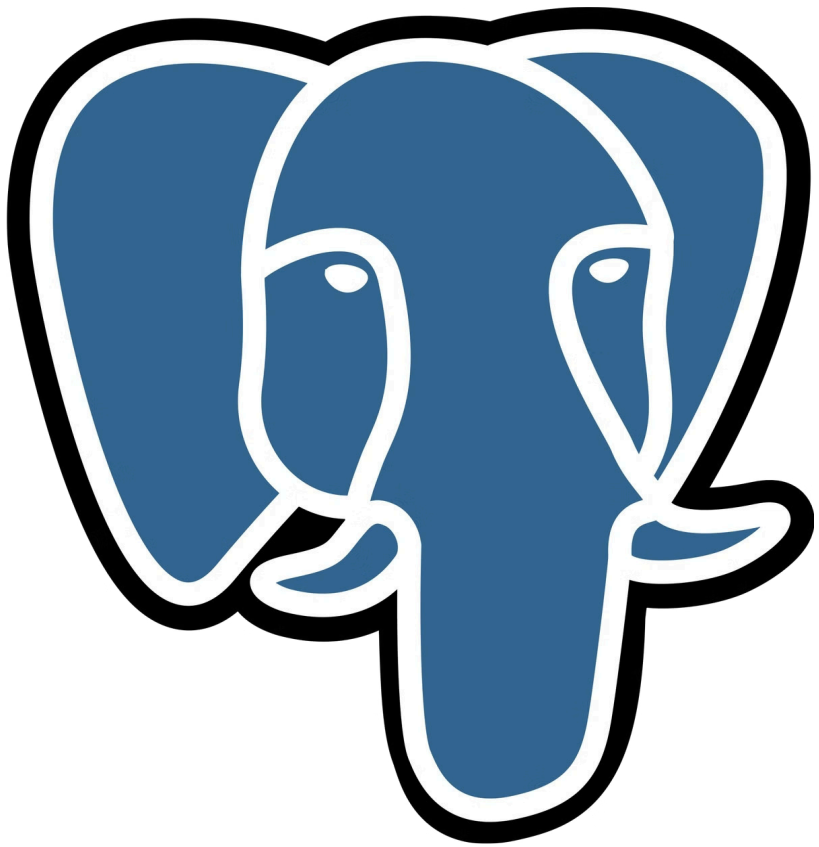
🎓 Clase 2: Filtros, Condicionales y fuentes de información

👋 **Bienvenidos al segundo módulo**

En esta sesión, exploraremos operadores de texto, matemáticos, lógicos y de tiempo. Además, veremos el uso del WHERE y CASE WHEN. Finalmente, importaremos data de Python a Postgress 🚀



✍️ **Profesor:** Yoseph Ayala Valencia

🌟 ¡Prepárate para dominar los fundamentos de SQL y llevarlos al siguiente nivel con Python! 🐍

Funciones de texto

- Una función de texto en SQL es una función que se utiliza para manipular y trabajar con cadenas de texto (también conocidas como cadenas de caracteres).
- Estas funciones permiten realizar diversas operaciones sobre los datos de tipo texto, como concatenar cadenas, cambiar mayúsculas a minúsculas, extraer subcadenas, calcular la longitud de una cadena, y más.

