

# Reference guide: SQL

## Google Cybersecurity Certificate

---

### Table of contents

[Query a database](#)

[Apply filters to SQL queries](#)

[Join tables](#)

[Perform calculations](#)

---

### Query a database

The `SELECT`, `FROM`, and `ORDER BY` keywords are used when retrieving information from a database.

#### **FROM**

Indicates which table to query; required to perform a query

```
FROM employees
```

Indicates to query the `employees` table

#### **ORDER BY**

Sequences the records returned by a query based on a specified column or columns

```
ORDER BY department
```

Sorts the records in ascending order by the `department` column; `ORDER BY department ASC` also sorts the records in ascending order by the `department` column

```
ORDER BY city DESC
```

Sorts the records in descending order by the `city` column

```
ORDER BY country, city
```

Sorts the records in ascending order by multiple columns; first sorts the output by `country`, and for records with the same `country`, sorts them based on `city`

## **SELECT**

Indicates which columns to return; required to perform a query

```
SELECT employee_id
```

Returns the `employee_id` column

```
SELECT *
```

Returns all columns in a table

## Apply filters to SQL queries

`WHERE` and the other SQL keywords and characters that follow are used when applying filters to SQL queries.

## **AND**

Specifies that both conditions must be met simultaneously in a filter that contains two conditions

```
WHERE region = 5 AND country = 'USA'
```

Returns all records with a value in the `region` column of `5` and a value in the `country` column of `'USA'`

## **BETWEEN**

Filters for numbers or dates within a range; `BETWEEN` is followed by the first value to include in the range, the `AND` operator, and the last value to include in the range

```
WHERE hiredate BETWEEN '2002-01-01' AND '2003-01-01'
```

Returns all records with a value in the `hiredate` column that is between `'2002-01-01'` and `'2003-01-01'`

## = (equal to)

Used in filters to return only the records that contain a value in a specified column that is equal to a particular value

```
WHERE birthdate = '1980-05-15'
```

Returns all records with a value in the `birthdate` column that equals '1980-05-15'

## > (greater than)

Used in filters to return only the records that contain a value in a specified column that is greater than a particular value

```
WHERE birthdate > '1970-01-01'
```

Returns all records with a value in the `birthdate` column that is greater than '1970-01-01'

## >= (greater than or equal to)

Used in filters to return only the records that contain a value in a specified column that is greater than or equal to a particular value

```
WHERE birthdate >= '1965-06-30'
```

Returns all records with a value in the `birthdate` column that is greater than or equal to '1965-06-30'

## < (less than)

Used in filters to return only the records that contain a value in a specified column that is less than a particular value

```
WHERE date < '2023-01-31'
```

Returns all records with a value in the `date` column that is less than '2023-01-31'

## **<= (less than or equal to)**

Used in filters to return only the records that contain a value in a specified column that is less than or equal to a particular value

```
WHERE date <= '2020-12-31'
```

Returns all records with a value in the `date` column that is less than or equal to '2020-12-31'

## **LIKE**

Used with `WHERE` to search for a pattern in a column

```
WHERE title LIKE 'IT%'
```

Returns all records with a value in the `title` column that matches the pattern of 'IT%'

```
WHERE state LIKE 'N_'
```

Returns all records with a value in the `state` column that matches the pattern of 'N\_'

## **NOT**

Negates a condition

```
WHERE NOT country = 'Mexico'
```

Returns all records with a value in the `country` column that is not 'Mexico'

## **<> (not equal to)**

Used in filters to return only the records that contain a value in a specified column that is not equal to a particular value; `!=` also used as an operator for not equal to

```
WHERE date <> '2023-02-28'
```

Returns all records with a value in the `date` column that is not equal to '2023-02-28'

## != (not equal to)

Used in filters to return only the records that contain a value in a specified column that is not equal to a particular value; <> also used as an operator for not equal to

```
WHERE date != '2023-05-14'
```

