ACTIVIDAD RETORNO

Estimados aprendices vamos a realizar un repaso de creación de una base de datos, creación de tablas, inserción de datos, actualizar datos, eliminar datos y demás componentes vistos en las formaciones anteriores.

MODELO ENTIDAD RELACION

Diagrama

Descripción generada automáticamente

Con este modelo entidad relación van a crear una base de datos con 20 registros, además van a realizar consultas en diferentes tablas creadas, actualizar 5 productos, 5 ventas, buscar diferentes clientes y productos con diferentes letras y nombres y 5 clientes, realizar la eliminación de diferentes ventas y producto.

Todo este ejercicio lo van a realizar en SQL y en visual studio code sus respectivos resultados en Word.

SENTENCIAS EN SQL

**SHOW DATABASES;**

**CREATE DATABASE “ NOMBRE DE LA BASE DE DATOS” ;**

**INSERT INTO Nombre de la tabla VALUES (Información a incluir );**

**SELECT \* FROM Nombre de la tabla ;**

**UPDATE <Nombre\_de\_la\_tabla> SET <Columna\_a\_modificar>= ‘Lo que vamos a actualizar;**

**DELETE FROM <Nombre de la tabla>;**

**DELETE FROM <Nombre de la tabla> WHERE <identificador de registro> =**

**<valor>;**

**SELECT \* FROM Nombre de la tabla WHERE Identificador de registro**

Una vez hecho este ejercicio práctico de conocimiento seguiremos con las siguiente actividad .

**-** **Base de datos de demostración – trabajaremos con la de la guía**

**pasada ----**

**CARGAR EL ARCHIVO .SQL Y EL WORD CON LOS PANTALLAZOS AL REPO COMO RELACION\_SQL EN UN ARCHIVO ZIP**

* Diagrama

  Descripción generada automáticamente**MODELO – MAQUETADO.**
* **CÓDIGO**

CREATE DATABASE FACTURACIÓN;

USE FACTURACIÓN;

CREATE TABLE cliente(

id\_cliente VARCHAR (30) UNIQUE PRIMARY KEY, nombre VARCHAR (25) UNIQUE NOT NULL, apellido VARCHAR (25) UNIQUE NOT NULL,

direccion VARCHAR(20) NOT NULL, telefono VARCHAR(20) NOT NULL, correo VARCHAR(50) UNIQUE NOT NULL,

municipio VARCHAR(20), departamento VARCHAR(20),

pais VARCHAR (20) CHECK (pais='Colombia'), -- CHECK permite limitar a una respuesta

creado TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE factura(

num\_factura VARCHAR (20) UNIQUE PRIMARY KEY,

id\_cliente VARCHAR (30),

id\_producto VARCHAR (30),

creado TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (id\_cliente) REFERENCES cliente(id\_cliente),

);

CREATE TABLE productos(

id\_producto VARCHAR (30) UNIQUE PRIMARY KEY, nombre VARCHAR (25) NOT NULL,

precio INT (25) NOT NULL, stock INT (25) NOT NULL,

num\_factura VARCHAR (20),

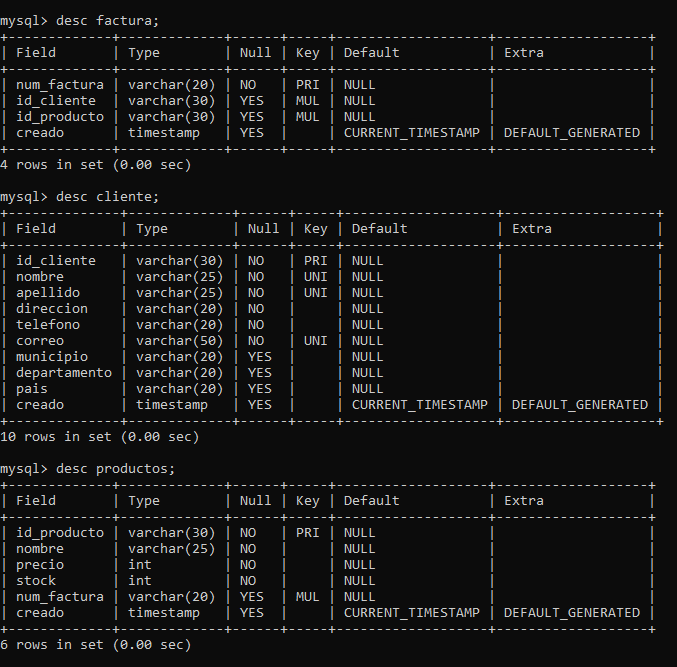
creado TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (num\_factura) REFERENCES factura(num\_factura)

