

Organizing Data

Scenario

The database operations team has created a relational database named **world** containing three tables: **city**, **country**, and **countrylanguage**. You help write a few queries to group records for analysis by using both the **GROUP BY** and **OVER** clauses.

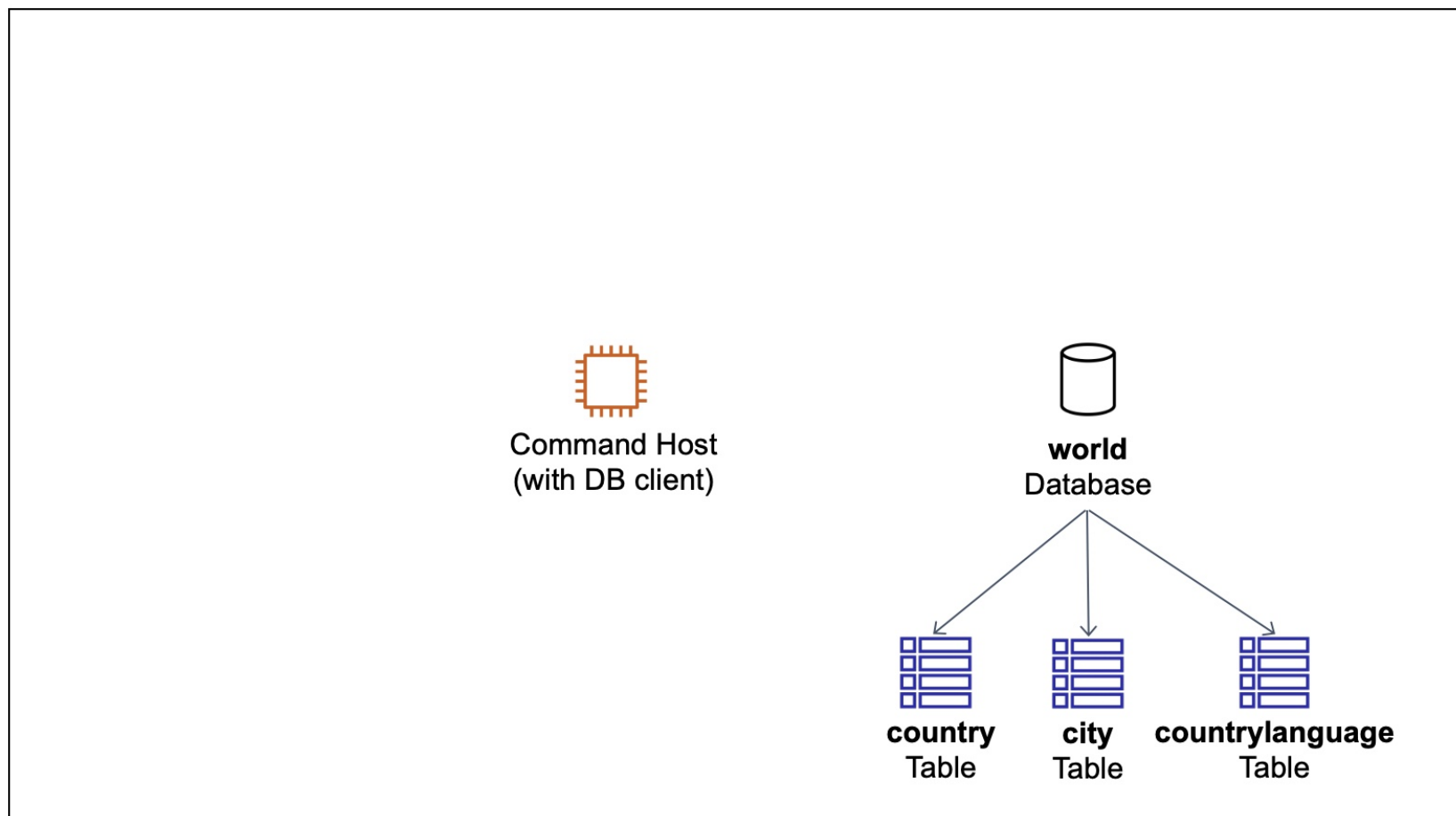
Lab overview and objectives

This lab demonstrates how to use some common database functions with the **GROUP BY** and **OVER** clauses.

After completing this lab, you should be able to:

