

## ✓ 1. SELECT (consultar datos)

```
SELECT columnas  
FROM tabla;
```

---

## ✓ 2. WHERE (filtro)

```
SELECT *  
FROM tabla  
WHERE condición;
```

---

## ✓ 3. Operadores lógicos

AND, OR, NOT

Ejemplo:

```
WHERE edad > 18 AND ciudad = 'Madrid'
```

---

## ✓ 4. ORDER BY (ordenar)

```
SELECT *  
FROM tabla  
ORDER BY columna ASC|DESC;
```

---

## ✓ 5. LIMIT (limitar resultados)

```
SELECT *  
FROM tabla  
LIMIT 10;
```

---

## ✓ 6. GROUP BY (agrupar)

```
SELECT columna, COUNT(*)
```

```
FROM tabla  
GROUP BY columna;
```

---

## ✓ 7. HAVING (filtro después de GROUP BY)

```
SELECT columna, COUNT(*)  
FROM tabla  
GROUP BY columna  
HAVING COUNT(*) > 1;
```

---

## ✓ 8. JOIN (unir tablas)

### INNER JOIN (lo más común)

```
SELECT *  
FROM tabla1 t1  
JOIN tabla2 t2  
ON t1.id = t2.id;
```

### LEFT JOIN

```
SELECT *  
FROM tabla1 t1  
LEFT JOIN tabla2 t2  
ON t1.id = t2.id;
```

---

## ✓ 9. Funciones de agregación

COUNT(), SUM(), AVG(), MIN(), MAX()

Ejemplo:

```
SELECT COUNT(*), AVG(salario)  
FROM empleados;
```

---

## ✓ 10. INSERT (insertar datos)

```
INSERT INTO tabla (col1, col2)
VALUES (val1, val2);
```

---

## ✓ 11. UPDATE (modificar datos)

```
UPDATE tabla
SET col1 = valor
WHERE condición;
```

---

## ✓ 12. DELETE (borrar datos)

```
DELETE FROM tabla
WHERE condición;
```

---

## ✓ 13. CREATE TABLE (crear tabla)

```
CREATE TABLE tabla (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(50),
    edad INTEGER
);
```

---

## ✓ 14. ALTER TABLE (modificar tabla)

```
ALTER TABLE tabla
ADD COLUMN nueva_columna tipo;
```

---

## ✓ 15. DROP TABLE (borrar tabla)

```
DROP TABLE tabla;
```

---

## ✓ 16. Índices (para acelerar consultas)

```
CREATE INDEX idx_nombre  
ON tabla (columna);
```

---

## ✓ 17. Vistas (views)

```
CREATE VIEW vista AS  
SELECT columnas  
FROM tabla  
WHERE condición;
```

---

## Resumen de comandos más usados

| Comando         | Uso             |
|-----------------|-----------------|
| SELECT          | Consultar       |
| INSERT          | Insertar        |
| UPDATE          | Modificar       |
| DELETE          | Borrar          |
| CREATE<br>TABLE | Crear tabla     |
| ALTER<br>TABLE  | Modificar tabla |
| DROP<br>TABLE   | Borrar tabla    |

## ✓1 TO\_CHAR( ) (formatear fechas y números)

Convierte fechas o números a texto con formato.

### 📌 Ejemplo fecha

```
TO_CHAR(fecha, 'DD/MM/YYYY')
```

### 📌 Ejemplo número

```
TO_CHAR(1234.56, '9G999D99')
```

📌 *Nota:* No existe `to_string()` en SQL estándar, eso es de otros lenguajes. En SQL se usa `TO_CHAR()`.

---

## ✓2 Funciones de ventana (window functions)

Estas funciones se usan con `OVER()`.

---

### ♦ RANK( )

Asigna un **rango** dentro de un grupo ordenado.

```
RANK() OVER (ORDER BY salario DESC)
```

📌 Si hay empates, el rango se repite y deja huecos.

