Taller Mysql

Este taller está diseñado para profundizar en el manejo y optimización de bases de datos MySQL. A través de ejercicios prácticos, se explorarán temas avanzados para reforzar el conocimiento en normalización, joins, consultas complejas, subconsultas, procedimientos almacenados, funciones definidas por el usuario y triggers.

BASE DE DATOS:

```
-- Creación de la base de datos
CREATE DATABASE vtaszfs;
USE vtaszfs;
-- Tabla Clientes
CREATE TABLE Clientes (
id INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100),
email VARCHAR(100) UNIQUE
    );
-- Tabla UbicacionCliente
CREATE TABLE UbicacionCliente (
id INT PRIMARY KEY AUTO_INCREMENT,
cliente_id INT,
direccion VARCHAR(255),
ciudad VARCHAR(100),
estado VARCHAR(50),
codigo_postal VARCHAR(10),
pais VARCHAR(50),
FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);
-- Tabla Empleados
CREATE TABLE Empleados (
id INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100),puesto VARCHAR(50),
salario DECIMAL(10, 2),
fecha_contratacion DATE
);
-- Tabla Proveedores
CREATE TABLE Proveedores (
id INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100),
contacto VARCHAR(100),
telefono VARCHAR(20),
direccion VARCHAR(255)
);
-- Tabla TiposProductos
CREATE TABLE TiposProductos (
id INT PRIMARY KEY AUTO_INCREMENT,
tipo_nombre VARCHAR(100),
descripcion TEXT
);
-- Tabla Productos
CREATE TABLE Productos (
id INT PRIMARY KEY AUTO_INCREMENT,
```

```
nombre VARCHAR(100),
precio DECIMAL(10, 2),
proveedor_id INT,
tipo_id INT,
FOREIGN KEY (proveedor_id) REFERENCES Proveedores(id),
FOREIGN KEY (tipo_id) REFERENCES TiposProductos(id)
);
-- Tabla Pedidos
CREATE TABLE Pedidos (
id INT PRIMARY KEY AUTO_INCREMENT,
cliente_id INT,
fecha DATE,
total DECIMAL(10, 2),
FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);
-- Tabla DetallesPedido
CREATE TABLE DetallesPedido (
id INT PRIMARY KEY AUTO_INCREMENT,
pedido_id INT,
producto_id INT,
cantidad INT,
precio DECIMAL(10, 2),
FOREIGN KEY (pedido_id) REFERENCES Pedidos(id),
FOREIGN KEY (producto_id) REFERENCES Productos(id)
);
```

Normalización

1. Crear una tabla HistorialPedidos que almacene cambios en los pedidos.

```
CREATE TABLE historialpedidos(
id INT AUTO_INCREMENT PRIMARY KEY,
id_pedidos INT,
fecha_cambio DATE,
total_cambio DECIMAL (10, 2),
id_empleado INT,
CONSTRAINT id_pedidos_fk FOREIGN KEY (id_pedidos) REFERENCES Pedidos(id),
CONSTRAINT id_empleados_fk FOREIGN KEY (id_empleados) REFERENCES Empleados(id)

);
```

2. Evaluar la tabla Clientes para eliminar datos redundantes y normalizar hasta 3NF.

```


CREATE TABLE TipoDocumento (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(50) UNIQUE
);

CREATE TABLE TipoEmail (
id INT PRIMARY KEY AUTO_INCREMENT,
```

```
tipo_email VARCHAR(50) UNIQUE
);

CREATE TABLE Clientes (
id INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100),
tipo_documento_id INT,
tipo_email_id INT,
CONSTRAINT fk_tipo_documento FOREIGN KEY (tipo_documento_id) REFERENCES
TipoDocumento(id),
CONSTRAINT fk_tipo_email_id FOREIGN KEY (tipo_email_id) REFERENCES TipoEmail(id)
);
```

3. Separar la tabla Empleados en una tabla de DatosEmpleados y otra para Puestos .

```
CREATE TABLE Puesto (
id INT AUTO_INCREMENT PRIMARY KEY,
nombre_puesto VARCHAR(50)

)

CREATE TABLE DatosEmpleados (
id INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(50),
salario DECIMAL(10,2),
fecha_contratacion DATE,
id_puesto INT,
CONSTRAINT id_puesto_fk FOREIGN KEY (id_puesto) REFERENCES Puesto(id)

);
```

4. Revisar la relación Clientes y UbicacionCliente para evitar duplicación de datos.

Aquí debemos hacer tablas nuevas para ciudad, estado y país, esto con el fin de manejar varias direcciones si el cliente posee varias.

```
CREATE TABLE UbicacionCliente (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT,
  direccion VARCHAR(200),
  ciudad_id INT,
  codigo_postal VARCHAR(10),
  CONSTRAINT fk_cliente FOREIGN KEY (cliente_id) REFERENCES Clientes(id),
  CONSTRAINT fk_ciudad FOREIGN KEY (ciudad_id) REFERENCES Ciudades(id)
);


CREATE TABLE Paises (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) UNIQUE
);
```

```
CREATE TABLE Estados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50),
  pais_id INT,
  CONSTRAINT fk_pais FOREIGN KEY (pais_id) REFERENCES Paises(id)
);


CREATE TABLE Ciudades (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50),
  estado_id INT,
  CONSTRAINT fk_estado FOREIGN KEY (estado_id) REFERENCES Estados(id)
);
```

5. Normalizar Proveedores para tener ContactoProveedores en otra tabla.

```
CREATE TABLE Proveedores (
id INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(50),
direccion VARCHAR(255)
);

CREATE TABLE ContactoProveedores(
id INT AUTO_INCREMENT PRIMARY KEY,
id_proveedor INT,
nombre_contacto VARCHAR(50),
telefono VARCHAR(20),
ciudades_id INT,
CONSTRAINT fk_ciudades_id FOREIGN KEY(ciudades_id) REFERENCES Ciudades(id),
CONSTRAINT fk_proveedor_id FOREIGN KEY (id_proveedor) REFERENCES Proveedores(id)

);
```

6. Crear una tabla de Telefonos para almacenar múltiples números por cliente.

```
CREATE TABLE telefonos (
id INT AUTO_INCREMENT PRIMARY KEY,
telefono VARCHAR(20),
id_cliente INT,
CONSTRAINT fk_id_cliente FOREIGN KEY (id_cliente) REFERENCES Clientes(id)
);
```

7. Transformar TiposProductos en una relación categórica jerárquica.

```
CREATE TABLE Categorias (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre_categoria VARCHAR(100)
);

CREATE TABLE Subcategorias (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  categoria_id INT,
  nombre_subcategoria VARCHAR(100),
  descripcion TEXT,
  CONSTRAINT fk_categoria FOREIGN KEY (categoria_id) REFERENCES Categorias(id)
);


CREATE TABLE Productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  precio DECIMAL(10, 2),
  proveedor_id INT,
  subcategoria_id INT,
  CONSTRAINT fk_proveedor FOREIGN KEY (proveedor_id) REFERENCES Proveedores(id),
  CONSTRAINT fk_subcategoria FOREIGN KEY (subcategoria_id) REFERENCES
Subcategorias(id)
);
```

8. Normalizar Pedidos y DetallesPedido para evitar inconsistencias de precios.

```
CREATE TABLE Pedidos (
id INT PRIMARY KEY AUTO_INCREMENT,
cliente_id INT,
fecha DATE,
total DECIMAL(10, 2),
FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);

CREATE TABLE DetallesPedido (
id INT PRIMARY KEY AUTO_INCREMENT,
pedido_id INT,
producto_id INT,
cantidad INT,
precio_unidad DECIMAL(10, 2),
FOREIGN KEY (pedido_id) REFERENCES Pedidos(id),
FOREIGN KEY (producto_id) REFERENCES Productos(id)
);
```

9. Usar una relación de muchos a muchos para Empleados y Proveedores

```
CREATE TABLE EmpleadosProveedores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  empleado_id INT,
  proveedor_id INT,
  rol VARCHAR(50),
  CONSTRAINT fk_empleado FOREIGN KEY (empleado_id) REFERENCES DatosEmpleados(id),
  CONSTRAINT proveedor_fk FOREIGN KEY (proveedor_id) REFERENCES Proveedores(id)
);
```

10. Convertir la tabla UbicacionCliente en una relación genérica de Ubicaciones .

```
CREATE TABLE Ubicaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  direccion VARCHAR(200),
  ciudad VARCHAR(80),
  estado VARCHAR(50),
  codigo_postal VARCHAR(10),
  pais VARCHAR(50)
);

CREATE TABLE UbicacionEntidad (
  id INT AUTO_INCREMENT PRIMARY KEY,
  ubicacion_id INT,
  tipo_entidad ENUM('Cliente', 'Proveedor', 'Empleado'),
  entidad_id INT,
  CONSTRAINT fk_ubicacion FOREIGN KEY (ubicacion_id) REFERENCES Ubicaciones(id)
);
```

INGRESO DE DATOS:

Se ingresaron solo 5 usuarios, cada uno con diferente tipo de documento de identidad y de email.

```
INSERT INTO TipoDocumento (tipo) VALUES ('CC'),('CE'),('PASAPORTE'),('TI');
INSERT INTO TipoEmail (tipo_email) VALUES ('Personal'), ('Empresarial'),
('Otro');
INSERT INTO Clientes (nombre, tipo_documento_id, tipo_email_id)
    VALUES
    ('María Fernanda López', 1, 1),
    ('Carlos Andrés Rodríguez', 2, 2),
    ('Ana Sofía Ramírez', 3, 1),
    ('Juan Camilo Martínez', 1, 3),
    ('Laura Valentina Torres', 2, 2);
```

