Curso: Nota:

# ACTIVIDAD

2: Traer 5 ejemplos de cada una de las funciones que explicaron y traer ejemplos de cada una de las exposiciones, todo escrito para la proxima clase.

## Funciones SQL

### GROUP BY

1º Contar el número de clientes por ciudad.

```
SELECT ciudad, COUNT(cliente_id) AS total_clientes
FROM clientes
GROUP BY ciudad;
```

2º Calcular el promedio de ventas por vendedor

```
SELECT vendedor_id, AVG(monto_venta) AS promedio_ventas
FROM ventas
GROUP BY vendedor_id;
```

3º Obtener el total de productos en stock por categoria

```
SELECT categoria, SUM(cantidad_stock) AS total_stock
FROM productos
GROUP BY categoria;
```

4º Encontrar el precio máximo de un producto por proveedor

```
SELECT proveedor_id, MAX(precio_unitario) AS precio_maximo
FROM productos
GROUP BY proveedor_id;
```

5º Listar el número de pedidos por fecha

```
SELECT fecha_pedido, COUNT(pedido_id) AS numero_pedidos
FROM pedidos
GROUP BY fecha_pedido;
```

## JOIN

1º obtener los nombres de clientes y sus pedidos

```
SELECT c.nombre_cliente, p.pedido_id, p.fecha_pedido
FROM clientes c
INNER JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

2º Listar los productos y las categorías a las que pertenece

```
SELECT pr.nombre_producto, ca.nombre_categoria
FROM Productos pr
INNER JOIN categorias ca ON pr.categoria_id = ca.categoria_id;
```

3º Mostrar los empleados y los departamentos en los que trabajan

```
SELECT e.nombre_empleado, d.nombre_departamento
FROM empleados e
INNER JOIN departamentos d ON e.departamento_id =
d.departamento_id;
```

4º Ver los de las ventas junto con los nombres de los productos vendidos.

```
SELECT v.venta_id, pr.nombre_producto, dv.cantidad,
dv.precio_unitario
FROM ventas v
INNER JOIN detalles_venta dv ON v.venta_id = dv.venta_id
INNER JOIN productos pr ON dv.producto_id = pr.producto_id;
```

5º ~~(Hattin)~~ Relacionar autores con sus libros

```
SELECT a.nombre_autor, l.titulo_libro
FROM autores a
INNER JOIN libros l ON a.autor_id = l.autor_id;
```

## HAVING

1º clientes con más de 3 pedidos

```
SELECT cliente_id, COUNT(pedido_id) AS total_pedidos
FROM pedidos
GROUP BY cliente_id
HAVING COUNT(pedido_id) > 3;
```

2º Categorías con un stock total superior a 100 unidades

```
SELECT categoria, SUM(cantidad_stock) AS stock_total
FROM Productos
GROUP BY Categoria
HAVING SUM(cantidad_stock) > 100;
```

3º Vendedores cuyas ventas promedio superan los $500

```
SELECT vendedor_id, AVG (monto_venta) AS promedio_ventas
FROM Ventas
GROUP BY vendedor_id
HAVING AVG (monto_venta) > 500;
```

4º Ciudades con un número de clientes entre 5 y 10

```
SELECT ciudad, COUNT (cliente_id) AS numero_clientes
FROM clientes
GROUP BY ciudad
HAVING COUNT (cliente_id) BETWEEN 5 AN 10;
```

5º Departamentos con un salario promedio de empleados superior a $60000

```
SELECT departamento_id, AVG (salario) AS salario_promedio
FROM empleados
GROUP BY departamento_id
HAVING AVG (salario) > 60000;
```

LIMIT

1º Obtener los 10 clientes más recientes

```
SELECT *
FROM clientes
ORDER BY fecha-registro DESC
LIMIT 10;
```

2º Mostrar los 5 más caros,

```
SELECT nombre_producto, precio_unitario
FROM Productos
ORDER BY precio_unitario DESC
LIMIT 5;
```

3º Obtener 3 pedidos aleatorios

```
SELECT *
FROM pedidos
ORDER BY RAND()
LIMIT 3;
```

4º Listar los primeros 20 resultados de una búsqueda

```
SELECT *
FROM articulos
WHERE titulo LIKE '%SQL%'
LIMIT 20;
```

5º Recuperar el empleado con el salario más bajo

```
SELECT nombre_empleado, salario
FROM empleados
ORDER BY salario ASC
LIMIT 1;
```

SELECT

1º seleccionar todas las columnas de la tabla 'productos'

```
SELECT *
FROM productos;
```

2º seleccionar el nombre y correo electronico de los clientes

```
SELECT nombre_cliente, email
FROM clientes;
```

3º seleccionar productos cuyo precio sea mayor a $50

```
SELECT nombre_producto, precio_unitario
FROM productos
WHERE precio_unitario > 50;
```

4º Contar el número total de empleados

```
SELECT COUNT (*) AS total_empleados
FROM empleados;
```

5º seleccionar los distintos paises de los proveedores

```
SELECT DISTINCT pais
FROM proveedores;
```

ORDER BY

1º Ordenas clientes por nombres alfabeticamente

```
SELECT nombre_cliente, ciudad
FROM clientes
ORDER BY nombre_cliente ASC;
```

2º Ordenar productos por precio de mayor a menor