---

### ♦ DENSE\_RANK( )

Igual que `RANK()` pero **sin huecos** en empates.

```
DENSE_RANK() OVER (ORDER BY salario DESC)
```

---

### ♦ ROW\_NUMBER( )

Da un número único por fila, sin empates.

`ROW_NUMBER() OVER (ORDER BY salario DESC)`

---

### ✓ 3 **OVER()** (claves de las window functions)

**OVER()** define:

- cómo se ordena
- cómo se agrupa (si hay PARTITION)

#### Ejemplo básico

```
SELECT nombre,  
       salario,  
       RANK() OVER (ORDER BY salario DESC) AS rango  
FROM empleados;
```

---

### ✓ 4 **PARTITION BY** (dentro de OVER)

Agrupar dentro de la ventana.

#### Ejemplo: ranking por departamento

```
RANK() OVER (PARTITION BY departamento_id ORDER BY salario DESC)
```

📌 Esto crea un ranking **por cada departamento**.

---

### 🧠 Resumen rápido (chuleta)

| Función                   | Qué hace                          |
|---------------------------|-----------------------------------|
| <code>TO_CHAR()</code>    | Formatea fechas/números a texto   |
| <code>RANK()</code>       | Ranking con huecos si hay empates |
| <code>DENSE_RANK()</code> | Ranking sin huecos                |

|                           |                              |
|---------------------------|------------------------------|
| <code>ROW_NUMBER()</code> | Número de fila único         |
| <code>OVER()</code>       | Define ventana de la función |
| <code>PARTITION BY</code> | Agrupar dentro de la ventana |

---



## Frase perfecta para examen

“Las funciones de ventana (`RANK`, `ROW_NUMBER`, etc.) se usan con `OVER()` y permiten calcular valores sobre conjuntos de filas sin agruparlas con `GROUP BY`.”

# FUNCIONES COMUNES EN SQL (PostgreSQL)

---

## ✓ 1 **EXTRACT()**

Extrae partes de una fecha (año, mes, día, etc.)

```
EXTRACT(YEAR FROM fecha)
EXTRACT(MONTH FROM fecha)
EXTRACT(DAY FROM fecha)
```

 Devuelve **número**, no texto.

---

## ✓ 2 **UPPER()**

Convierte texto a mayúsculas.

```
UPPER('hola') --> 'HOLA'
```

---

## ✓ 3 **LOWER()**

Convierte texto a minúsculas.

```
LOWER('HOLA') --> 'hola'
```

---

## ✓ 4 **INITCAP()**

Convierte el texto a formato **Título** (primera letra en mayúscula).

```
INITCAP('hola mundo') --> 'Hola Mundo'
```

---



## ✓ 5 TRIM()

Elimina espacios al inicio y final.

```
TRIM('  hola  ') --> 'hola'
```

---

## ✓ 6 LENGTH()

Devuelve la longitud de un texto.

```
LENGTH('hola') --> 4
```

---

## ✓ 7 SUBSTRING()

Extrae parte de un texto.

```
SUBSTRING('hola', 1, 2) --> 'ho'
```

---

## ✓ 8 CONCAT()

Une texto (como un + en otros lenguajes).

```
CONCAT('Hola', ' ', 'Mundo') --> 'Hola Mundo'
```

---

## ✓ 9 COALESCE()

Devuelve el primer valor **no NULL**.

```
COALESCE(NULL, 'valor', 'otro') --> 'valor'
```

---

## Resumen rápido (chuleta)

| Función | Para qué sirve |
|---------|----------------|
|---------|----------------|

EXTRACT() Extraer partes de fecha

UPPER() Convertir a mayúsculas

LOWER() Convertir a minúsculas

INITCAP() Primera letra  
mayúscula

TRIM() Quitar espacios

LENGTH() Longitud de texto

SUBSTRING  
( ) Subcadena

CONCAT() Unir texto

COALESCE(  
) Sustituir NULL