Ingresamos ahora las categorias de los productos:

```
INSERT INTO Categorias (nombre_categoria) VALUES ('Electrónica')('Cocina')
('Automovil')('Aseo');
```

Ingresamos las subcategorias :

```
INSERT INTO Subcategorias (nombre_subcategoria, categoria_id) VALUES
('Celulares', 1),('Televisores', 1),('Utensilios de cocina', 2),('Llantas', 3),
('Accesorios para auto', 3),('Limpieza general', 4);
```

Ingresamos los proveedores

```
INSERT INTO Proveedores (nombre) VALUES
    ('TecnoDistribuciones S.A.S'),
    ('ElectroMundo Ltda.'),
    ('Importaciones Celulares y Más'),
    ('Distribuidora Global Tech')
```

Ingresamos los productos

```
INSERT INTO Productos (nombre, precio, proveedor_id, subcategoria_id) VALUES
('iPhone 14 Pro', 5600000.00, 1, 1),('Samsung Galaxy S22', 4200000.00, 1, 1),
   ('Xiaomi Redmi Note 11', 980000.00, 1, 1),
    ('Motorola Edge 30', 1500000.00, 1, 1),
    ('Huawei Nova 11', 2200000.00, 1, 1);
    ('LG Smart TV 55"', 499000.00, 2, 2),
    ('Sony Bravia 50"', 600000.00, 2, 2),
    ('Set de cuchillos inox', 39999.00, 4, 3),
    ('Accesorios para auto - Kit limpieza', 45000.00, 4, 4)
```

Ingresamos los pedidos

```
INSERT INTO Pedidos (cliente_id, fecha, total) VALUES
(1, '2025-06-01', 1049000.00),
(2, '2025-06-03', 600000.00),
(3, '2025-06-05', 84999.00),
(4, '2025-06-07', 39999.00),
(5, '2025-06-09', 45000.00),
(2, '2025-06-10', 1140000.00),
(1, '2025-06-12', 499000.00),
(3, '2025-06-13', 159000.00),
(5, '2025-06-15', 600000.00),
(4, '2025-06-17', 165000.00);
```

Joins

1. **Obtener la lista de todos los pedidos con los nombres de clientes usando INNER JOIN .**

```
SELECT
    Pedidos.id AS pedido_id,
    Clientes.nombre AS cliente,
    Pedidos.fecha,
    Pedidos.total
FROM
    Pedidos
INNER JOIN
    Clientes ON Pedidos.cliente_id = Clientes.id;
```

```
+-----------+---------------------------+------------+------------+
| pedido_id | cliente                   | fecha      | total      |
+-----------+---------------------------+------------+------------+
|         1 | María Fernanda López      | 2025-06-01 | 1049000.00 |
|         7 | María Fernanda López      | 2025-06-12 |  499000.00 |
|         2 | Carlos Andrés Rodríguez   | 2025-06-03 |  600000.00 |
|         6 | Carlos Andrés Rodríguez   | 2025-06-10 | 1140000.00 |
|         3 | Ana Sofía Ramírez         | 2025-06-05 |   84999.00 |
|         8 | Ana Sofía Ramírez         | 2025-06-13 |  159000.00 |
|         4 | Juan Camilo Martínez      | 2025-06-07 |   39999.00 |
|        10 | Juan Camilo Martínez      | 2025-06-17 |  165000.00 |
|         5 | Laura Valentina Torres    | 2025-06-09 |   45000.00 |
|         9 | Laura Valentina Torres    | 2025-06-15 |  600000.00 |
+-----------+---------------------------+------------+------------+
10 rows in set (0,00 sec)
```

2. Listar los productos y proveedores que los suministran con INNER JOIN

```sql
SELECT
    Productos.nombre AS producto,
    Productos.precio,
    Proveedores.nombre AS proveedor
FROM
    Productos
INNER JOIN
    Proveedores ON Productos.proveedor_id = Proveedores.id;
```

```
+-----------------------------------+------------+---------------------------+
| producto                          | precio     | proveedor                 |
+-----------------------------------+------------+---------------------------+
| iPhone 14 Pro                     | 5600000.00 | TecnoDistribuciones S.A.S |
| Samsung Galaxy S22                | 4200000.00 | TecnoDistribuciones S.A.S |
| Xiaomi Redmi Note 11              |  980000.00 | TecnoDistribuciones S.A.S |
| Motorola Edge 30                  | 1500000.00 | TecnoDistribuciones S.A.S |
| Huawei Nova 11                    | 2200000.00 | TecnoDistribuciones S.A.S |
| Limpiador multiusos               |   15000.00 | TecnoDistribuciones S.A.S |
| LG Smart TV 55"                   |  499000.00 | ElectroMundo Ltda.        |
| Sony Bravia 50"                   |  600000.00 | ElectroMundo Ltda.        |
| Set de cuchillos inox             |   39999.00 | Distribuidora Global Tech |
| Accesorios para auto - Kit limpieza |   45000.00 | Distribuidora Global Tech |
+-----------------------------------+------------+---------------------------+
10 rows in set (0,00 sec)
```

3. Mostrar los pedidos y las ubicaciones de los clientes con LEFT JOIN .

```sql
SELECT
    p.id AS pedido_id,
    c.nombre AS cliente,
    p.fecha,
    p.total,
    uc.direccion,
    ci.nombre AS ciudad,
    es.nombre AS estado,
    pa.nombre AS pais,
    uc.codigo_postal
FROM Pedidos p
LEFT JOIN Clientes c ON p.cliente_id = c.id
```

```sql
LEFT JOIN UbicacionCliente uc ON uc.cliente_id = c.id
LEFT JOIN Ciudades ci ON uc.ciudad_id = ci.id
LEFT JOIN Estados es ON ci.estado_id = es.id
LEFT JOIN Paises pa ON es.pais_id = pa.id;
```

```
+-----------+-------------------------+------------+------------+-----------------------+---------------+-------------+---------------+--------------+
| pedido_id | cliente                 | fecha      | total      | direccion             | ciudad        | estado      | pais          | codigo_postal |
+-----------+-------------------------+------------+------------+-----------------------+---------------+-------------+---------------+--------------+
|         1 | María Fernanda López    | 2025-06-01 | 1049000.00 | Carrera 10 #45-67     | Bogotá        | Cundinamarca | Colombia      | 110111       |
|         2 | Carlos Andrés Rodríguez | 2025-06-03 |  600000.00 | Calle 23 #12-34       | Medellín      | Antioquia    | Colombia      | 050021       |
|         3 | Ana Sofía Ramírez       | 2025-06-05 |   84999.00 | Avenida 7 #89-12      | Los Ángeles   | California   | Estados Unidos | 44100       |
|         4 | Juan Camilo Martínez    | 2025-06-07 |   39999.00 | Calle 5 #67-89        | San Francisco | California   | Estados Unidos | 01234       |
|         5 | Laura Valentina Torres  | 2025-06-09 |   45000.00 | Boulevard Central 123 | Buenos Aires  | Buenos Aires | Argentina     | 1000         |
|         6 | Carlos Andrés Rodríguez | 2025-06-10 | 1140000.00 | Calle 23 #12-34       | Medellín      | Antioquia    | Colombia      | 050021       |
|         7 | María Fernanda López    | 2025-06-12 |  499000.00 | Carrera 10 #45-67     | Bogotá        | Cundinamarca | Colombia      | 110111       |
|         8 | Ana Sofía Ramírez       | 2025-06-13 |  159000.00 | Avenida 7 #89-12      | Los Ángeles   | California   | Estados Unidos | 44100       |
|         9 | Laura Valentina Torres  | 2025-06-15 |  600000.00 | Boulevard Central 123 | Buenos Aires  | Buenos Aires | Argentina     | 1000         |
|        10 | Juan Camilo Martínez    | 2025-06-17 |  165000.00 | Calle 5 #67-89        | San Francisco | California   | Estados Unidos | 01234       |
+-----------+-------------------------+------------+------------+-----------------------+---------------+-------------+---------------+--------------+
10 rows in set (0,00 sec)
```

4. Consultar los empleados que han registrado pedidos, incluyendo empleados sin pedidos (LEFT JOIN).

```sql
SELECT
    e.id AS empleado_id,
    e.nombre,
    COUNT(p.id) AS total_pedidos
FROM Empleados e
LEFT JOIN Pedidos p ON p.empleado_id = e.id
GROUP BY e.id, e.nombre
ORDER BY total_pedidos DESC;
```

```
+-------------+--------------------+----------------+
| empleado_id | nombre             | total_pedidos |
+-------------+--------------------+----------------+
|           1 | Ana Pérez          |              2 |
|           2 | Carlos Gómez       |              2 |
|           3 | María Rodríguez    |              2 |
|           4 | Luis Martínez      |              2 |
|           5 | Sandra López       |              2 |
+-------------+--------------------+----------------+
5 rows in set (0.00 sec)
```

5. Obtener el tipo de producto y los productos asociados con INNER JOIN .

```sql
SELECT
    sc.nombre_subcategoria AS tipo_producto,
    p.nombre AS producto,
    p.precio
FROM Productos p
INNER JOIN Subcategorias sc ON p.subcategoria_id = sc.id
ORDER BY sc.nombre_subcategoria, p.nombre;
```

tipo_producto	producto	precio
Accesorios para auto	Accesorios para auto - Kit limpieza	45000.00
Celulares	Huawei Nova 11	2200000.00
Celulares	iPhone 14 Pro	5600000.00
Celulares	Motorola Edge 30	1500000.00
Celulares	Samsung Galaxy S22	4200000.00
Celulares	Xiaomi Redmi Note 11	980000.00
Limpieza general	Limpiador multiusos	15000.00
Televisores	LG Smart TV 55"	499000.00
Televisores	Sony Bravia 50"	600000.00
Utensilios de cocina	Set de cuchillos inox	39999.00

