ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
CARRERA DE SOFTWARE

Bases de Datos 1

Informe de tarea No. 3

1. DATOS GENERALES:

TEMA: Informe de SQL: DDL

NOMBRE: [redacted] **CODIGO(S):** [redacted]

GRUPO No.: N/A

FECHA DE ENTREGA: 04/05/2024

2. OBJETIVO:

El objetivo de la tarea para el caso de "Órdenes de Compra" es documentar de manera exhaustiva la estructura y definición de la base de datos utilizada en el sistema de gestión de órdenes de compra.

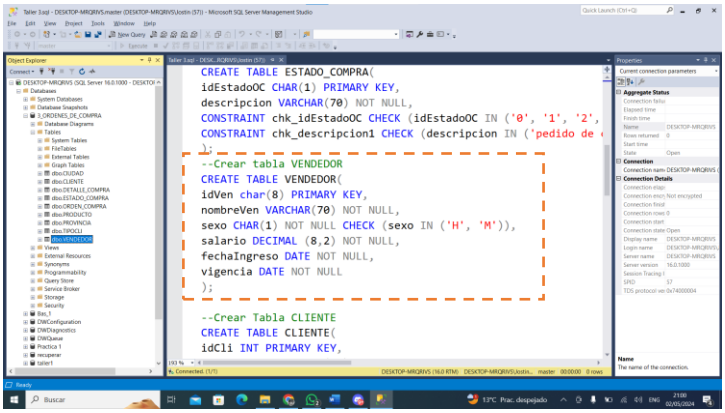
3. ACTIVIDADES DESARROLLADAS:

Crear las tablas de la base de datos (según esquema indicado) utilizando SQL en el RDBMS (CREATE).

Análisis del Caso:
Se realizó una revisión exhaustiva del caso "Órdenes de Compra" para comprender los requisitos del sistema y la estructura de la base de datos necesaria para su implementación.
Identificación de las Entidades Principales:
Se identificaron las entidades principales del sistema a partir del análisis del caso. Las claves primarias primero y después las hijas siguiendo este orden

TABLAS PADRES	TABLAS HIJAS
PROVINCIA TIPOCLI ESTADO_COMPRA VENDEDOR PRODUCTO	ORDEN_COMPRA CLIENTE DETALLE_COMPRA CIUDAD

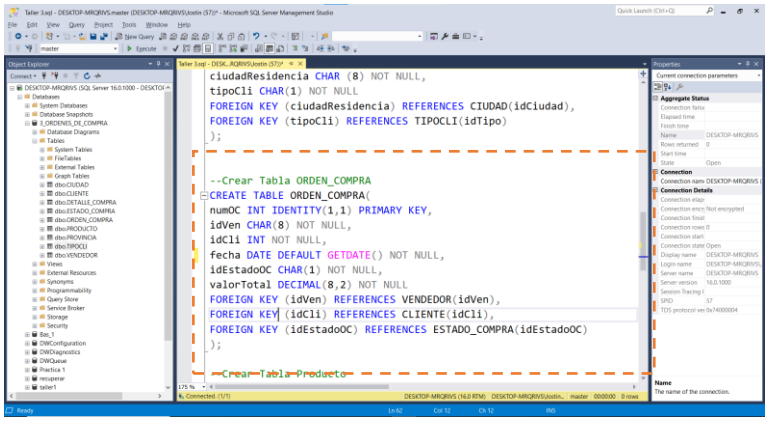
Tabla Vendedor:
La tabla Vendedor almacena información sobre los vendedores que participan en el sistema de órdenes de compra. Cada entrada en esta tabla representa un vendedor y sus atributos incluyen un identificador único (idVen), el nombre del vendedor (nombreVen), su sexo, salario, fecha de ingreso (fechaIngreso), y la vigencia.



```
CREATE TABLE ESTADO_COMPRA(
idEstadoOC CHAR(1) PRIMARY KEY,
descripcion VARCHAR(70) NOT NULL,
CONSTRAINT chk_idEstadoOC CHECK (idEstadoOC IN ('0', '1', '2',
CONSTRAINT chk_descripcion1 CHECK (descripcion IN ('pedido de
);
--Crear tabla VENDEDOR
CREATE TABLE VENDEDOR(
idVen char(8) PRIMARY KEY,
nombreVen VARCHAR(70) NOT NULL,
sexo CHAR(1) NOT NULL CHECK (sexo IN ('H', 'M')),
salario DECIMAL (8,2) NOT NULL,
fechaIngreso DATE NOT NULL,
vigencia DATE NOT NULL
);
--Crear Tabla CLIENTE
CREATE TABLE CLIENTE(
idCli INT PRIMARY KEY,
```

Tabla Orden Compra:
La tabla Orden_Compra registra cada orden de compra realizada en el sistema. Cada registro en esta tabla contiene detalles sobre una orden específica, como el número de orden de compra (numOC), el identificador del vendedor asociado (idVen), el identificador del cliente (idCli), la fecha de la orden, el estado de la orden (idEstadoOC), y el valor total de la compra.