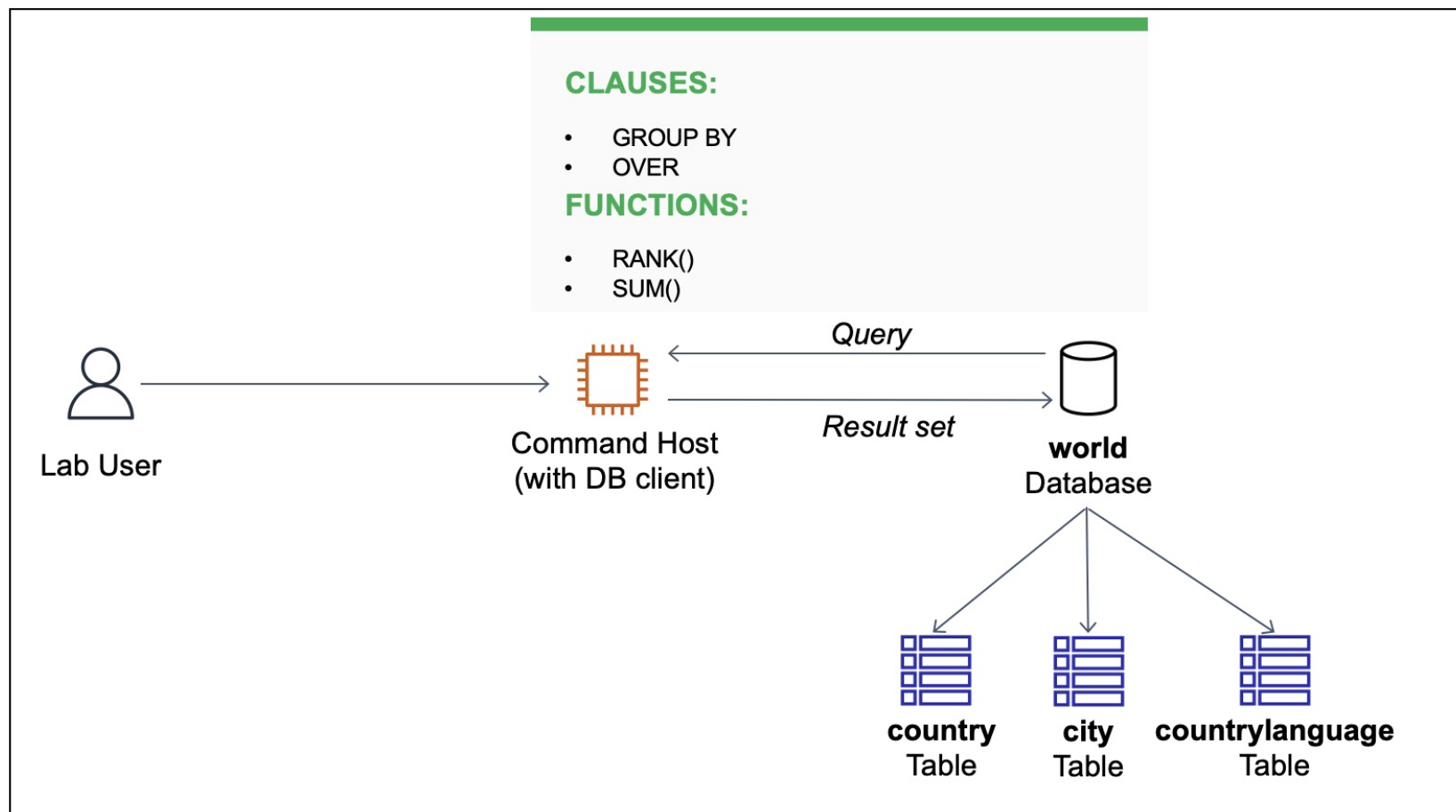
- Use the **GROUP BY** clause with the aggregate function **SUM()**
- Use the **OVER** clause with the **RANK()** window function
- Use the **OVER** clause with the aggregate function **SUM()** and the **RANK()** window function

When you start the lab, the following resources are already created for you:



A Command Host instance and world database containing three tables

At the end of this lab, you would have used both the **GROUP BY** and **OVER** clauses with some common database operators:



A lab user is connected to a database instance. It also displays some commonly used SQL clauses and database functions.

Sample data in this course is taken from Statistics Finland, general regional statistics, February 4, 2022.

Duration

This lab requires approximately **45 minutes** to complete.

AWS service restrictions




In this lab environment, access to AWS services and service actions might be restricted to the ones that you need to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that this lab describes.

Accessing the AWS Management Console

1. At the upper-right corner of these instructions, choose ▶ **Start Lab**

Troubleshooting tip: If you get an **Access Denied** error, close the error box, and choose ▶ **Start Lab** again.

2. The following information indicates the lab status:

- A red circle next to **AWS**  at the upper-left corner of this page indicates that the lab has not been started.
- A yellow circle next to **AWS**  at the upper-left corner of this page indicates that the lab is starting.
- A green circle next to **AWS**  at the upper-left corner of this page indicates that the lab is ready.

Wait for the lab to be ready before proceeding.

3. At the top of these instructions, choose the green circle next to **AWS** 

This option opens the AWS Management Console in a new browser tab. The system automatically sign you in.

Tip: If a new browser tab does not open, a banner or icon at the top of your browser will indicate that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop-ups**.

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you should be able to see both browser tabs at the same time so that you can follow the lab steps.

⚠ Do not change the lab Region unless specifically instructed to do so.

Task 1: Connect to the Command Host

In this task, you connect to an instance containing a database client, which is used to connect to a database. This instance is referred to as the Command Host.

5. In the AWS Management Console, choose the  **Services** menu. Under **Compute**, choose **EC2**.

6. In the navigation pane, choose **Instances**.

7. Next to the instance labelled **Command Host**, select the check box ☒ and then choose **Connect**.

Note: If you do not see the **Command Host**, the lab is possibly still being provisioned, or you may be using another Region.