6. Listar todos los clientes y el número de pedidos realizados con COUNT y GROUP BY

```sql
SELECT
    c.id AS cliente_id,
    c.nombre AS cliente,
    COUNT(p.id) AS total_pedidos
FROM Clientes c
LEFT JOIN Pedidos p ON p.cliente_id = c.id
GROUP BY c.id, c.nombre
ORDER BY total_pedidos DESC;
```

cliente_id	cliente	total_pedidos
1	María Fernanda López	2
2	Carlos Andrés Rodríguez	2
3	Ana Sofía Ramírez	2
4	Juan Camilo Martínez	2
5	Laura Valentina Torres	2

5 rows in set (0,00 sec)

7. Combinar Pedidos y Empleados para mostrar qué empleados gestionaron pedidos específicos

```sql
SELECT
    p.id AS pedido_id,
    p.fecha,
    p.total,
    e.id AS empleado_id,
    e.nombre AS empleado
FROM Pedidos p
INNER JOIN Empleados e ON p.empleado_id = e.id
ORDER BY p.fecha DESC;
```

pedido_id	fecha	total	empleado_id	empleado
10	2025-06-17	165000.00	5	Sandra López
9	2025-06-15	600000.00	4	Luis Martínez
8	2025-06-13	159000.00	3	María Rodríguez
7	2025-06-12	499000.00	2	Carlos Gómez
6	2025-06-10	1140000.00	1	Ana Pérez
5	2025-06-09	45000.00	5	Sandra López
4	2025-06-07	39999.00	4	Luis Martínez
3	2025-06-05	84999.00	3	María Rodríguez
2	2025-06-03	600000.00	2	Carlos Gómez
1	2025-06-01	1049000.00	1	Ana Pérez

10 rows in set (0,00 sec)

8. Mostrar productos que no han sido pedidos (usando RIGHT JOIN)

```sql
SELECT
    p.id AS pedido_id,
    pr.id AS producto_id,
    pr.nombre AS producto
FROM Pedidos p
RIGHT JOIN DetallesPedido dp ON dp.pedido_id = p.id
RIGHT JOIN Productos pr ON dp.producto_id = pr.id
WHERE p.id IS NULL
ORDER BY pr.nombre;
```

pedido_id	producto_id	producto
NULL	21	Accesorios para auto - Kit limpieza
NULL	17	Huawei Nova 11
NULL	13	iPhone 14 Pro
NULL	18	LG Smart TV 55"
NULL	22	Limpiador multiusos
NULL	16	Motorola Edge 30
NULL	14	Samsung Galaxy S22
NULL	20	Set de cuchillos inox
NULL	19	Sony Bravia 50"
NULL	15	Xiaomi Redmi Note 11

10 rows in set (0,00 sec)

9.Mostrar el total de pedidos y ubicación de clientes usando múltiples JOIN

```sql
SELECT
    c.id AS cliente_id,
    c.nombre AS cliente,
    COUNT(p.id) AS total_pedidos,
    uc.direccion,
    ci.nombre AS ciudad,
    uc.codigo_postal
FROM Clientes c
LEFT JOIN Pedidos p ON p.cliente_id = c.id
LEFT JOIN UbicacionCliente uc ON uc.cliente_id = c.id
LEFT JOIN Ciudades ci ON uc.ciudad_id = ci.id
```

```
GROUP BY c.id, c.nombre, uc.direccion, ciudad, uc.codigo_postal
ORDER BY total_pedidos DESC;
```

```
+------------+--------------------------+--------------+----------------------+--------------+--------------+
| cliente_id | cliente                  | total_pedidos | direccion            | ciudad       | codigo_postal |
+------------+--------------------------+--------------+----------------------+--------------+--------------+
|          1 | María Fernanda López     |            2 | Carrera 10 #45-67    | Bogotá       | 110111       |
|          2 | Carlos Andrés Rodríguez  |            2 | Calle 23 #12-34      | Medellín     | 050021       |
|          3 | Ana Sofía Ramírez        |            2 | Avenida 7 #89-12     | Los Ángeles  | 44100        |
|          4 | Juan Camilo Martínez     |            2 | Calle 5 #67-89       | San Francisco | 01234        |
|          5 | Laura Valentina Torres   |            2 | Boulevard Central 123 | Buenos Aires | 1000         |
+------------+--------------------------+--------------+----------------------+--------------+--------------+
5 rows in set (0,00 sec)
```

10. Unir Proveedores , Productos , y TiposProductos para un listado completo de inventario.

```
SELECT
    p.id AS producto_id,
    p.nombre AS producto,
    p.precio,
    sc.nombre_subcategoria AS subcategoria,
    pr.nombre AS proveedor
FROM Productos p
LEFT JOIN Subcategorias sc ON p.subcategoria_id = sc.id
LEFT JOIN Proveedores pr ON p.proveedor_id = pr.id
ORDER BY p.nombre;
```

```
+-------------+-----------------------------------+------------+---------------------+--------------------------+
| producto_id | producto                          | precio     | subcategoria        | proveedor                |
+-------------+-----------------------------------+------------+---------------------+--------------------------+
|          21 | Accesorios para auto - Kit limpieza |   45000.00 | Accesorios para auto | Distribuidora Global Tech |
|          17 | Huawei Nova 11                    | 2200000.00 | Celulares           | TecnoDistribuciones S.A.S |
|          13 | iPhone 14 Pro                     | 5600000.00 | Celulares           | TecnoDistribuciones S.A.S |
|          18 | LG Smart TV 55"                   |  499000.00 | Televisores         | ElectroMundo Ltda.       |
|          22 | Limpiador multiusos               |   15000.00 | Limpieza general    | TecnoDistribuciones S.A.S |
|          16 | Motorola Edge 30                  | 1500000.00 | Celulares           | TecnoDistribuciones S.A.S |
|          14 | Samsung Galaxy S22                | 4200000.00 | Celulares           | TecnoDistribuciones S.A.S |
|          20 | Set de cuchillos inox             |   39999.00 | Utensilios de cocina | Distribuidora Global Tech |
|          19 | Sony Bravia 50"                   |  600000.00 | Televisores         | ElectroMundo Ltda.       |
|          15 | Xiaomi Redmi Note 11              |  980000.00 | Celulares           | TecnoDistribuciones S.A.S |
+-------------+-----------------------------------+------------+---------------------+--------------------------+
```

Consultas Simples:

1. **Seleccionar todos los productos con precio mayor a $50**

```
SELECT * FROM Productos
WHERE precio > 50;
```

```
+----+-----------------------------------+------------+--------------+-----------------+
| id | nombre                            | precio     | proveedor_id | subcategoria_id |
+----+-----------------------------------+------------+--------------+-----------------+
| 13 | iPhone 14 Pro                     | 5600000.00 |            1 |               1 |
| 14 | Samsung Galaxy S22                | 4200000.00 |            1 |               1 |
| 15 | Xiaomi Redmi Note 11              |  980000.00 |            1 |               1 |
| 16 | Motorola Edge 30                  | 1500000.00 |            1 |               1 |
| 17 | Huawei Nova 11                    | 2200000.00 |            1 |               1 |
| 18 | LG Smart TV 55"                   |  499000.00 |            2 |               2 |
| 19 | Sony Bravia 50"                   |  600000.00 |            2 |               2 |
| 20 | Set de cuchillos inox             |   39999.00 |            4 |               3 |
| 21 | Accesorios para auto - Kit limpieza |   45000.00 |            4 |               4 |
| 22 | Limpiador multiusos               |   15000.00 |            1 |               5 |
+----+-----------------------------------+------------+--------------+-----------------+
```

2. Consultar clientes registrados en una ciudad específica

```sql
SELECT c.*
FROM Clientes c
JOIN UbicacionCliente uc ON c.id = uc.cliente_id
JOIN Ciudades ci ON uc.ciudad_id = ci.id
WHERE ci.nombre = 'Bogotá';
```

```
+----+-------------------------+--------------------+----------------+
| id | nombre                  | tipo_documento_id | tipo_email_id |
+----+-------------------------+--------------------+----------------+
|  1 | María Fernanda López    |                  1 |              1 |
+----+-------------------------+--------------------+----------------+
```

3. Mostrar empleados contratados en los últimos 5 años

```sql
SELECT * FROM Empleados
WHERE fecha_contratacion >= DATE_SUB(CURDATE(), INTERVAL 5 YEAR);
```

```
+----+-------------------+-------------------+------------+--------------------+
| id | nombre            | puesto            | salario    | fecha_contratacion |
+----+-------------------+-------------------+------------+--------------------+
|  1 | Ana Pérez         | Gerente           | 3500000.00 | 2022-01-15         |
|  2 | Carlos Gómez      | Vendedor          | 1800000.00 | 2023-03-20         |
|  3 | María Rodríguez   | Administrador     | 2500000.00 | 2021-11-05         |
|  4 | Luis Martínez     | Soporte Técnico   | 1600000.00 | 2023-06-01         |
|  5 | Sandra López      | Vendedor          | 1700000.00 | 2022-09-12         |
+----+-------------------+-------------------+------------+--------------------+
```

