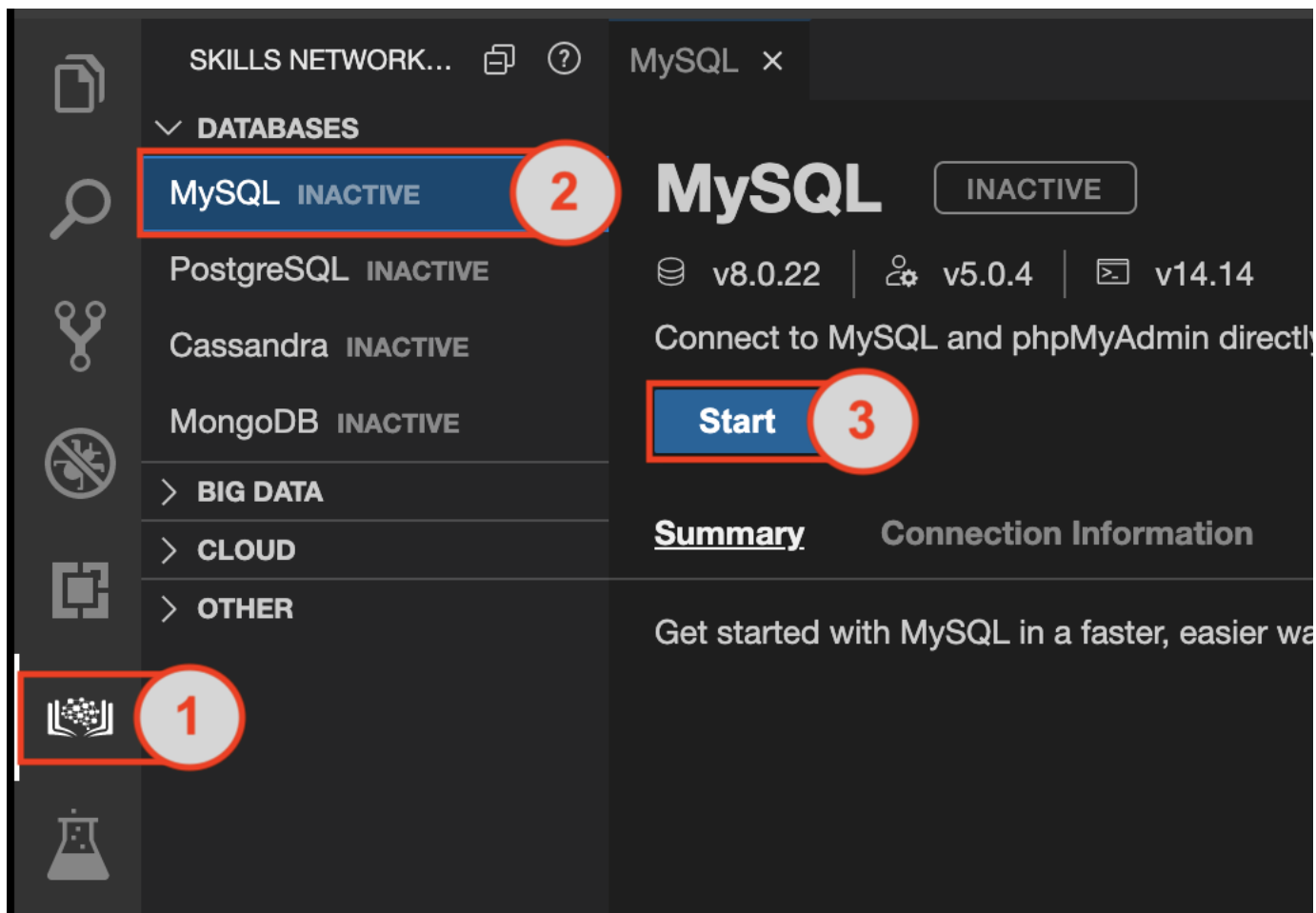
## Exercise 1: Manage MySQL user accounts and roles

In this example exercise, you will go through an example on how to manage MySQL user accounts and roles using phpMyAdmin.

User management is the process of controlling which users are allowed to connect to the MySQL server and what permissions they have on each database. phpMyAdmin does not handle user management, rather it passes the username and password on to MySQL, which then determines whether a user is permitted to perform a particular action. Within phpMyAdmin, administrators have full control over creating users, viewing and editing privileges for existing users, and removing users.

1. Go to **Skills Network Toolbox** by clicking the following icon from the side by side launched Cloud IDE.
2. From the **Databases** drop-down menu, click **MySQL** to open the MySQL service session tab.

3. Click the **Start** button and wait until MySQL service session gets launched.



The MySQL server will take a few moments to start. Once it is ready, you will see the green “Active” label near the top of the window.

SKILLS NETWORK...

MySQL x

▼ DATABASES

MySQL **ACTIVE**

PostgreSQL **INACTIVE**

Cassandra **INACTIVE**

MongoDB **INACTIVE**

> BIG DATA

> CLOUD

> OTHER

# MySQL

**ACTIVE**

v8.0.22 | v5.0.4 | v14.14

Connect to MySQL and phpMyAdmin directly

**Stop**

Summary Connection Information

Your database and phpMyAdmin server are ready for use. For more details on how to connect, see the [Getting Started](#) section.

**Username:**

**Password:**

You can manage MySQL via:

