Sololearn SQL introduction

Lun-Hsien Chang

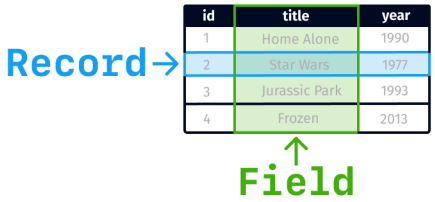
2024-12-24

Introduction to SQL

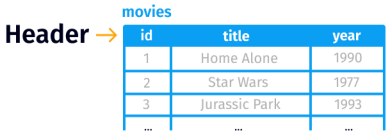
Getting Started with SQL

Working with Data

SQL code is used to send requests to a database. These requests are known as **queries**.



Most databases collect data in **tables**. Tables organize data in records and fields. Rows are records. Columns are fields



The **header** row is the top row of a table and includes the names for the different fields (columns).

The SELECT command is used to extract field data from a table. Create a query to extract the title field from the movies table 'SELECT title FROM movies'

Running SQL Queries

Data that can be stored in tables is called **structured data**.

Unstructured data is information that is difficult to store in tables.

Match the data with its category

sales table: structured

audio file: unstructured

SQL stands for **Structured Query Language**.

With SQL you'll be able to extract data from massive datasets with thousands of fields and records.

SQL is used to work with structured data in the form of tables

Relational Databases

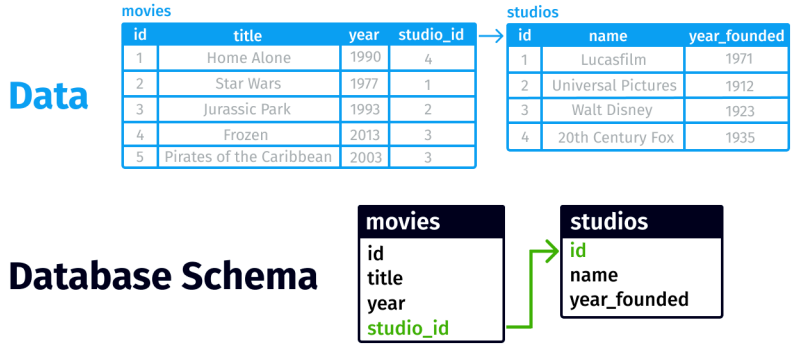
```
starting httpd help server ... done
```

The **relational database** is the most common type of database. The image shows a relational database.

The different tables in a relational database connect to each other using fields (columns) with values in common. These fields are called **keys**.

Debugging

Databases won't understand your queries if your SQL code contains mistakes and they will return an error message. Errors in computer code are known as bugs and you'll learn to identify and fix errors in this lesson.



Knowing how data is organized in the database will help you reduce errors.

A **schema** is a visual representation of how a database is organized, showing its tables, fields and keys. Arrows are used to show how the different tables are related.

The * symbol allows you to select **all the fields** in a table. This way you can avoid typos when listing field names.

Module 1 Quiz

The term for a request of information from a database is **query**

Match the term with the definition

- Error in code: bug
- Information request: query
- Visual representation of a database: schema

A relational database stores information in **tables**

What connects different tables in a relational database? **Common key fields**

Explain what the SQL code does 'SELECT name, age FROM users' **Extracts name and age fields from the users table**

Going deeper with SQL

Standards & Best Practices

For better collaboration, it's important to write code that is also easy for others to understand. In this lesson, you'll explore some of the best practices that will help you write efficient, readable SQL queries.

To explain their SQL queries, data professionals use **comments.** Complete the code with **double hyphens** (–) to add a comment

'– extract movie titles'

```
SELECT title FROM movies
```

If you need to make comments with multiple lines, you can use **/* … */ block comments**.

Data professionals use uppercase for the SQL command words and lowercase for tables and fields names. This makes the SQL code easier to read and consistent across projects. Code a query that follows the best practices

'– Extract all fields'

```
SELECT * FROM movies
```

Select all the best practices- Lower case for field and table names, Uppercase for command words

Sorting Data

Sorting consists in putting data in a meaningful order

The **ORDER BY** command is used to sort the extracted data in the results table.

