

Hands-on Lab: Monitoring and Optimizing Your Databases in MySQL

Estimated time needed: 45 minutes

In this lab, you'll learn how to monitor and optimize your database in MySQL with the help of phpMyAdmin.

Objectives

After completing this lab, you will be able to:

- Maintain the health and performance of your database with phpMyAdmin
- Identify the difference between an unoptimized and optimized database
- Optimize your database with phpMyAdmin by applying best practices

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.



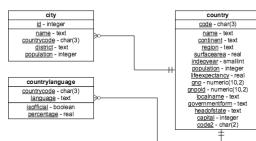
To complete this lab, you will utilize the MySQL relational database service available as part of the IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database Used in this Lab

The World database used in this lab comes from the following source: <https://dev.mysql.com/doc/world-source/> under CC BY 4.0 License with Copyright 2021 - Statistics Final.

You will be using 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 EER diagram shows the schema of the World database:



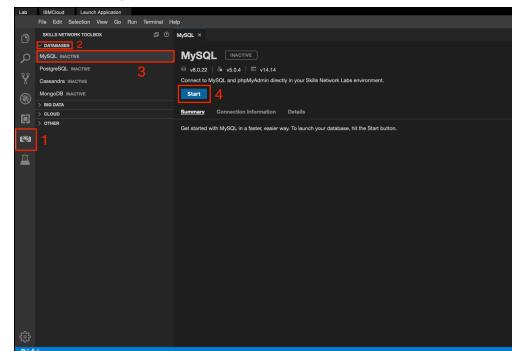
The first row is the table name, the second is the primary key, and the remaining items are any additional attributes.

Exercise 1: Create Your Database

To begin, we'll create the *World* database and load the necessary data.

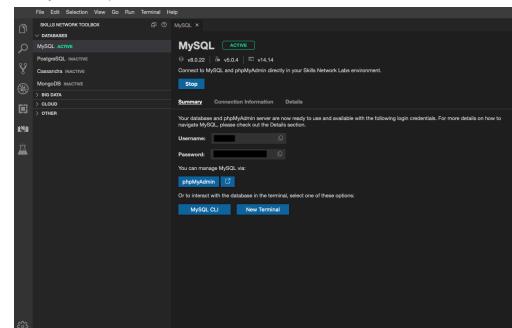
1. First, we'll launch MySQL. We can do this by navigating to the Skills Network Toolbox, selecting Databases and then MySQL.

Press Start. This will start a session of MySQL in SN Labs.



The **Inactive** label will change to **Starting**. This may take a few moments.

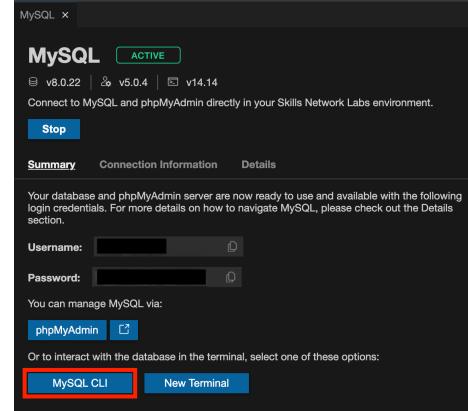
When it changes to **Active**, it means your session has started.



2. In the menu bar, select Terminal > New Terminal. This will open the Terminal:

To download the zip file containing the *sql* file for the *World* database, and paste the following into the Terminal:
1. https://cf-courses-data.4sofa.courses.cloud-object-storage.appdomain.cloud/b8e-08d21cb-04111b70c0/datasets/world_mysql_script.sql

3. Next, initiate a MySQL command-line session by clicking MySQL CLI from the MySQL tab. Doing this will allow us to create a table in MySQL via the command-line.



4. In the terminal, create a new database called *world*.

► Hint (Click Here)

Solution (Click Here)

In the Terminal, we can create a new database *world* with the following:

1. `CREATE DATABASE world;`

► Hint (Click Here)

Solution (Click Here)

To see the newly created *world* database, we can enter the following into the Terminal:

1. `USE world;`

► Hint (Click Here)

Solution (Click Here)

5. Now, we'll want to use the newly created database, *world*. How can we go about doing that?

► Hint (Click Here)

Solution (Click Here)

To use the newly created *world* database, we can enter the following into the Terminal:

1. `USE world;`

► Hint (Click Here)

Solution (Click Here)

6. In order to complete the database creation process, we'll want to execute the MySQL script containing the data that we downloaded. This file will create tables and insert data into those tables.