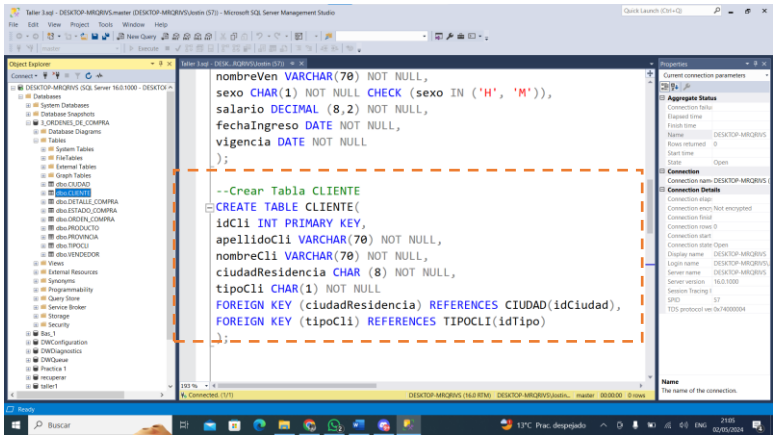
Nota: numOC debe ser auto-incremental



```
ciudadResidencia CHAR (8) NOT NULL,
tipoCli CHAR(1) NOT NULL
FOREIGN KEY (ciudadResidencia) REFERENCES CIUDAD(idCiudad),
FOREIGN KEY (tipoCli) REFERENCES TIPOCLI(idTipo)
);

--Crear Tabla ORDEN_COMPRA
CREATE TABLE ORDEN_COMPRA(
numOC INT IDENTITY(1,1) PRIMARY KEY,
idVen CHAR(8) NOT NULL,
idCli INT NOT NULL,
fecha DATE DEFAULT GETDATE() NOT NULL,
idEstadoOC CHAR(1) NOT NULL,
valorTotal DECIMAL(8,2) NOT NULL
FOREIGN KEY (idVen) REFERENCES VENDEDOR(idVen),
FOREIGN KEY (idCli) REFERENCES CLIENTE(idCli),
FOREIGN KEY (idEstadoOC) REFERENCES ESTADO_COMPRA(idEstadoOC)
);
--Crear Tabla Producto
```

Tabla Cliente:
La tabla Cliente almacena información sobre los clientes que realizan órdenes de compra en el sistema. Cada fila en esta tabla representa un cliente y sus atributos incluyen un identificador único (idCli), el apellido del cliente (apellidoCli), nombre del cliente (nombreCli), su ciudad de residencia (ciudadResidencia), y el tipo de cliente (idTipo).



```
nombreVen VARCHAR(70) NOT NULL,
sexo CHAR(1) NOT NULL CHECK (sexo IN ('H', 'M')),
salario DECIMAL (8,2) NOT NULL,
fechaIngreso DATE NOT NULL,
vigencia DATE NOT NULL
);

--Crear Tabla CLIENTE
CREATE TABLE CLIENTE(
idCli INT PRIMARY KEY,
apellidoCli VARCHAR(70) NOT NULL,
nombreCli VARCHAR(70) NOT NULL,
ciudadResidencia CHAR (8) NOT NULL,
tipoCli CHAR(1) NOT NULL
FOREIGN KEY (ciudadResidencia) REFERENCES CIUDAD(idCiudad),
FOREIGN KEY (tipoCli) REFERENCES TIPOCLI(idTipo)
);
```

Tabla TipoCli:
La tabla TipoCli contiene los diferentes tipos de clientes que pueden realizar órdenes de compra en el sistema. Cada registro en esta tabla describe un tipo de cliente y sus atributos incluyen un identificador único (idTipo) y una descripción del tipo de cliente.

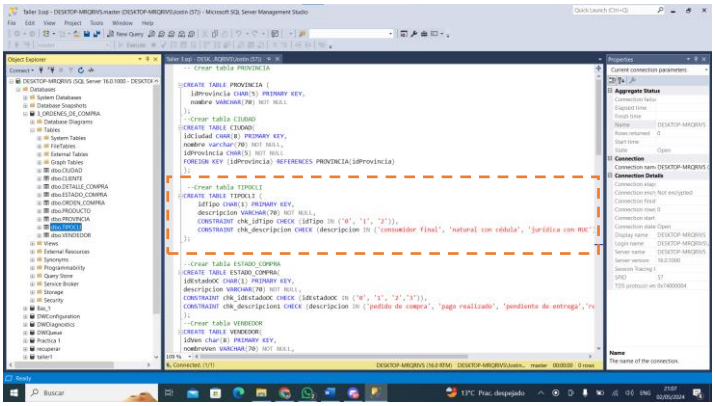


Tabla Detalle_Compra:
La tabla Detalle_Compra guarda los detalles de cada producto comprado en una orden de compra. Cada fila en esta tabla representa un artículo específico comprado en una orden, con detalles como el número de orden de compra (numOC), el identificador del producto (idProducto), la cantidad comprada, y el subtotal de la compra.