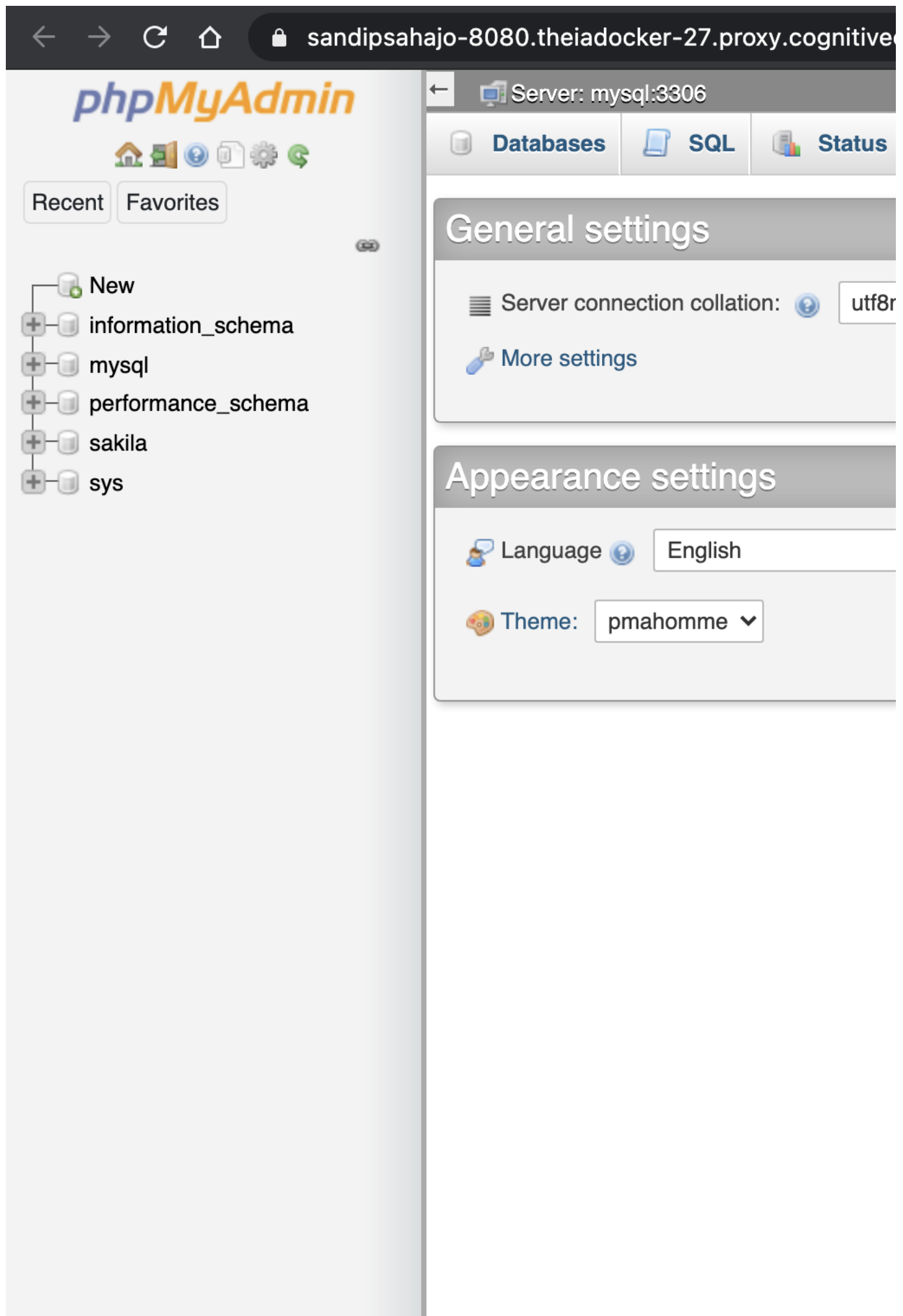
**phpMyAdmin**

Or to interact with the database in the terminal:

**MySQL CLI** **New Terminal**

**NOTE:** Whenever you are required to enter your MySQL service session password from the MySQL service session tab at any step of the lab, copy the password by clicking on the small copy button on the right of the password block. Paste the password into the terminal using **Ctrl + V** (Mac: **⌘ + V**), and press **Enter** on the keyboard. For security reasons, you will not see the password as it is entered on the terminal.

3. Click **phpMyAdmin** button from the mysql service session tab. You will see the phpMyAdmin GUI tool.



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

The screenshot shows the phpMyAdmin interface. On the left sidebar, the 'New' button is highlighted with a red box. The main area shows the 'Databases' tab. In the 'Create database' section, the database name 'world' is entered in the text field, and 'utf8mb4\_0900\_ai\_ci' is selected from the collation dropdown. The 'Create' button is highlighted with a red box. Below this, a table lists existing databases:

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

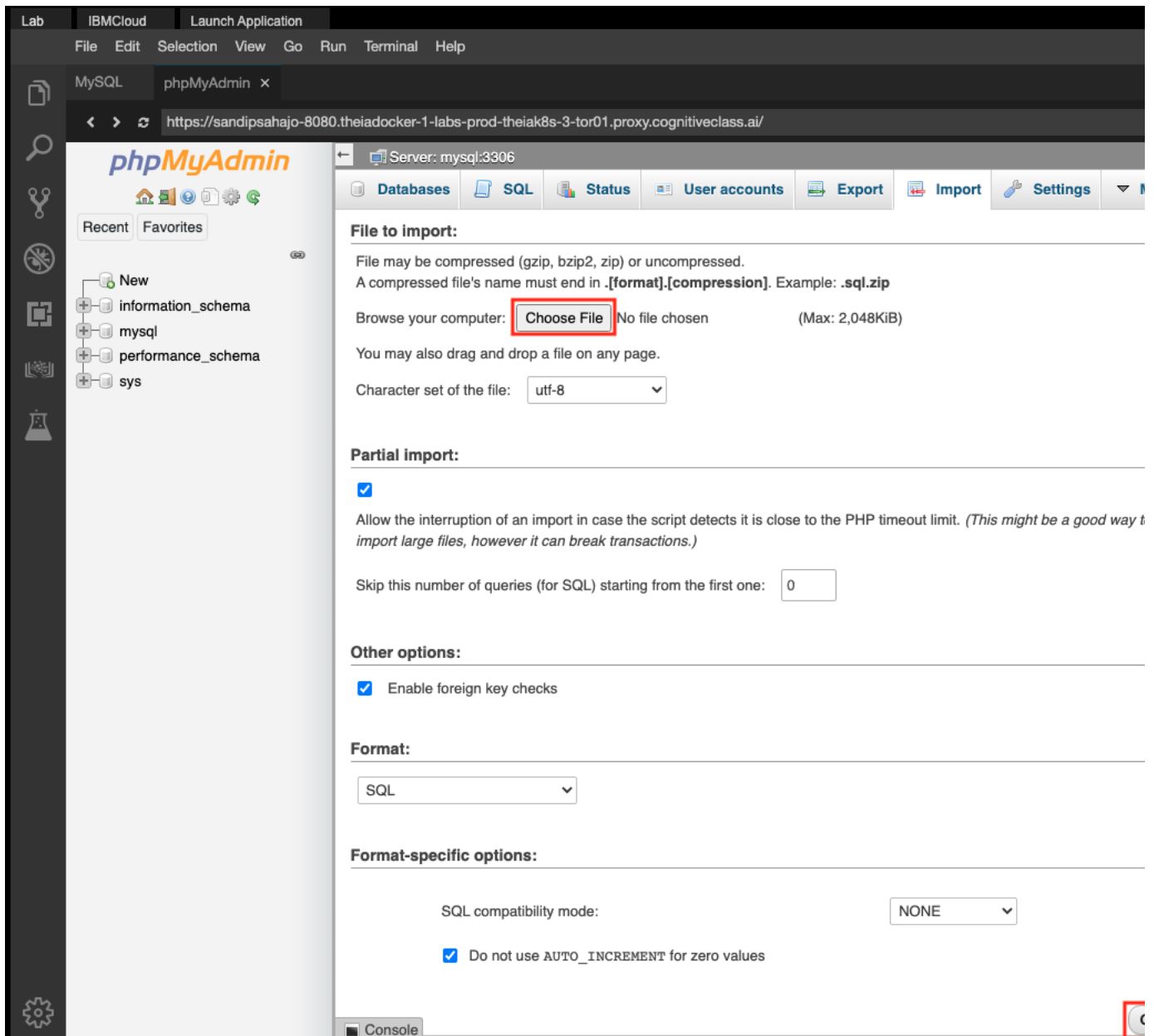
With selected: [Drop](#)

