Clase N° 11: Bases de datos

mongoDB.

Microsoft SQL Server

MySQL

Crombie

Funciones Básicas

Funciones de Agregación:

- ➔ COUNT(): Cuenta filas.
- ➔ SUM(): Suma valores numéricos.
- ➔ AVG(): Calcula promedio.
- ➔ MIN() y MAX(): Encuentran el valor mínimo y máximo.

```
SELECT
  COUNT(*) AS Total,
  SUM(price) AS TotalPrice,
  AVG(price) AS AvgPrice,
  MIN(price) AS MinPrice,
  MAX(price) AS MaxPrice
FROM products;
```

Agrupación con GROUP BY y HAVING

- El comando GROUP BY organiza los datos en grupos basados en una o más columnas, mientras que HAVING permite **filtrar** estos grupos basados en condiciones.
- **GROUP BY**: Ayuda a segmentar los datos para calcular estadísticas dentro de cada grupo.
- **HAVING**: Funciona como un filtro para los datos agrupados. A diferencia de WHERE, que filtra antes de agrupar, HAVING se usa después de aplicar funciones de agregación como COUNT o SUM.

Crombie

Agrupación con GROUP BY y HAVING

- ➔ GROUP BY: Agrupa datos en subconjuntos.
- ➔ HAVING: Filtra resultados después de la agrupación.

```
SELECT category, COUNT(*) AS ProductCount
FROM products
GROUP BY category
HAVING COUNT(*) > 5;
```