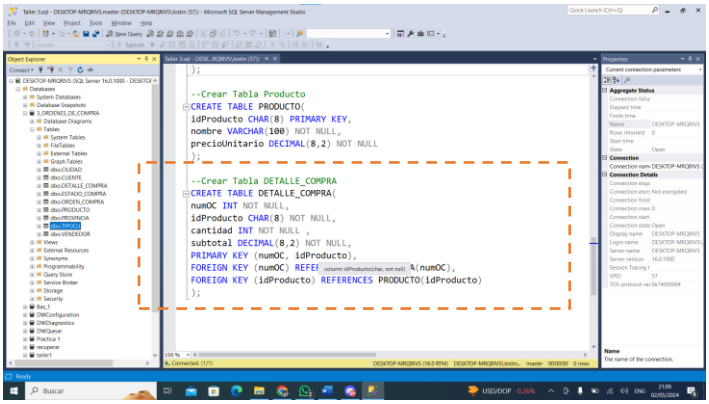


Tabla Producto:
La tabla Producto contiene información sobre los productos disponibles para su compra en el sistema. Cada entrada en esta tabla representa un producto y sus atributos incluyen un identificador único (idProducto), el nombre del producto, y su precio unitario.

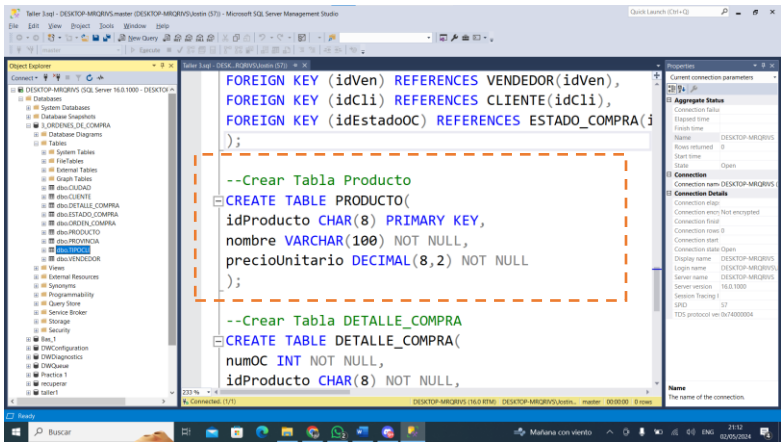


Tabla Estado_Compra:
La tabla Estado_Compra registra los diferentes estados posibles de una orden de compra en el sistema. Cada fila en esta tabla describe un estado y sus atributos incluyen un identificador único (idEstadoOC) y una descripción del estado.

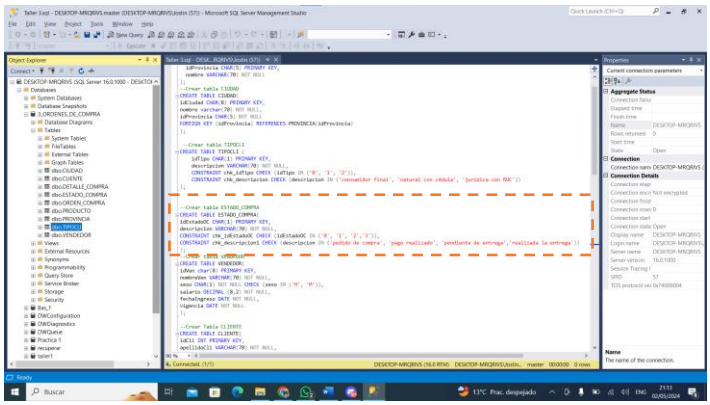


Tabla Provincia:
La tabla Provincia almacena información sobre las provincias a las que pertenecen las ciudades en el sistema. Cada entrada en esta tabla representa una provincia y sus atributos incluyen un identificador único (idProvincia) y el nombre de la provincia.