► Hint (Click Here)

Solution (Click Here)

To execute the SQL script file that we downloaded, we can use the following command:

► Hint (Click Here)

Solution (Click Here)

```

1-1
-- SOURCE world_myndi_script.sql;
-- This may take about thirty seconds or so.

7 Wait a few database creation, we'll want to take a look at the tables inside it to get a general idea of the data we have. What tables do we have?

8 Hint (Click Here)
9 Solution (Click Here)

We can show the tables in the database with the SHOW TABLES command:

```

```

1-1
-- SHOW TABLES;
-- Output

```

As you can see, there are a total of three tables in the world database: city, country and countrylanguage.

Exercise 2: Monitor Your Database

Database monitoring refers to the tasks related to reviewing the operational status of your database. Monitoring is critical in maintaining the health and performance of your database because it'll help you identify issues in a timely manner. In doing so, you'll be able to avoid problems such as database outages that affect your users.

With phpMyAdmin, we can take a look at how our database is doing and how we can improve it.

1 In the MySQL tab, select button next to phpMyAdmin to access the tool in a new tab.

The screenshot shows the MySQL monitor interface. At the top, it displays connection details: v8.0.22, v5.0.4, v14.14. Below this, a message says "Connect to MySQL and phpMyAdmin directly in your Skills Network Labs environment." There are "Stop" and "Start" buttons. Under "Summary", there are tabs for "Connection Information" and "Details". A note says "Your database and phpMyAdmin server are now ready to use and available with the following login credentials. For more details on how to navigate MySQL, please check out the Details section." It shows "Username" and "Password" fields. A "phpMyAdmin" link is highlighted with a red box. Below it, a note says "Or to interact with the database in the terminal, select one of these options: MySQL CLI, New Terminal".

2 On the phpMyAdmin homepage, you'll notice a left sidebar and right panel.

On the left sidebar, you'll see all your databases, including the one we just created, world.

For now, we'll want to focus on the right panel to see more information about our servers and databases.

Select Status.

The screenshot shows the phpMyAdmin Status page. At the top, there's a navigation bar with tabs: Databases, SQL, Status (which is selected and highlighted with a red box), User accounts, Export, Import, Settings, Binary log, Replication, Variables,Charsets, Engines, and Plugins. The main content area is divided into several sections: General settings, Appearance settings, Database server, Web server, and phpMyAdmin. The General settings section shows "Server connection collation: utf8mb4_unicode_ci". The Appearance settings section shows "Language: English" and "Theme: pmahomme". The Database server section lists: Server: mysql via TCP/IP, Server type: MySQL, Server connection: SSL is not being used, Server version: 8.0.22 - MySQL Community Server - GPL, Protocol version: 10, User: root@172.25.0.3, and Server charset: UTF-8 Unicode (utf8mb4). The Web server section lists: Apache/2.4.38 (Debian), Database client version: libmysql - mysqld 7.4.15, PHP extension: mysqli curl mbstring, and PHP version: 7.4.15. The phpMyAdmin section lists: Version information: 5.0.4, latest stable version: 5.1.1, Documentation, Official Homepage, Contribute, Get support, List of changes, and License.

3 There are several subpages on the Status page: Server, Processes, Query Statistics, All status variables, Monitor and Advisor. These subpages will give you an inside look at the current processes on your server.

The screenshot shows the phpMyAdmin Server subpage. At the top, there's a navigation bar with tabs: Databases, SQL, Status, User accounts, Export, Import, Settings, Binary log, Replication, Variables,Charsets, Engines, and Plugins. Below this, there are tabs for Server, Processes, Query statistics, All status variables, Monitor, and Advisor. A message says "Network traffic since startup: 10.3 MiB". A table shows "Traffic" statistics: Received 1.7 MiB, Sent 8.7 MiB, Total 10.3 MiB; and "Connections" statistics: Max. concurrent connections 4, Failed attempts 203, Aborted 0, Total 1,325. A note says "This MySQL server works as master in replication process."

Replication status

Master status

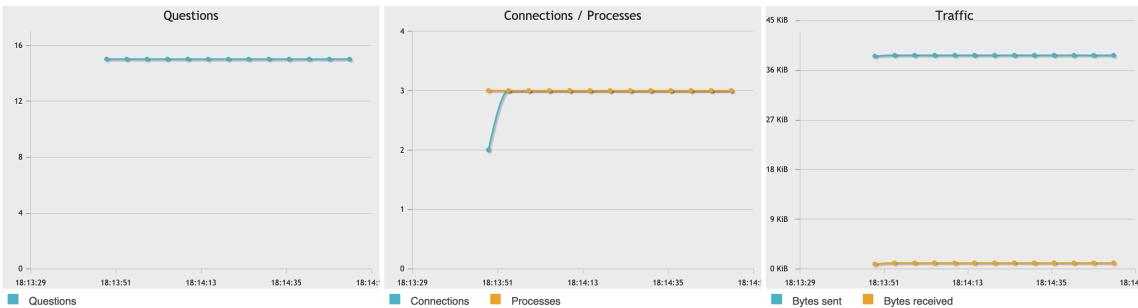
Variable	Value
File	b1nlog.000002
Position	1690985
Binlog_Do_DB	
Binlog_Ignore_DB	

Feel free to explore the rest of the subpages to gain a better understanding of the current status of your server.