Función (PostgreSQL)	Explicación	Ejemplo (PostgreSQL)
`CONCAT`	Combina dos o más cadenas de texto en una sola.	`SELECT CONCAT('Hello', ' ', 'World');` → `Hello World`
`UPPER`	Convierte todos los caracteres de una cadena a mayúsculas.	`SELECT UPPER('sql commands');` → `SQL COMMANDS`
`LOWER`	Convierte todos los caracteres de una cadena a minúsculas.	`SELECT LOWER('SQL Commands');` → `sql commands`
`SUBSTRING`	Extrae una subcadena de una cadena a partir de una posición inicial y una longitud especificada.	`SELECT SUBSTRING('Database' FROM 1 FOR 4);` → `Data`
`LENGTH`	Devuelve la longitud de una cadena de texto.	`SELECT LENGTH('SQL');` → `3`
`LEFT`	Devuelve el número especificado de caracteres desde el principio de una cadena.	`SELECT LEFT('DataScience', 4);` → `Data`
`RIGHT`	Devuelve el número especificado de caracteres desde el final de una cadena.	`SELECT RIGHT('DataScience', 7);` → `Science`
`TRIM`	Elimina espacios en blanco (u otros caracteres especificados) desde el inicio y/o el final de una cadena.	`SELECT TRIM(' SQL ');` → `SQL`
`STRING_AGG`	Combina valores de una columna en una sola cadena, separados por un delimitador especificado.	`SELECT STRING_AGG(column_name, ', ') FROM table_name;` → `value1, value2, value3`

Funciones matemáticas

- Las funciones matemáticas en SQL son operaciones predefinidas que se utilizan para realizar cálculos numéricos en los datos.
- Estas funciones permiten manipular, transformar y analizar valores numéricos dentro de una base de datos.
- Son ampliamente utilizadas para realizar operaciones como redondeo, cálculo de potencias, logaritmos, raíces cuadradas, y más.

Función (PostgreSQL)	Explicación	Ejemplo (PostgreSQL)
`ABS`	Devuelve el valor absoluto de un número, es decir, convierte los números negativos en positivos.	`SELECT ABS(-10);` → 10
`ROUND`	Redondea un número a un número específico de decimales.	`SELECT ROUND(5.678, 2);` → 5.68
`CEILING`	Redondea un número hacia arriba al entero más cercano.	`SELECT CEILING(4.2);` → 5
`FLOOR`	Redondea un número hacia abajo al entero más cercano.	`SELECT FLOOR(4.8);` → 4
`SQRT`	Devuelve la raíz cuadrada de un número.	`SELECT SQRT(16);` → 4
`POWER`	Eleva un número a la potencia de otro número.	`SELECT POWER(2, 3);` → 8
`EXP`	Devuelve el resultado de elevar e (la base de los logaritmos naturales) a la potencia de un número.	`SELECT EXP(1);` → 2.718281828459045
`LN`	Devuelve el logaritmo natural de un número, es decir, el logaritmo en base e.	`SELECT LN(10);` → 2.302585092994046

Funciones de tiempo

- Las funciones de tiempo en SQL son funciones que permiten trabajar con datos relacionados con fechas y horas.
- Estas funciones se utilizan para manipular, extraer, calcular diferencias y realizar transformaciones sobre valores de fecha y hora en una base de datos.
- Son esenciales para gestionar y analizar datos temporales, como eventos, registros de transacciones, planificación, y reportes basados en tiempo.

Función (PostgreSQL)	Explicación	Ejemplo (PostgreSQL)
`AGE`	Calcula la diferencia entre dos fechas, devolviendo el resultado en años, meses, días, horas, minutos, y segundos.	`SELECT AGE('2023-12-31'::date, '2023-01-01'::date);` → 364 days
`CURRENT_DATE`	Devuelve la fecha actual del sistema en formato 'YYYY-MM-DD'.	`SELECT CURRENT_DATE;` → '2024-08-11'
`CURRENT_TIMESTAMP`	Devuelve la fecha y hora actuales del sistema.	`SELECT CURRENT_TIMESTAMP;` → '2024-08-11 14:35:00'
`INTERVAL` + `+` (DATEADD)	Añade un intervalo específico (días, meses, años, etc.) a una fecha.	`SELECT '2024-08-01'::date + INTERVAL '10 days';` → '2024-08-11'
`TO_CHAR`	Convierte un valor de un tipo de datos a otro, comúnmente utilizado para formatear fechas.	`SELECT TO_CHAR(CURRENT_TIMESTAMP, 'DD/MM/YYYY');` → '11/08/2024'
`EXTRACT`	Extrae una parte específica de una fecha, como año, mes, día, etc.	`SELECT EXTRACT(YEAR FROM '2024-08-11'::date);` → 2024
`EXTRACT`	Extrae el mes de una fecha.	`SELECT EXTRACT(MONTH FROM '2024-08-11'::date);` → 8
`EXTRACT`	Extrae el día de una fecha.	`SELECT EXTRACT(DAY FROM '2024-08-11'::date);` → 11
`DATE_TRUNC`	Devuelve la última fecha del mes en que cae una fecha determinada.	`SELECT DATE_TRUNC('MONTH', '2024-08-11'::date) + INTERVAL '1 MONTH' - INTERVAL '1 day';` → '2024-08-31'
`EXTRACT`	Devuelve una parte específica de una fecha (como semana, año, mes, etc.).	`SELECT EXTRACT(WEEK FROM '2024-08-11'::date);` → 32