);

ALTER TABLE factura ADD FOREIGN KEY(id\_producto) REFERENCES productos(id\_producto);

* **ESTRUCTURA**



# - Base de datos de demostración

Realizamos inserción de registros:

## -- CLIENTE

INSERT INTO cliente (id\_cliente, nombre, apellido, direccion, telefono, correo, municipio, departamento, pais)

## VALUES ('900876543-0', 'JUAN', 'GOMEZ', 'CALLE 13', '8608978', 'USER1@USER1.COM', 'QUEBRADAS', 'RISARALDA', 'COLOMBIA');

INSERT INTO cliente (id\_cliente, nombre, apellido, direccion, telefono, correo, municipio, departamento, pais)

## VALUES ('800876543-0', 'CAMILO', 'VELEZ', 'CALLE 11', '87098765', 'USER2@USER2.COM', 'ORITO', 'PUTUMAYO', 'COLOMBIA');

INSERT INTO cliente (id\_cliente, nombre, apellido, direccion, telefono, correo, municipio, departamento, pais)

## VALUES ('1077654345', 'JUANA', 'DE ARCO', 'CALLE 10', '861876567', 'USER3@USER3.COM', 'BOGOTÁ', 'CUNDINAMARCA', 'COLOMBIA');

-- FACTURA

INSERT INTO factura (num\_factura, id\_cliente) VALUES ('A-001', '900876543-0');

INSERT INTO factura (num\_factura, id\_cliente) VALUES ('A-002', '1077654345');

INSERT INTO factura (num\_factura, id\_cliente) VALUES ('A-003', '800876543-0');

-- No se insertan valores para la columna id\_producto, debido a que, al ser llave foránea, estos valores deben primero crearse en la tabla original de productos.

## --PRODUCTOS

INSERT INTO productos (id\_producto, nombre, precio, stock, num\_factura) VALUES ('PRO-001', 'PORTÁTIL', '2800000', '70', 'A-001');

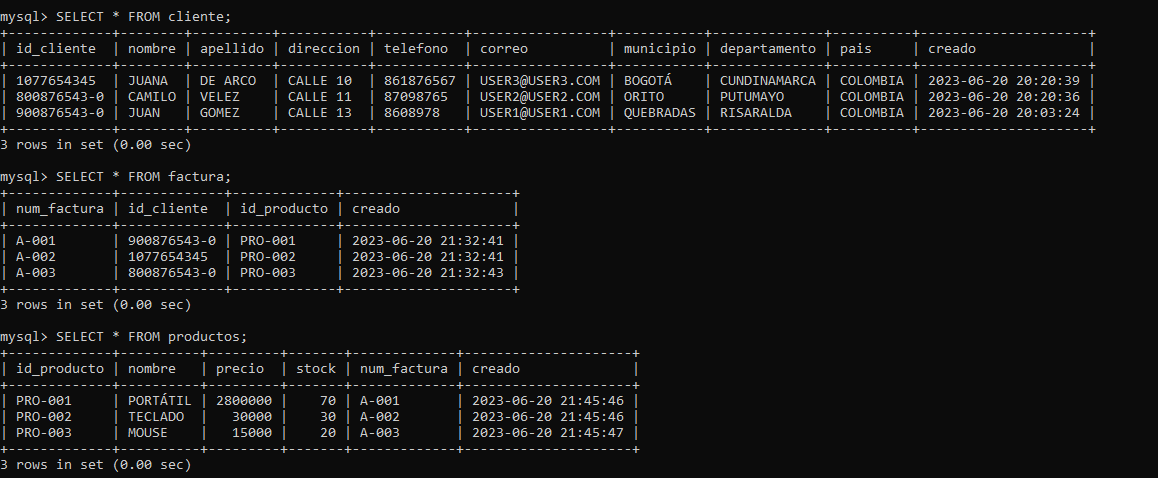
INSERT INTO productos (id\_producto, nombre, precio, stock, num\_factura) VALUES ('PRO-002', 'TECLADO', '30000', '30', 'A-002');

INSERT INTO productos (id\_producto, nombre, precio, stock, num\_factura) VALUES ('PRO-003', 'MOUSE', '15000', '20', 'A-003');

-- Ahora vamos a actualizar la columna id\_producto de la tabla factura.

