TEMA 2.3

Por Eduardo Felipe Poot Chairez

Se tienen 2 tablas:

```
CREATE TABLE categorias (
    id_categoria INT PRIMARY KEY,
    nombre_categoria VARCHAR(50)
);

CREATE TABLE productos (
    id_producto INT PRIMARY KEY,
    id_categoria INT,
    nombre_producto VARCHAR(100),
    precio DECIMAL(10, 2),
    stock INT,
    FOREIGN KEY (id_categoria) REFERENCES categorias(id_categoria)
);
```

Se insertan datos en sus respectivas tablas:

```
INSERT INTO categorias VALUES
(1, 'Electrónica'),
(2, 'Ropa'),
(3, 'Alimentos'),
(4, 'Juguetes');

INSERT INTO productos VALUES
(1, 1, 'Laptop', 18000.00, 5),
(2, 1, 'Mouse', 350.00, 20),
(3, 2, 'Camisa', 450.00, 15),
(4, 3, 'Arroz', 40.00, 100),
(5, NULL, 'Producto sin categoría', 999.00, 1);
```

- SELECT que relacione dos tablas por medio de clave primaria y clave foránea (No utilizar JOIN).

SELECT categorias.nombre_categoria, productos.nombre_producto, productos.precio, productos.stock
FROM categorias, productos
WHERE categorias.id_categoria = productos.id_categoria;

nombre_categoria	nombre_producto	precio	stock
Electrónica	Laptop	18000.00	5
Electrónica	Mouse	350.00	20
Ropa	Camisa	450.00	15
Alimentos	Arroz	40.00	100

- SELECT con BETWEEN

Consulta para ver los productos que cuestan entre 300 y 1000.

SELECT *
FROM productos
WHERE precio BETWEEN 300 AND 1000;

id_producto	id_categoria	nombre_producto	precio	stock
2	1	Mouse	350.00	20
3	2	Camisa	450.00	15

- SELECT con LIKE

Consulta para ver los productos que comiencen con L.

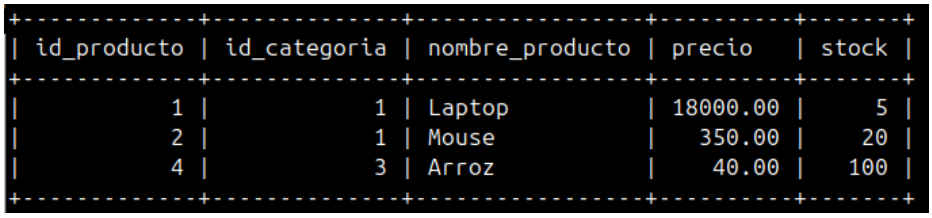
SELECT *
FROM productos
WHERE nombre_producto LIKE 'L%';

id_producto	id_categoria	nombre_producto	precio	stock
1	1	Laptop	18000.00	5

- SELECT con IN

Consulta para determinar los productos que tengan como id de categoría 1 o 3 (Se usa el IN en vez del OR para optimizar la escritura de la consulta).

```
SELECT *
FROM productos
WHERE id_categoria IN (1, 3);
```



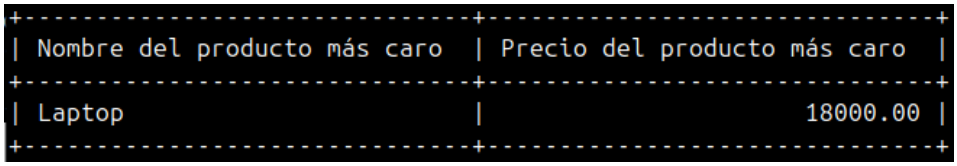
id_producto	id_categoria	nombre_producto	precio	stock
1	1	Laptop	18000.00	5
2	1	Mouse	350.00	20
4	3	Arroz	40.00	100

- SELECT con MAX

La función de agregación `MAX` se utiliza para determinar el valor máximo dentro de un conjunto de registros.

