GUIA CONSULTAS SQL

LINA PAOLA WILCHES GOMEZ N° 1023242747

LAURA JULIETH GUERRERO ROA N° 1053282200

MODELADO DE LOS ARTEFACTOS DEL SOFTWARE.

ESTUPIÑAN FINO DEMETRIO MAURICIO

SERVICIO NACIONAL DE APRENDIZAJE (SENA)

BOGOTA D.C

2025

FUNCIONES DE SQL

FUNCIONES DE MANIPULACIÓN DE CADENAS DE TEXTO

UPPER(cadena): Convierte todos los caracteres de la cadena especificada a mayúsculas.

Ejemplo:

```
SELECT UPPER(nombre) FROM producto; (Mostrará los nombres de los productos en mayúsculas).
```

LOWER(cadena): Convierte todos los caracteres de la cadena especificada a minúsculas.

Ejemplo:

```
SELECT LOWER(nombre) FROM producto; (Mostrará los nombres de los productos en minúsculas).
```

SUBSTRING(cadena, inicio, longitud): Extrae una subcadena de la cadena especificada, comenzando en la posición inicio (el primer carácter es la posición 1) y con una longitud de longitud caracteres.

Ejemplo:

```
SELECT SUBSTRING(nombre, 1, 2) FROM fabricante; (Mostrará los dos primeros caracteres del nombre de cada fabricante).
```

LIKE 'patrón': Se utiliza en la cláusula WHERE para buscar filas que coincidan con un patrón específico. Se utilizan comodines:

%: Representa cero o más caracteres. Ejemplo:

- WHERE nombre LIKE 'S%'; (Nombres que comienzan con 'S').
- WHERE nombre LIKE '%e'; (Nombres que terminan con 'e').
- WHERE nombre LIKE '%w%'; (Nombres que contienen 'w').
- WHERE nombre LIKE '%Portátil%'; (Nombres que contienen 'Portátil').

_: Representa exactamente un carácter. Ejemplo:

```
WHERE nombre LIKE '_ _ _ _'; (Nombres de exactamente 4 caracteres).
```

FUNCIONES NUMÉRICAS

ROUND(número, decimales): Redondea el número al número de decimales especificado. Si se omite decimales o es 0, se redondea al entero más cercano.

Ejemplo:

```
SELECT ROUND(precio) FROM producto; (Mostrará el precio redondeado al entero más cercano).
```

TRUNCATE(número, decimales): Trunca el número al número de decimales especificado, eliminando los decimales sin redondear.

Ejemplo:

```
SELECT TRUNCATE(precio, 0) FROM producto; (Mostrará el precio sin decimales).
```

FUNCIONES DE AGREGACIÓN (PARA RESUMIR DATOS)

COUNT(*): Cuenta el número total de filas en una tabla o el número de filas que cumplen una condición.

Ejemplo:

- SELECT COUNT(*) FROM producto; (Cuenta el número total de productos).
- SELECT COUNT(codigo) FROM producto WHERE precio >= 180; (Cuenta los productos con precio mayor o igual a 180).
- COUNT(DISTINCT columna): Cuenta el número de valores distintos en una columna específica.
- SELECT COUNT(DISTINCT codigo_fabricante) FROM producto; (Cuenta el número de fabricantes distintos que tienen productos).

AVG(columna): Calcula el valor promedio de los valores en una columna numérica.

Ejemplo:

```
SELECT AVG(precio) FROM producto; (Calcula el precio promedio de todos los productos).
```

MIN(columna): Encuentra el valor mínimo en una columna específica.

Ejemplo:

```
SELECT MIN(precio) FROM producto; (Encuentra el precio más bajo).
```

MAX(columna): Encuentra el valor máximo en una columna específica.

Ejemplo:

```
SELECT MAX(precio) FROM producto; (Encuentra el precio más alto).
```

SUM(columna): Calcula la suma de todos los valores en una columna numérica.