Cláusula WHERE

La cláusula WHERE en SQL permite aplicar condiciones para filtrar los datos devueltos por una consulta SELECT. Usando operadores como =, !=, <, >, <=, y >=, los usuarios pueden seleccionar solo los registros que cumplan con los criterios especificados.

Ejemplo:

```
SELECT *

FROM Empleados

WHERE Departamento = 'Ventas' AND Salario > 50000;
```

Explicación:

- `SELECT *`: Selecciona todas las columnas.
- `FROM Empleados`: Indica la tabla de la que se obtendrán los datos.
- `WHERE Departamento = 'Ventas'`: Filtra los registros para que solo se incluyan aquellos donde el departamento sea 'Ventas'.
- `AND Salario > 50000`: Añade una condición adicional para que solo se incluyan los empleados con un salario mayor a 50,000.

Operadores Lógicos y de Comparación

1 Operadores de Comparación

Permiten comparar valores: =, !=, <, >, <=, >=

2 Operadores Lógicos

Combinan múltiples condiciones: AND, OR, NOT

Aplicación de Múltiples Condiciones

Operador AND

Devuelve registros que cumplan con todas las condiciones.

```
SELECT *

FROM Empleados

WHERE Departamento = 'Ventas' AND
Salario > 50000;
```

Operador OR

Devuelve registros que cumplan con al menos una de las condiciones.

```
SELECT *

FROM Empleados

WHERE Departamento = 'Ventas' OR
Salario > 50000;
```

Operador NOT

Devuelve registros que no cumplen con la condición especificada.

```
SELECT *

FROM Empleados

WHERE NOT Departamento = 'Ventas';
```

Cláusula CASE WHEN

- La cláusula `CASE WHEN` se utiliza para realizar evaluaciones condicionales dentro de una consulta SQL.
- Permite devolver diferentes resultados basados en condiciones específicas, similar a una estructura `if-else` en otros lenguajes de programación.

Sintaxis básica

```sql
SELECT columna1,
       columna2,
       CASE
           WHEN condición1 THEN resultado1
           WHEN condición2 THEN resultado2
           ELSE resultado_default
       END AS alias_columna
FROM tabla;
```

Ejemplo

Supongamos que tienes una tabla `Empleados` con una columna `Salario` y deseas clasificar los empleados en diferentes categorías según su salario.

```sql
SELECT Nombre,
       Salario,
       CASE
           WHEN Salario > 80000 THEN 'Alto'
           WHEN Salario BETWEEN 50000 AND 80000 THEN 'Medio'
           ELSE 'Bajo'
       END AS Categoria_Salario
FROM Empleados;
```

Explicación:

- `CASE`: Inicia la expresión condicional.
- `WHEN Salario > 80000 THEN 'Alto'`: Si el salario es mayor a 80,000, se clasifica como 'Alto'.
- `WHEN Salario BETWEEN 50000 AND 80000 THEN 'Medio'`: Si el salario está entre 50,000 y 80,000, se clasifica como 'Medio'.
- `ELSE 'Bajo'`: Para todos los otros casos (salarios menores a 50,000), se clasifica como 'Bajo'.
- `AS Categoria_Salario`: Alias para la columna resultante.