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

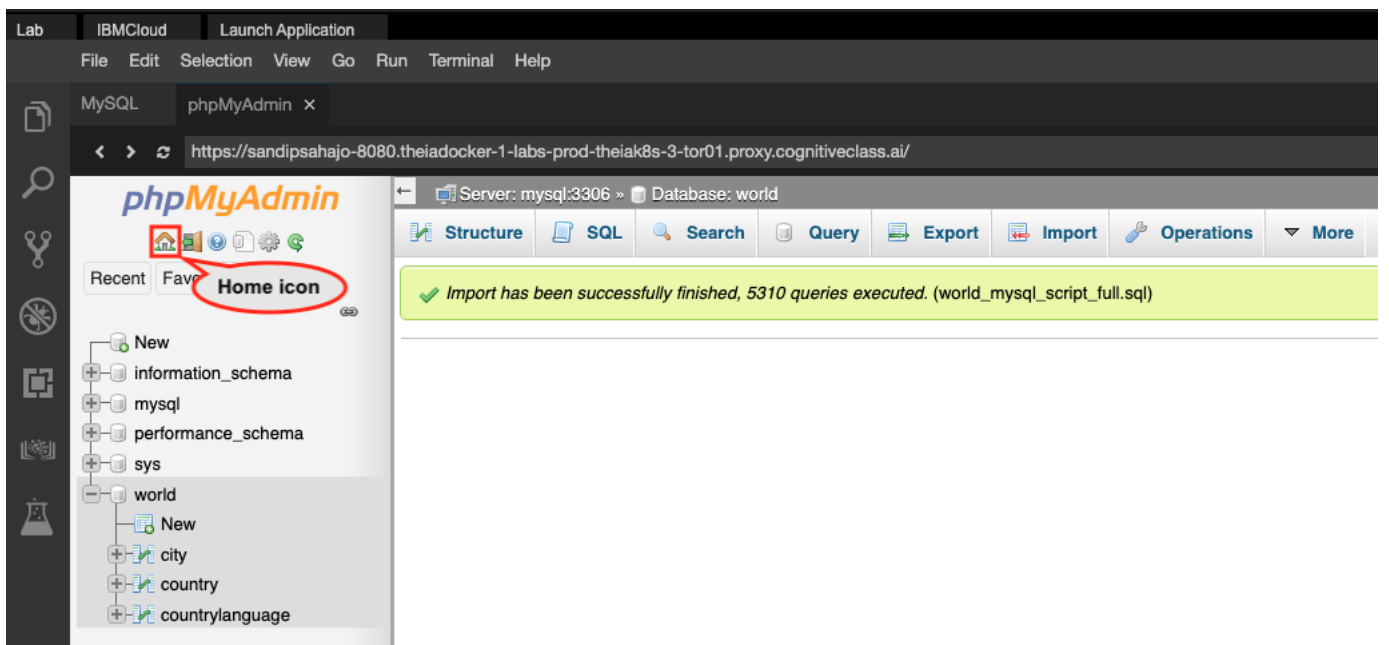
- [Enable statistics](#)

5. Go to the **Import** tab. Upload the following sql script file using the **Choose File** button (first download the following sql script file to your local computer storage). Then click **Go** button at the bottom. You will be notified when the import successfully gets finished. Click the **Home** icon.

- [world\\_mysql\\_script\\_full.sql](#)