If you'd like a refresher on the information provided by each subpage, you can revisit the previous reading, [Monitoring and Optimizing Your Database in MySQL](#).

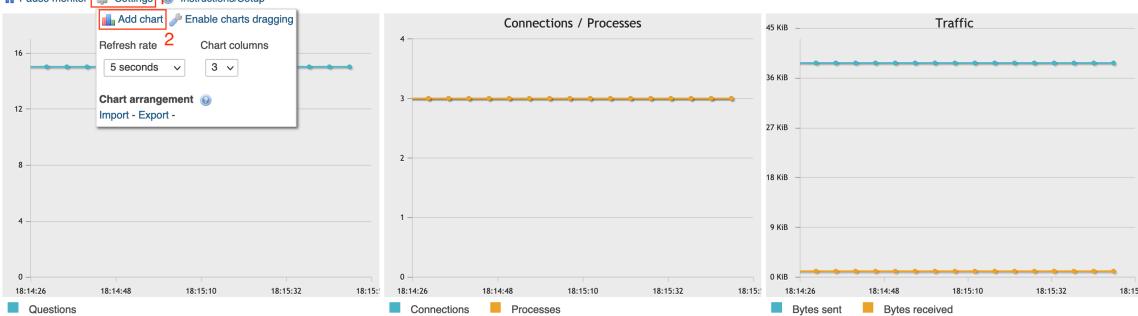
In this lab, we'll be taking a look at the **Monitor** subpage.

4 From the available sections, click **Monitor**.

Pause monitor Settings Instructions/Setup


When you first arrive at this page, you may only have these three charts: Questions, Connections / Processes and Traffic. We can add a few more graphs to help track important metrics.

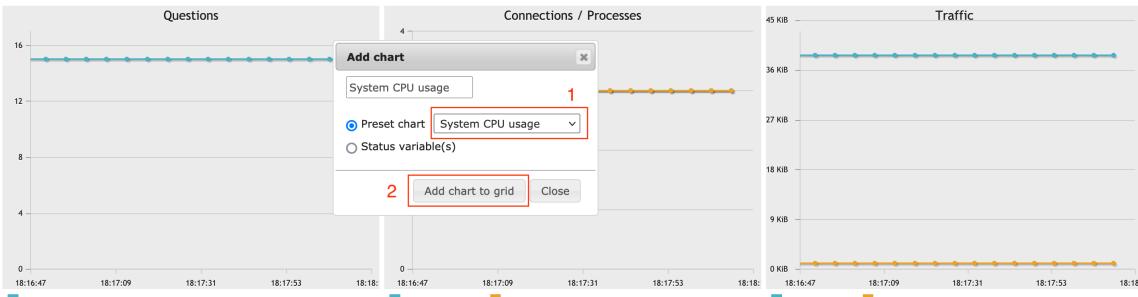
5. Select Settings then Add chart.

 Pause monitor Settings Instructions/Setup


6. In the dialog box, you can choose a chart based on a status variable or select Preset chart. In this case, we'll be adding two new charts.

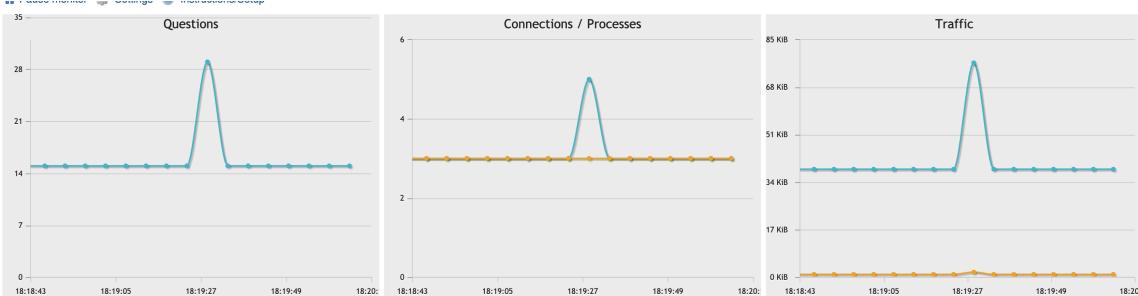
First, let's select System CPU usage.

Press Add chart to grid.

 Pause monitor Settings Instructions/Setup


Now, you should see the System CPU Usage following chart.

