**Running Your First SQL Query**

In this section, we'll run our first **SQL query** based on a **common security analyst task**: determining which computer has been assigned to a specific employee.

**Understanding the Employees Table**

The **employees table** contains multiple columns, but we’re focusing on these two: **employee\_id** (unique ID for each employee) and **device\_id** (ID of the assigned computer).

**Basic SQL Query Syntax**

SQL queries use two essential keywords: **SELECT** (specifies which columns to retrieve) and **FROM** (specifies the table to query).

To retrieve employee and device IDs, use:  
**SELECT employee\_id, device\_id FROM employees;**

* **SELECT employee\_id, device\_id** → Retrieves only these two columns.
* **FROM employees** → Specifies that the data comes from the **employees** table.
* **;** → Marks the end of the statement.

**SQL is case-insensitive**, but keywords are typically written in **uppercase** for readability.

**Retrieving All Columns with SELECT \***

If we need more details, such as **department, username, or office location**, we can retrieve **all columns** using:

**SELECT \* FROM employees;**

The **\*** wildcard tells SQL to **return every column** from the table.

**Next Steps: Adding Filters**

Now that we can retrieve data, the next step is to **filter results** to find specific employees or devices efficiently. Let’s dive into that next!

# Query a database

In this reading, you’ll review those basic SQL queries and learn a new keyword that will help you organize your output. You'll also learn about the Chinook database, which this course uses for queries in readings and quizzes.

## Basic SQL query

There are two essential keywords in any SQL query: SELECT and FROM. You will use these keywords every time you want to query a SQL database. Using them together helps SQL identify what data you need from a database and the table you are returning it from.

The video demonstrated this SQL query:

SELECT employee\_id, device\_id

FROM employees;

In readings and quizzes, this course uses a sample database called the Chinook database to run queries. The Chinook database includes data that might be created at a digital media company. A security analyst employed by this company might need to query this data.  For example, the database contains eleven tables, including an employees table, a customers table, and an invoices table. These tables include data such as names and addresses.

As an example, you can run this query to return data from the customers table of the Chinook database:



### ****SELECT****

The SELECT keyword indicates which columns to return. For example, you can return the customerid column from the Chinook database with

SELECT customerid

You can also select multiple columns by separating them with a comma. For example, if you want to return both the customerid and city columns, you should write SELECT customerid, city.

If you want to return all columns in a table, you can follow the SELECT keyword with an asterisk (\*). The first line in the query will be SELECT \*.

**Note:** Although the tables you're querying in this course are relatively small, using SELECT \* may not be advisable when working with large databases and tables; in those cases, the final output may be difficult to understand and might be slow to run.

### ****FROM****

The SELECT keyword always comes with the FROM keyword. FROM indicates which table to query. To use the FROM keyword, you should write it after the SELECT keyword, often on a new line, and follow it with the name of the table you’re querying. If you want to return all columns from the customers table, you can write:

SELECT \*

FROM customers;

When you want to end the query here, you put a semicolon (;) at the end to tell SQL that this is the entire query.

**Note:** Line breaks are not necessary in SQL queries, but are often used to make the query easier to understand. If you prefer, you can also write the previous query on one line as

SELECT \* FROM customers;

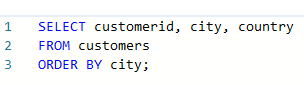
## ORDER BY

Database tables are often very complicated, and this is where other SQL keywords come in handy. ORDER BY is an important keyword for organizing the data you extract from a table.

ORDER BY sequences the records returned by a query based on a specified column or columns. This can be in either ascending or descending order.

### ****Sorting in ascending order****

To use the ORDER BY keyword, write it at the end of the query and specify a column to base the sort on. In this example, SQL will return the customerid, city, and country columns from the customers table, and the records will be sequenced by the city column:



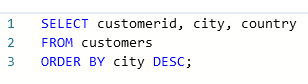
The ORDER BY keyword sorts the records based on the column specified after this keyword. By default, as shown in this example, the sequence will be in ascending order. This means

* if you choose a column containing numeric data, it sorts the output from the smallest to largest. For example, if sorting on customerid, the ID numbers are sorted from smallest to largest.
* if the column contains alphabetic characters, such as in the example with the city column, it orders the records from the beginning of the alphabet to the end.

### ****Sorting in descending order****

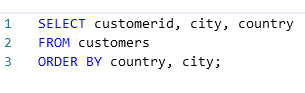
You can also use the ORDER BY with the DESC keyword to sort in descending order. The DESC keyword is short for "descending" and tells SQL to sort numbers from largest to smallest, or alphabetically from Z to A. This can be done by following ORDER BY with the DESC keyword. For example, you can run this query to examine how the results differ when DESC is applied:

Now, cities at the end of the alphabet are listed first.



### ****Sorting based on multiple columns****

You can also choose multiple columns to order by. For example, you might first choose the country and then the city column. SQL then sorts the output by country, and for rows with the same country, it sorts them based on city. You can run this to explore how SQL displays this:



## Key takeaways

SELECT and FROM are important keywords in SQL queries. You use SELECT to indicate which columns to return and FROM to indicate which table to query. You can also include ORDER BY in your query to organize the output. These foundational SQL skills will support you as you move into more advanced queries.