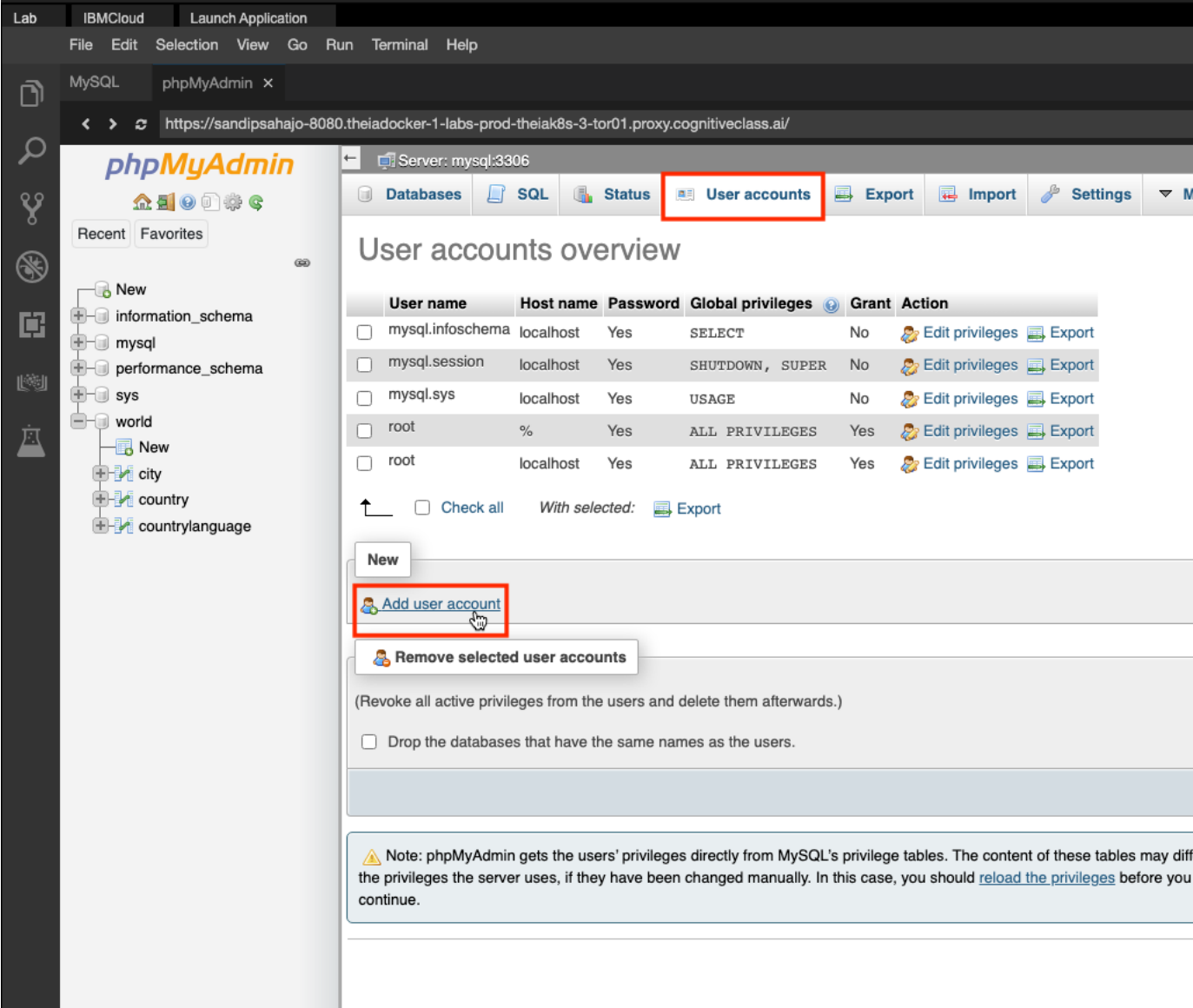


The screenshot shows the phpMyAdmin interface with the 'Import' tab selected. The left sidebar displays a tree view of databases: 'New', 'information\_schema', 'mysql', 'performance\_schema', and 'sys'. The main content area is titled 'File to import:' and includes instructions on file formats (gzip, bzip2, zip) and a 'Choose File' button highlighted with a red box. Below this, there are options for 'Partial import' (checked), 'Other options' (checked for 'Enable foreign key checks'), 'Format' (set to 'SQL'), and 'Format-specific options' (checked for 'Do not use AUTO\_INCREMENT for zero values'). The 'Character set of the file' is set to 'utf-8'. A 'Console' tab is visible at the bottom.



The screenshot shows the phpMyAdmin interface with the 'Structure' tab selected. The left sidebar displays a tree view of databases, including 'New', 'information\_schema', 'mysql', 'performance\_schema', 'sys', and 'world'. The 'world' database is expanded, showing tables: 'New', 'city', 'country', and 'countrylanguage'. The main content area shows a green success message: 'Import has been successfully finished, 5310 queries executed. (world\_mysql\_script\_full.sql)'. A red circle highlights the 'Home icon' in the top navigation bar.

6. Now you will create a user account with custom role "db\_owner". Usually a user with db\_owner role has all global privileges and access to all existing databases. Go to the **User accounts** tab and click **Add user account**.



7. Fill the **Login Information** as shown in following image (enter your own password). Under **Global privileges**, click **Check all**. Scroll down and click **Go**.

Lab

IBMCLOUD

Launch Application

File

Edit

Selection

View

Go

Run

Terminal

Help

MySQL

phpMyAdmin x

<

>

↺

https://sandipsahajo-8080.theiadocker-1-labs-prod-theiak8s-3-tor01.proxy.cognitiveclass.ai/

phpMyAdmin

Server: mysql:3306

Databases

SQL

Status

User accounts

Export

Import

Settings

M

Recent

Favorites

New

information\_schema

mysql

performance\_schema

sys

world

New

city

country

countrylanguage

Add user account

Login Information

User name: Use text field: db\_owner

Host name: Local localhost

Password: Use text field: ..... Strength: Extremely weak

Re-type: .....

Authentication Plugin Caching sha2 authentication

Generate password: Generate

Database for user account

Create database with same name and grant all privileges.

Grant all privileges on wildcard name (username\\_%).