CPU usage Memory usage Disk usage Network usage Group



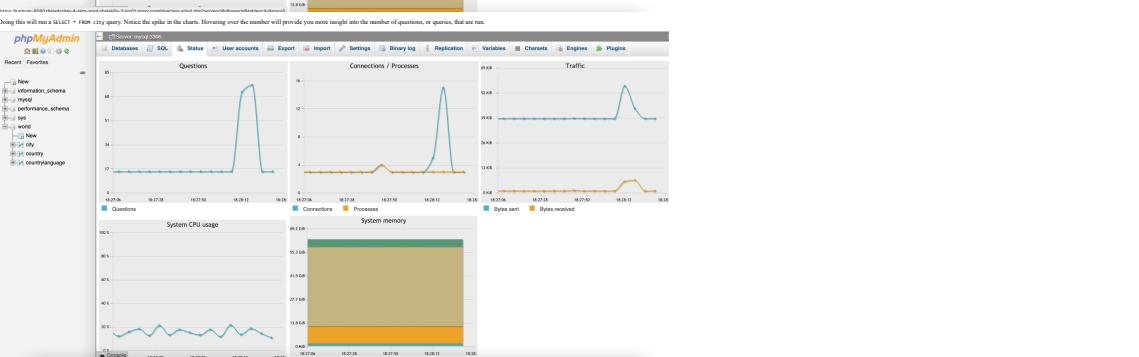
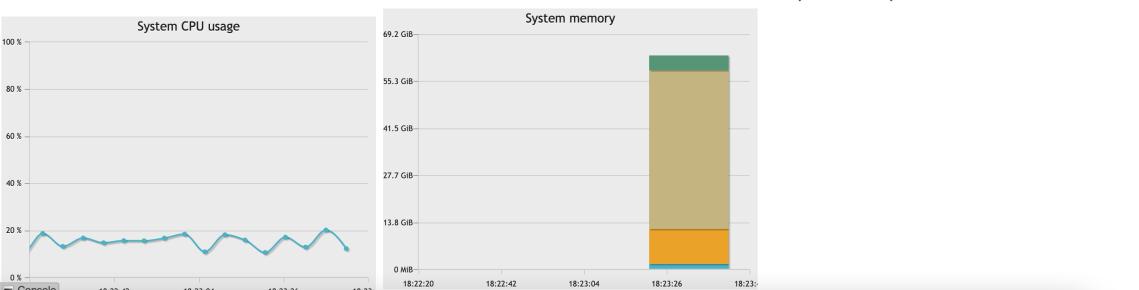
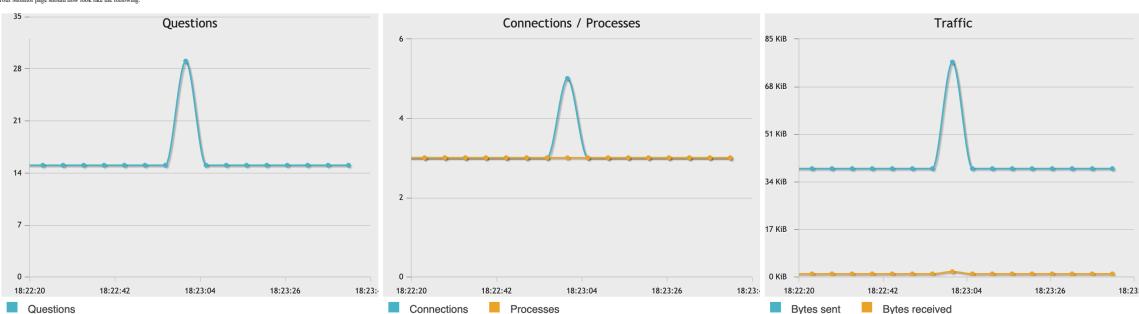
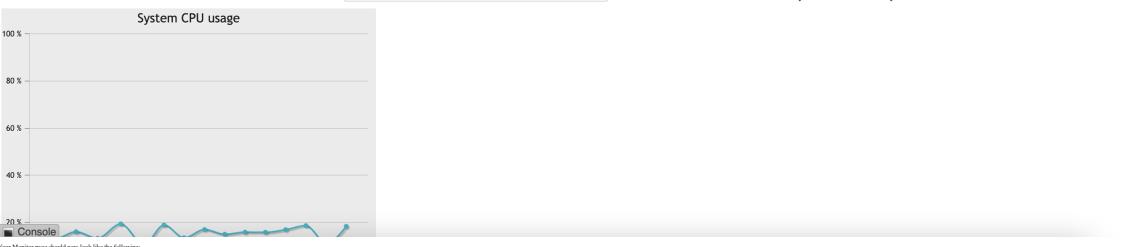
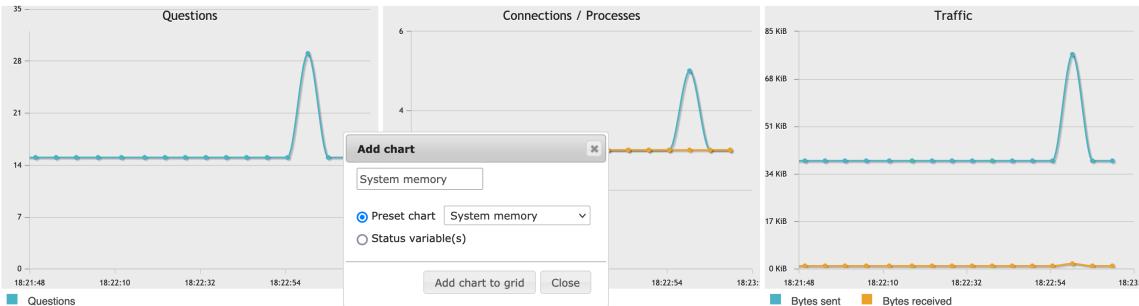
Notice how there was a spike in your charts. Recall that Questions includes any queries run in the background by phpMyAdmin. In this case, the interaction was MySQL bringing up the System CPU Usage chart.