En esta consulta, se obtiene el precio más alto registrado en la tabla de productos:

```
SELECT nombre_producto AS 'Nombre del producto más caro', precio AS 'Precio del producto más caro'
FROM productos
WHERE precio = (SELECT MAX(precio) FROM productos);
```

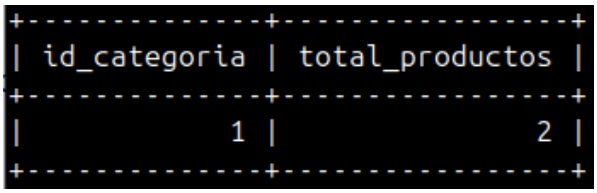


Nombre del producto más caro	Precio del producto más caro
Laptop	18000.00

- SELECT con HAVING

El HAVING funciona como el WHERE pero para datos agrupados y en este caso la consulta determina qué id de categoría tiene más de un producto (esto se hace con el COUNT que cuenta el número de registros).

```
SELECT id_categoria, COUNT(*) AS total_productos
FROM productos
GROUP BY id_categoria
HAVING COUNT(*) > 1;
```

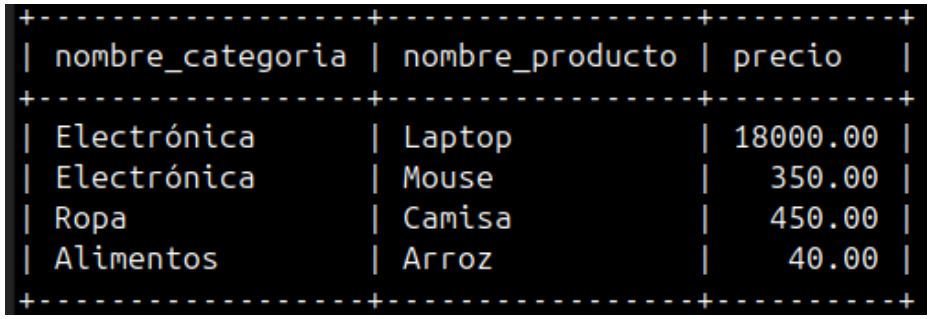


id_categoria	total_productos
1	2

- INNER JOIN

Consulta para mostrar todas las categorías , puesto que el `INNER JOIN` permite combinar registros de dos tablas únicamente cuando existe coincidencia entre las claves relacionadas.

```
SELECT categorias.nombre_categoria, productos.nombre_producto, productos.precio
FROM categorias
INNER JOIN productos
ON categorias.id_categoria = productos.id_categoria;
```

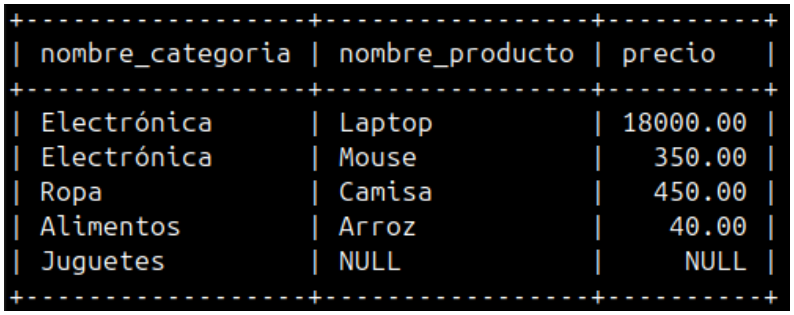


nombre_categoria	nombre_producto	precio
Electrónica	Laptop	18000.00
Electrónica	Mouse	350.00
Ropa	Camisa	450.00
Alimentos	Arroz	40.00

- LEFT JOIN

Consulta para mostrar todas las categorías aunque no tengan productos, ya que el `LEFT JOIN` devuelve todos los registros de la tabla izquierda, aun cuando no exista una coincidencia en la tabla derecha.

```
SELECT categorias.nombre_categoria, productos.nombre_producto, productos.precio
FROM categorias
LEFT JOIN productos
ON categorias.id_categoria = productos.id_categoria;
```