Returns all records with a value in the `date` column that is not equal to '2023-05-14'

## OR

Specifies that either condition can be met in a filter that contains two conditions

```
WHERE country = 'Canada' OR country = 'USA'
```

Returns all records with a value in the `country` column of either 'Canada' or 'USA'

## % (percentage sign)

Substitutes for any number of other characters; used as a wildcard in a pattern that follows `LIKE`

```
'a%'
```

Represents a pattern consisting of the letter 'a' followed by zero or more characters

```
'%a'
```

Represents a pattern consisting of zero or more characters followed by the letter 'a'

```
'%a%'
```

Represents a pattern consisting of the letter 'a' surrounded by zero or more characters on each side

## \_ (underscore)

Substitutes for one other character; used as a wildcard in a pattern that follows `LIKE`

'a\_'

Represents a pattern consisting of the letter 'a' followed by one character

'a\_\_'

Represents a pattern consisting of the letter 'a' followed by two characters

'\_a'

Represents a pattern consisting of one character followed by the letter 'a'

'\_a\_'

Represents a pattern consisting of the letter 'a' surrounded by one character on each side

## WHERE

Indicates the condition for a filter; must be used to begin a filter

```
WHERE title = 'IT Staff'
```

Returns all records that contain 'IT Staff' in the title column; WHERE is placed before the condition of title = 'IT Staff' to create the filter

## Join tables

The following SQL keywords are used to join tables.

### FULL OUTER JOIN

Returns all records from both tables; the column used to join the tables is specified following FULL OUTER JOIN with syntax that includes ON and equal to (=)

```
SELECT *  
FROM employees  
FULL OUTER JOIN machines ON employees.device_id =  
machines.device_id;
```

Returns all records from the employees table and machines table; uses the device\_id column to join the two tables

## INNER JOIN

Returns records matching on a specified column that exists in more than one table; the column used to join the tables is specified following `INNER JOIN` with syntax that includes `ON` and equal to (`=`)

```
SELECT *  
FROM employees  
INNER JOIN machines ON employees.device_id =  
machines.device_id;
```

Returns all records that have a value in the `device_id` column in the `employees` table that matches a value in the `device_id` column in the `machines` table

## LEFT JOIN

Returns all the records of the first table, but only returns records of the second table that match on a specified column; the first (or left) table appears directly after the keyword `FROM`; the column used to join the tables is specified following `LEFT JOIN` with syntax that includes `ON` and equal to (`=`)

```
SELECT *  
FROM employees  
LEFT JOIN machines ON employees.device_id =  
machines.device_id;
```

Returns all records from the `employees` table but only the records from the `machines` table that have a value in the `device_id` column that matches a value in the `device_id` column in the `employees` table

## RIGHT JOIN

Returns all of the records of the second table, but only returns records from the first table that match on a specified column; the second (or right) table appears directly after the `RIGHT JOIN` keyword; the column used to join the tables is specified following `RIGHT JOIN` with syntax that includes `ON` and equal to (`=`)

```
SELECT *  
FROM employees  
RIGHT JOIN machines ON employees.device_id =  
machines.device_id;
```

Returns all records from the `machines` table but only the records from the `employees` table that have a value in the `device_id` column that matches a value in the `device_id` column in the `machines` table

## Perform calculations

The following SQL keywords are aggregate functions and are helpful when performing calculations.

### AVG

Returns a single number that represents the average of the numerical data in a column; placed after `SELECT`

```
SELECT AVG(height)
```

Returns the average height from all records that have a value in the `height` column

### COUNT

Returns a single number that represents the number of records returned from a query; placed after `SELECT`

```
SELECT COUNT(firstname)
```

Returns the number of records that have a value in the `firstname` column

### SUM

Returns a single number that represents the sum of the numerical data in a column; placed after `SELECT`

```
SELECT SUM(cost)
```

Returns the sum of costs from all records that have a value in the `cost` column