7. Repeat the process to add the System memory chart.

[Home \(Click Here\)](#)

[Solutions \(Click Here\)](#)

When adding the chart, the dialog box should look like the following:

[Pause monitor](#) [Settings](#) [Instructions/Setup](#)


To confirm that we ran the query, we can navigate to the new tab we opened and take a look at the success message:

Showing rows 0 - 24 (4006 total, Query took 0.0007 seconds.)

```
SELECT * FROM `city`
```

	ID	Name	CountryCode	District	Population
<input type="checkbox"/>	1	Kabul	AFG	Kabul	1780000
<input type="checkbox"/>	2	Qandahar	AFG	Qandahar	237500
<input type="checkbox"/>	3	Herat	AFG	Herat	186800
<input type="checkbox"/>	4	Mazar-e-Sharif	AFG	Balkh	127800
<input type="checkbox"/>	5	Amsterdam	NLD	Noord-Holland	731200
<input type="checkbox"/>	6	Rotterdam	NLD	Zuid-Holland	593321
<input type="checkbox"/>	7	Haag	NLD	Zuid-Holland	440900
<input type="checkbox"/>	8	Utrecht	NLD	Utrecht	234323
<input type="checkbox"/>	9	Eindhoven	NLD	Noord-Brabant	201843
<input type="checkbox"/>	10	Tilburg	NLD	Noord-Brabant	193238
<input type="checkbox"/>	11	Groningen	NLD	Groningen	172701
<input type="checkbox"/>	12	Breda	NLD	Noord-Brabant	160398
<input type="checkbox"/>	13	Apolloorn	NLD	Gelderland	153491
<input type="checkbox"/>	14	Nijmegen	NLD	Gelderland	152463
<input type="checkbox"/>	15	Enschede	NLD	Overijssel	149544

Since phpMyAdmin limits queries to displaying the first 25 rows (similar to the LIMIT clause), the query did not take long to load.

The Status page is helpful in monitoring the health and performance of your server and database. Particularly with the Monitor page's charts, you'll be able to see real-time information on how your databases are doing. If there are unexpected irregularities, such as a sudden spike in queries or a high volume of incoming traffic, then that may point to a problem that needs to be attended to.

Exercise 3: Optimize Your Database

Database optimization refers to maximizing the speed and efficiency of retrieving data from your database. When you optimize your database, you are improving its performance, which can also improve the performance of slow queries. By optimizing your database, you'll improve the experience for both yourself and your users.

We can see this in practice with the following scenario:

Imagine that you are the organizer of a sports tournament with a total of 120 participating teams. There's three pieces of information that you've collected from each team: their (unique) team number, their three-letter team code for the system, and the date of the team's creation.

How can we go about optimizing a database containing that information? Let's take a look!

Task A: Create Your Table

1. To start, we'll create a database in phpMyAdmin.

On the left sidebar, select the New button to create a new database.

The screenshot shows the phpMyAdmin interface with the 'General settings' tab selected. In the center, there's a 'Database server' section with details like 'Server connection: utf8mb4_unicod_ci'. Below it is a 'Web server' section with 'Apache/4.8.30 (Ubuntu)' and PHP version information. At the bottom, the 'phpMyAdmin' section lists version information and links for documentation, support, and license.

2. On the Databases page, you can enter a Database name and leave it as the UTF-8 encoding. UTF-8 is the most commonly used character encoding for content or data.