By default, data is sorted in ascending order. You can use the explicit ASC keyword to clarify and make your queries more readable, particularly when writing complex queries. Complete the query and make it explicit for the reader that the result table will be sorted in ascending order

```
SELECT * FROM customers ORDER BY age ASC
```

Your extracted data will be sorted in ascending order by default. If you need the data sorted in descending order (from largest to smallest) you need to add the **DESC** keyword. Complete the query to sort data in descending order

```
SELECT * FROM customers ORDER BY age DESC
```

Limiting Data

Breaking large sets of data into smaller parts speeds up loading times and helps show people only what they need. In this lesson, you will learn how to control the number of records extracted from a database.

The **LIMIT** keyword extracts a limited number of records. Complete the SQL query to extract the first 10 records.

```
SELECT * FROM employees LIMIT 10
```

LIMIT is usually combined with ORDER BY to extract the records that rank the highest/lowest. Complete to select the 10 most expensive products

```
SELECT * FROM products ORDER BY price DESC LIMIT 10
```

The OFFSET parameter is used with LIMIT to skip a number of records. `OFFSET 2` will skip the first 2 rows and start extracting from the 3rd record

```
SELECT title FROM movies LIMIT 3 OFFSET 2
```

Data Types

Data comes in different shapes and forms. Computers treat different types of data in different ways. In this lesson, you'll start working with different types of data.

When data comes in the form of numbers you can do calculations with the values. You can include calculations in the results table. You can give columns a temporary name (or alias) using the **AS** keyword

'SELECT name, price+delivery AS TOTAL FROM sales'

```
SELECT name, price+delivery AS TOTAL FROM sales
```

How many columns will the resulting table contain?

```
SELECT name, price, price+delivery FROM sales
```

The column name for `price + delivery` will depend on the SQL database system you are using. In **MySQL**, **SQLite,** the column will be unnamed and may appear as something like `price+delivery` unless explicitly aliased

A piece of text data is technically called a string. An operation that you can do with strings is concatenation. Concatenation joins strings together. Run the code to see the results table.

```
SELECT CONCAT(first_name, last_name) AS full_name FROM employees
```

Filtering Data

Databases can contain large amounts of data. Most likely, you don't want to see everything at the same time. In this lesson, you'll learn to filter data that meets a condition.

WHERE is used to extract records that meet a condition. Complete the query to extract the studio records with name='Walt Disney'

```
SELECT * FROM studio WHERE name = 'Walt Disney
```

Module 2 Quiz

Query Techniques

Pattern Matching

You can automate manual and tedious tasks with code. This can be particularly helpful in fields like marketing and finance. In this lesson, you'll learn how to find specific patterns in data to speed up tasks and free up time.

The LIKE keyword is used with the WHERE command to search for patterns in string values. The **% symbol** can replace any number of characters (one, multiple or none) in a string to create patterns. Complete the query to extract all the Avengers comics

```
SELECT * FROM comics WHERE title LIKE 'The Avengers%'
```

The % special symbol is known as a wildcard and is used to create patterns. You can use patterns to extract email addresses that share the same domain.

The % wildcard can be used in any part of the pattern and as many times as needed. Extract all the titles that contain the word Avengers:

```
SELECT * FROM comics WHERE title LIKE '%Avengers%
```

Advanced Pattern Matching

In this lesson, you'll learn various techniques to enhance your pattern matching skills and make your queries more effective.

The **underscore symbol _** is another wildcard and represents 1 single character only.

Match the wildcard symbol with the number of characters it represents

- any number of characters: '%'
- 1 single character: '_'

You can include multiple queries in your SQL code. You just need to separate them with a **semicolon(;)**

```
SELECT name, code FROM products;

SELECT name FROM products WHERE code LIKE 'A_B_';
```

Patterns are case-sensitive. `LOWER()` and `UPPER()` commands are used to convert strings to lower or upper-case.

SQL Conditions

You can use conditions to write more precise queries and extract only the data you need. In this lesson you'll learn how.