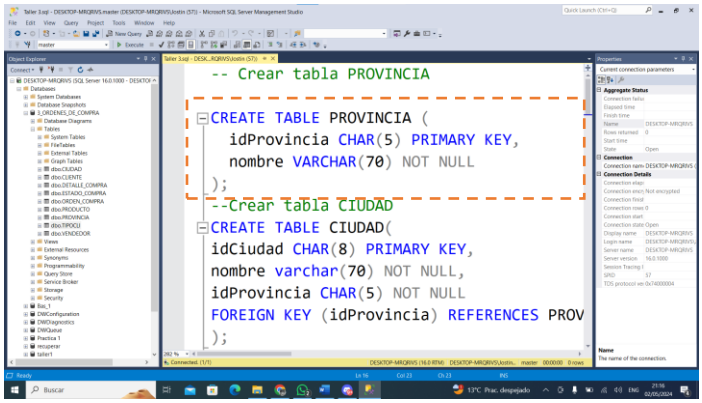
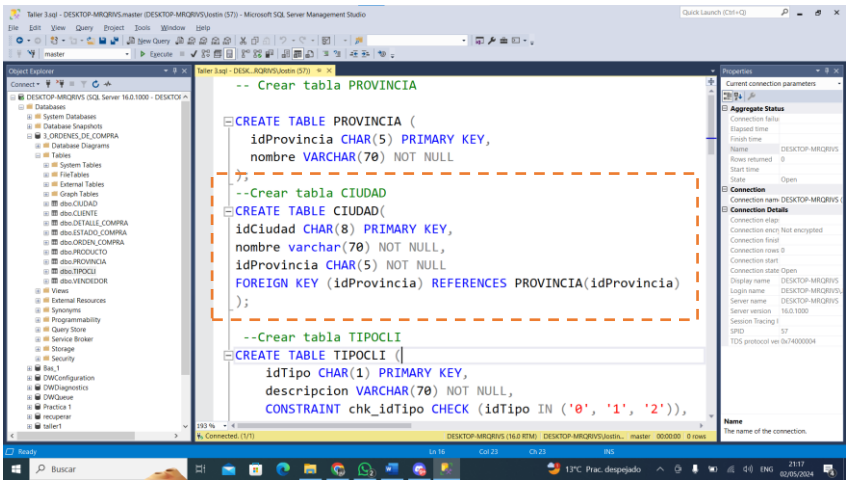
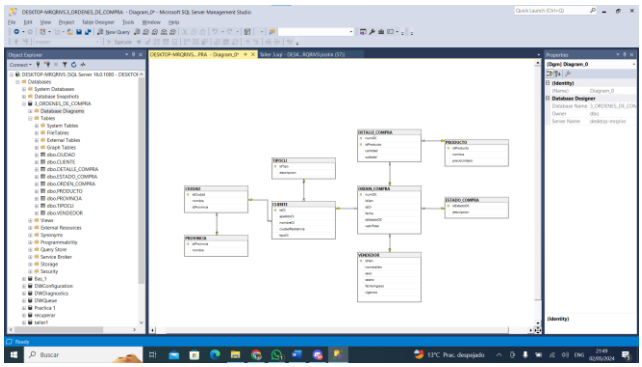


Tabla Ciudad:
La tabla Ciudad contiene datos sobre las ciudades presentes en el sistema. Cada registro en esta tabla representa una ciudad y sus atributos incluyen un identificador único (idCiudad), el nombre de la ciudad y el identificador de la provincia a la que pertenece.



Referencias entre las Tablas:
CIUDAD hace referencia a PROVINCIA.
CLIENTE hace referencia a CIUDAD.
CLIENTE hace referencia a TIPOCLI.
ORDEN_COMPRA hace referencia a VENDEDOR.
ORDEN_COMPRA hace referencia a CLIENTE.
ORDEN_COMPRA hace referencia a ESTADO_COMPRA.
DETALLE_COMPRA hace referencia a ORDEN_COMPRA.
DETALLE_COMPRA hace referencia a PRODUCTO.



Restricciones:

VENDEDOR:
Sexo: H: hombre , M: mujer

```
sexo CHAR(1) NOT NULL CHECK (sexo IN ('H', 'M')),
```

ESTADO_COMPRA
idEstadoOC: 0,1,2,3
descripcion: 0: pedido de compra , 1: pago realizado , 2: pendiente de entrega , 3: realizada la entrega

```
CONSTRAINT chk_idEstadoOC CHECK (idEstadoOC IN ('0', '1', '2','3')),
CONSTRAINT chk_descripcion1 CHECK (descripcion IN ('pedido de compra', 'pago realizado', 'pendiente de entrega','realizada la entrega'))
);
```

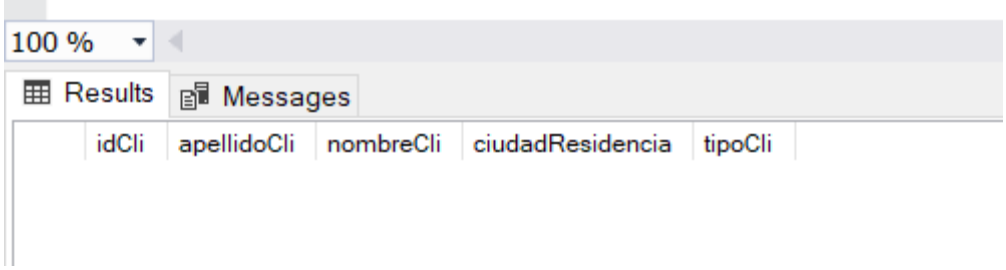
TIPOCLI
idTipo: 0, 1, 2
descripcion: 0: consumidor final , 1: natural con cédula , 2: jurídica con RUC

```
    CONSTRAINT chk_idTipo CHECK (idTipo IN ('0', '1', '2')),
    CONSTRAINT chk_descripcion CHECK (descripcion IN ('consumidor final', 'natural con cédula', 'jurídica con RUC'))
);
```