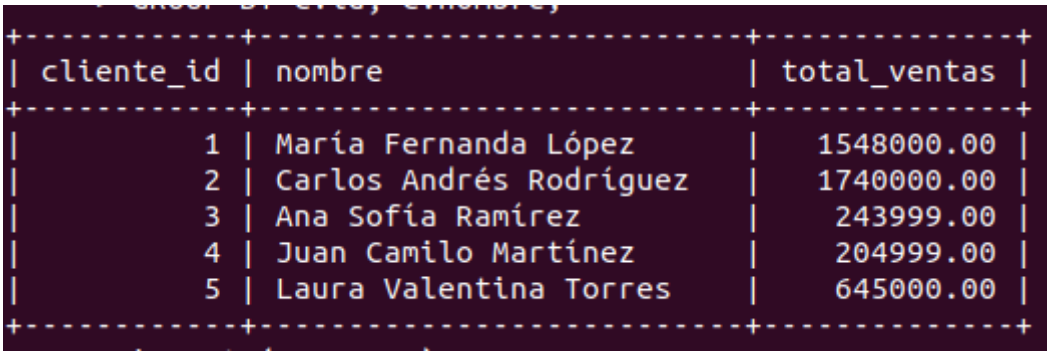
4. Seleccionar proveedores que suministran más de 5 productos

```sql
SELECT pr.id, pr.nombre, COUNT(p.id) AS total_productos
FROM Proveedores pr
JOIN Productos p ON p.proveedor_id = pr.id
GROUP BY pr.id, pr.nombre
HAVING COUNT(p.id) > 5;
```

```
+----+----------------------------+-----------------+
| id | nombre                     | total_productos |
+----+----------------------------+-----------------+
|  1 | TecnoDistribuciones S.A.S  |               6 |
+----+----------------------------+-----------------+
```

6. Calcular el total de ventas por cada cliente

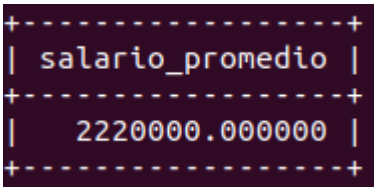
```sql
SELECT
    c.id AS cliente_id,
    c.nombre,
    SUM(p.total) AS total_ventas
FROM Clientes c
JOIN Pedidos p ON p.cliente_id = c.id
GROUP BY c.id, c.nombre;
```



cliente_id	nombre	total_ventas
1	María Fernanda López	1548000.00
2	Carlos Andrés Rodríguez	1740000.00
3	Ana Sofía Ramírez	243999.00
4	Juan Camilo Martínez	204999.00
5	Laura Valentina Torres	645000.00

7. Mostrar el salario promedio de los empleados

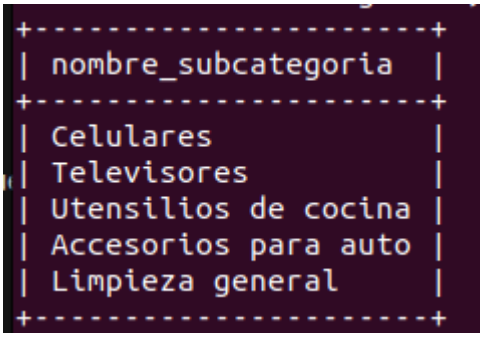
```sql
SELECT AVG(salario) AS salario_promedio
FROM Empleados;
```



salario_promedio
2220000.000000

8. Consultar el tipo de productos disponibles

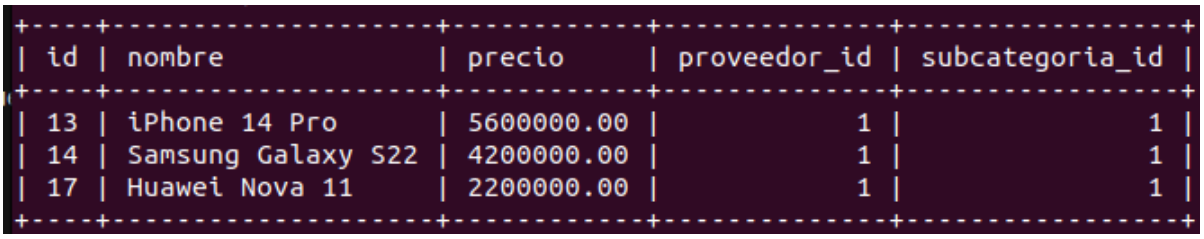
```sql
SELECT DISTINCT nombre_subcategoria
FROM Subcategorias;
```



nombre_subcategoria
Celulares
Televisores
Utensilios de cocina
Accesorios para auto
Limpieza general

9. Seleccionar los 3 productos más caros

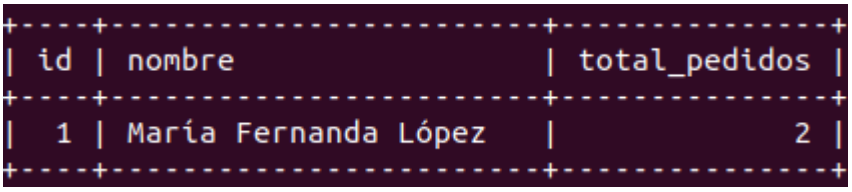
```sql
SELECT *
FROM Productos
ORDER BY precio DESC
LIMIT 3;
```



id	nombre	precio	proveedor_id	subcategoria_id
13	iPhone 14 Pro	5600000.00	1	1
14	Samsung Galaxy S22	4200000.00	1	1
17	Huawei Nova 11	2200000.00	1	1

10. Consultar el cliente con el mayor número de pedidos

```sql
SELECT
    c.id,
    c.nombre,
    COUNT(p.id) AS total_pedidos
FROM Clientes c
JOIN Pedidos p ON p.cliente_id = c.id
GROUP BY c.id, c.nombre
ORDER BY total_pedidos DESC
LIMIT 1;
```



id	nombre	total_pedidos
1	María Fernanda López	2

Consultas Multitabla

1. Listar todos los pedidos y el cliente asociado.

```sql
SELECT p.id AS pedido_id, p.fecha, c.id AS cliente_id, c.nombre AS cliente
FROM Pedidos p
JOIN Clientes c ON p.cliente_id = c.id;
```

pedido_id	fecha	cliente_id	cliente
1	2025-06-01	1	María Fernanda López
7	2025-06-12	1	María Fernanda López
2	2025-06-03	2	Carlos Andrés Rodríguez
6	2025-06-10	2	Carlos Andrés Rodríguez
3	2025-06-05	3	Ana Sofía Ramírez
8	2025-06-13	3	Ana Sofía Ramírez
4	2025-06-07	4	Juan Camilo Martínez
10	2025-06-17	4	Juan Camilo Martínez
5	2025-06-09	5	Laura Valentina Torres
9	2025-06-15	5	Laura Valentina Torres

2. Mostrar la ubicación de cada cliente en sus pedidos

```sql
SELECT p.id AS pedido_id, c.nombre AS cliente, uc.direccion, ci.nombre AS ciudad
FROM Pedidos p
JOIN Clientes c ON p.cliente_id = c.id
LEFT JOIN UbicacionCliente uc ON c.id = uc.cliente_id
LEFT JOIN Ciudades ci ON uc.ciudad_id = ci.id;
```

pedido_id	cliente	direccion	ciudad
1	María Fernanda López	Carrera 10 #45-67	Bogotá
7	María Fernanda López	Carrera 10 #45-67	Bogotá
2	Carlos Andrés Rodríguez	Calle 23 #12-34	Medellín
6	Carlos Andrés Rodríguez	Calle 23 #12-34	Medellín
3	Ana Sofía Ramírez	Avenida 7 #89-12	Los Ángeles
8	Ana Sofía Ramírez	Avenida 7 #89-12	Los Ángeles
4	Juan Camilo Martínez	Calle 5 #67-89	San Francisco
10	Juan Camilo Martínez	Calle 5 #67-89	San Francisco
5	Laura Valentina Torres	Boulevard Central 123	Buenos Aires
9	Laura Valentina Torres	Boulevard Central 123	Buenos Aires

3. Listar productos junto con el proveedor y tipo de producto

```sql
SELECT
    p.id AS producto_id,
    p.nombre AS producto,
    pr.nombre AS proveedor,
    sc.nombre_subcategoria AS tipo_producto
FROM Productos p
LEFT JOIN Proveedores pr ON p.proveedor_id = pr.id
LEFT JOIN Subcategorias sc ON p.subcategoria_id = sc.id;
```

producto_id	producto	proveedor	tipo_producto
13	iPhone 14 Pro	TecnoDistribuciones S.A.S	Celulares
14	Samsung Galaxy S22	TecnoDistribuciones S.A.S	Celulares
15	Xiaomi Redmi Note 11	TecnoDistribuciones S.A.S	Celulares
16	Motorola Edge 30	TecnoDistribuciones S.A.S	Celulares
17	Huawei Nova 11	TecnoDistribuciones S.A.S	Celulares
18	LG Smart TV 55"	ElectroMundo Ltda.	Televisores
19	Sony Bravia 50"	ElectroMundo Ltda.	Televisores
20	Set de cuchillos inox	Distribuidora Global Tech	Utensilios de cocina
21	Accesorios para auto - Kit limpieza	Distribuidora Global Tech	Accesorios para auto
22	Limpiador multiusos	TecnoDistribuciones S.A.S	Limpieza general