You are already familiar with the >, < and = comparison operators. There are more comparison operators you can use to make your queries more precise.

- greater than or equal to: `>=`
- less than or equal to: `<=`
- not equal to: '<>'

When running a filtering query. Only those records that make the condition true will be extracted. The code below will result in 2 tables. The first one helps us to see which records meet the condition. The result column contains t or f. The second one is the result of the filtering query.

```
/*The result column contains t or f*/
SELECT year, year > 2000 AS result FROM movies;

/*The second one is the result of the filtering query*/
SELECT title FROM movies WHERE year > 2000;
```

Data Aggregation

We often make better decisions using data from multiple different sources. In this lesson, you'll learn to summarize data with aggregation operations.

MAX() is an example of an aggregation operation. What's the maximum value for the year field?

The result of an aggregation operation is a single numerical value. **MIN()** is another example of an aggregation operation. What's the result of this operation?

COUNT() is another aggregation operation. It counts the number of records

SUM() is another aggregation operation. It produces the total sum of the values in a numerical field.

AVG() is another example of an aggregation operation. It calculates the average value

Mixing Things Up

The combination of aggregation and filtering is frequently used in the analysis of data. In this lesson, you'll learn to run SQL queries that summarize the data that meets a condition.

When running queries that involve different operations, filtering happens first. What's the result of the following query?

```
SELECT MIN(year) FROM movies WHERE year > 2000
```

Module 3 Quiz

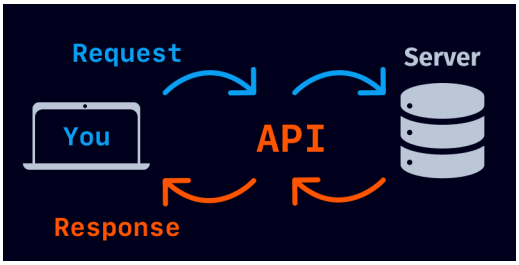
Data Analysis

Analyzing Data

Raw (or untreated) data needs to be processed and analyzed so it can be used to make decisions. This process involves different steps. In this lesson, you'll learn about the first steps in the data analysis process, starting with questions.

The number of records changes as new data is added to the database. The schema doesn't show the number of records in the tables.

Once your question has been defined, the next step is usually collecting data to answer it.



A lot of data lives on the Internet. Servers are computers that are always listening for requests of information. You can request information from a database server through an API (Application Programming Interface). There are APIs to request data from Facebook, Youtube, Wikipedia and pretty much every other platform.

Grouping

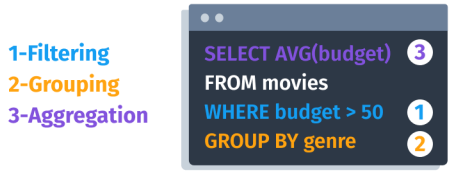
Data grouping is a powerful tool when working with large datasets. Grouping allows you to collect and see data in a new way, to answer more complex questions. In this lesson, you'll learn to group information to see patterns and relationships.

GROUP BY allows you to organize similar data into categories. It's combined with aggregations to compute key metrics for a group of records.

```
SELECT genre, AVG(budget)
FROM movies
GROUP BY genre;
```

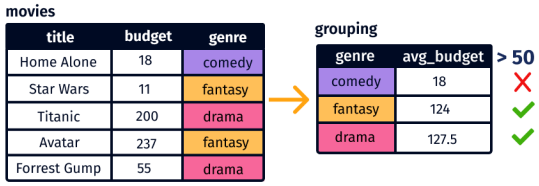
Grouping and Filtering

In this lesson, you'll learn to combine grouping and filtering in the same query! This will let you make better decisions from your data.



You can combine **GROUP BY** with **WHERE** filters. Data is filtered first, then grouped.