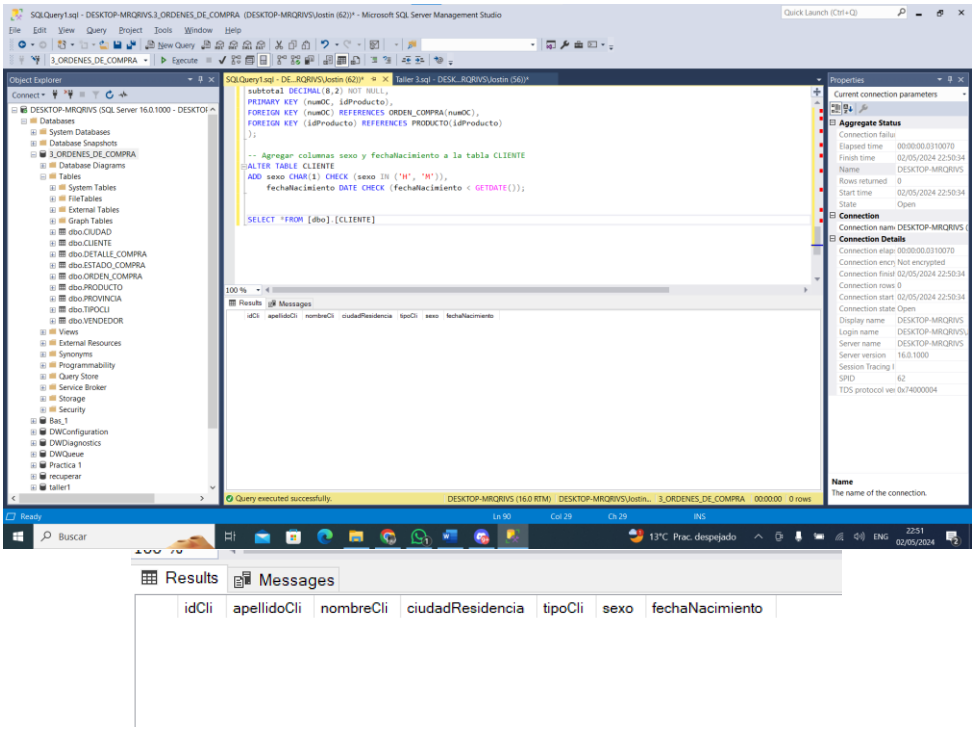
Otros: los que considere

```
--Crear Tabla ORDEN_COMPRA
CREATE TABLE ORDEN_COMPRA(
numOC INT IDENTITY(1,1) PRIMARY KEY,
idVen CHAR(8) NOT NULL,
idCli INT NOT NULL,
fecha DATE DEFAULT GETDATE() NOT NULL,
idEstadoOC CHAR(1) NOT NULL,
valorTotal DECIMAL(8,2) NOT NULL
FOREIGN KEY (idVen) REFERENCES VENDEDOR(idVen),
FOREIGN KEY (idCli) REFERENCES CLIENTE(idCli),
FOREIGN KEY (idEstadoOC) REFERENCES ESTADO_COMPRA(idEstadoOC)
);
```

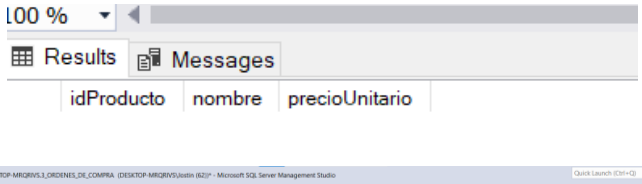
2. Modificar la estructura de una de las tablas de la base de datos (ALTER)

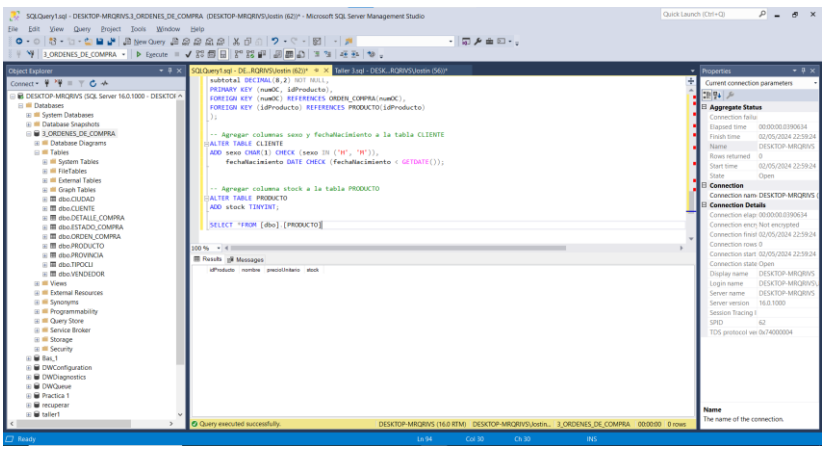


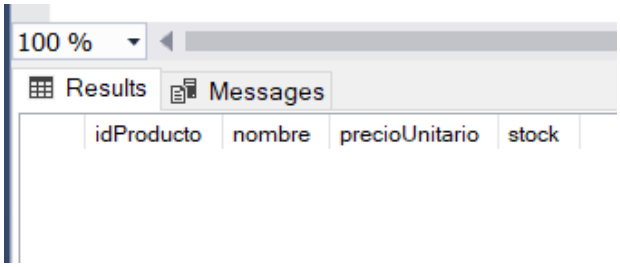
Agrega dos columnas (considerar check de validación) a la tabla cliente: sexo y fechaNacimiento