4. Consultar todos los empleados que gestionan pedidos de clientes en una ciudad específica

```sql
SELECT DISTINCT e.id, e.nombre
FROM Empleados e
JOIN Pedidos p ON p.empleado_id = e.id
JOIN Clientes c ON p.cliente_id = c.id
JOIN UbicacionCliente uc ON uc.cliente_id = c.id
JOIN Ciudades ci ON uc.ciudad_id = ci.id
WHERE ci.nombre = 'Bogotá';  -- Cambia por la ciudad que desees
```

id	nombre
1	Ana Pérez
2	Carlos Gómez

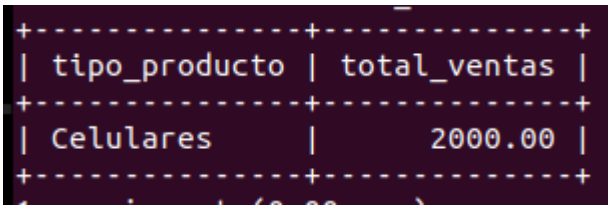
6. Obtener la cantidad total de pedidos por cliente y ciudad

```sql
SELECT c.id AS cliente_id, c.nombre, ci.nombre AS ciudad, COUNT(p.id) AS
total_pedidos
FROM Clientes c
JOIN Pedidos p ON p.cliente_id = c.id
JOIN UbicacionCliente uc ON uc.cliente_id = c.id
JOIN Ciudades ci ON uc.ciudad_id = ci.id
GROUP BY c.id, c.nombre, ci.nombre;
```

cliente_id	nombre	ciudad	total_pedidos
1	María Fernanda López	Bogotá	2
2	Carlos Andrés Rodríguez	Medellín	2
3	Ana Sofía Ramírez	Los Ángeles	2
4	Juan Camilo Martínez	San Francisco	2
5	Laura Valentina Torres	Buenos Aires	2

8. Mostrar el total de ventas agrupado por tipo de producto.

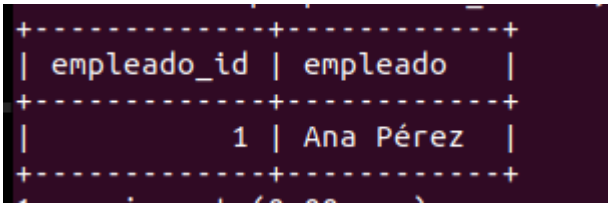
```sql
SELECT
    sc.nombre_subcategoria AS tipo_producto,
    SUM(dp.cantidad * dp.precio_unidad) AS total_ventas
FROM DetallesPedido dp
JOIN Productos p ON dp.producto_id = p.id
JOIN Subcategorias sc ON p.subcategoria_id = sc.id
GROUP BY sc.nombre_subcategoria
ORDER BY total_ventas DESC;
```



tipo_producto	total_ventas
Celulares	2000.00

9. **Listar empleados que gestionan pedidos de productos de un proveedor específico**!

```sql
SELECT DISTINCT
    e.id AS empleado_id,
    e.nombre AS empleado
FROM Empleados e
JOIN Pedidos p ON p.empleado_id = e.id
JOIN DetallesPedido dp ON dp.pedido_id = p.id
JOIN Productos prd ON dp.producto_id = prd.id
WHERE prd.proveedor_id = 2;
```



empleado_id	empleado
1	Ana Pérez

10. **Obtener el ingreso total de cada proveedor a partir de los productos vendidos**

```sql
SELECT
    pr.id AS proveedor_id,
    pr.nombre AS proveedor,
    SUM(dp.cantidad * dp.precio_unidad) AS ingreso_total
FROM Proveedores pr
JOIN Productos p ON p.proveedor_id = pr.id
JOIN DetallesPedido dp ON dp.producto_id = p.id
GROUP BY pr.id, pr.nombre
ORDER BY ingreso_total DESC;
```

proveedor_id	proveedor	ingreso_total
1	TecnoDistribuciones S.A.S	2000.00
2	ElectroMundo Ltda.	100.00

Subconsultas

1. **Producto más caro en cada categoría**

```sql
SELECT *
FROM Productos p
WHERE precio = (
    SELECT MAX(p2.precio)
    FROM Productos p2
    JOIN Subcategorias sc ON p2.subcategoria_id = sc.id
    WHERE sc.categoria_id = (
        SELECT sc2.categoria_id FROM Subcategorias sc2 WHERE sc2.id =
p.subcategoria_id
    )
);
```

id	nombre	precio	proveedor_id	subcategoria_id
13	iPhone 14 Pro	5600000.00	1	1
20	Set de cuchillos inox	39999.00	4	3
21	Accesorios para auto - Kit limpieza	45000.00	4	4
22	Limpiador multiusos	15000.00	1	5

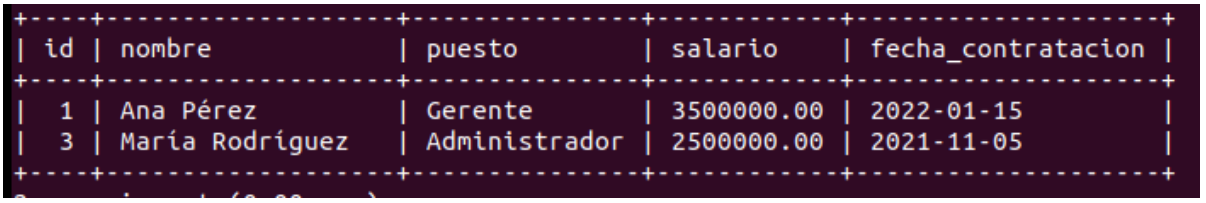
2. Cliente con mayor total en pedidos

```sql
SELECT *
FROM Clientes
WHERE id = (
    SELECT cliente_id
    FROM Pedidos
    GROUP BY cliente_id
    ORDER BY SUM(total) DESC
    LIMIT 1
);
```

id	nombre	tipo_documento_id	tipo_email_id
2	Carlos Andrés Rodríguez	2	2

3. Empleados que ganan más que el salario promedio

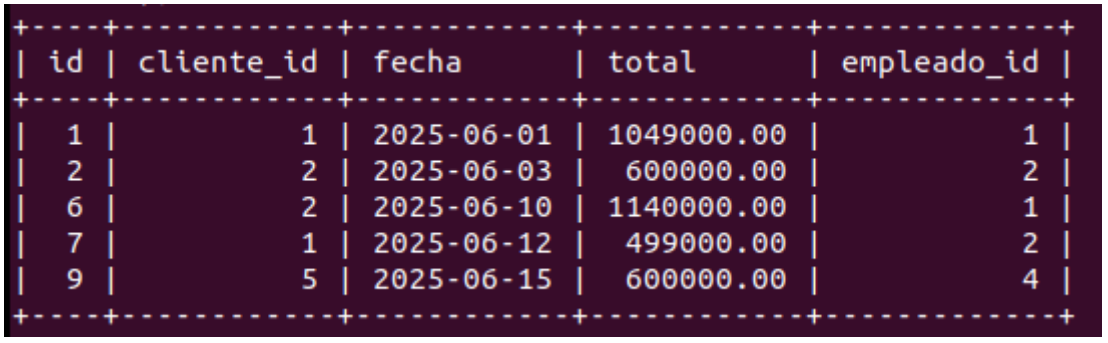
```sql
SELECT *
FROM Empleados
WHERE salario > (
    SELECT AVG(salario)
    FROM Empleados
);
```



id	nombre	puesto	salario	fecha_contratacion
1	Ana Pérez	Gerente	3500000.00	2022-01-15
3	María Rodríguez	Administrador	2500000.00	2021-11-05

5. Pedidos cuyo total es mayor al promedio de todos los pedidos

```sql
SELECT *
FROM Pedidos
WHERE total > (
    SELECT AVG(total)
    FROM Pedidos
);
```



id	cliente_id	fecha	total	empleado_id
1	1	2025-06-01	1049000.00	1
2	2	2025-06-03	600000.00	2
6	2	2025-06-10	1140000.00	1
7	1	2025-06-12	499000.00	2
9	5	2025-06-15	600000.00	4

6. 3 proveedores con más productos

```sql
SELECT proveedor_id, COUNT(*) AS total_productos
FROM Productos
GROUP BY proveedor_id
ORDER BY total_productos DESC
LIMIT 3;
```

proveedor_id	total_productos
1	7
2	2
4	2

7. **Productos con precio superior al promedio de su tipo (subcategoría)**

```sql
SELECT *
FROM Productos p
WHERE precio > (
    SELECT AVG(p2.precio)
    FROM Productos p2
    WHERE p2.subcategoria_id = p.subcategoria_id
);
```

id	nombre	precio	proveedor_id	subcategoria_id
13	iPhone 14 Pro	5600000.00	1	1
14	Samsung Galaxy S22	4200000.00	1	1
19	Sony Bravia 50"	600000.00	2	2

8. **Clientes que han realizado más pedidos que la media**

```sql
SELECT *
FROM Clientes
WHERE id IN (
    SELECT cliente_id
    FROM Pedidos
    GROUP BY cliente_id
    HAVING COUNT(*) >= (
        SELECT AVG(cnt)
        FROM (
            SELECT COUNT(*) AS cnt
            FROM Pedidos
            GROUP BY cliente_id
        ) AS sub
    )
);
```

```
+----+--------------------------+------------------+---------------+
| id | nombre                   | tipo_documento_id | tipo_email_id |
+----+--------------------------+------------------+---------------+
|  1 | María Fernanda López     |                1 |             1 |
|  2 | Carlos Andrés Rodríguez  |                2 |             2 |
|  3 | Ana Sofía Ramírez        |                3 |             1 |
|  4 | Juan Camilo Martínez     |                1 |             3 |
|  5 | Laura Valentina Torres   |                2 |             2 |
+----+--------------------------+------------------+---------------+
```

9. Encontrar productos cuyo precio es mayor que el promedio de todos los productos

```sql
SELECT *
FROM Productos
WHERE precio > (
    SELECT AVG(precio)
    FROM Productos
);
```

```
    -> );
+----+--------------------+------------+--------------+-----------------+
| id | nombre             | precio     | proveedor_id | subcategoria_id |
+----+--------------------+------------+--------------+-----------------+
| 13 | iPhone 14 Pro      | 5600000.00 |            1 |               1 |
| 14 | Samsung Galaxy S22 | 4200000.00 |            1 |               1 |
| 16 | Motorola Edge 30   | 1500000.00 |            1 |               1 |
| 17 | Huawei Nova 11     | 2200000.00 |            1 |               1 |
+----+--------------------+------------+--------------+-----------------+
4 rows in set (0.00 sec)
```