Global privileges

Check all

Data

Structure

Administration

SELECT

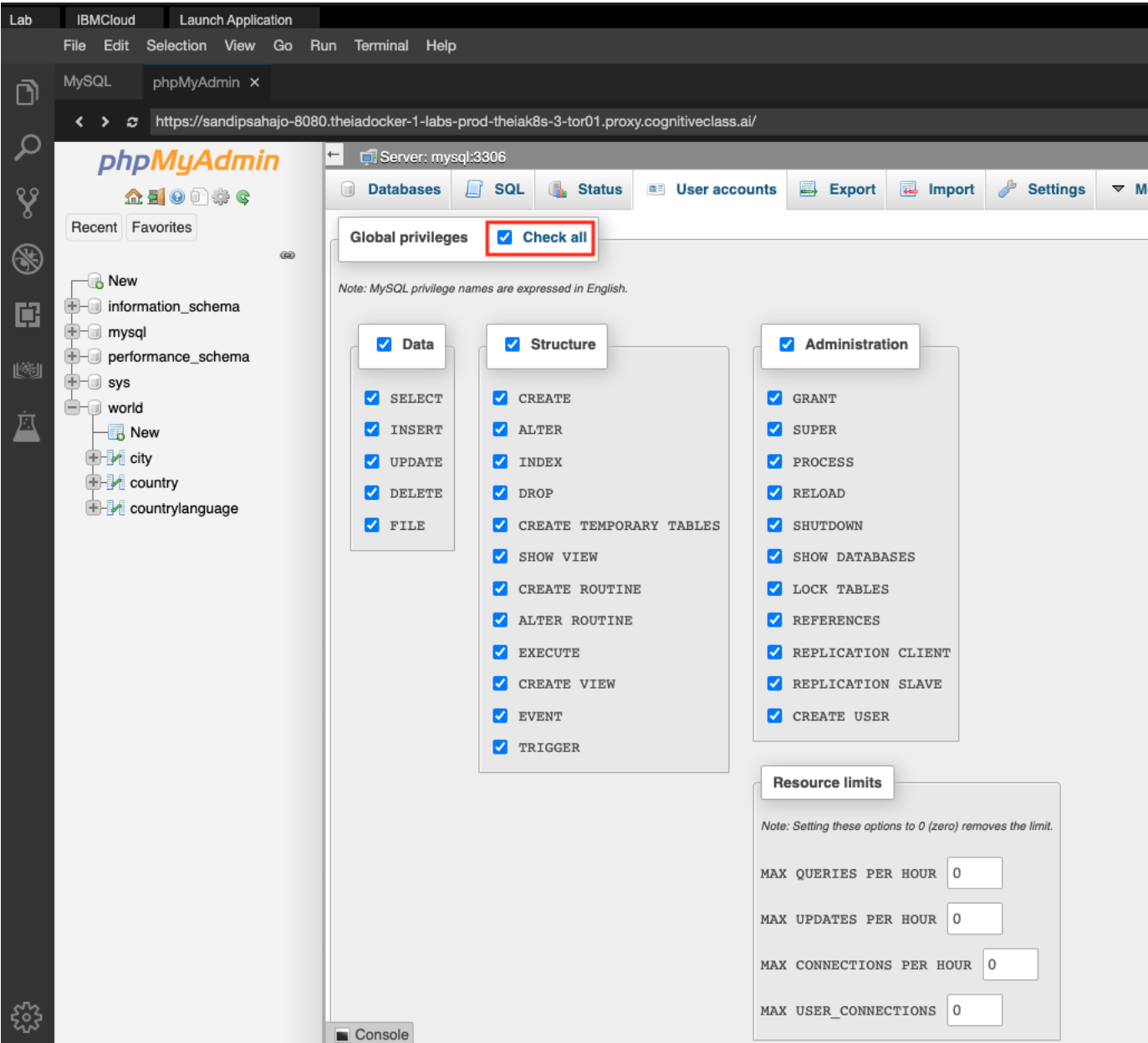
CREATE

GRANT

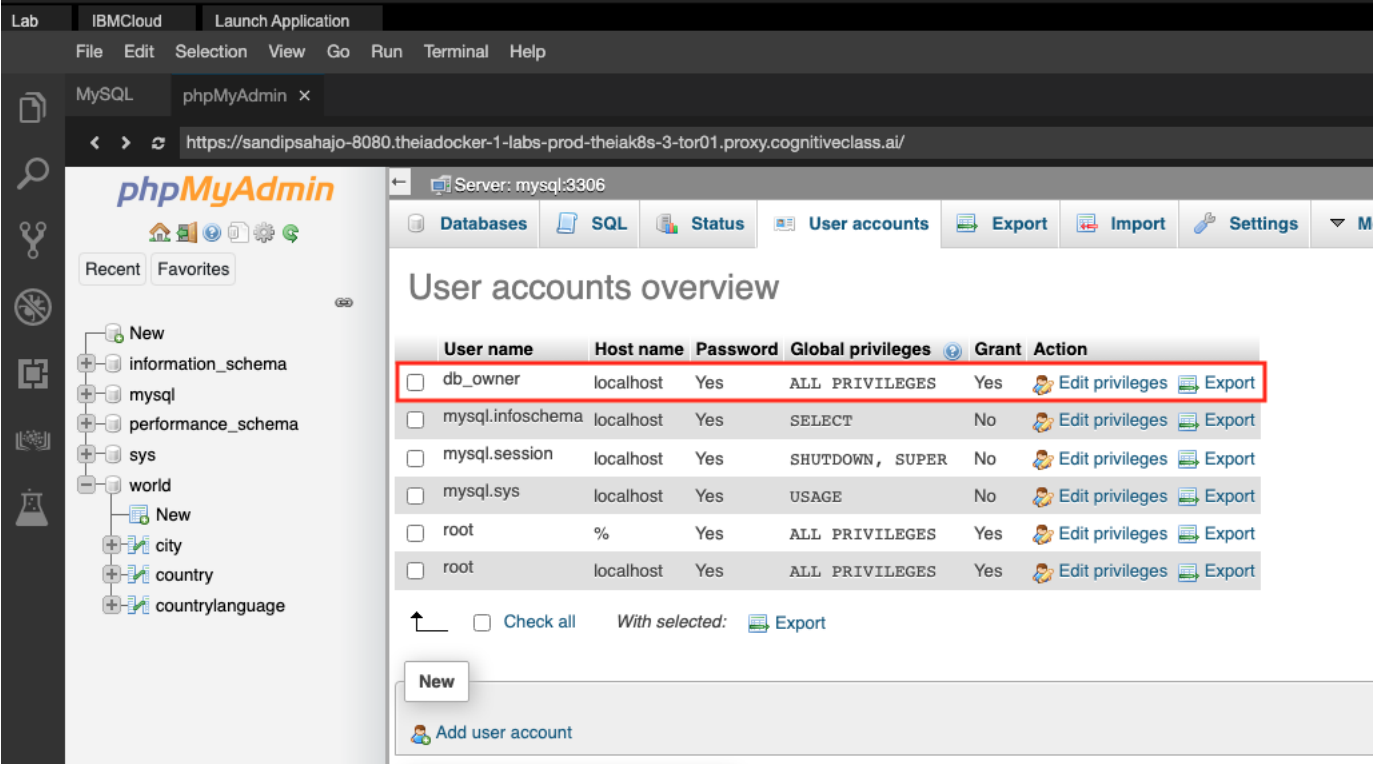
ALTER

SUPER





8. You have successfully created a user account with appropriate privileges.

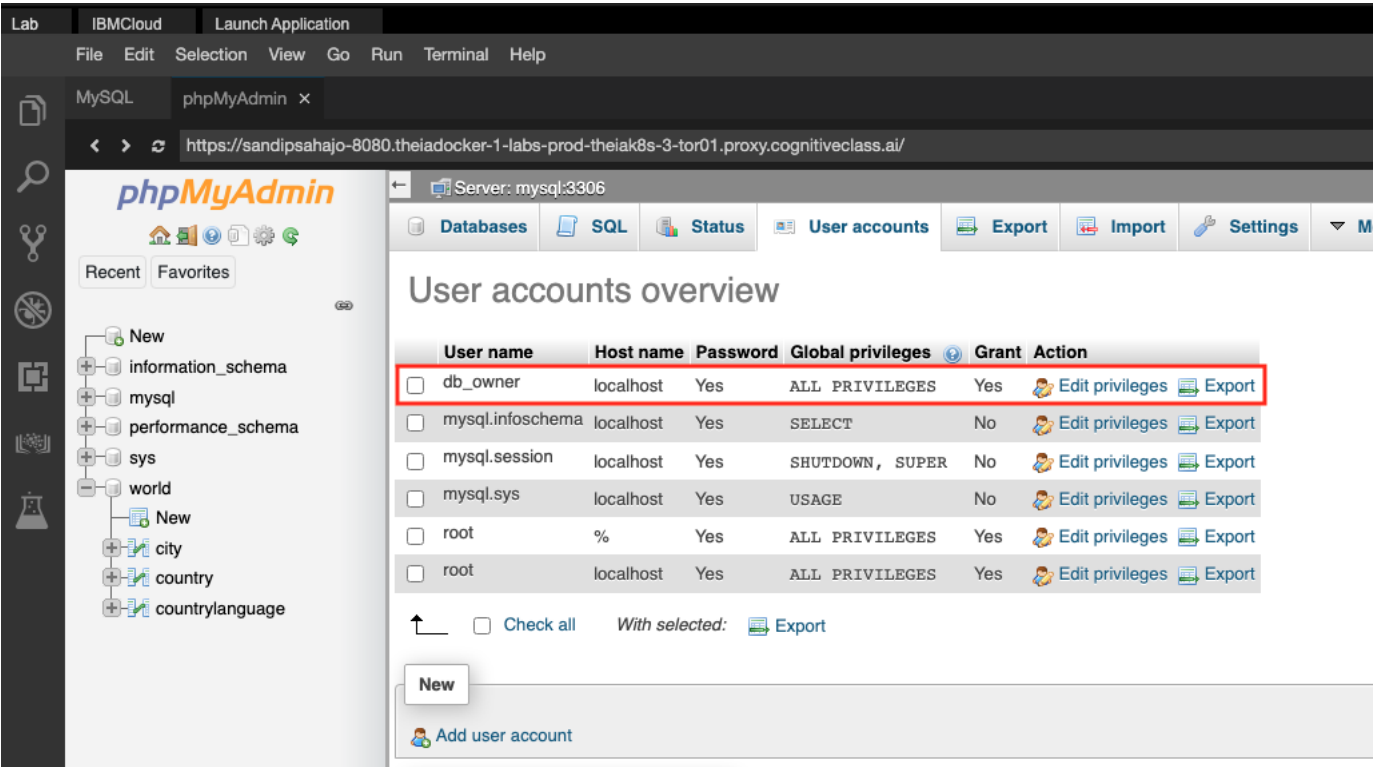


Exercise 2: Control access to MySQL databases and their objects

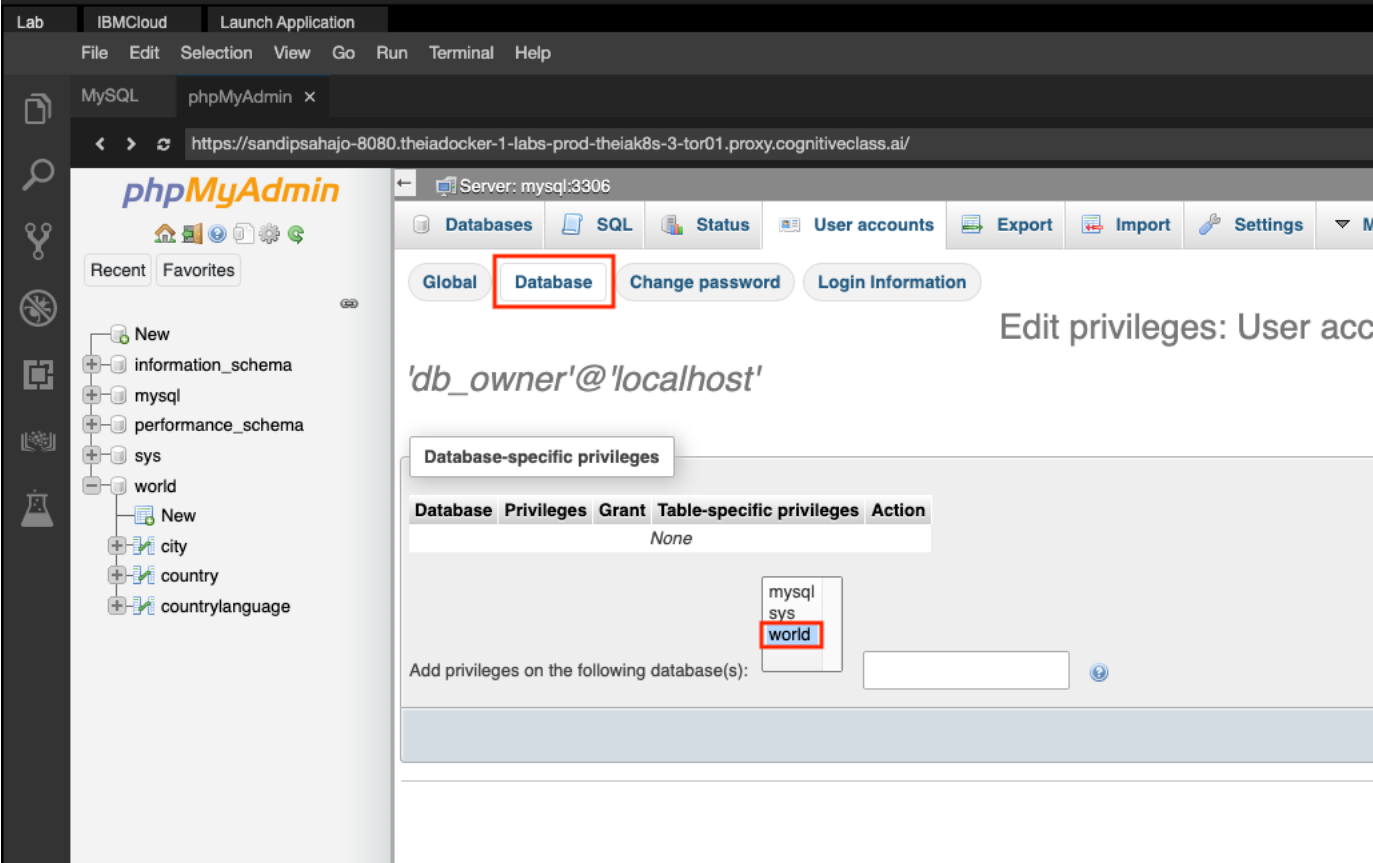
In this example exercise, you will learn how to control access to MySQL databases and their objects.

Making an exception to the user definition of db\_owner role you created earlier, you will modify privileges of this user so that this user won't be able to update other columns except a specific column of a specific table of a specific database. You will restrict db\_owner from updating all the other columns except the column **Population** of the table **city** of the database **world**.

1. Go to **Home > User accounts** tab. Click **Edit privileges** option of **db\_owner** user name.



2. Under **Database** sub-tab, select **world** database and click **Go**.



3. Under **Database-specific privileges**, select **Check all** and click **Go** at the bottom.

Lab | IBMCloud | Launch Application

File Edit Selection View Go Run Terminal Help

MySQL phpMyAdmin x

https://sandipsahajo-8080.theiadocker-1-labs-prod-theiak8s-3-tor01.proxy.cognitiveclass.ai/

