

Travaux pratiques : fonctions intégrées



Temps estimé nécessaire : 20 minutes

Dans SQL, vous pouvez accéder à de nombreuses fonctions intégrées qui peuvent être utilisées pour obtenir plus de variété dans notre analyse de données. Ces fonctions incluent des fonctions d'agrégation (comme MAX, MIN, SUM et AVG), des fonctions de chaîne (comme LENGTH, UCASE et LCASE), des fonctions scalaires (comme ROUND) et une variété de fonctions de date également. Dans cet onglet, vous pourrez vous entraîner à les utiliser toutes.

Logiciel utilisé dans ce laboratoire

Dans ce laboratoire, vous utiliserez [MySQL](#). MySQL est un système de gestion de base de données relationnelle (SGBDR) conçu pour stocker, manipuler et récupérer des données efficacement.



Pour réaliser ce laboratoire, vous utiliserez le service de base de données relationnelle MySQL disponible dans le cadre d'IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs est un environnement de laboratoire virtuel utilisé dans ce cours.

Objectifs

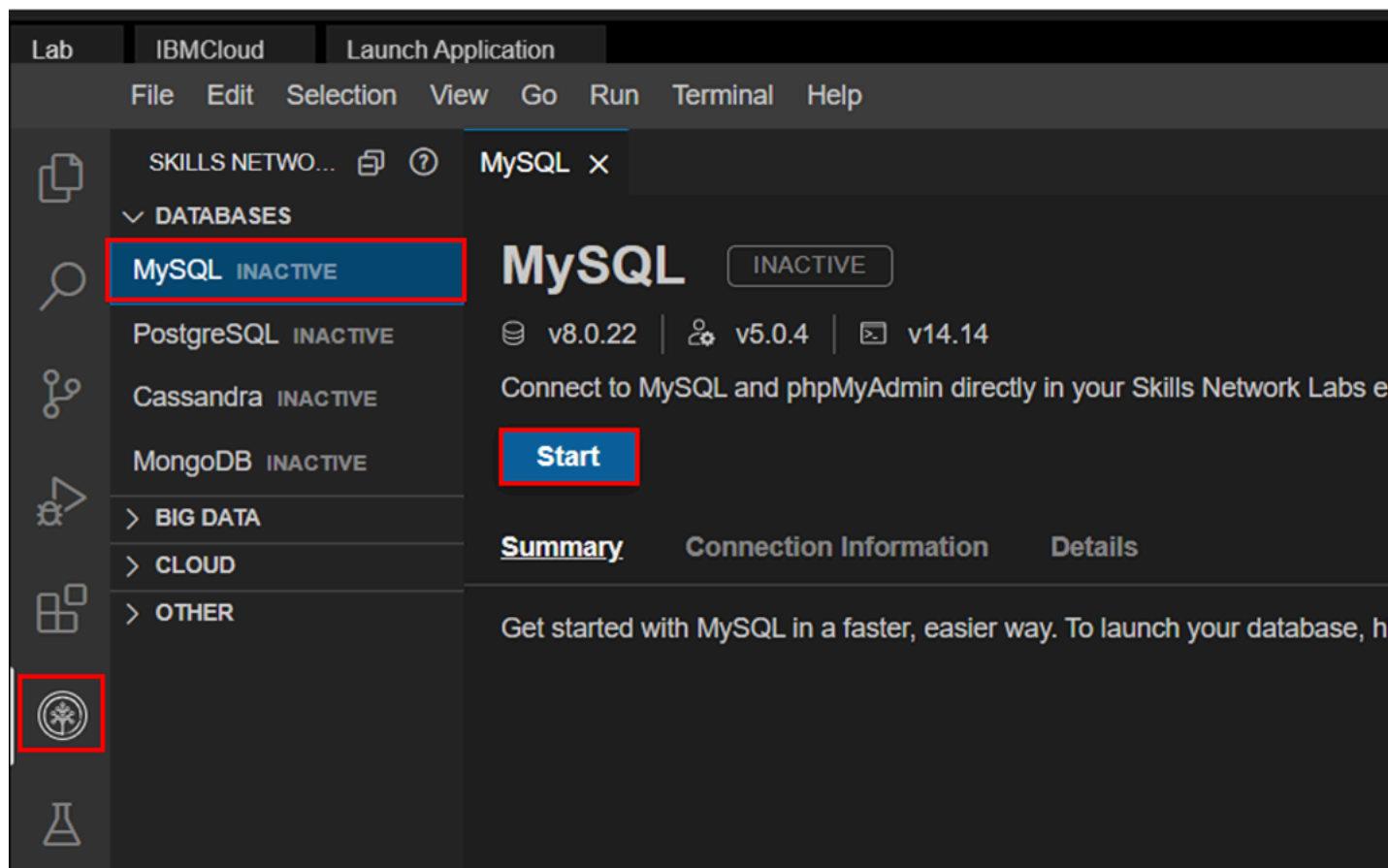
Après avoir terminé ce laboratoire, vous serez capable de composer des requêtes dans phpMyAdmin avec MySQL en utilisant :