Ejemplo:

```
SELECT SUM(precio) FROM producto; (Calcula la suma de todos los precios).
```

CLÁUSULAS Y OPERADORES

LEFT JOIN tabla2 ON condición: Devuelve todas las filas de la tabla izquierda (tabla1) y las filas coincidentes de la tabla derecha (tabla2). Si no hay coincidencia en la tabla derecha, se devuelven valores NULL para las columnas de la tabla derecha.

Ejemplo:

```
SELECT f.nombre, p.nombre FROM fabricante f LEFT JOIN producto p ON f.codigo =
p.codigo_fabricante;
```

WHERE condición: Filtra las filas devueltas por la consulta basándose en una condición específica.

Ejemplo:

```
SELECT nombre FROM producto WHERE precio <= 120;
```

ORDER BY columna [ASC | DESC]: Ordena el resultado de la consulta por una o más columnas. ASC (ascendente) es el valor predeterminado. DESC ordena de forma descendente.

Ejemplo:

```
SELECT nombre FROM fabricante ORDER BY nombre ASC;

SELECT nombre, precio FROM producto ORDER BY nombre ASC, precio DESC;
```

LIMIT (número): Limita el número de filas devueltas por la consulta. número especifica el número máximo de filas.

Ejemplo:

```
SELECT nombre FROM fabricante LIMIT 5; (Devuelve las primeras 5 filas).
```

DISTINCT: Elimina las filas duplicadas del resultado de la consulta, mostrando solo valores únicos en las columnas seleccionadas.

Ejemplo:

```
SELECT DISTINCT codigo_fabricante FROM producto;
```

BETWEEN valor1 AND valor2: Se utiliza en la cláusula WHERE para seleccionar filas donde el valor de una columna está dentro de un rango inclusivo especificado por valor1 y valor2.

Ejemplo:

```
WHERE precio BETWEEN 60 AND 200;
```

IN (valor1, valor2, ...): Se utiliza en la cláusula WHERE para seleccionar filas donde el valor de una columna coincide con alguno de los valores de una lista especificada.

Ejemplo:

```
WHERE codigo_fabricante IN (1, 3, 5);
```

ON: Especifica la condición para combinar filas de dos o más tablas en una operación JOIN. Define cómo se relacionan las filas de las tablas involucradas.

Ejemplo:

```
SELECT p.nombre AS nombre_producto, f.nombre AS nombre_fabricante FROM Producto p
INNER JOIN Fabricante f ON p.codigo_fabricante = f.codigo;
```

WHERE EXISTS: Verifica si una subconsulta devuelve alguna fila. Si sí, la condición es TRUE.

Ejemplo:

```
SELECT f.nombre FROM fabricante f WHERE EXISTS (SELECT * FROM producto p
WHERE p.codigo_fabricante = f.codigo);
```

WHERE NOT EXISTS: Verifica si una subconsulta no devuelve ninguna fila. Si no devuelve filas, la condición es TRUE.

Ejemplo:

```
SELECT f.nombre FROM fabricante f WHERE NOT EXISTS (SELECT * FROM producto p
WHERE p.codigo_fabricante = f.codigo);
```

GROUP BY columna1, columna2, ...: Agrupa las filas que tienen los mismos valores en una o más columnas en filas de resumen. Se utiliza a menudo con funciones de agregación para calcular estadísticas por grupo.

Ejemplo:

```
SELECT nombre_fabricante, COUNT(codigo) FROM fabricante LEFT JOIN producto ON fabricante.codigo = producto.codigo_fabricante GROUP BY nombre_fabricante;
```

HAVING: Filtra los resultados de los grupos creados por la cláusula

GROUP BY. Se utiliza para aplicar condiciones a los grupos en sí, en lugar de a las filas individuales (para eso se usa WHERE).

Ejemplo:

```
GROUP BY nombre_fabricante HAVING COUNT(codigo) >= 2;
```