Let's call our database `tournament_teams`.

Press Create.

The screenshot shows the 'Databases' page with the 'tournament_teams' database listed. It has 1 table: 'teams'. The table has columns: team_no (INT), team_code (CHAR 100), and creation_date (CHAR 100). There are checkboxes for 'Check id', 'With select', and 'Drop'.

Now, we can make a table. Let's call it `teams`.

Read the information we want to store in this database: team number, team code and team creation date. We can make these columns for that.

The screenshot shows the 'Structure' tab for the 'teams' table. It has 3 columns: 'team_no' (INT), 'team_code' (CHAR 100), and 'creation_date' (CHAR 100). There are checkboxes for 'Check id', 'With select', and 'Drop'.

No tables found in database.

The screenshot shows the 'Create table' page for the 'teams' table. It has 3 columns: 'team_no' (INT), 'team_code' (CHAR 100), and 'creation_date' (CHAR 100). There are checkboxes for 'Check id', 'With select', and 'Drop'.

Press Go.

We can now add in the column names and data types.

We'll enter the following information into the table. Below are only the values that you'll need to fill out. Anything else can be left as is:

Name	Type	Length/Value
team_no	INT	
team_code	CHAR 100	
creation_date	CHAR 100	

Since the team number is unique, with each of the 120 teams having their own number ranging from 1 to 120, that will be our primary key. It is a numeric data type, so we use INT for now.

The team code and creation date are both set as the CHAR data type. Notice that for the CHAR type, we must include the length of the value. To be safe, we've set that to 100.

You might be thinking that these data types aren't the best choice for this database. You'd be correct! Given what we know, there are better choices for each of the entries, but let's first take a look at an unoptimized database and how we can optimize it.

When all the values are entered, press Save.

Table name: teams Add 1 column(s) Go

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A.I.	Comments	Virtuality
team_no	INT		None			✓	---	✓		
team_code	CHAR	100	None			✓	---	✓		
creation_date	CHAR	100	None			✓	---	✓		

Structure Table comments: Collation: Storage Engine: InnoDB

PARTITION definition: Partition by: (Expression or column) Partitions:

Preview SQL Save

On the Structure page, select the checkbox next to team_no and click Primary.

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure **Relation view**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input checked="" type="checkbox"/> 1	team_no	int		No	None				<input type="button" value="Change"/> <input type="button" value="Drop"/> More
<input type="checkbox"/> 2	team_code	char(100)	utf8mb4_0900_ai_ci	No	None				<input type="button" value="Change"/> <input type="button" value="Drop"/> More
<input type="checkbox"/> 3	creation_date	char(100)	utf8mb4_0900_ai_ci	No	None				<input type="button" value="Change"/> <input type="button" value="Drop"/> More

Check all With selected:

This will set team_no as the primary key of the table. If you were successful in doing so, you'll receive a success message, like the following:

Your SQL query has been executed successfully.

ALTER TABLE `teams` ADD PRIMARY KEY(`team_no`);

[Edit inline] [Edit] [Create PHP code]

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure **Relation view**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	team_no	int		No	None				<input type="button" value="Change"/> <input type="button" value="Drop"/> More
<input type="checkbox"/> 2	team_code	char(100)	utf8mb4_0900_ai_ci	No	None				<input type="button" value="Change"/> <input type="button" value="Drop"/> More
<input type="checkbox"/> 3	creation_date	char(100)	utf8mb4_0900_ai_ci	No	None				<input type="button" value="Change"/> <input type="button" value="Drop"/> More

Check all With selected:

Print Move columns Normalize

Add 1 column(s) after creation_date Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
<input type="button" value="Edit"/> <input type="button" value="Drop"/> PRIMARY	PRIMARY	BTREE	Yes	No	team_no	0	A	No	

Create an index on 1 columns Go

Notice that team_no was automatically added as a PRIMARY index as well.

Let's import the data! This sample teams data has been prepared specifically for this lab and is composed of team numbers, team code names, and the creation date of these teams.

You can download the file by clicking the following link: tournament_teams_data.sql.

To import it, navigate to Import, then select Browse and find the file you just downloaded.

Server: mysql:3306 Database: tournament_teams Table: teams

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

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

tournament_teams_data.sql (Max: 2.048KB)

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

Character set of the file: utf-8

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, however it can break transactions.)

Skip this number of queries (for SQL) starting from the first one: 0

Other options:

Enable foreign key checks

Format:

SQL

Format-specific options:

SQL compatibility mode: NONE

Do not use AUTO_INCREMENT for zero values

Go

Now, we can take a look at the imported data by navigating back to Browse. This automatically loads a SELECT * FROM teams statement, which will load the first 25 rows from the teams table.