```
SELECT nombre_producto, precio_unitario
FROM Productos
ORDER BY precio_unitario DESC;
```

3. Ordenar pedidos por fecha, y luego por cliente_id

```
SELECT pedido_id, fecha_pedido, cliente_id
FROM Pedidos
ORDER BY fecha_pedido ASC, cliente_id DESC;
```

4. Empleados ordenados por salario de forma descendente

```
SELECT nombre_empleado, salario
FROM empleados
ORDER BY salario DESC;
```

5. Libros ordenados por título de la A a la z

```
SELECT titulo_libros, autor_id
FROM Libros
ORDER BY titulo_libro ASC;
```

COUNT

1. Contar el número total de productos

```
SELECT COUNT(*) AS total_productos
FROM productos;
```

2. Contar el número de pedidos realizados por el Cliente con ID 101

```
SELECT COUNT (pedido_id) AS pedidos_cliente_101
FROM pedidos
WHERE cliente_id = 101;
```

3. Contar cuantos empleados tienen un cargo 'de 'Gerente'

```
SELECT COUNT (empleado_id) AS numeros_gerentes
FROM empleados
WHERE cargo = 'Gerente';
```

4. Contar el numero de ciudades unicas en la tabla de clientes

```
SELECT COUNT (DISTINCT ciudad) AS ciudades_unicas
FROM clientes;
```

5. Contar el numero de ventas con un monto superior a $1000

```
SELECT COUNT (venta_id) AS ventas_grandes
FROM Ventas
WHERE monto_venta >1000
```

## SUM

1. Calcular el total de ventas de la tabla 'ventas'

```
SELECT SUM (monto_venta) AS ventab_totales
FROM Ventas;
```

2. Calcular el total de stock de productos en la categoria 'electronica'

```
SELECT SUM (cantidad_stock) As stock_electronica
FROM productos
WHERE categoria = 'electronica';
```

3. Sumar el salario de todos los empleados del departamento de 'Ventas'

```
SELECT SUM (salario) AS total_salarios_ventas
FROM empleados
WHERE departamento = 'Ventas';
```

4. Calcular el total de los precios unitarios de los productos

```
SELECT SUM (precio_unitario) AS suma_precios_unitarios
FROM productos;
```

5. Obtener la suma de las cantidades de productos en un pedido especifico (ID 2001)

```
SELECT SUM (cantidad) AS cantidad_total_pedido_2001
FROM detalles_pedido
WHERE pedido_id = 2001;
```

## AVG

1. Calcular el precio promedio de los productos

```
SELECT AVG (precio_unitario) AS precio_promedio
FROM productos;
```

2. Calcular el monto promedio de los pedidos

```
SELECT AVG (monto_total) AS promedio_monto_pedidos
FROM pedidos;
```

3. obtener el salario promedio de los empleados

```
SELECT AVG (salario) AS salario_promedio_empleados
FROM empleados;
```

4. Calcular el promedio de calificación de los productos

```
SELECT AVG (calificacion) AS calificacion_promedio
FROM productos
WHERE calificacion IS NOT NULL;
```

DISTINCT

1º Listar las ciudades únicas donde residen los clientes

```
SELECT DISTINCT ciudad
FROM clientes;
```

2º Obtener la lista de las categorías de productos sin duplicación

```
SELECT DISTINCT categoría
FROM productos;
```

3º Mostrar los cargos únicos de los empleados

```
SELECT DISTINCT cargo
FROM empleados;
```

4º Listar los países de los proveedores sin repeticiones

```
SELECT DISTINCT país
FROM proveedores;
```

5º Ver las fechas únicas en las que se realizaron pedidos

```
SELECT DISTINCT fecha_pedido
FROM pedidos;
```

## TEMAS DE LAS EXPOSICIONES

- **llave primaria:** Es una columna o conjunto de columnas en una tabla que contiene valores únicos para cada fila.

Ejemplo:

cliente_id en una tabla de clientes:

```
CREATE TABLE Clientes (
    cleente_id  INT PRIMARY KEY,
    nombre      VARCHAR (100),
    email       VARCHAR (100) UNIQUE
);
```

- **llave foranea:** es una columna o conjunto de columnas en una tabla que hace referencia a la llave primaria de otra tabla.

Ejemplo

cliente_id en la tabla pedidos referenciado a cliente_id en clientes

```
CREATE TABLE Pedidos (

    pedido_id   INT PRIMARY KEY,
    cliente_id  INT,
    fecha_pedido DATE,
    FOREIGN KEY (cliente_id) REFERENCES clientes
    (cliente_id

);
```

- **TRUNCATE:** se utiliza para eliminar rápidamente todas las filas de una tabla, dejando la estructura de la tabla intacta.

Ejemplo

Vaciar Completamente la tabla logs

```
TRUNCATE TABLE Lgs;
```

- **DELETE:** se utiliza para eliminar una o más filas existentes de una tabla.

Ejemplo

Eliminar un cliente específico por su ID

```
DELETE FROM Clientes
WHERE cliente_id = 105;
```

- **UPDATE:** se utiliza para modificar los datos existentes en una tabla.

Ejemplo

Actualizar el correo electrónico de un cliente especefico

```
UPDATE Clientes
SET email = 'nuevo.correo@example.com'
WHERE cliente_id = 101;
```