- Fonctions d'agrégation
- Fonctions scalaires
- Fonctions de chaîne
- Fonctions de date

Créer la base de données

Cliquez sur **Skills Network Toolbox** dans le ruban de l'interface de programmation. Dans la section **Base de données**, cliquez sur **MySQL**.

Pour démarrer le moteur MySQL, cliquez sur **Démarrer**.



Une fois **MySQL** démarré, cliquez sur le bouton **phpMyAdmin** pour ouvrir **phpMyAdmin** dans la même fenêtre.

The screenshot shows the IBM Cloud Skills Network Labs interface. At the top, there are tabs for 'Lab', 'IBMCLOUD', and 'Launch Application'. Below these is a menu bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The main content area has a dark theme. On the left, there is a sidebar with icons for file management, search, and other tools. The main area displays the 'MySQL' status as 'ACTIVE' with a green button. Below this, it shows versions for MySQL (v8.0.22), phpMyAdmin (v5.0.4), and the lab environment (v14.14). A message states: 'Connect to MySQL and phpMyAdmin directly in your Skills Network Labs environment.' There is a blue 'Stop' button. Below this, there are tabs for 'Summary', 'Connection Information', and 'Details'. The 'Summary' tab is selected, showing a message: '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.' The login credentials are listed: 'Username: malikas' and 'Password: [REDACTED]'. Below this, it says 'You can manage MySQL via:' and there are two buttons: 'phpMyAdmin' (highlighted with a red box) and a button with an external link icon. At the bottom, it says 'Or to interact with the database in the terminal, select one of these options:' and there are two buttons: 'MySQL CLI' and 'New Terminal'.

Vous verrez l'outil GUI phpMyAdmin.

The screenshot shows the phpMyAdmin web interface. The browser's address bar displays the URL: `sandipsahajo-8080.theiadocker-27.proxy.cognitiveclas`. The phpMyAdmin logo is at the top left of the main content area. Below the logo are navigation icons for home, server status, help, search, settings, and a refresh button. There are two tabs: 'Recent' and 'Favorites'. The left sidebar contains a tree view of databases: 'New' (with a green plus icon), 'information_schema', 'mysql', 'performance_schema', 'sakila', and 'sys'. Each database entry has a plus icon to its left. The right pane shows the 'Server: mysql:3306' header. Below this are three tabs: 'Databases', 'SQL', and 'Status'. The 'General settings' section is active, showing 'Server connection collation: utf8mb4' and a 'More settings' link. The 'Appearance settings' section is also visible, showing 'Language: English' and 'Theme: pmahomme'.

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**.

Leave the default **utf8** encoding. UTF-8 is the most commonly used character encoding for content or data.

Databases

SQL

Status

User accounts

Export

Import

Settings

Binary log

Re

Databases

Create database

▼

Create

Database	Collation	Master replication	Action
<input type="checkbox"/> information_schema	utf8_general_ci	✔ Replicated	Check privileges
<input type="checkbox"/> mysql	utf8mb4_0900_ai_ci	✔ Replicated	Check privileges
<input type="checkbox"/> performance_schema	utf8mb4_0900_ai_ci	✔ Replicated	Check privileges
<input type="checkbox"/> sys	utf8mb4_0900_ai_ci	✔ Replicated	Check privileges
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**

Create the PETRESCUE table

Rather than create the table manually by typing the DDL commands in the SQL editor, you will execute a script containing the create table command.