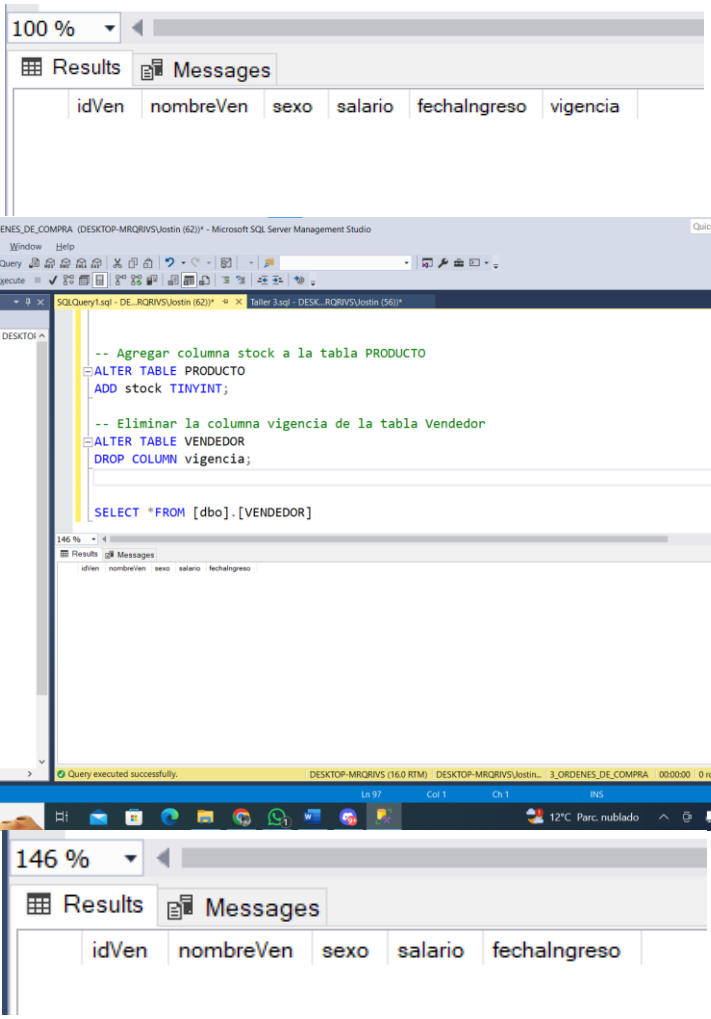
Agregar la columna stock en la tabla producto