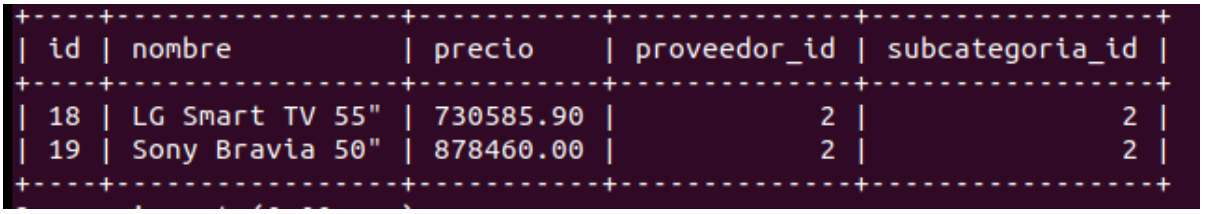
10. Mostrar empleados cuyo salario es menor al promedio del departamento

```sql
SELECT *
FROM Empleados e
WHERE salario < (
    SELECT AVG(e2.salario)
    FROM Empleados e2
    WHERE e2.puesto = e.puesto
);
```

```
+----+---------------+----------+------------+--------------------+
| id | nombre        | puesto   | salario    | fecha_contratacion |
+----+---------------+----------+------------+--------------------+
|  5 | Sandra López  | Vendedor | 1700000.00 | 2022-09-12         |
+----+---------------+----------+------------+--------------------+
1 row in set (0.00 sec)
```

Procedimientos almacenados

1.Actualizar el precio de todos los productos de un proveedor



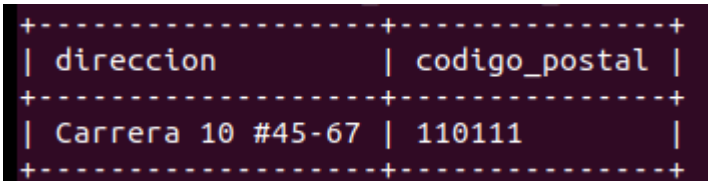
id	nombre	precio	proveedor_id	subcategoria_id
18	LG Smart TV 55"	730585.90	2	2
19	Sony Bravia 50"	878460.00	2	2

```
DELIMITER //

CREATE PROCEDURE actualizar_precios_proveedor(
    IN p_proveedor_id INT,
    IN p_porcentaje DECIMAL(5,2)
)
BEGIN
    UPDATE Productos
    SET precio = precio * (1 + p_porcentaje / 100)
    WHERE proveedor_id = p_proveedor_id;
END;
//

DELIMITER ;
```

2. Un procedimiento que devuelva la dirección de un cliente por ID.



direccion	codigo_postal
Carrera 10 #45-67	110111

```
DELIMITER //

CREATE PROCEDURE obtener_direccion_cliente(
    IN p_cliente_id INT
)
BEGIN
    SELECT uc.direccion, uc.codigo_postal
    FROM UbicacionCliente uc
    WHERE uc.cliente_id = p_cliente_id;
END;
//

DELIMITER ;
```

3. Registrar un pedido nuevo y sus detalles

```
mysql> SELECT * FROM DetallesPedido ORDER BY id DESC LIMIT 1;
+----+-----------+-------------+----------+----------------+
| id | pedido_id | producto_id | cantidad | precio_unidad |
+----+-----------+-------------+----------+----------------+
|  2 |         1 |          18 |        1 |         100.00 |
+----+-----------+-------------+----------+----------------+
```

```
DELIMITER //

CREATE PROCEDURE registrar_pedido(
    IN p_cliente_id INT,
    IN p_empleado_id INT,
    IN p_fecha DATE,
    IN p_total DECIMAL(10,2),
    IN p_producto_id INT,
    IN p_cantidad INT,
    IN p_precio DECIMAL(10,2)
)
BEGIN
    DECLARE v_pedido_id INT;

    INSERT INTO Pedidos (cliente_id, fecha, total, empleado_id)
    VALUES (p_cliente_id, p_fecha, p_total, p_empleado_id);

    SET v_pedido_id = LAST_INSERT_ID();

    INSERT INTO DetallesPedido (pedido_id, producto_id, cantidad, precio_unidad)
    VALUES (v_pedido_id, p_producto_id, p_cantidad, p_precio);
END;
//

DELIMITER ;
```

4. Calcular el total de ventas de un cliente

```
DELIMITER //

CREATE PROCEDURE total_ventas_cliente(
    IN p_cliente_id INT
)
BEGIN
    SELECT SUM(total) AS total_ventas
    FROM Pedidos
    WHERE cliente_id = p_cliente_id;
END;
//

DELIMITER ;
```

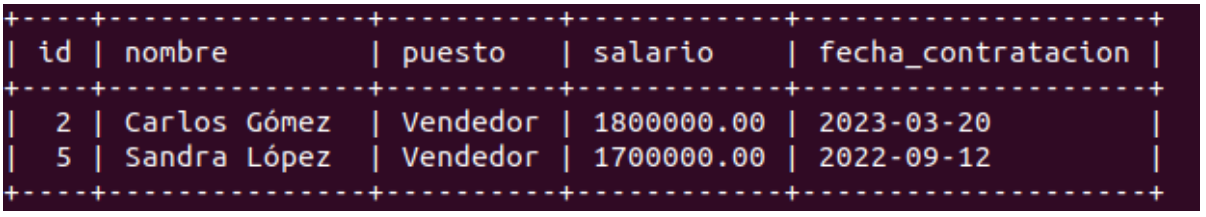
5. **Obtener los empleados por puesto**

```
DELIMITER //

CREATE PROCEDURE empleados_por_puesto(
    IN p_puesto VARCHAR(50)
)
BEGIN
    SELECT * FROM Empleados
    WHERE puesto = p_puesto;
END;
//

DELIMITER ;
```

6. **Actualizar el salario de empleados por puesto**



id	nombre	puesto	salario	fecha_contratacion
2	Carlos Gómez	Vendedor	1800000.00	2023-03-20
5	Sandra López	Vendedor	1700000.00	2022-09-12

```
DELIMITER //

CREATE PROCEDURE actualizar_salario_por_puesto(
    IN p_puesto VARCHAR(50),
    IN p_incremento DECIMAL(5,2)
)
BEGIN
    UPDATE Empleados
    SET salario = salario * (1 + p_incremento / 100)
    WHERE puesto = p_puesto;
END;
//

DELIMITER ;
```

7. **Listar pedidos entre dos fechas**

id	cliente_id	fecha	total	empleado_id
1	1	2025-06-01	1049000.00	1
2	2	2025-06-03	600000.00	2
3	3	2025-06-05	84999.00	3
4	4	2025-06-07	39999.00	4
5	5	2025-06-09	45000.00	5
6	2	2025-06-10	1140000.00	1

```sql
DELIMITER //

CREATE PROCEDURE pedidos_entre_fechas(
    IN p_inicio DATE,
    IN p_fin DATE
)
BEGIN
    SELECT * FROM Pedidos
    WHERE fecha BETWEEN p_inicio AND p_fin;
END;
//

DELIMITER ;
```

8. Aplicar descuento a productos de una categoría

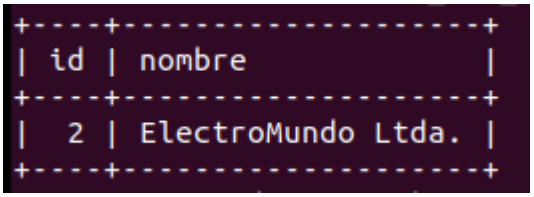
id	nombre	precio
1	Laptop	850.00
13	iPhone 14 Pro	4760000.00
14	Samsung Galaxy S22	3570000.00
15	Xiaomi Redmi Note 11	833000.00
16	Motorola Edge 30	1275000.00
17	Huawei Nova 11	1870000.00
18	LG Smart TV 55"	620998.02
19	Sony Bravia 50"	746691.00

```sql
DELIMITER //

CREATE PROCEDURE aplicar_descuento_categoria(
    IN p_categoria_id INT,
    IN p_descuento DECIMAL(5,2)
)
BEGIN
    UPDATE Productos p
    JOIN Subcategorias sc ON p.subcategoria_id = sc.id
    SET p.precio = p.precio * (1 - p_descuento / 100)
    WHERE sc.categoria_id = p_categoria_id;
END;
//
```

```
DELIMITER ;
```

9. Listar todos los proveedores de un tipo de producto (subcategoría)



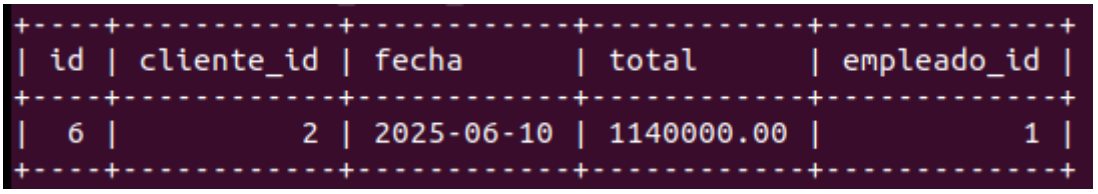
```
+----+--------------------+
| id | nombre             |
+----+--------------------+
|  2 | ElectroMundo Ltda. |
+----+--------------------+
```

```
DELIMITER //

CREATE PROCEDURE proveedores_por_tipo(
    IN p_subcategoria_id INT
)
BEGIN
    SELECT DISTINCT pr.id, pr.nombre
    FROM Proveedores pr
    JOIN Productos p ON p.proveedor_id = pr.id
    WHERE p.subcategoria_id = p_subcategoria_id;
END;
//

DELIMITER ;
```

