Reference guide: SQL

Google Cybersecurity Certificate

Table of contents

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

Índice de Contenidos

---

1. Consultar una Base de Datos

Para recuperar información de una base de datos, se utilizan las palabras clave **SELECT**, **FROM** y **ORDER BY**.

FROM
Indica la tabla que se va a consultar. Es necesario para realizar una consulta.

```
FROM empleados
```

ORDER BY
Ordena los registros devueltos por una consulta según una o varias columnas.

```
ORDER BY departamento
```

Ordena los registros de forma ascendente por la columna `departamento`. También se puede usar `ASC` explícitamente para ascendente:

```
ORDER BY departamento ASC
```

Para ordenar de manera descendente:

```
ORDER BY ciudad DESC
```

Para ordenar por múltiples columnas:

```
ORDER BY país, ciudad
```

SELECT
Indica qué columnas se devolverán en el resultado. Es obligatorio en una consulta.

```
SELECT id_empleado
```
Para devolver todas las columnas de la tabla:

```
SELECT *
```

---

2. Aplicar Filtros a las Consultas SQL

Los filtros en SQL se aplican utilizando la cláusula **WHERE** y otros operadores.

AND
Especifica que ambas condiciones deben cumplirse simultáneamente en un filtro que contenga dos condiciones.

```
WHERE región = 5 AND país = 'USA'
```

BETWEEN
Filtra números o fechas dentro de un rango.

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

= (igual a)
Filtra los registros que contienen un valor específico en una columna.

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

> (mayor que)
Filtra los registros que contienen un valor mayor que un valor específico.

```
WHERE fecha_nacimiento > '1970-01-01'
```

>= (mayor o igual a)
Filtra los registros que contienen un valor mayor o igual a un valor específico.

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

< (menor que)
Filtra los registros que contienen un valor menor que un valor específico.

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

<= (menor o igual a)
Filtra los registros que contienen un valor menor o igual a un valor específico.

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

LIKE
Busca un patrón en una columna.

```
WHERE título LIKE 'IT%'
```

NOT
Niega una condición.

```
WHERE NOT país = 'México'
```

<> o != (no igual a)
Filtra los registros que no tienen un valor específico en una columna.

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

OR
Especifica que cualquiera de las condiciones puede cumplirse.

```
WHERE país = 'Canadá' OR país = 'USA'
```

% (porcentaje)
Sustituye cualquier cantidad de caracteres, utilizado como comodín en patrones con **LIKE**.

```
WHERE nombre LIKE 'a%'
```

_ (guion bajo)
Sustituye un solo carácter, utilizado como comodín en patrones con **LIKE**.

```
WHERE nombre LIKE 'a_'
```

---

3. Unir Tablas

Para combinar información de varias tablas, se utilizan las siguientes palabras clave SQL:

FULL OUTER JOIN
Devuelve todos los registros de ambas tablas.

```
SELECT *
FROM empleados
FULL OUTER JOIN dispositivos ON empleados.id_dispositivo =
dispositivos.id_dispositivo;
```

INNER JOIN
Devuelve solo los registros que tienen valores coincidentes en ambas tablas.

```
SELECT *
FROM empleados
INNER JOIN dispositivos ON empleados.id_dispositivo =
dispositivos.id_dispositivo;
```

LEFT JOIN
Devuelve todos los registros de la primera tabla y solo aquellos de la segunda que

coincidan.

```
SELECT *
FROM empleados
LEFT JOIN dispositivos ON empleados.id_dispositivo =
dispositivos.id_dispositivo;
```

RIGHT JOIN
Devuelve todos los registros de la segunda tabla y solo aquellos de la primera que coincidan.

```
SELECT *
FROM empleados
RIGHT JOIN dispositivos ON empleados.id_dispositivo =
dispositivos.id_dispositivo;
```

---

4. Realizar Cálculos

Los siguientes operadores agregados son útiles para realizar cálculos sobre los datos:

AVG
Devuelve el promedio de los datos numéricos en una columna.

```
SELECT AVG(altura)
```

COUNT
Devuelve el número de registros devueltos por una consulta.

```
SELECT COUNT(nombre)
```

SUM
Devuelve la suma de los datos numéricos en una columna.

```
SELECT SUM(costo)
```

Este formato proporciona una guía más estructurada y visualmente clara para trabajar con consultas SQL y comprender cómo aplicar filtros, unir tablas y realizar cálculos en una base de datos.