

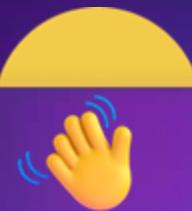
Clase 11. Bootcamp de SQL profesional

# Condiciones de consulta - Parte 2

Keyla Dolores



# Hola, soy Keyla Dolores



## PUNTOS DE CONTACTO

LinkedIn: [Keyla Dolores Méndez](#)

Blog: [www.keyladolores.com](#)

Twitter: @KeylaDoloresM



## EXPERIENCIA

Data Architect

Máster en Big Data & Analytics

Speaker Azure y Big Data

Blogger

Microsoft Certified Trainer

## CERTIFICACIONES



## CONTENIDO





# ¿Qué aprenderemos hoy?

- Trabajar con fechas en consultas SQL
- Aplicar condiciones en WHERE con JOINs
- Aprender buenas prácticas en consultas avanzadas

# Empecemos...



# ¿Qué aprendimos en la sesión anterior?

- INNER JOIN, LEFT JOIN, RIGHT JOIN
- Cláusula WHERE en consultas simples
- Uso de NULL en consultas

# ¿Por qué importan las fechas en SQL?

- 📌 En reportes y análisis
- 📌 Para filtrar datos recientes o históricos
- 📌 Para calcular diferencias de tiempo

# Tipos de Datos de Fecha

- DATE
- TIMESTAMP
- TIMESTAMPTZ

```
sql
```

```
SELECT NOW(), CURRENT_DATE, CURRENT_TIME;
```

# Filtrando datos por Fecha

- Filtrar un período específico

sql

```
SELECT *
FROM pedidos
WHERE fecha_pedido BETWEEN '2024-01-01' AND '2024-12-31';
```

- Fechas relativas con INTERVAL

sql

```
SELECT *
FROM pedidos
WHERE fecha_pedido >= CURDATE() - INTERVAL 30 DAY;
```

# Extrayendo información de Fechas

- Extraer Año, Mes y Día:

sql

```
SELECT fecha_pedido,  
       YEAR(fecha_pedido) AS año,  
       MONTH(fecha_pedido) AS mes  
FROM pedidos;
```

- Calcular diferencias de fechas:

sql

```
SELECT  
       DATEDIFF(CURDATE(), '2022-01-01') AS dias_diferencia;
```

# Generando Fechas Automáticamente

- Crear un rango de fechas automáticamente:

```
sql
```

```
WITH RECURSIVE meses AS (
    SELECT DATE('2024-01-01') AS fecha
    UNION ALL
    SELECT DATE_ADD(fecha, INTERVAL 1 MONTH)
    FROM meses WHERE fecha < '2024-12-01'
)
SELECT * FROM meses;
```

# ¿Cómo calcular acumulados por mes?

- 📌 Ventas acumuladas mes a mes:

```
sql
```

```
SELECT  
    fecha_venta,  
    SUM(total) OVER (ORDER BY fecha_venta) AS total_acumulado  
FROM ventas;
```

# Ejercicios con Fechas

 **Reto:** Obtener la cantidad de ventas por mes en 2024.

# ¿Cómo funcionan los JOINs en SQL?

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN

sql

SELECT

c.nombre, p.producto

FROM clientes c

INNER JOIN pedidos p ON c.id\_cliente = p.id\_cliente;

# WHERE con JOINs

- 📌 Filtrando resultados después del JOIN

sql

```
SELECT c.nombre, p.nombre AS producto, pe.fecha_pedido
FROM clientes c
JOIN pedidos pe ON c.id_cliente = pe.id_cliente
JOIN detalles_pedido dp ON pe.id_pedido = dp.id_pedido
JOIN productos p ON dp.id_producto = p.id_producto
WHERE YEAR(pe.fecha_pedido) = 2024;
```

# Manejo de valores NULL en JOINs

- ➡ LEFT JOIN puede traer valores NULL
- ➡ Uso de COALESCE() para manejar valores faltantes

sql

```
SELECT
    c.nombre, COALESCE(p.producto, 'Sin pedidos') AS producto
FROM clientes c
LEFT JOIN pedidos p ON c.id_cliente = p.id_cliente;
```

# JOINs con Fechas – Caso Práctico



## Enunciado:

- Listar clientes que hicieron compras en los últimos 3 meses.
- Mostrar productos que no han sido vendidos en los últimos 6 meses.

sql

```
SELECT GENERATE_SERIES('2024-01-01', '2024-12-31', '1 month') AS meses;
```

# JOINS con Subconsultas

- 📌 ¿Cuándo usar subqueries con JOINs?

sql

```
SELECT nombre
FROM clientes
WHERE id_cliente IN (
    SELECT id_cliente FROM pedidos
    WHERE fecha_pedido >= CURDATE() - INTERVAL 3 MONTH
);
```

# Reto #1

- 📌 Consulta a desarrollar:
  - 📍 Clientes que han comprado todos los productos de una categoría.
  - 📍 Mostrar la cantidad de ventas por mes en 2024.

## Reto #2 – Ejercicio Real

- 💡 Caso Real: Control de Asistencia en una Empresa
  - ◆ Un empleado puede fichar su entrada y salida varias veces al día.
  - ◆ Queremos calcular **horas trabajadas por día**.

**Pregunta:** ¿Cómo diseñar la consulta?

- 💡 **Pista:** Se necesita calcular la diferencia de horas por cada día.

# Conclusiones

- Existen diferentes tipos (DATE, TIMESTAMP, TIMESTAMPTZ).
- WHERE se usa después del JOIN para filtrar los resultados.
- Siempre verificar tipos de datos en columnas de fechas antes de hacer comparaciones.
  
- Analizar el impacto de JOINs en el volumen de datos y tiempos de respuesta.

# ¿Preguntas?