UPDATE factura SET id\_producto = 'PRO-001' WHERE num\_factura = 'A-001'; UPDATE factura SET id\_producto = 'PRO-002' WHERE num\_factura = 'A-002'; UPDATE factura SET id\_producto = 'PRO-003' WHERE num\_factura = 'A-003';

Ahora tenemos lo siguiente:



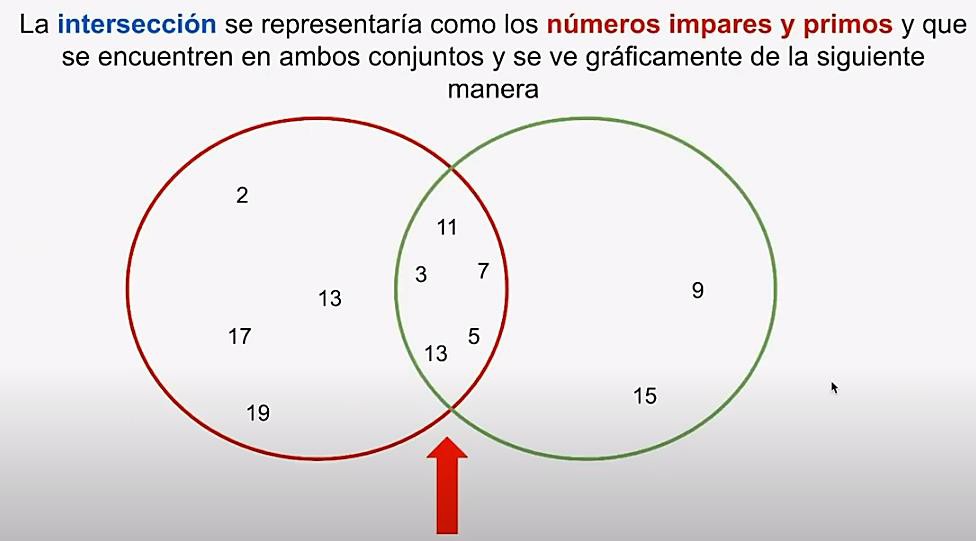
**TEORIA DE CONJUNTOS**

**Ejemplos usando diagramas de BEN**

Diagrama, Diagrama de Venn

Descripción generada automáticamente

**CASO 1.**



**SET DE DATOS RESULTANTE: [3, 5, 7, 11, 13]**

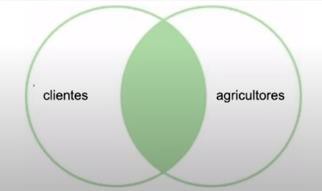
**CASO 2.**

Diagrama, Diagrama de Venn

Descripción generada automáticamente

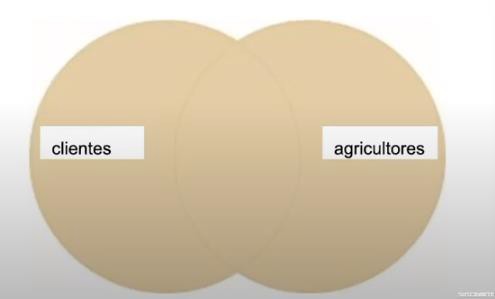
**SET DE DATOS RESULTANTE: [2, 3, 5, 7, 9, 11, 13, 15, 17, 19]**

Lo anterior nos explica cómo funcionan los JOIN, es decir la unión de tablas, de una manera visual, antes de pasar a SQL.