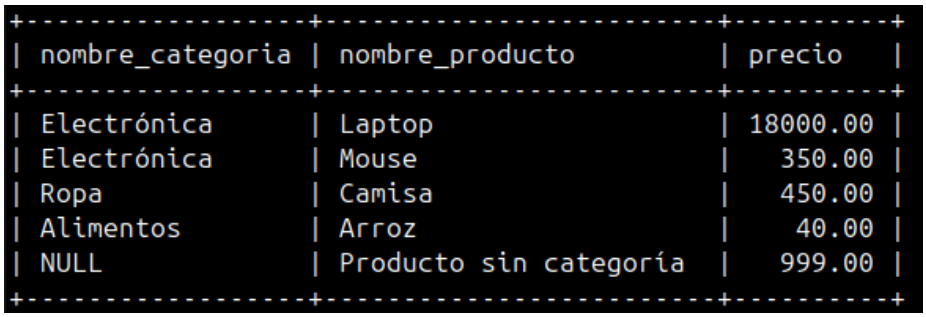


nombre_categoria	nombre_producto	precio
Electrónica	Laptop	18000.00
Electrónica	Mouse	350.00
Ropa	Camisa	450.00
Alimentos	Arroz	40.00
Juguetes	NULL	NULL

- RIGHT JOIN

Consulta para mostrar todos los productos aunque no tengan categoría, puesto que el `RIGHT JOIN` funciona de manera similar al `LEFT JOIN`, pero prioriza los registros de la tabla derecha.

```
SELECT categorias.nombre_categoria, productos.nombre_producto, productos.precio
FROM categorias
RIGHT JOIN productos
ON categorias.id_categoria = productos.id_categoria;
```

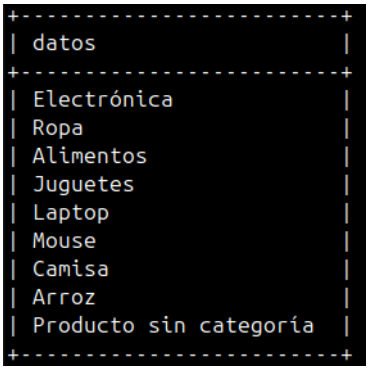


nombre_categoria	nombre_producto	precio
Electrónica	Laptop	18000.00
Electrónica	Mouse	350.00
Ropa	Camisa	450.00
Alimentos	Arroz	40.00
NULL	Producto sin categoría	999.00

- UNION

La operación `UNION` permite combinar los resultados de dos consultas en una sola columna, eliminando registros duplicados.
En esta consulta se muestran los nombres de los datos de la tabla de categoría y productos en una sola columna.

```
SELECT nombre_categoria AS datos
FROM categorias
UNION
SELECT nombre_producto
FROM productos;
```

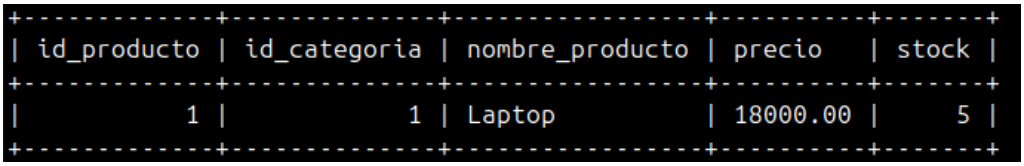


datos
Electrónica
Ropa
Alimentos
Juguetes
Laptop
Mouse
Camisa
Arroz
Producto sin categoría

- SUBCONSULTA

En esta consulta se utiliza una subconsulta para calcular el precio promedio de los productos mediante la función `AVG`.

```
SELECT *
FROM productos
WHERE precio > (SELECT AVG(precio)
FROM productos);
```



id_producto	id_categoria	nombre_producto	precio	stock
1	1	Laptop	18000.00	5