Import has been successfully finished, 120 queries executed. (tournament_teams_data.sql)

Server: mysql:3306 > Database: tournament_teams > Table: teams

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 24 (120 total, Query took 0.0006 seconds.)

SELECT * FROM `teams`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	team_no	team_code	creation_date
Edit	1	ACE	2021-01-01
Edit	2	FAX	2021-01-03
Edit	3	RED	2021-01-04
Edit	4	MAZ	2021-01-05
Edit	5	AMS	2021-01-06
Edit	6	ROT	2022-02-10
Edit	7	HAS	2021-01-08
Edit	8	BLU	2021-01-09
Edit	9	EIN	2021-01-10
Edit	10	TIL	2021-01-11
Edit	11	GRO	2021-01-12
Edit	12	BRE	2021-01-13
Edit	13	APE	2021-01-14
Edit	14	NIJ	2021-01-15
Edit	15	ENS	2021-01-16
Edit	16	HAR	2021-01-17

The limit is implemented by phpMyAdmin, but you can easily show all rows by checking the Show all checkbox.
You will receive a warning asking if you'd like to see all rows. Since there's only 120 rows in this example, you can click OK. In larger datasets, loading all the rows will likely crash your browser.

Server: mysql:3306 > Database: tournament_teams > Table: teams

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 119 (120 total, Query took 0.0008 seconds.)

SELECT * FROM `teams`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: All Filter rows: Search this table Sort by key: None

+ Options

	team_no	team_code	creation_date
Edit	1	ACE	2021-01-01
Edit	2	FAX	2021-01-03
Edit	3	RED	2021-01-04
Edit	4	MAZ	2021-01-05
Edit	5	AMS	2021-01-06
Edit	6	ROT	2022-02-10
Edit	7	HAS	2021-01-08
Edit	8	BLU	2021-01-09
Edit	9	EIN	2021-01-10
Edit	10	TIL	2021-01-11
Edit	11	GRO	2021-01-12
Edit	12	BRE	2021-01-13
Edit	13	APE	2021-01-14
Edit	14	NIJ	2021-01-15
Edit	15	ENS	2021-01-16
Edit	16	HAR	2021-01-17
Edit	17	LAM	2021-01-18

We can now confirm that all 120 rows have been loaded.

On the left sidebar, select the tournament_teams database.

phpMyAdmin

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Designer

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
teams	Browse Structure Search Insert Empty Drop	120	InnoDB	utf8mb4_0900_ai_ci	64.0 Kib	-
	Sum	120	InnoDB	utf8mb4_0900_ai_ci	64.0 Kib	0 B

Print Data dictionary

Create table

Name: Number of columns: 4 Go

With this view, you can see all the tables in the database, including the teams table that we just created. As you can see, the size of the database is currently 64 Kib, that is, 64 kilobytes. While this is a small size, do remember that this would not be the case with larger datasets as this sample table only has 120 entries.

Keep this number in mind as we start optimizing this table.

Task B: Optimize Your Data Types

Under Action for the teams table, select Structure. This will bring you back to the Structure page.

Next to team_no, click Change. This function is helpful when you need to make adjustments to a column, for any reason.

Server: mysql:3306 > Database: tournament_teams > Table: teams

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Triggers](#)

[Table structure](#) [Relation view](#)

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	team_no	int		No	None				Change Drop More
2	team_code	char(100)	utf8mb4_0900_ai_ci	No	None				Change Drop More
3	creation_date	char(100)	utf8mb4_0900_ai_ci	No	None				Change Drop More

Check all With selected: [Browse](#) [Change](#) [Drop](#) [Primary](#) [Unique](#) [Index](#) [Fulltext](#)

2 In this editor, let's change the Type to one that would be better suited for this column.

[Hint \(Click Here\)](#)

[Solution \(Click Here\)](#)

If you were thinking TINYINT, you would be correct! As we know the maximum signed value of TINYINT is 127, this data type would work for our maximum number of 120 teams.

Please note, if you'd like to add an unsigned integer type, you can do that under the Attributes column. In this case, we only have 120 teams, so we are within the TINYINT maximum signed value range.

Your entry should now look like this:

Server: mysql:3306 > Database: tournament_teams > Table: teams

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Triggers](#)

Name	Type	Length/Values	Default	Collation	Attributes	Null	A_I	Comments	Virtuality	Move column
team_no	TINYINT		None							

[Structure](#) [Preview SQL](#) [Save](#)

3 Change the data type for team_code.

[Hint \(Click Here\)](#)

[Solution \(Click Here\)](#)

If you were thinking CHAR with a length of 3, you would be correct! Since we know for sure that every team code is exactly 3 letters, the fixed length CHAR data type would be the most suitable in this case.