Download the script file [PETRESCUE-CREATE.sql](#)

Note: To download, right-click on the link above and click on **Save As** or **Save Link As** depending on your browser. Remember to save the file as a .sql file and not HTML.

Next, load the .sql file to your database using the **Import** option as shown in the image below.

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: PETRESCUE-CREATE.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. (The script will stop importing when it reaches the timeout limit.)

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

Other options:

☒ Enable foreign key checks

Format:

PETRESCUE-CREAT....sql HR_Database_Crea....sql

Upon execution, the table PETRESCUE will be created in the Mysql_Learners database and loaded with a set of values as well. The attributes of the PETRESCUE table are:

Column Name	Data Type	Description
ID	INTEGER	ID of the entry
ANIMAL	VARCHAR(20)	Type of animal
QUANTITY	INTEGER	Number of animals
COST	DECIMAL(6,2)	Cost incurred
RESCUEDATE	DATE	Date of Rescue

Once the table is loaded, you may open the sql editor to start executing the queries.

Aggregation Functions

1. Write a query that calculates the total cost of all animal rescues in the PETRESCUE table.

For this query, we will use the function `SUM(COLUMN_NAME)`. The output of this query will be the total value of all elements in the column. The query for this question can be written as:

1. 1

```
1. SELECT SUM(COST) FROM PETRESCUE;
```

You can further assign a label to the query `SUM_OF_COST`.

1. 1

```
1. SELECT SUM(COST) AS SUM_OF_COST FROM PETRESCUE;
```

2. Write a query that displays the maximum quantity of animals rescued (of any kind).

For this query, we will use the function `MAX(COLUMN_NAME)`. The output of this query will be the maximum value of all elements in the column. The query for this question can be written as:

1. 1

```
1. SELECT MAX(QUANTITY) FROM PETRESCUE;
```

Copied!

The query can easily be changed to display the minimum quantity using the MIN function instead.

1. 1

```
1. SELECT MIN(QUANTITY) FROM PETRESCUE;
```

Copied!

3. Write a query that displays the average cost of animals rescued.

For this query, we will use `AVG(COLUMN_NAME)`. The output of this query will be the average of all elements in the column. The query for this question can be written as

1. 1

```
1. SELECT AVG(COST) FROM PETRESCUE;
```

Copied!

Fonctions scalaires et fonctions de chaîne

1. Écrivez une requête qui affiche le coût intégral arrondi de chaque sauvetage.

Pour cette requête, nous utiliserons la fonction `ROUND(COLUMN_NAME, NUMBER_OF_DECIMALS)`. La sortie de cette requête sera la valeur de chaque élément de la colonne arrondie au nombre de décimales spécifié. Notez que le deuxième argument est facultatif et, s'il est omis, il aboutit à un arrondi à une valeur entière. La requête pour cette question peut être écrite comme suit :

1. 1

```
1. SELECT ROUND(COST) FROM PETRESCUE;
```

Copié!

La requête pourrait également être écrite comme suit :

1. 1

```
1. SELECT ROUND(COST, 0) FROM PETRESCUE;
```

Copié!

Si la question consistait à arrondir la valeur à 2 décimales, la requête deviendrait :

1. 1

```
1. SELECT ROUND(COST, 2) FROM PETRESCUE;
```

Copié!

2. Écrivez une requête qui affiche la longueur de chaque nom d'animal.

Pour cette requête, nous utiliserons la fonction `LENGTH(COLUMN_NAME)`. Le résultat de cette requête sera la longueur de chaque élément de la colonne. La requête pour cette question peut être écrite comme suit :

1. 1

```
1. SELECT LENGTH(ANIMAL) FROM PETRESCUE;
```

Copié!

3. Écrivez une requête qui affiche le nom de l'animal de chaque sauvetage en majuscules.

Pour cette requête, nous utiliserons la fonction `UCASE(COLUMN_NAME)`. Le résultat de cette requête sera chaque élément de la colonne en majuscules. La requête pour cette question peut être écrite comme suit :

1. 1

```
1. SELECT UCASE(ANIMAL) FROM PETRESCUE;
```

Copié!