10. Devolver el pedido de mayor valor



```
+----+------------+------------+------------+-------------+
| id | cliente_id | fecha      | total      | empleado_id |
+----+------------+------------+------------+-------------+
|  6 |          2 | 2025-06-10 | 1140000.00 |           1 |
+----+------------+------------+------------+-------------+
```

```
DELIMITER //

CREATE PROCEDURE pedido_mayor_valor()
BEGIN
    SELECT *
    FROM Pedidos
    ORDER BY total DESC
    LIMIT 1;
END;
//

DELIMITER ;
```

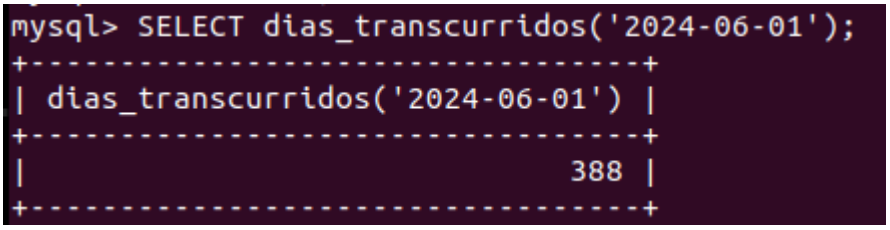
Funciones Definidas por el Usuario

1. Crear una función que reciba una fecha y devuelva los días transcurridos.

```
DELIMITER //

CREATE FUNCTION dias_transcurridos(fecha DATE)
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN DATEDIFF(CURDATE(), fecha);
END;
//

DELIMITER ;
```

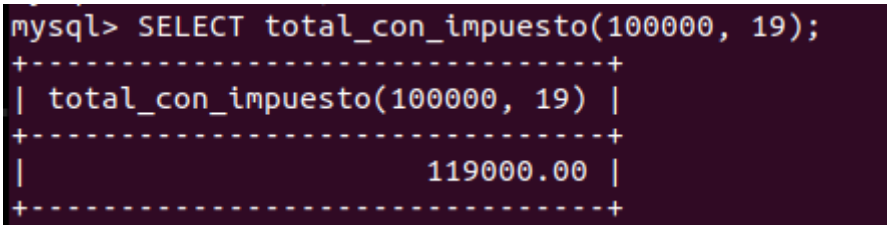


2. Crear una función para calcular el total con impuesto de un monto.

```
DELIMITER //

CREATE FUNCTION total_con_impuesto(monto DECIMAL(10,2), porcentaje DECIMAL(5,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    RETURN monto * (1 + porcentaje / 100);
END;
//

DELIMITER ;
```



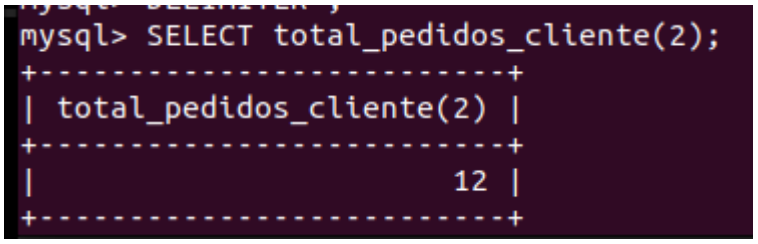
3. Una función que devuelva el total de pedidos de un cliente específico.

```
DELIMITER //

CREATE FUNCTION total_pedidos_cliente(cliente_id INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total INT;
```

```
    SELECT COUNT(*) INTO total
    FROM Pedidos
    WHERE cliente_id = cliente_id;
    RETURN total;
END;
//

DELIMITER ;
```

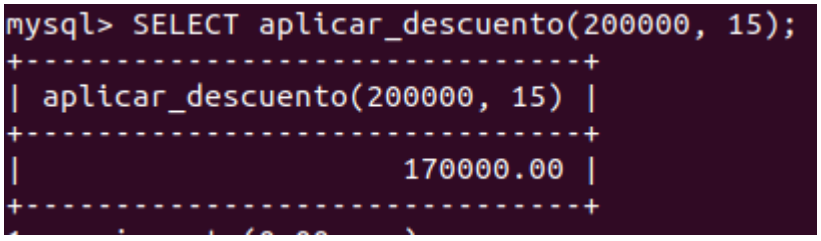


4. Crear una función para aplicar un descuento a un producto.

```
DELIMITER //

CREATE FUNCTION aplicar_descuento(precio DECIMAL(10,2), porcentaje DECIMAL(5,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    RETURN precio * (1 - porcentaje / 100);
END;
//

DELIMITER ;
```



5. Una función que indique si un cliente tiene dirección registrada.

```
DELIMITER //

CREATE FUNCTION cliente_con_direccion(p_cliente_id INT)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
    DECLARE tiene_direccion BOOLEAN;
    SELECT COUNT(*) > 0 INTO tiene_direccion
    FROM UbicacionCliente
```

```
    WHERE cliente_id = p_cliente_id;
    RETURN tiene_direccion;
END;
//

DELIMITER ;
```

```
mysql> SELECT cliente_con_direccion(3);
+--------------------------+
| cliente_con_direccion(3) |
+--------------------------+
|                        1 |
+--------------------------+
```

6. Crear una función que devuelva el salario anual de un empleado.

```
DELIMITER //

CREATE FUNCTION salario_anual(salario_mensual DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    RETURN salario_mensual * 12;
END;
//

DELIMITER ;
```

```
mysql> SELECT salario_anual(2500000);
+------------------------+
| salario_anual(2500000) |
+------------------------+
|             30000000.00 |
+------------------------+
```

7. Una función para calcular el total de ventas de un tipo de producto.

```
DELIMITER //

CREATE FUNCTION total_ventas_subcategoria(p_subcat_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2);
    SELECT SUM(dp.cantidad * dp.precio_unidad) INTO total
    FROM DetallesPedido dp
    JOIN Productos p ON dp.producto_id = p.id
    WHERE p.subcategoria_id = p_subcat_id;
    RETURN IFNULL(total, 0.00);
END;
//

DELIMITER ;
```

```
mysql> SELECT total_ventas_subcategoria(1);
+------------------------------+
| total_ventas_subcategoria(1) |
+------------------------------+
|                      2000.00 |
+------------------------------+
```

8. Crear una función para devolver el nombre de un cliente por ID.

```sql
DELIMITER //

CREATE FUNCTION nombre_cliente(p_id INT)
RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
    DECLARE nombre VARCHAR(100);
    SELECT c.nombre INTO nombre
    FROM Clientes c
    WHERE c.id = p_id;
    RETURN nombre;
END;
//

DELIMITER ;
```

```
mysql> SELECT nombre_cliente(4);
+------------------------+
| nombre_cliente(4)      |
+------------------------+
| Juan Camilo Martínez   |
+------------------------+
```

9. Una función que reciba el ID de un pedido y devuelva su total.

```sql
DELIMITER //

CREATE FUNCTION total_pedido(p_pedido_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2);
    SELECT SUM(dp.cantidad * dp.precio_unidad) INTO total
    FROM DetallesPedido dp
    WHERE dp.pedido_id = p_pedido_id;
    RETURN IFNULL(total, 0.00);
END;
//

DELIMITER ;
```

```
mysql> SELECT total_pedido(5);
+-----------------+
| total_pedido(5) |
+-----------------+
|            0.00 |
+-----------------+
```

10. Crear una función que indique si un producto está en inventario.

```sql
DELIMITER //

CREATE FUNCTION producto_en_stock(p_id INT)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
    DECLARE en_stock BOOLEAN;
    SELECT CASE WHEN p.id IS NOT NULL THEN TRUE ELSE FALSE END INTO en_stock
    FROM Productos p
    WHERE p.id = p_id;
    RETURN en_stock;
END;
//

DELIMITER ;
```

```
mysql> SELECT producto_en_stock(1);
+----------------------+
| producto_en_stock(1) |
+----------------------+
|                    1 |
+----------------------+
```

Triggers

1. Crear un trigger que registre en HistorialSalarios cada cambio de salario de empleados.

```sql
CREATE TABLE IF NOT EXISTS HistorialSalarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    empleado_id INT,
    salario_anterior DECIMAL(10,2),
    salario_nuevo DECIMAL(10,2),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```sql
DELIMITER //

CREATE TRIGGER tr_salario_update
BEFORE UPDATE ON Empleados
FOR EACH ROW
```

```
BEGIN
    IF OLD.salario <> NEW.salario THEN
        INSERT INTO HistorialSalarios (empleado_id, salario_anterior,
salario_nuevo)
        VALUES (OLD.id, OLD.salario, NEW.salario);
    END IF;
END;
//

DELIMITER ;
```

2. Crear un trigger que evite borrar productos con pedidos activos.

```
DELIMITER //

CREATE TRIGGER tr_prevent_delete_producto
BEFORE DELETE ON Productos
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1 FROM DetallesPedido WHERE producto_id = OLD.id
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar un producto con pedidos
activos';
    END IF;
END;
//

DELIMITER ;
```

3. Un trigger que registre en HistorialPedidos cada actualización en Pedidos .

```
CREATE TABLE IF NOT EXISTS HistorialPedidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT,
    fecha_modificacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    total_anterior DECIMAL(10,2),
    total_nuevo DECIMAL(10,2)
);
```