8. For **Connect to instance**, choose the **Session Manager** tab.

9. Choose **Connect** to open a terminal window.

Note: If the **Connect** button is not available, wait for a few minutes and try again.

10. To configure the terminal to access all required tools and resources, run the following command:

```
sudo su
cd /home/ec2-user/
```

i Tips:

- Copy and paste the command into the Session Manager terminal window.
- If you are using a Windows system, press Shift+Ctrl+v to paste the command.

11. To connect to the database server, run the following command in the terminal. A password was configured when the database was installed.

```
mysql -u root --password='re:St@rt!9'
```

i Tip: At any stage of the lab, if the Sessions Manager window is not responsive or if you need to reconnect to the database instance, then follow these steps:

- Close the Sessions Manager window, and try to reconnect using the previous steps.
- Run the following commands in the terminal.

```
sudo su  
cd /home/ec2-user/  
mysql -u root --password='re:St@rt!9'
```

Task 2: Query the world database

In this task, you query the **world** database using various **SELECT** statements and database functions.

12. To show the existing databases, enter the following command in the terminal.

```
SHOW DATABASES;
```

Verify that a database named **world** is available. If the **world** database is not available, contact your instructor.

13. To review the table schema, data, and number of rows in the **country** table, enter the following query.

```
SELECT * FROM world.country;
```

14. To return a list of records where the **Region** is Australia and New Zealand, run the following query. This query includes an **ORDER BY** clause (which a previous lab introduced) that arranges the results by **Population** in descending order.

```
SELECT Region, Name, Population FROM world.country WHERE Region = 'Australia and New Zealand' ORDER BY Population desc;
```

15. You can use the **GROUP BY** clause to group related records together. The following example starts by filtering records using a condition where the region is equal to Australia and New Zealand. The results are then grouped together by using a **GROUP BY** clause. The **SUM()** function is then applied to the grouped results to generate a total population for that region. Run the following query in your terminal.

```
SELECT Region, SUM(Population) FROM world.country WHERE Region = 'Australia and New Zealand' GROUP BY Region ORDER BY SUM(Population) desc;
```

This query returns a **SUM()** of the **Population** for the Australia and New Zealand region. Because the **WHERE** clause is filtered by **Region**, only the Australia and New Zealand records are aggregated.

16. The following example uses a windowing function to generate a running total by adding the **Population** of the first record to the **Population** of the second record and subsequent records. This query uses the **OVER()** clause to group the records by **Region** and uses the **SUM()** function to aggregate the records. The output displays the population of a country along side a running total of the region. Run the following query in your terminal.

```
SELECT Region, Name, Population, SUM(Population) OVER(partition by Region ORDER BY Population) as 'Running Total' FROM world.country WHERE Region = 'Australia and New Zealand';
```

17. The following query groups the records by **Region** and orders them by **Population** with the **OVER()** clause. This query also includes the **RANK()** function to generate a rank number indicating the position of each record in the result set. The **RANK()** function is useful when dealing with large groups of records. Run the following query in your terminal.

```
SELECT Region, Name, Population, SUM(Population) OVER(partition by Region ORDER BY Population) as 'Running Total', RANK() over(partition by region ORDER BY population) as 'Ranked' FROM world.country WHERE region = 'Australia and New Zealand';
```

Challenge

Write a query to rank the countries in each region by their population from largest to smallest.

- ▼ You have to determine whether to use either the **GROUP BY** or **OVER** grouping clause and either the **SUM()** or **RANK()** function.

```
SELECT Region, Name, Population, RANK() OVER(partition by Region ORDER BY Population desc) as 'Ranked' FROM world.country order by Region, Ranked;
```

Tip: Expand the question to reveal the solution.

Conclusion

👏 Congratulations! You have now successfully:

- Used the **GROUP BY** clause with the aggregate function **SUM()**
- Used the **OVER** clause with the **RANK()** window function
- Used the **OVER** clause with the aggregate function **SUM()** and the **RANK()** window function

Lab complete 🎓

18. Choose **■ End Lab** at the top of this page, and then select **Yes** to confirm that you want to end the lab.
19. An **Ended AWS Lab Successfully** message is briefly displayed indicating that the lab has ended.

Additional resources

- Country, city, and language data, Statistics Finland: The material was downloaded from Statistics Finland's interface service on February 4, 2022, with the license [CC BY 4.0](#). The original data source is available from [Statistics Finland](#).
- For more information about database functions and operators, see the following resources:
 - [GROUP By clause](#)
 - [OVER clause](#)
 - [SUM function](#)
 - [RANK function](#)
 - [SELECT statements](#)
 - [COUNT function](#)

For more information about AWS Training and Certification, see [AWS Training and Certification](#).

Your feedback is welcome and appreciated.

If you would like to share any suggestions or corrections, please provide the details in our [AWS Training and Certification Contact Form](#).

© 2022 Amazon Web Services, Inc. and its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.