Server: mysql:3306

Databases SQL Status User accounts Export Import Settings M

Database Table Routine

### Edit privileges: User account

#### 'db\_owner'@'localhost' - Database world

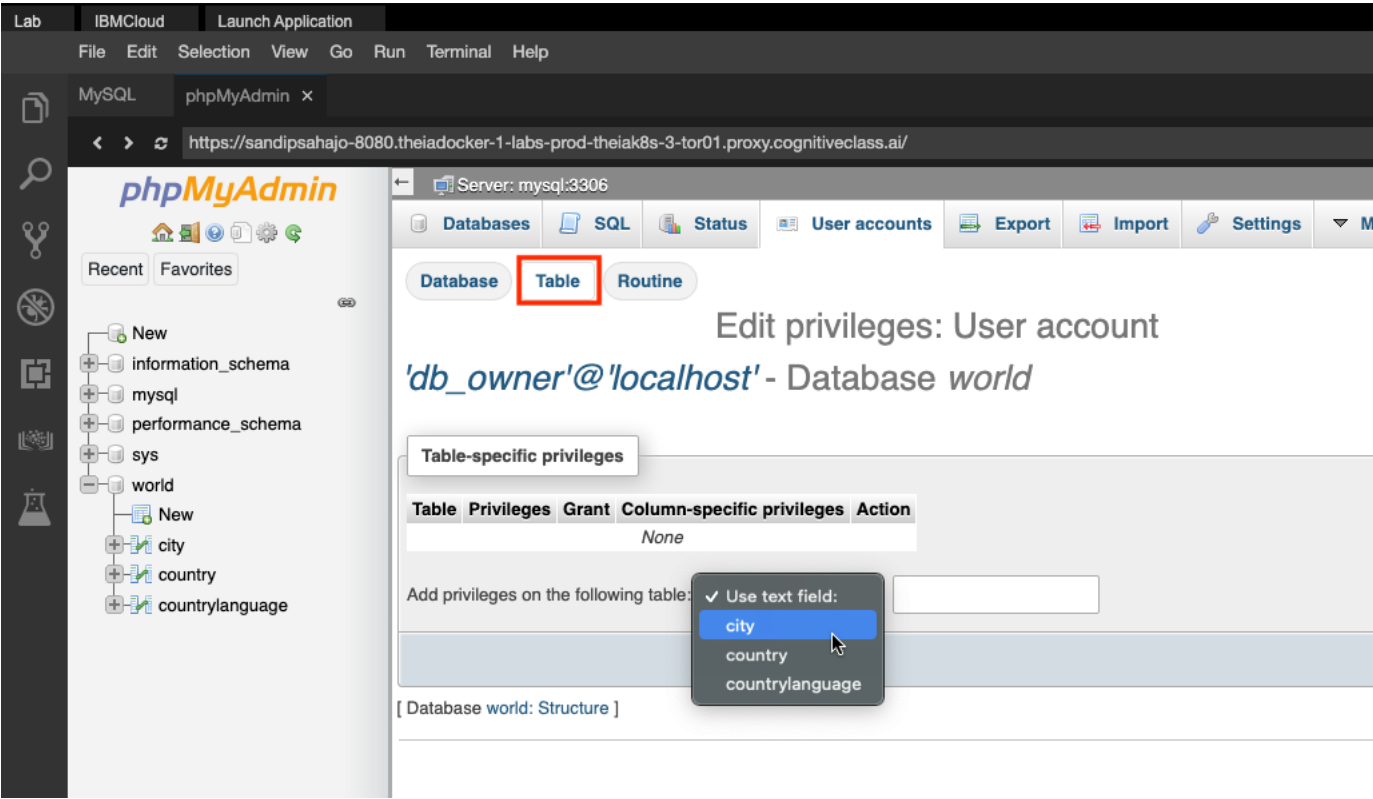
Database-specific privileges ☒ Check all

Note: MySQL privilege names are expressed in English.

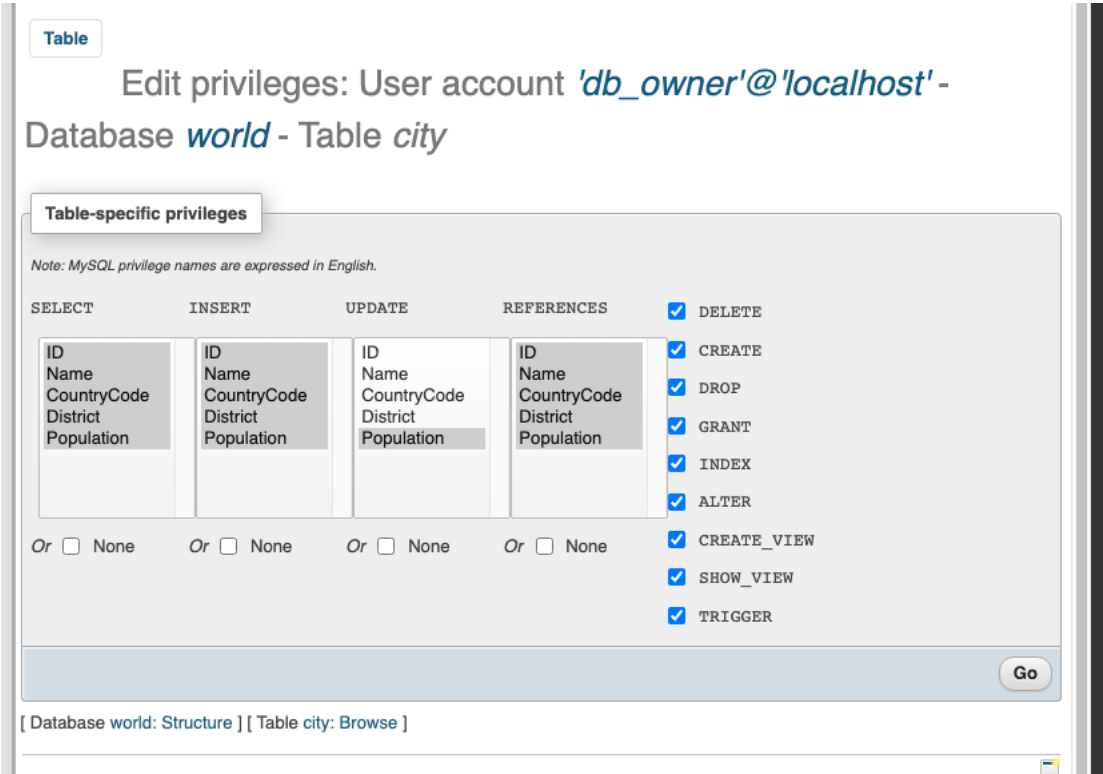
<input checked="" type="checkbox"/> Data	<input checked="" type="checkbox"/> Structure	<input checked="" type="checkbox"/> Administration
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> GRANT
<input checked="" type="checkbox"/> INSERT	<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> LOCK TABLES
<input checked="" type="checkbox"/> UPDATE	<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> REFERENCES
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP	
	<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	
	<input checked="" type="checkbox"/> SHOW VIEW	
	<input checked="" type="checkbox"/> CREATE ROUTINE	
	<input checked="" type="checkbox"/> ALTER ROUTINE	
	<input checked="" type="checkbox"/> EXECUTE	
	<input checked="" type="checkbox"/> CREATE VIEW	
	<input checked="" type="checkbox"/> EVENT	
	<input checked="" type="checkbox"/> TRIGGER	

[ Database world: Structure ]

4. Switch to **Table** sub-tab. Select the table **city** from the drop-down menu and click **Go**.



5. Under **Table-specific privileges**, configure all the SQL commands and their custom access to the columns of the table **city** as shown below. Then click **Go**. Such table-specific privilege configuration will restrict db\_owner from updating all the other columns except the column **Population** of the table **city** of the database **world**.



### Exercise 3: Secure data using encryption

In this example exercise, you will learn how to secure your data adding extra layer of security using data encryption. There may be certain parts of your database containing sensitive information which *should not* be stored in plain text. This is where encryption comes in.

You will implement encryption and decryption of a column in the world database using the official AES (Advanced Encryption Standard) algorithm. AES is a symmetric encryption where the same key is used to encrypt and decrypt the data. The AES standard permits various key lengths. By default, key length of 128-bits is used. Key lengths of 196 or 256 bits can be used. The key length is a trade off between performance and security. Let's get started.

1. Click the **MySQL CLI** button from the mysql service session tab.

Lab | IBMCloud | Launch Application

File | Edit | Selection | View | Go | Run | Terminal | Help

MySQL x phpMyAdmin

# MySQL

ACTIVE

v8.0.22 | v5.0.4 | v14.14

Connect to MySQL and phpMyAdmin directly in your Skills Network Labs environment

Stop

Summary | Connection Information | Details

Your database and phpMyAdmin server are now ready to use and available via the terminal. For more details on how to navigate MySQL, please check out the Details section.