'SELECT genre, AVG(budget) FROM movies WHERE budget > 50 GROUP BY genre;'



HAVING allows you to filter data that has been grouped. The following code will group first, then filter

```
SELECT genre, AVG(budget)
FROM movies
GROUP BY genre
HAVING AVG(budget) > 50;
```

Complete the query to identify the authors that have published more than 3 books

```sql
SELECT author, COUNT(*)
FROM books
GROUP BY author
HAVING COUNT(*) > 3;
```

When a query contains both GROUP BY and HAVING data is grouped first, then filtered

Complete the query to calculate average salary per department first, then show departments with an average greater than 5000

```sql
SELECT department, AVG(salary)
FROM employees
GROUP BY department
HAVING AVG(salary)> 5000;
```

Why `HAVING` Cannot Refer to an Alias in `SELECT`?

Consider the following SQL query:

```sql
SELECT
    department,
    COUNT(employee_id) AS total_employees
FROM employees
GROUP BY department
HAVING total_employees > 10;
```

This query will result in an error because `HAVING` cannot directly reference the alias `total_employees`. The `HAVING` clause is processed before the `SELECT` clause, so SQL doesn't recognize `total_employees` when it is used in `HAVING`.

To resolve this, you should use the aggregation expression directly in the `HAVING` clause, like this:

```sql
SELECT
    department,
    COUNT(employee_id) AS total_employees
FROM employees
GROUP BY department
HAVING COUNT(employee_id) > 10;
```

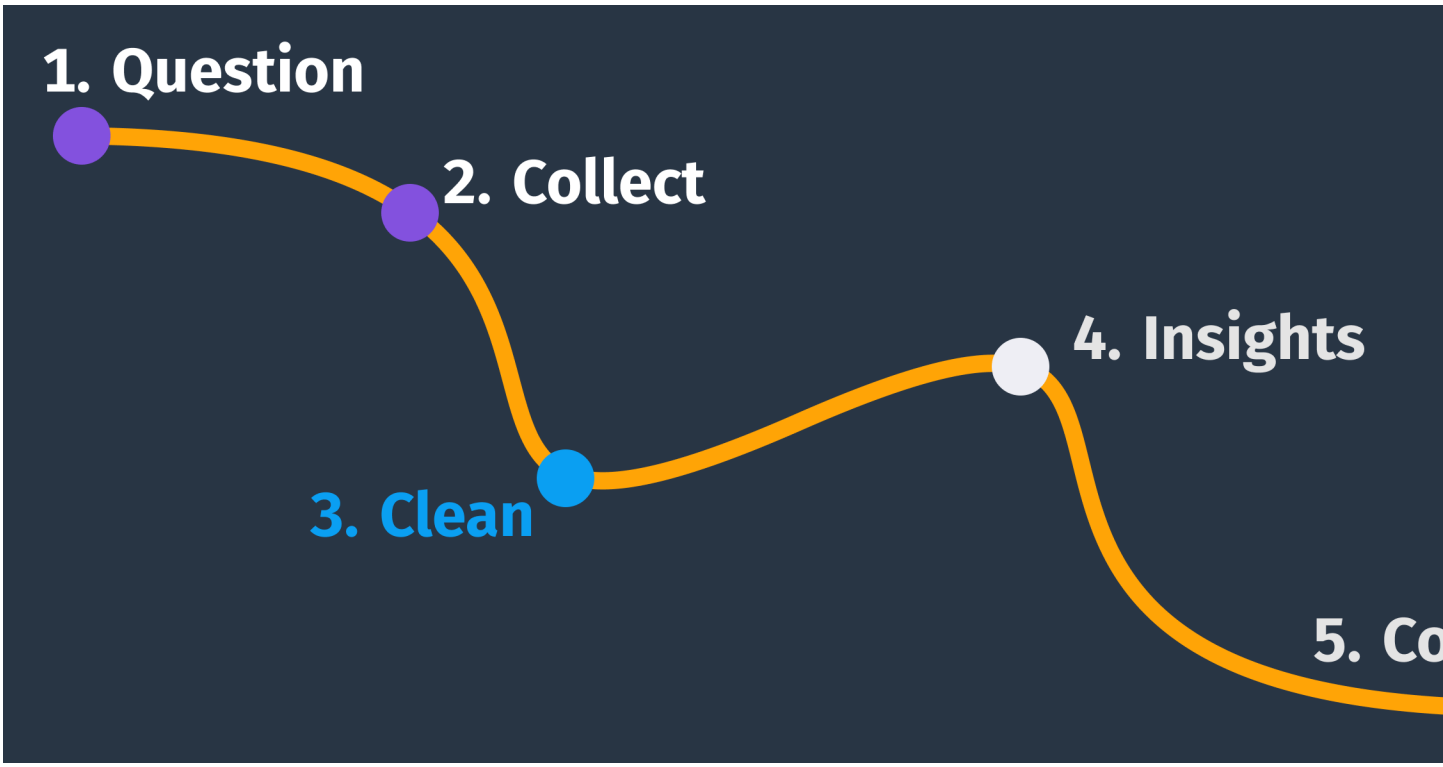
SQL processes the clauses in a specific order