Your entry should now look like this:

Server: mysql:3306 > Database: tournament_teams > Table: teams

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Triggers](#)

Name	Type	Length/Values	Default	Collation	Attributes	Null	A_I	Comments	Virtuality	Move column
team_code	CHAR	3	None	utf8mb4_0900_ai_ci						

[Structure](#) [Preview SQL](#) [Save](#)

4 Finally, let's change the creation_date.

[Hint \(Click Here\)](#)

[Solution \(Click Here\)](#)

If you were thinking DATE, you would be correct! As we know, DATE is optimized to contain date values, which is exactly what creation_date is.

Your entry should now look like this:

Server: mysql:3306 > Database: tournament_teams > Table: teams

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Triggers](#)

Name	Type	Length/Values	Default	Collation	Attributes	Null	A_I	Comments	Virtuality	Move column
creation_date	DATE		None	utf8mb4_0900_ai_ci						

[Structure](#) [Preview SQL](#) [Save](#)

5 After updating your data type, your Structure page should look like the following:

Server: mysql:3306 > Database: tournament_teams > Table: teams

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Triggers](#)

✓ Table teams has been altered successfully.

```
ALTER TABLE `teams` CHANGE `creation_date` `creation_date` DATE NOT NULL;
```

[\[Edit inline\]](#) [\[Edit\]](#) [\[Create PHP code\]](#)

[Table structure](#) [Relation view](#)

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	team_no	tinyint		No	None				Change Drop More
2	team_code	char(3)	utf8mb4_0900_ai_ci	No	None				Change Drop More
3	creation_date	date		No	None				Change Drop More

Check all With selected: [Browse](#) [Change](#) [Drop](#) [Primary](#) [Unique](#) [Index](#) [Fulltext](#)

As you can see, we still have 120 entries! Great! So, let's take a look at the size of this table.

[Returns to the tournament_teams database in the left sidebar](#)

Server: mysql:3306 > Database: tournament_teams

[Structure](#) [SQL](#) [Search](#) [Query](#) [Export](#) [Import](#) [Operations](#) [Privileges](#) [Routines](#) [Events](#) [Triggers](#) [Designer](#)

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
teams	Browse Structure Search Insert Empty Drop	120	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	–
1 table	Sum	120	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	0 B

Check all With selected: [▼](#)

Upon first glance, we can see the size of the database has reduced to 16 Kib, four times smaller than the original unoptimized database!

[8 We can take optimization a step further by selecting the teams table and, in the dropdown, selecting optimize table. This is equivalent to performing the OPTIMIZE TABLE function in the command-line interface. This command recognizes the table data and indexes to reduce storage and make accessing the table more efficient. The exact changes made to the table depend on the storage engine in use.](#)

With the InnoDB engine, which is what we are currently using, the table is actually rebuilt, with the index statistics updated to free unused space.

The screenshot shows the phpMyAdmin interface for the 'tournament_teams' database. The 'teams' table is selected. A context menu is open over the table name, with the 'Optimize table' option highlighted by a red box. Other options in the menu include Copy table, Show create, Export, Delete data or table, Empty, Drop, Table maintenance, Analyze table, Check table, Checksum table, Repair table, Add prefix to table, Replace table prefix, and Copy table with prefix.

Your SQL query has been executed successfully.

```
OPTIMIZE TABLE `teams`
```

[Edit inline] [Edit] [Create PHP code]

+ Options

Table	Op	Msg_type	Msg_text
tournament_teams.teams	optimize	note	Table does not support optimize, doing recreate + ...
tournament_teams.teams	optimize	status	OK

Query results operations

Print

Filters

Containing the word:

Action

Table	Action	Rows	Type	Collation	Size	Overhead
teams	<input type="checkbox"/> Browse <input type="checkbox"/> Structure <input type="checkbox"/> Search <input type="checkbox"/> Insert <input type="checkbox"/> Empty <input type="checkbox"/> Drop	120	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
1 table	<input type="checkbox"/> Sum	120	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	0 B

Check all

Name: Number of columns:

That is what the returned table tells us: InnoDB "optimizes" tables by recreating and analyzing them, and that the optimization had been completed.

Conclusion

Congratulations! You now know how to monitor and optimize your database with the help of the handy tool, phpMyAdmin. Now, you can apply what you have learned to any database that you create or modify in the future.

Author(s)

Kathy An

Other Contributor(s)

Rev. Alpna

Changelog

Date	Version	Changed by	Change Description
2021-10-12 1:08	1.0	Kathy An	Created initial version
2021-10-12 1:08	1.0	Kathy An	Added first section
2021-04-04 1:2	1.2	Rahul Jaiswal	Updated Markdown file

© IBM Corporation 2023. All rights reserved.