**INNER JOIN:** Devuelve todas las filas cuando hay al menos una coincidencia en ambas tablas.



Tabla

Descripción generada automáticamente

Diagrama, Diagrama de Venn

Descripción generada automáticamente



**LEFT JOIN:** Devuelve todas las filas de la tabla de la izquierda, y las filas coincidentes de la tabla de la derecha.

Un dibujo de una cara con ojos y boca

Descripción generada automáticamente con confianza baja



**RIGHT JOIN:** Devuelve todas las filas de la tabla de la derecha, y las filas coincidentes de la tabla de la izquierda.

Tabla

Descripción generada automáticamente

**OUTER JOIN:** Devuelve todas las filas de las dos tablas, la izquierda y la derecha. También se llama FULL OUTER JOIN.

# Con las variantes INNER, LEFT, RIGHT y FULL de consultas multi-tabla somos capaces de obtener registros relacionados y los registros relacionados + los registros no relacionados en uno de los dos lados o en ambos.

**Ejercicio 1.**

Obtener todos los clientes que tangan por lo menos una factura.

# Estructura:

**SELECT**

**\***

**FROM** <Nombre de la tabla de origen de datos>

**INNER JOIN** <Nombre de la tabla a unir> **ON** <tabla origen.campo en común>

=<tabla a unir.campo en común >; Teoría de conjuntos Intercepción = A

Cliente Factura

A

SELECT

-> \*

-> FROM cliente

-> INNER JOIN factura ON cliente.id\_cliente = factura.id\_cliente;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

# Ejercicio 2.

Obtener todos los clientes con sus facturas, mostrando también los clientes que no tienen factura. (teoría de conjuntos, lado izquierdo)

# Estructura:

**SELECT**

**\***

**FROM** <Nombre de la tabla de origen de datos>

**LEFT JOIN** <Nombre de la tabla a unir> **ON** <tabla origen.campo en común>

=<tabla a unir.campo en común >; Teoría de conjuntos lado izquierdo = A

Cliente Factura

A

1. Obtener todos los clientes con sus facturas:

SELECT

\*

FROM cliente

LEFT JOIN factura ON cliente.id\_cliente = factura.id\_cliente;

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Mostrando también los clientes que no tienen factura.

SELECT

\*

FROM cliente

LEFT JOIN factura ON cliente.id\_cliente = factura.id\_cliente WHERE factura.id\_cliente IS NULL;

Texto

Descripción generada automáticamente

Como todos los clientes tienen factura, no nos retorna datos.

# Ejercicio 3.

Obtener todas las facturas de clientes, mostrando también las facturas que no tienen cliente asociado. (teoría de conjuntos, lado derecho)

# Estructura:

**SELECT**

**\***

**FROM** <Nombre de la tabla de origen de datos>

**RIGHT JOIN** <Nombre de la tabla a unir> **ON** <tabla origen.campo en común>

=<tabla a unir.campo en común >; Teoría de conjuntos lado derecho = A

Cliente Factura

A

1. Obtener todas las facturas de clientes.

SELECT

\*

FROM cliente

RIGHT JOIN factura ON cliente.id\_cliente = factura.id\_cliente;

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Obtener las facturas que no tienen cliente asociado.

SELECT

\*

FROM cliente

RIGHT JOIN factura ON cliente.id\_cliente = factura.id\_cliente WHERE factura.id\_cliente IS NULL;

Texto

Descripción generada automáticamente

La sintaxis de OUTER JOIN o FULL OUTER JOIN no existen en MySQL, pero se puede conseguir el mismo resultado de diferentes formas, esta es una:

SELECT

\*

FROM cliente

LEFT JOIN factura ON cliente.id\_cliente = factura.id\_cliente;

UNION SELECT

\*

FROM clientes

RIGHT JOIN factura ON cliente.id\_cliente = factura.id\_cliente;

TALLER

1. Maquete un modelo entidad relación con por lo menos 4 tablas, declare llaves primarias y foráneas.

Diagrama

Descripción generada automáticamenteEjemplo:

1. Desarrolle el modelo en la terminal.
2. Realice la inserción de por lo menos 15 registros, en cada una de las tablas.
3. Aplique teoría de conjuntos: Inner, Left y Right.
4. Cargue el archivo .sql y los pantallazos en Word o PDF.