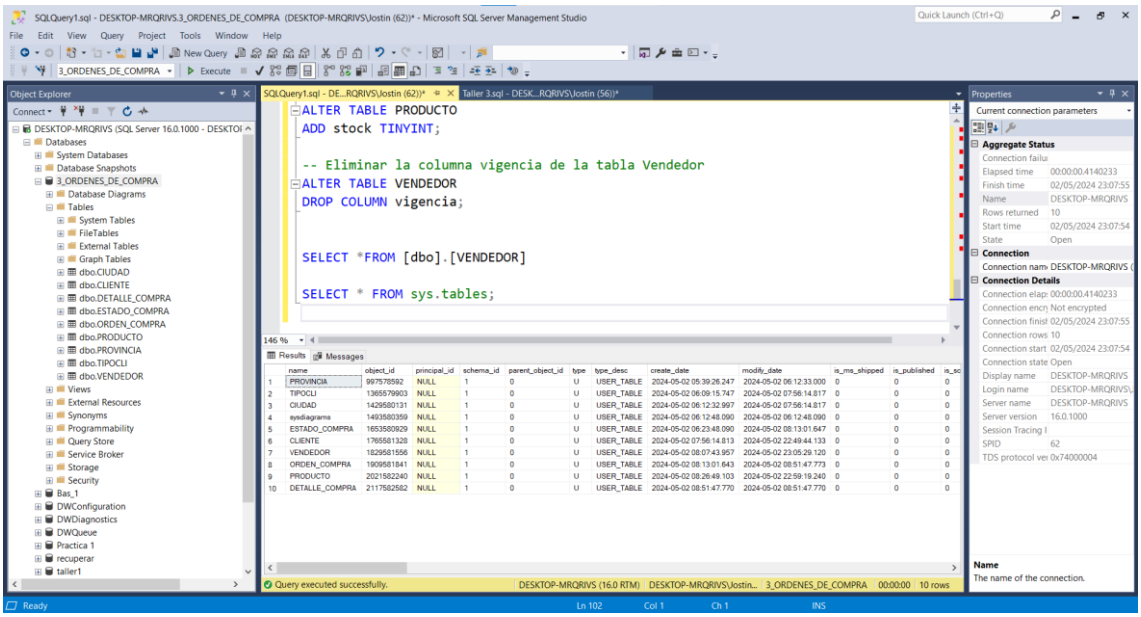




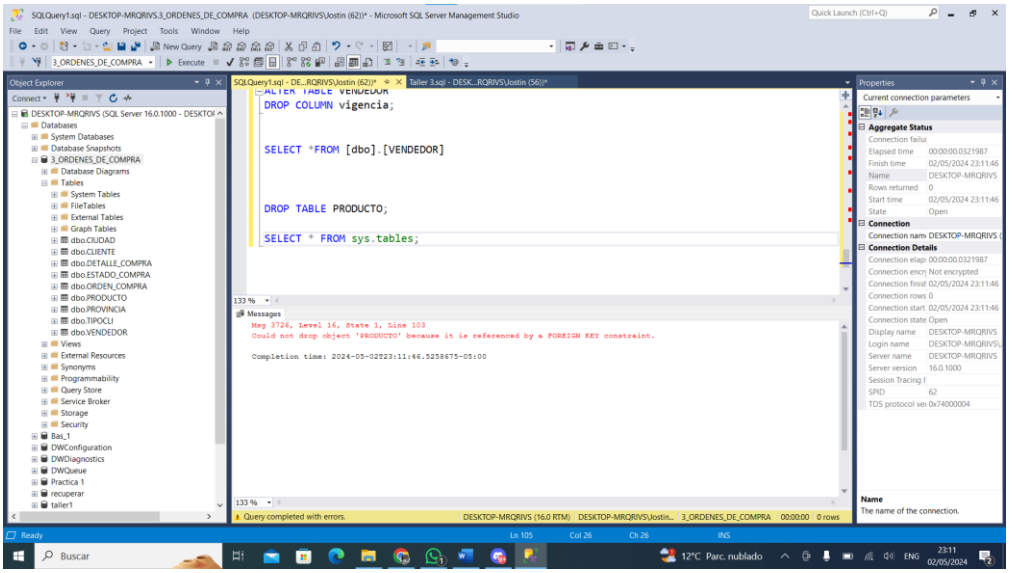
3. Eliminar la columna vigencia de la tabla vendedor



4. Eliminar una de las tablas (DROP) creadas



El mensaje de error "Msg 3726, Level 16, State 1" indica que la operación de eliminación de la tabla 'PRODUCTO' no pudo completarse debido a una restricción de clave externa (FOREIGN KEY constraint). Esta restricción impide la eliminación directa de la tabla debido a que existe una relación de referencia entre esta y al menos una otra tabla en la base de datos. Sin embargo, a continuación se presenta el código necesario para llevar a cabo la eliminación de la tabla 'PRODUCTO':



5. RESULTADOS OBTENIDOS

Se ha creado el código SQL con éxito para la creación de las tablas necesarias en el contexto del caso "Órdenes de Compra". Todas las tablas han sido creadas con sus respectivas referencias y restricciones de integridad, asegurando la coherencia y consistencia de los datos en la base de datos.

Además, se ha demostrado la capacidad de agregar y eliminar columnas de las tablas existentes utilizando la sentencia ALTER TABLE, lo que permite una flexibilidad adicional en la gestión de la estructura de la base de datos.