**Username:** davidpaster2

**Password:** MTg4OTctZGF2aWRw

You can manage MySQL via:

phpMyAdmin

Or to interact with the database in the terminal, select one of these options:

MySQL CLI | New Terminal

2. First, you will need to hash your passphrase (consider your passphrase is **My secret passphrase**) with a specific hash length (consider your hash length is **512**) using a hash function (here you will use hash function from **SHA-2** family). It is good practice to hash the passphrase you use, since storing the passphrase in

plaintext is a significant security vulnerability. Use the following command in the terminal to use the SHA2 algorithm to hash your passphrase and assign it to the variable `key_str`:

```
1. 1
1. SET @key_str = SHA2('My secret passphrase', 512);
```

Copied!

3. Now, let's take a look at the `world` database. First, you will want to connect to the database by entering the following command in the CLI:

```
1. 1
1. USE world;
```

Copied!

4. Next, let's take a quick look at the `countrylanguage` table in our database with the following command:

```
1. 1
1. SELECT * FROM countrylanguage LIMIT 5;
```

Copied!

```
theia@theiadocker-davidpaster2: /home/project ×

Database changed
mysql> SELECT * FROM countrylanguage LIMIT 5;
+-----+-----+-----+-----+
| CountryCode | Language | IsOfficial | Percentage |
+-----+-----+-----+-----+
| ABW        | Dutch   | T          | 5.3        |
| ABW        | English | F          | 9.5        |
| ABW        | Papiamentto | F      | 76.7       |
| ABW        | Spanish | F          | 7.4        |
| AFG        | Balochi | F          | 0.9        |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> █
```

For *demonstration purposes*, **suppose** that the last column in the table, labeled *Percentage* contains sensitive data, such as a citizen's passport number. Storing such sensitive data in plain text is an enormous security concern, so let's go ahead and encrypt that column.

5. To encrypt the *Percentage* column, we will first want to convert the data in the column into binary byte strings of length 255 by entering the following command into the CLI:

```
1. 1
1. ALTER TABLE countrylanguage MODIFY COLUMN Percentage varbinary(255);
```

Copied!

6. Now to actually encrypt the *Percentage* column, we use the AES encryption standard and our hashed passphrase to execute the following command:

```
1. 1
1. UPDATE countrylanguage SET Percentage = AES_ENCRYPT(Percentage, @key_str);
```

Copied!

7. Let's go ahead and see if the column was successfully encrypted by taking another look at the *countrylanguage* table. We again run the same command as in step 4:

```
1. 1
1. SELECT * FROM countrylanguage LIMIT 5;
```

Copied!

```
theia@theiadocker-davidpaster2: /home/project ×

mysql> SELECT * FROM countrylanguage LIMIT 5;
+-----+-----+-----+-----+
| CountryCode | Language | IsOfficial | Percentage |
+-----+-----+-----+-----+
| ABW         | Dutch    | T          | d0L0D00e0!0h~ |
| ABW         | English  | F          | 0rG0000000(0R0KK |
| ABW         | Papiamentto | F        | 100Â0;00ž0m |
| ABW         | Spanish  | F          | K0000A0 o00 |
| AFG         | Balochi  | F          | 00'0hB0zI0Hr |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> █
```

As you can see, the data on the *Percentage* column is encrypted and completely illegible.

8. The supposedly sensitive data is now encrypted and secured from prying eyes. However, we should still have a way to access the encrypted data when needed. To do this, we use the `AES_DECRYPT` command, and since AES is symmetric, we use the same key for both encryption and decryption. In our case, recall that the key was a passphrase which was hashed and stored in the variable `key_str`. Suppose we need to access the sensitive data in that column. We can do so by entering the following command in the CLI:

```
1. 1
1. SELECT cast(AES_DECRYPT(Percentage, @key_str) as char(255)) FROM countrylanguage;
```

Copied!

```
theia@theiadocker-davidpaster2: /home/project ×

5 rows in set (0.01 sec)

mysql> SELECT cast(AES_DECRYPT(Percentage, @key_str) as char(2
+-----+-----+-----+-----+
| cast(AES_DECRYPT(Percentage, @key_str) as char(255)) |
+-----+-----+-----+-----+
| 5.3 |
| 9.5 |
| 76.7 |
| 7.4 |
| 0.9 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

## Practice Exercise: Control access to MySQL databases and their objects

In this practice exercise, you will get to put what you learned to use and modify privileges for a user.

Scenario: You will modify privileges of the user **db\_owner** you created in example exercise A such a way that this user won't be able to insert, update and delete any column of a specific table **country** of the **world** database.

▼ Hint (Click Here)

Follow example exercise 2. If necessary create the user account following example exercise 1.

▼ Solution (Click Here)



1. Go to **Home** > **User accounts** tab. Click **Edit privileges** option of **db\_owner** user name.
2. Under **Database** sub-tab, select **world** database and click **Go**.
3. Under **Database-specific privileges**, select **Check all** and click **Go** at the bottom.
4. Switch to **Table** sub-tab. Select the table **country** from the drop-down menu and click **Go**.
5. Under **Table-specific privileges**, configure all the SQL commands and their custom access to the columns of the table **city** as shown in the following image. Then click **Go**. Such table-specific privilege configuration will restrict db\_owner from inserting, updating and deleting any column of a specific table **country** of the **world** database.

Table

Edit privileges: User account 'db\_owner'@'localhost' - Database world - Table country

Table-specific privileges

Note: MySQL privilege names are expressed in English.

SELECT

Code  
Name  
Continent  
Region  
SurfaceArea  
IndepYear  
Population  
LifeExpectancy

Or ☐ None

INSERT

Code  
Name  
Continent  
Region  
SurfaceArea  
IndepYear  
Population  
LifeExpectancy

Or ☒ None

UPDATE

Code  
Name  
Continent  
Region  
SurfaceArea  
IndepYear  
Population  
LifeExpectancy

Or ☒ None

REFERENCES

Code  
Name  
Continent  
Region  
SurfaceArea  
IndepYear  
Population  
LifeExpectancy

Or ☐ None

☐ DELETE

☒ CREATE

☒ DROP

☒ GRANT

☒ INDEX

☒ ALTER

☒ CREATE\_VIEW

☒ SHOW\_VIEW

☒ TRIGGER

Go

[ Database world: Structure ] [ Table country: Browse ]

Congratulations! You have completed this lab, and you are ready for the next topic.

Author(s)

- [Sandip Saha Joy](#)

Other Contributor(s)

- [David Pasternak](#)

Changelog

Date	Version	Changed by	Change Description
2021-07-13	1.0	Sandip Saha Joy	Created initial version
2021-10-05	1.1	David Pasternak	Updated instructions
2021-10-13	1.2	Steve Hord	Copy edits
2021-07-13	1.3	Jaskomal Natt	Updated copyright date