Tout aussi facilement, l'utilisateur pourrait demander une représentation en minuscules, et la requête serait modifiée en :

1. 1

```
1. SELECT LCASE(ANIMAL) FROM PETRESCUE;
```

Copié!

Date Functions

1. Write a query that displays the rescue date.

For this query, we will use the function `DAY(COLUMN_NAME)`. The output of this query will be only the DAY part of the date in the column. The query for this question can be written as:

1. 1

```
1. SELECT DAY(RESCUEDATE) FROM PETRESCUE;
```

Copied!

In case the query was asking for MONTH of rescue, the query would change to:

```
1. 1
```

```
1. SELECT MONTH(RESCUEDATE) FROM PETRESCUE;
```

Copied!

In case the query was asking for YEAR of rescue, the query would change to:

```
1. 1
```

```
1. SELECT YEAR(RESCUEDATE) FROM PETRESCUE;
```

Copied!

2. Animals rescued should see the vet within three days of arrival. Write a query that displays the third day of each rescue.

For this query, we will use the function `DATE_ADD(COLUMN_NAME, INTERVAL Number Date_element)`. Here, the quantity in `Number` and in `Date_element` will combine to form the interval to be added to the date in the column. For the given question, the query would be:

```
1. 1
```

```
1. SELECT DATE_ADD(RESCUEDATE, INTERVAL 3 DAY) FROM PETRESCUE
```

Copied!

If the question was to add 2 months to the date, the query would change to:

```
1. 1
```

```
1. SELECT DATE_ADD(RESCUEDATE, INTERVAL 2 MONTH) FROM PETRESCUE
```

Copied!

Similarly, we can retrieve a date before the one given in the column by a given number using the function `DATE_SUB`. By modifying the same example, the following query would provide the date 3 days before the rescue.

```
1. 1
```

```
1. SELECT DATE_SUB(RESCUEDATE, INTERVAL 3 DAY) FROM PETRESCUE
```

Copied!

3. Write a query that displays the length of time the animals have been rescued, for example, the difference between the current date and the rescue date.

For this query, we will use the function `DATEDIFF(Date_1, Date_2)`. This function calculates the difference between the two given dates and gives the output in number of days. For the given question, the query would be:

```
1. 1
```

```
1. SELECT DATEDIFF(CURRENT_DATE, RESCUEDATE) FROM PETRESCUE
```

Copied!

`CURRENT_DATE` is also an inbuilt function that returns the present date as known to the server.

To present the output in a `YYYY-MM-DD` format, another function `FROM_DAYS(number_of_days)` can be used. This function takes a number of days and returns the required formatted output. The query above would thus be modified to

```
1. 1
```

```
1. SELECT FROM_DAYS(DATEDIFF(CURRENT_DATE, RESCUEDATE)) FROM PETRESCUE
```

Copied!

Practice Problems

1. Write a query that displays the average cost of rescuing a single dog. Note that the cost per dog would not be the same in different instances.

- ▶ [Click here for a hint](#)
- ▶ [Click here for the solution](#)

2. Write a query that displays the animal name in each rescue in uppercase without duplications.

- ▶ [Click here for a hint](#)
- ▶ [Click here for the solution](#)

3. Write a query that displays all the columns from the `PETRESCUE` table where the animal(s) rescued are cats. Use **cat** in lowercase in the query.

- ▶ [Click here for a hint](#)
- ▶ [Click here for the solution](#)

4. Write a query that displays the number of rescues in the 5th month.

- ▶ [Click here for a hint](#)
- ▶ [Click here for the solution](#)

5. The rescue shelter is supposed to find good homes for all animals within 1 year of their rescue. Write a query that displays the ID and the target date.

- ▶ [Click here for a hint](#)
- ▶ [Click here for Solution](#)

Conclusion

Congratulations on completing this lab.

You are now able to:

- Use aggregation functions to calculate total, maximum, minimum, and average values of numerical attributes.
- Use scalar functions to round a floating value to the desired number of decimal places.
- Use string functions to convert text into upper or lower cases.
- Use date operations to manipulate data columns with the attribute as date.

Author(s)

[Lakshmi Holla](#)

[Malika Singla](#)

[Abhishek Gagneja](#)

© IBM Corporation 2023. All rights reserved.