A continuación, se adjunta el código SQL utilizado, así como cualquier mensaje de error o confirmación relevante. Este documento proporciona una referencia útil para futuras revisiones o modificaciones en el sistema de gestión de órdenes de compra.

```
-- Crear tabla PROVINCIA
CREATE TABLE PROVINCIA (
  idProvincia CHAR(5)  PRIMARY KEY,
  nombre VARCHAR(70)  NOT NULL
);
--Crear tabla CIUDAD
CREATE TABLE CIUDAD(
idCiudad  CHAR(8) PRIMARY KEY,
nombre varchar(70) NOT NULL,
idProvincia CHAR(5) NOT NULL
FOREIGN KEY (idProvincia) REFERENCES PROVINCIA(idProvincia)
);
 --Crear tabla TIPOCLI
CREATE TABLE TIPOCLI (
   idTipo CHAR(1)  PRIMARY KEY,
   descripcion VARCHAR(70)  NOT NULL,
   CONSTRAINT chk_idTipo CHECK (idTipo IN ('0', '1', '2')),
   CONSTRAINT chk_descripcion CHECK (descripcion IN ('consumidor final', 'natural con cédula', 'jurídica con RUC'))
);
--Crear tabla ESTADO_COMPRA
CREATE TABLE ESTADO_COMPRA(
idEstadoOC CHAR(1) PRIMARY KEY,
descripcion VARCHAR(70)  NOT NULL,
CONSTRAINT chk_idEstadoOC CHECK (idEstadoOC IN ('0', '1', '2','3')),
CONSTRAINT chk_descripcion1 CHECK (descripcion IN ('pedido de compra', 'pago realizado', 'pendiente de
entrega','realizada la entrega'))
);
--Crear tabla VENDEDOR
CREATE TABLE VENDEDOR(
idVen char(8) PRIMARY KEY,
nombreVen VARCHAR(70)  NOT NULL,
sexo CHAR(1)  NOT NULL CHECK (sexo IN ('H', 'M')),
salario  DECIMAL (8,2) NOT NULL,
fechaIngreso  DATE NOT NULL,
vigencia DATE NOT NULL
);
--Crear Tabla CLIENTE
CREATE TABLE CLIENTE(
idCli INT PRIMARY KEY,
apellidoCli  VARCHAR(70)  NOT NULL,
nombreCli VARCHAR(70)  NOT NULL,
ciudadResidencia  CHAR (8) NOT NULL,
tipoCli  CHAR(1) NOT NULL
FOREIGN KEY (ciudadResidencia)  REFERENCES CIUDAD(idCiudad),
FOREIGN KEY (tipoCli) REFERENCES TIPOCLI(idTipo)
);
--Crear Tabla ORDEN_COMPRA
CREATE TABLE ORDEN_COMPRA(
numOC INT IDENTITY(1,1) PRIMARY KEY,
```

```
idVen CHAR(8) NOT NULL,
idCli INT NOT NULL,
fecha DATE NOT NULL,
idEstadoOC CHAR(1) NOT NULL,
valorTotal DECIMAL(8,2) NOT NULL
FOREIGN KEY (idVen) REFERENCES VENDEDOR(idVen),
FOREIGN KEY (idCli) REFERENCES CLIENTE(idCli),
FOREIGN KEY (idEstadoOC) REFERENCES ESTADO_COMPRA(idEstadoOC)
);
--Crear Tabla Producto
CREATE TABLE PRODUCTO(
idProducto CHAR(8) PRIMARY KEY,
nombre VARCHAR(100)  NOT NULL,
precioUnitario DECIMAL(8,2) NOT NULL
);
--Crear Tabla DETALLE_COMPRA
CREATE TABLE DETALLE_COMPRA(
numOC INT NOT NULL,
idProducto CHAR(8) NOT NULL,
cantidad INT NOT NULL ,
subtotal DECIMAL(8,2) NOT NULL,
PRIMARY KEY (numOC, idProducto),
FOREIGN KEY (numOC) REFERENCES ORDEN_COMPRA(numOC),
FOREIGN KEY (idProducto) REFERENCES PRODUCTO(idProducto)
);

-- Agregar columnas sexo y fechaNacimiento a la tabla CLIENTE
ALTER TABLE CLIENTE
ADD sexo CHAR(1) CHECK (sexo IN ('H', 'M')),
   fechaNacimiento DATE CHECK (fechaNacimiento < GETDATE());

-- Agregar columna stock a la tabla PRODUCTO
ALTER TABLE PRODUCTO
ADD stock TINYINT;

-- Eliminar la columna vigencia de la tabla Vendedor
ALTER TABLE VENDEDOR
DROP COLUMN vigencia;

SELECT *FROM [dbo].[VENDEDOR]

DROP TABLE PRODUCTO;

SELECT * FROM sys.tables;
```

6. CONCLUSIONES

Durante el proceso de diseño y creación de la base de datos para el caso "Órdenes de Compra", se han identificado varios aspectos clave que destacan tanto los éxitos como los desafíos enfrentados.

En primer lugar, el proceso de creación de tablas se llevó a cabo de manera satisfactoria, demostrando una comprensión profunda de los requisitos del sistema y una habilidad para traducir estos requisitos en estructuras de base de datos funcionales la utilización adecuada de las sentencias SQL para crear tablas, definir claves primarias y foráneas, así como establecer restricciones de integridad, contribuyó a la creación de una base de datos robusta y coherente.

Sin embargo, también se encontraron desafíos significativos durante el proceso la aparición de errores, como la imposibilidad de eliminar una tabla debido a restricciones de clave externa, destacó la importancia de comprender completamente las relaciones entre las tablas y planificar cuidadosamente cualquier modificación estructural estos errores subrayan la necesidad de una atención meticulosa a los detalles y una comprensión profunda de los principios de diseño de bases de datos relacionales.

A pesar de estos desafíos, el proceso también estuvo marcado por numerosos éxitos. La capacidad para agregar y eliminar columnas de las tablas existentes utilizando la sentencia ALTER TABLE demostró una flexibilidad considerable en la gestión de la estructura de la base de datos. Este nivel de flexibilidad es esencial en entornos en los que los requisitos del sistema pueden cambiar con el tiempo.

7. RECOMENDACIONES

Planificación Detallada: Antes de comenzar a crear la base de datos, es fundamental realizar una planificación detallada que incluya la identificación de todas las entidades, atributos y relaciones necesarias. Esto ayuda a evitar errores y garantiza que la base de datos refleje con precisión las necesidades del sistema.

Comprensión Profunda de las Relaciones: Es esencial comprender a fondo las relaciones entre las diferentes tablas, especialmente cuando se trata de restricciones de clave externa. Una comprensión clara de estas relaciones ayuda a evitar errores como la imposibilidad de eliminar tablas debido a restricciones de clave externa.

Documentación Completa: Mantener una documentación completa y actualizada de la estructura de la base de datos, incluyendo todas las tablas, columnas, relaciones y restricciones, facilita la comprensión y el mantenimiento del sistema para los desarrolladores y administradores.

Formación Continua: Mantenerse al día con las últimas tendencias y mejores prácticas en diseño de bases de datos mediante la formación continua y la participación en comunidades profesionales puede mejorar las habilidades y la eficacia en la gestión de bases de datos.