4. Crear un trigger que actualice el inventario al registrar un pedido.

```
DELIMITER //

CREATE TRIGGER tr_update_stock
AFTER INSERT ON DetallesPedido
```

```
FOR EACH ROW
BEGIN
    UPDATE Productos
    SET stock = stock - NEW.cantidad
    WHERE id = NEW.producto_id;
END;
//

DELIMITER ;
```

5. Un trigger que evite actualizaciones de precio a menos de $1.

```
DELIMITER //

CREATE TRIGGER tr_precio_minimo
BEFORE UPDATE ON Productos
FOR EACH ROW
BEGIN
    IF NEW.precio < 1 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El precio no puede ser menor a $1';
    END IF;
END;
//

DELIMITER ;
```

6. Crear un trigger que registre la fecha de creación de un pedido en HistorialPedidos .

```
DELIMITER //

CREATE TRIGGER tr_nuevo_pedido
AFTER INSERT ON Pedidos
FOR EACH ROW
BEGIN
    INSERT INTO HistorialPedidos (pedido_id, total_anterior, total_nuevo)
    VALUES (NEW.id, 0, NEW.total);
END;
//

DELIMITER ;
```

7. Un trigger que mantenga el precio total de cada pedido en Pedidos .

```
DELIMITER //

CREATE TRIGGER tr_total_pedido
```

```
AFTER INSERT ON DetallesPedido
FOR EACH ROW
BEGIN
    UPDATE Pedidos
    SET total = (
        SELECT SUM(cantidad * precio_unidad)
        FROM DetallesPedido
        WHERE pedido_id = NEW.pedido_id
    )
    WHERE id = NEW.pedido_id;
END;
//

DELIMITER ;
```

8. Crear un trigger para validar que UbicacionCliente no esté vacío al crear un cliente.

```

```

9. Un trigger que registre en LogActividades cada modificación en Proveedores .

```
DELIMITER //

CREATE TRIGGER tr_log_update_proveedores
AFTER UPDATE ON Proveedores
FOR EACH ROW
BEGIN
    INSERT INTO LogActividades (entidad, entidad_id, accion, usuario)
    VALUES ('Proveedor', OLD.id, 'Actualización', USER());
END;
//

DELIMITER ;
```

10. Crear un trigger que registre en HistorialContratos cada cambio en Empleados .

```
CREATE TABLE IF NOT EXISTS HistorialContratos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    empleado_id INT,
    puesto_anterior VARCHAR(50),
    puesto_nuevo VARCHAR(50),
    salario_anterior DECIMAL(10,2),
    salario_nuevo DECIMAL(10,2),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
DELIMITER //

CREATE TRIGGER tr_historial_contratos
BEFORE UPDATE ON Empleados
FOR EACH ROW
BEGIN
    IF OLD.puesto <> NEW.puesto OR OLD.salario <> NEW.salario THEN
        INSERT INTO HistorialContratos (
            empleado_id, puesto_anterior, puesto_nuevo,
            salario_anterior, salario_nuevo
        )
        VALUES (
            OLD.id, OLD.puesto, NEW.puesto,
            OLD.salario, NEW.salario
        );
    END IF;
END;
//

DELIMITER ;
```

Busquedas Avanzadas

1. Función de Descuento por Categoría de Producto

```
DELIMITER //

CREATE FUNCTION CalcularDescuento(subcat_id INT, precio DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE nombre_subcat VARCHAR(100);
    DECLARE precio_descuento DECIMAL(10,2);

    -- Obtener el nombre de la subcategoría
    SELECT nombre_subcategoria INTO nombre_subcat
    FROM Subcategorias
    WHERE id = subcat_id;

    -- Aplicar descuento si es Electrónica
    IF nombre_subcat = 'Electrónica' THEN
        SET precio_descuento = precio * 0.9;
    ELSE
        SET precio_descuento = precio;
    END IF;

    RETURN precio_descuento;
END;
//

DELIMITER ;
```

2. Función para Obtener la Edad de un Cliente y Filtrar Clientes Mayores de Edad

```
DELIMITER //

CREATE FUNCTION CalcularEdad(fecha_nac DATE)
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN TIMESTAMPDIFF(YEAR, fecha_nac, CURDATE());
END;
//

DELIMITER ;
```

3. Función de Cálculo de Impuesto y Consulta de Productos con Precio Final

```
DELIMITER //

CREATE FUNCTION CalcularImpuesto(precio DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    RETURN precio * 1.15;
END;
//

DELIMITER ;
```

4. Función para Calcular el Total de Pedidos de un Cliente

```
DELIMITER //

CREATE FUNCTION TotalPedidosCliente(clienteId INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2);

    SELECT SUM(total) INTO total
    FROM Pedidos
    WHERE cliente_id = clienteId;

    RETURN IFNULL(total, 0);
END;
//

DELIMITER ;
```

5. Función para Calcular el Salario Anual de un Empleado

```
DELIMITER //

CREATE FUNCTION SalarioAnual(salario_mensual DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    RETURN salario_mensual * 12;
END;
//

DELIMITER ;
```

6. Función de Bonificación y Consulta de Salarios Ajustados

```
DELIMITER //

CREATE FUNCTION Bonificacion(salario DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    RETURN salario * 0.10;
END;
//

DELIMITER ;
```

7. Función para Calcular Días Transcurridos Desde el Último Pedido

```
DELIMITER //

CREATE FUNCTION DiasDesdeUltimoPedido(clienteId INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE dias INT;

    SELECT DATEDIFF(CURDATE(), MAX(fecha)) INTO dias
    FROM Pedidos
    WHERE cliente_id = clienteId;

    RETURN IFNULL(dias, 9999);  -- Si no tiene pedidos, devuelve un valor grande
END;
//

DELIMITER ;
```

8. Función para Calcular el Total en Inventario de un Producto

```
DELIMITER //

CREATE FUNCTION TotalInventarioProducto(precio DECIMAL(10,2), cantidad INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    RETURN precio * cantidad;
END;
//

DELIMITER ;
```

9. Creación de un Historial de Precios de Productos

```
DELIMITER //

CREATE TRIGGER RegistroCambioPrecio
BEFORE UPDATE ON Productos
FOR EACH ROW
BEGIN
    IF OLD.precio <> NEW.precio THEN
        INSERT INTO HistorialPrecios (producto_id, precio_anterior, precio_nuevo)
        VALUES (OLD.id, OLD.precio, NEW.precio);
    END IF;
END;
//

DELIMITER ;
```

10. Procedimiento para Generar Reporte de Ventas Mensuales por Empleado

```
DELIMITER //

CREATE PROCEDURE ReporteVentasMensuales(IN mes INT, IN anio INT)
BEGIN
    SELECT
        e.id AS empleado_id,
        e.nombre AS empleado,
        SUM(p.total) AS total_ventas
    FROM Empleados e
    JOIN Pedidos p ON p.empleado_id = e.id
    WHERE MONTH(p.fecha) = mes AND YEAR(p.fecha) = anio
    GROUP BY e.id, e.nombre
    ORDER BY total_ventas DESC;
END;
//
```

```
DELIMITER ;
```

11. Subconsulta para Obtener el Producto Más Vendido por Cada Proveedor

```
SELECT
    pr.nombre AS proveedor,
    p.nombre AS producto,
    SUM(dp.cantidad) AS cantidad_vendida
FROM Productos p
JOIN Proveedores pr ON p.proveedor_id = pr.id
JOIN DetallesPedido dp ON dp.producto_id = p.id
GROUP BY pr.id, p.id
HAVING SUM(dp.cantidad) = (
    SELECT MAX(SUM_INNER)
    FROM (
        SELECT SUM(dp2.cantidad) AS SUM_INNER
        FROM Productos p2
        JOIN DetallesPedido dp2 ON dp2.producto_id = p2.id
        WHERE p2.proveedor_id = pr.id
        GROUP BY p2.id
    ) AS subquery
);
```

12. Función para Calcular el Estado de Stock de un Producto

```
DELIMITER //

CREATE FUNCTION EstadoStock(cantidad INT)
RETURNS VARCHAR(10)
DETERMINISTIC
BEGIN
    DECLARE estado VARCHAR(10);

    IF cantidad >= 100 THEN
        SET estado = 'Alto';
    ELSEIF cantidad >= 50 THEN
        SET estado = 'Medio';
    ELSE
        SET estado = 'Bajo';
    END IF;

    RETURN estado;
END;
//

DELIMITER ;
```

13. Trigger para Control de Inventario en Pedidos

```
DELIMITER //

CREATE TRIGGER ActualizarInventario
BEFORE INSERT ON DetallesPedido
FOR EACH ROW
BEGIN
    DECLARE stock_actual INT;

    -- Obtener el stock disponible del producto
    SELECT cantidad INTO stock_actual
    FROM Productos
    WHERE id = NEW.producto_id;

    -- Verificar si hay suficiente stock
    IF stock_actual < NEW.cantidad THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Stock insuficiente para completar el pedido';
    ELSE
        -- Disminuir el stock
        UPDATE Productos
        SET cantidad = cantidad - NEW.cantidad
        WHERE id = NEW.producto_id;
    END IF;
END;
//

DELIMITER ;
```

14. Procedimiento para Generar Informe de Clientes Inactivos

```
DELIMITER //

CREATE PROCEDURE ClientesInactivos()
BEGIN
    SELECT c.id AS cliente_id, c.nombre AS cliente
    FROM Clientes c
    WHERE c.id NOT IN (
        SELECT DISTINCT cliente_id
        FROM Pedidos
        WHERE fecha >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    );
END;
//

DELIMITER ;
```