SQL processes the clauses in a specific order:

1. `FROM`
2. `WHERE`

3. GROUP BY
4. HAVING
5. SELECT

Cleaning Data



Raw data can come to you with quality issues. Data cleaning is a key step in the data analysis process. In this lesson, you'll learn to fix errors in the data that has been collected.

Duplicated data is a very common issue. It refers to additional copies of the same data. Including duplicated data in your calculations and analysis can result in incorrect data insights.

You can use GROUP BY in combination with HAVING to check for duplicates in the data.

```
SELECT id, COUNT(id)
FROM employees
GROUP BY id
HAVING COUNT(id) > 1;
```

You can group by multiple fields to check for data duplication across multiple fields

```
SELECT id, name
FROM employees
GROUP BY id, name
HAVING COUNT(id) > 1;
```

Duplicated data can significantly impact the quality and accuracy of your analysis. Use **DISTINCT** to eliminate duplicate values

```
SELECT DISTINCT name FROM employees;
```

NULL is used to indicate that a data **value is missing** and does not exist in the database. NULL values are not shown in result tables. You can check if your data contains missing values. Use **IS NULL** in combination with WHERE to find missing values

```
SELECT *
FROM movies
WHERE genre IS NULL
```

Similarly, you can extract non-null values using **IS NOT NULL**. This will filter null values out.

```
SELECT *
FROM movies
WHERE genre IS NOT NULL

/*Check multiple variables are not missing*/
SELECT *
FROM your_table
WHERE column1 IS NOT NULL
  AND column2 IS NOT NULL
  AND column3 IS NOT NULL;

/*Another way to check multiple variables are not missing
<> means not equal*/
WHERE column1 <> ''
  AND column2 <> '';

/*You cannot use this expression like SAS PROC SQL*/
/*WHERE column1 not='' AND column2 not=''*/
```

Filtering out missing data is one way to deal with incomplete data. Different approaches are used depending on where the data is coming from or the type of analysis that you need to perform.

Fixing Data Types

Data comes in different shapes and forms. Data in the incorrect format can cause issues. In this lesson, you'll learn to detect and fix errors in data types.

The data type that stores one of two possible values (True or False) is called **boolean**.

Making Sense of Data

Extracting insights from cleaned data consists in finding hidden patterns and trends that help us make data-driven decisions. In this lesson, you'll dive deeper into the final steps of the data analysis process.

Booleans help you write more complex and effective queries to answer questions about the data. Operations involving boolean values are known as **logical operations**.

The **AND operation** results in a True value only when all the values are True at the same time. What's the result of this AND logical operation?

```
True AND True
```

You can use logical operations to filter data that meets multiple conditions. The AND operation results in a True value only when all conditions are True.

```
SELECT title
FROM books
WHERE qty >= 5
  AND year > 1950
```

The **OR logical operation** results in a True value if at least one of the conditions is True. What's the result of this OR logical operation?

'True OR False'

Data visualization helps you spot trends and patterns in your data. A programming language like Python lets you create powerful visual aids to communicate your findings! The perfect next step to complement your new SQL skills!

The last step of the data analysis process is **communicating** your findings. This is where your communication skills will come in handy. At this stage, data visualization and data storytelling will help you turn data insights into actions.

Module 4 Quiz

Certificate

[Credential URL](https://www.sololearn.com/certificates/CC-BJHLUH2I)