Clase 7: Bases de datos

2024

SQL

1. GROUP BY
Descripción:
La cláusula GROUP BY se utiliza en SQL para agrupar filas que tienen los mismos valores en columnas especificadas. Es útil principalmente en consultas que utilizan funciones de agregación (como SUM, AVG, MAX, MIN, COUNT) para realizar cálculos o resúmenes de los datos agrupados.

```
-- Calcular el número total de libros vendidos por categoría
SELECT Categoria, COUNT(*) AS TotalLibros
FROM Libros
JOIN Ventas ON Libros.ID_Libro = Ventas.ID_Libro
GROUP BY Categoria;
```

SQL

2. DISTINCT
Descripción:
DISTINCT se utiliza para eliminar duplicados en los resultados de una consulta. Esta cláusula es muy útil cuando solo quieres ver valores únicos en una columna o un conjunto de columnas.

```
-- Listar todos los autores únicos en la base de datos
SELECT DISTINCT Autor
FROM Libros;
```

SQL

3. HAVING
Descripción:
HAVING es similar a WHERE, pero se utiliza para filtrar los datos agrupados creados por GROUP BY. HAVING solo se puede utilizar en consultas que también usan GROUP BY.

```
-- Encontrar categorías cuyos libros tengan un precio promedio superior a 15 euros
SELECT Categoria, AVG(Precio) AS PrecioPromedio
FROM Libros
GROUP BY Categoria
HAVING AVG(Precio) > 15;
```

SQL

4. TOP
Descripción:
TOP se utiliza para especificar el número de registros a retornar en una consulta. Es muy útil cuando necesitas una muestra de datos o los primeros registros según un criterio específico. Nota: TOP es específico de algunos sistemas como Microsoft SQL Server. En otros sistemas como MySQL, se utilizaría LIMIT.

```
-- Obtener los tres libros más caros
SELECT TOP 3 *
FROM Libros
ORDER BY Precio DESC;
```

Ejercicios SQL

Ejercicio 1: Agrupación y Suma
Enunciado:
Escribe una consulta SQL que muestre el total de ventas (en euros) por cada categoría de libros. Utiliza GROUP BY para agrupar los libros por categoría y SUM para calcular el total de ventas.

SQL

```
SELECT Categoria, SUM(Precio *
Cantidad) AS TotalVentas
FROM Libros
JOIN Ventas ON Libros.ID_Libro =
Ventas.ID_Libro
GROUP BY Categoria;
```

SQL

Ejercicio 2: Uso de DISTINCT
Enunciado:
Genera una lista única de ciudades de los clientes que han comprado libros. Utiliza la cláusula DISTINCT para asegurarte de que cada ciudad solo aparezca una vez en la lista.

SQL

```
SELECT DISTINCT Ciudad
FROM Clientes;
```

);

Ejercicio 3: Filtrado con HAVING

Enunciado:

Encuentra las categorías de libros cuyo número total de copias vendidas exceda 3 unidades. Debes utilizar GROUP BY para agrupar los resultados por categoría y HAVING para filtrar los grupos.

SQL

```
SELECT Categoria, SUM(Cantidad) AS
TotalCopiasVendidas
FROM Ventas
JOIN Libros ON Ventas.ID_Libro =
Libros.ID_Libro
GROUP BY Categoria
HAVING SUM(Cantidad) > 3;
```

SQL

Ejercicio 4: Consulta con TOP

Enunciado: Selecciona los detalles de las cinco ventas más grandes en términos de cantidad de libros vendidos. Ordena los resultados de mayor a menor y utiliza la cláusula TOP para limitar los resultados.

SQL

```
SELECT TOP 5 ID_Venta, ID_Libro, ID_Cliente,
Cantidad
FROM Ventas
ORDER BY Cantidad DESC;
```

SQL

Ejercicio 5: Uso Combinado de Joins y Agregados **Enunciado:** Crea una consulta que muestre el nombre y el email de cada cliente junto con el total de dinero gastado en compras. Deberás sumar los precios de los libros multiplicados por las cantidades compradas y unir las tablas adecuadas para obtener los nombres y los emails de los clientes.	

SQL

```
SELECT Clientes.Nombre, Clientes.Email,
SUM(Libros.Precio * Ventas.Cantidad) AS
TotalGastado
FROM Ventas
JOIN Clientes ON Ventas.ID_Cliente =
Clientes.ID_Cliente
JOIN Libros ON Ventas.ID_Libro =
Libros.ID_Libro
GROUP BY Clientes.Nombre, Clientes.Email;
```

SQL

Ejercicio 6: Consulta Compleja con Joins **Enunciado:** Escribe una consulta que utilice INNER JOIN, LEFT JOIN, y RIGHT JOIN para mostrar todos los libros que han sido reservados al menos una vez, incluyendo el nombre del cliente que hizo la reserva y la fecha de la reserva. Si un libro no ha sido reservado, debería aparecer con el valor NULL en los campos de cliente y fecha.	

SQL

```
SELECT Libros.Titulo, Clientes.Nombre AS
NombreCliente, Ventas.Fecha AS
FechaReserva
FROM Libros
LEFT JOIN Ventas ON Libros.ID_Libro =
Ventas.ID_Libro
LEFT JOIN Clientes ON Ventas.ID_Cliente =
Clientes.ID_Cliente;
```

SQL

Ejercicio 7: Suma y Filtro

Enunciado: Determina qué autor ha generado más ingresos por ventas de sus libros. Agrupa los resultados por autor y utiliza SUM para calcular el total de ingresos, luego ordena el resultado para obtener el autor con mayores ingresos.

SQL

```
SELECT Autor, SUM(Libros.Precio *
Ventas.Cantidad) AS TotalIngresos
FROM Libros
JOIN Ventas ON Libros.ID_Libro =
Ventas.ID_Libro
GROUP BY Autor
ORDER BY TotalIngresos DESC;
```

MUCHAS GRACIAS!

inacap.cl