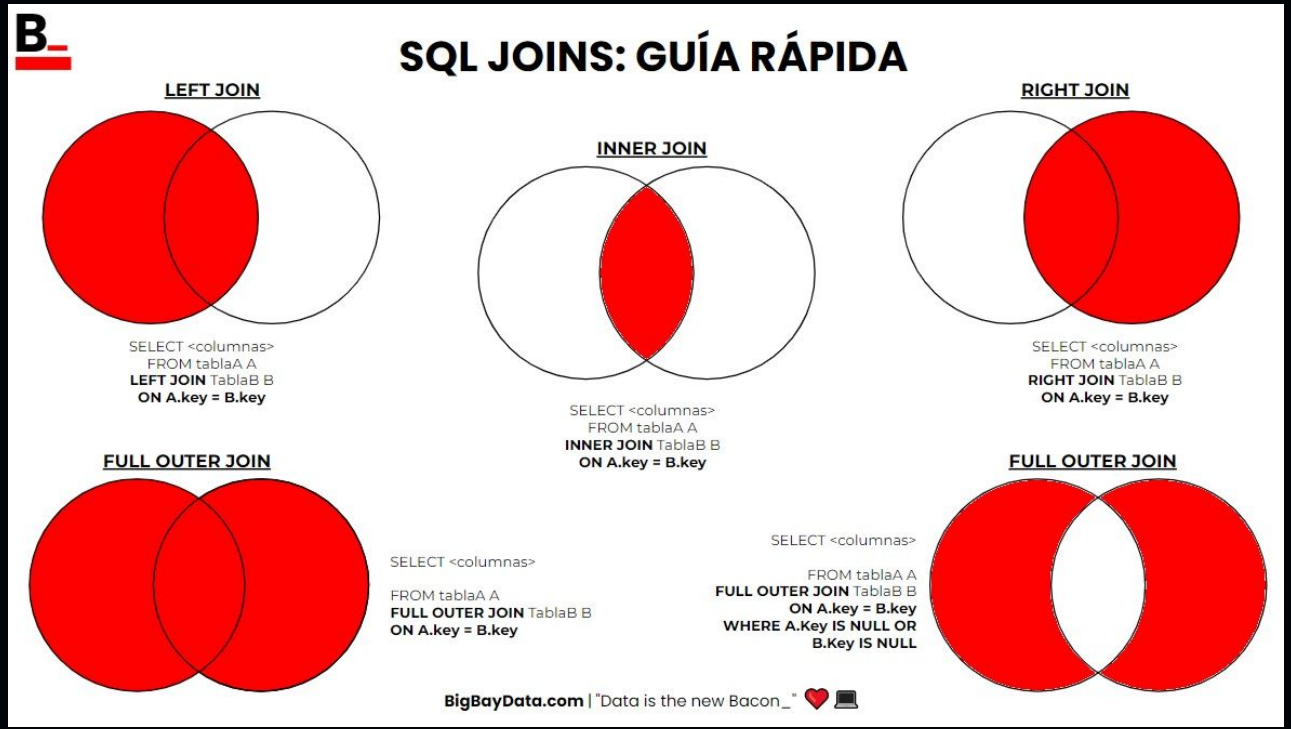
Crombie

Joins Básicos

- ➔ Relaciones entre Tablas: Conectar datos de múltiples tablas.
- ➔ Tipos de Joins:
  - INNER JOIN: Muestra coincidencias en ambas tablas.
  - LEFT JOIN: Todos los datos de la izquierda y coincidencias de la derecha.
  - RIGHT JOIN: Todos los datos de la derecha y coincidencias de la izquierda.
  - FULL OUTER JOIN: combina las filas de **dos tablas**. Pueden ser todas las que coincidan dada un criterio específico. O todas las que no, completando NULL.

```
SELECT
customers.name AS CustomerName,
orders.order_date
FROM customers
INNER JOIN orders ON customers.id = orders.customer_id;
```

Joins Básicos



Crombie

Diseño de Bases de Datos Relacionales

- ➔ El diseño de bases de datos relacionales asegura que los datos estén organizados de manera eficiente y sin redundancia. Los principios de normalización ayudan a estructurar las tablas para que sean consistentes y fáciles de mantener.

- ➔ 1FN (Primera Forma Normal): Los datos deben estar organizados en tablas y cada celda debe contener un solo valor. Por ejemplo, evitar columnas con listas de valores.
- ➔ 2FN (Segunda Forma Normal): Cada atributo no clave debe depender completamente de la clave primaria. Esto elimina dependencias parciales.
- ➔ 3FN (Tercera Forma Normal): Elimina dependencias transitivas, asegurando que cada atributo dependa solo de la clave primaria.

Diseño de Bases de Datos Relacionales

- Normalización:
  - 1NF: Eliminar datos repetidos.
  - 2NF: Dividir datos en múltiples tablas.
  - 3NF: Evitar dependencias no clave.
- Claves:
  - Primaria: Identifica filas únicas.
  - Foránea: Relaciona tablas.

Nivel de normalización	**Beneficio principal**
Sin normalizar	Datos en un solo lugar, pero redundante y problemático.
1FN	Cada celda contiene un solo valor.
2FN	Se eliminan dependencias parciales.
3FN	Se eliminan dependencias transitivas

Consultas Avanzadas

- **Subconsultas**: Son consultas anidadas dentro de otra consulta. Se usan para realizar operaciones más específicas, como encontrar clientes que hayan realizado pedidos recientes.

```
SELECT name
FROM customers
WHERE id IN (SELECT customer_id FROM orders WHERE order_date > '2024-01-01');
```

Crombie

Consultas Avanzadas

- **Uniones**: UNION combina resultados de varias consultas en una sola tabla, eliminando duplicados por defecto. INTERSECT devuelve datos comunes entre las consultas.

- **CASE**: Permite incluir lógica condicional en las consultas para categorizar o transformar datos.

```
SELECT name FROM customers
UNION
SELECT name FROM suppliers;
```

```
SELECT name,
      CASE
        WHEN total_spent > 1000 THEN 'VIP'
        ELSE 'Regular'
      END AS CustomerType
FROM customers;
```

Crombie

Práctica grupal

Encontrar al cliente **premium**:
Identificar a los clientes con un **gasto total mayor a $1500**, utilizando las funciones de agregación y JOINs.
Utilizando un **JOIN, GROUP BY, y HAVING**, escribe una consulta que obtenga el nombre y la ciudad de los clientes cuyo gasto total (suma de **TotalPrice**) sea mayor a $1500.

Preguntas:

¿Que ideas se les ocurren?

¿Qué tablas necesitamos?

¿Cuales son las relaciones?

Práctica

- Actividad 1: Análisis de Ventas
  - Usar funciones de agregación para obtener información como:
  - Número de productos vendidos por categoría.
  - Total de ingresos generados por cliente.

- Actividad 2: Relacionar Tablas
  - Usar un JOIN para combinar datos de las tablas orders y products.
  - Encontrar qué producto generó más ingresos: