UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SEMESTRE 2023-1

Caso de estudio

GRUPO 02

PROFESORA: M.I MARTHA LÓPEZ

PELCASTRE

INTEGRANTES:

Número de cuenta	Nombre	Número de lista
[redacted]	Hernández Solis Brandon	17
[redacted]	Rivera Morales Alan Adrián	27
[redacted]	Román Cruz Hugo	29

Fecha de entrega: 10 de enero de 2023

Caso de estudio de Telemarketing

Introducción

Es una estrategia de marketing directo que aprovecha que, hoy en día, todos tienen un teléfono. Busca crear conversaciones entre una empresa y sus clientes para ofrecer sus productos o servicio.

Por nuestra parte, somos parte de un trabajo de telemarketing sobre “Venta de peluches” de la empresa Disney.

Objetivos

Dentro del objetivo principal tenemos la venta de productos por vía de llamada telefónica, esto llevara una relación cliente vendedor en una sola llamada, podremos agregar una llamada vía internet pos si sucede algún accidente y se reportara a observaciones. Buscamos facilitar la comunicación con cliente y vendedor, por ende, trataremos que la llamada no dure mucho tiempo para no afectar nuestro tiempo y el del usuario.

Los clientes tomaremos sus datos con su teléfono que está marcando, nombre, apellidos, número de tarjeta de pago, domicilio y fecha de entrega. Nuestros vendedores tienen que identificarse con su nombre, su número telefónico, su id de vendedor, fecha de ingreso, correo electrónico y puesto que tiene.

Al realizar una compra, este se registrará en almacén para saber si tenemos producto o se solicitará más para hacer restablecimiento de peluches. Nuestro almacén tendrá que mandar informes de que peluches quedan, modelos, numero, fecha de ingreso y salida del modelo (en caso de ser exclusivo).

Nuestros informes los tendrá un observador que es el organizador de todo el almacén, las ventas las tendrá un contador, el marketing lo tendrá un especialista y el empleado general. No se puede tener dos trabajos al mismo tiempo por problemas de confianza y de la empresa.

Las ventas se guardarán con su id, fecha de pedido, fecha de salida, nombre del producto, código del producto y con la dirección de entrega. Las ventas solo se pueden realizar de lunes a jueves, los otros días solo se pueden hacer por internet.

La última sección, es la de atención a clientes y esta solo se podrá realizar si es llamada externa o solicitada por el cliente, esta atención solo solicita informe del detalle, fecha de lo ocurrido, nombre del cliente, número telefónico, correo y posibilidad de solución. Solo se podrá atender de lunes a viernes de un horario de 7 am a 7 pm.

Desarrollo

Para realizar la base de datos tuvimos que realizar varias etapas a lo largo del semestre, desde la conceptualización de la base de datos hasta el desarrollo de las consultas.

Al comienzo pensamos únicamente en la base de datos de una fábrica de peluches, la propuesta después se convirtió en la simulación de una base de datos de una tienda de peluches que además ofrece servicio en línea y teléfono. Además consideramos pertinente que parte del servicio contemplara la resolución de los problemas que se pudieran presentar.

Los tipos de usuario que hay en la base de datos son únicamente el empleado y el cliente. Cada uno con los datos estándar de cada uno de estos (nombre, apellido, domicilio, etc). Ya que el domicilio contiene varias entidades en ambos casos se va a guardar la información en tablas distintas.

Por el lado de los empleados habrán empleados para distintos roles, tales como almacenador, empleados que brindan atención al cliente, encargados de marketing, vendedores y contador. Dependiendo del rol habrá diferentes campos en su respectiva tabla

Una relación más de los empleados es la de los pagos, en la cual podemos encontrar los campos sueldo, descuentoPagos y bonos. Esta entidad a su vez se encuentra relacionada con la entidad de pagos, antes mencionadas.

Respecto a los clientes, sus relaciones son con la entidad suscripción y una entidad que a su vez se relaciona con el vendedor y la venta. Cada uno de estos contiene sus campos, los cuales pueden guardadr un historial, el código del producto, la fecha de pedido o cualquier información que consideremos útil para las demás relaciones que contiene la base de datos.

Parte importante de la base de datos es la rama de la base de datos que implica la entidad almacén, ya que esta tiene relación con el empleado que cumple la función de almacenador. Así como con el producto de nuestra tienda.

Respecto a nuestro producto consideramos que las entidades que debía contener eran peluchaesExclusivos, numerColeccion, tipo, fechaCreacion, idAlmacen.

Por ultimo para solucionar los errores que se podían presentar a los clientes creamos la entidad solicitudes, que tendría relación con el empleado que se encargara de estas, las cuales tenían que pasar por al relación de atención a cliente y la entidad revisa para relacionar a las otras dos.

Una vez reconocidas las entidades que queríamos dentro de nuestra base de datos, lo que restaba era encontrar las relaciones que necesitábamos realizar, para poder realizar nuestro diagrama. El siguiente paso fue analizar el tipo de datos que necesitábamos para cada campo, asi como identificar las llaves que íbamos a tener. Una vez con esto pudimos dar paso a las creaciones las tablas en sql sever.

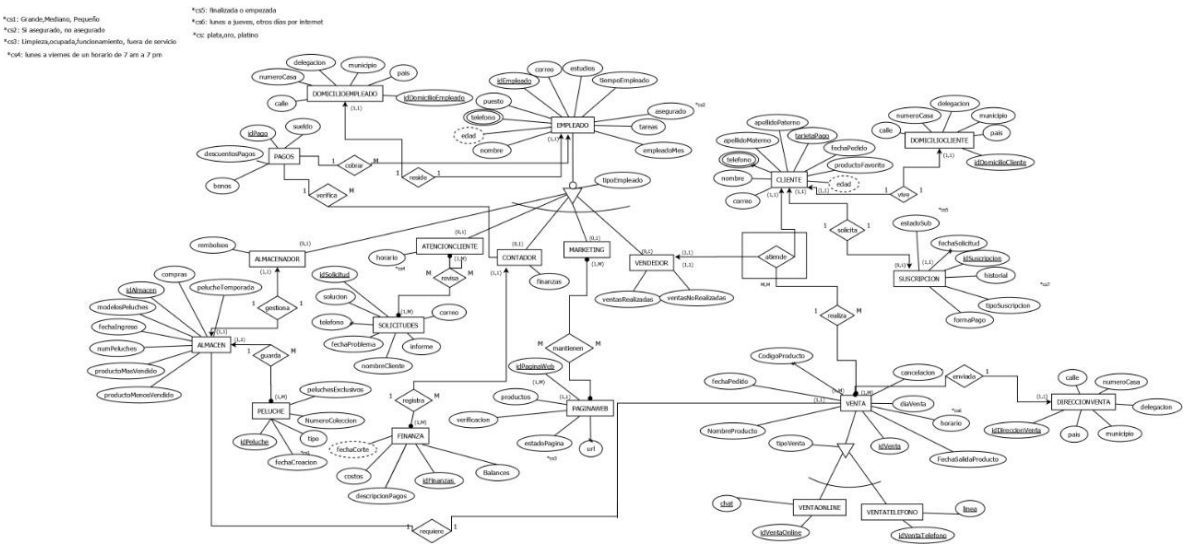
A la par de la realización de cada una de estas tareas intentamos completar las demás actividades, como es la lista de las entidades que tenemos. La creación las tablas no fue tan complicada apoyándonos con el mapa y de la lista de tipos de datos que utilizaríamos para cada una de las entidades necesarias de nuestra base de datos.

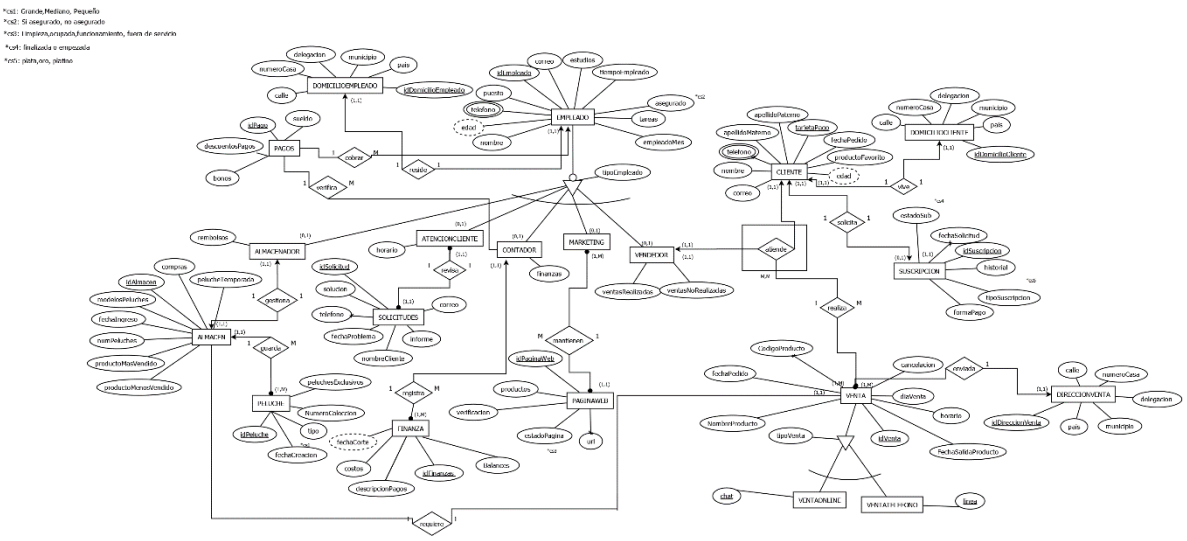
Ya con la estructura de nuestra base creada, lo que hicimos fue realizar el scrip que iba a iniciar la base de datos, es decir, meter información en nuestra base de datos para comprobar que se almacenara correctamente y ver si se cumplían las consideraciones que en un inicio se tenían planteadas para la base de datos.

Los miembros que desarrollamos la base consideramos que la parte más complicada del desarrollo fue la realización e los disparadores o triggers (ya que fue un tema que se concluyó bastante rapido). Por último se agregaron algunas consultas para obtener información de nuestra base de datos que nosotros consideramos podrían ser útiles.

ARCHIVO	NOMBRE	OBSERVACIONES
Diseño conceptual	Diseño_Conceptual.dia	El archivo contiene el diseño conceptual de nuestra base de datos Telemarketing
Diseño lógico	Diseño_Logico.dm1	Cuenta con el modelo relacional en una extensión que nos permite abrir ER/Studio. También tiene la generación del modelo físico para SQL Server.
DDL	crearBase.sql	Este script nos sirve para crear tablas, índices, esquemas y constraints de tipo CHECK.
DML	Dml.sql	Desde este archivo .sql guardamos la creación de triggers y stored procedures.
Script para cargar información.	cargaInicial.sql	Contiene toda la carga de información (inserts).

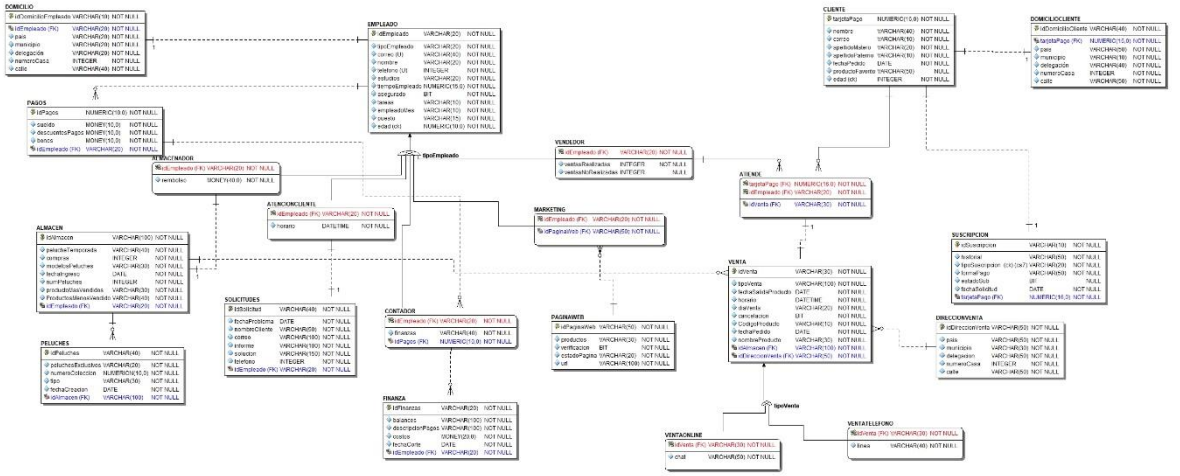
Diseño conceptual



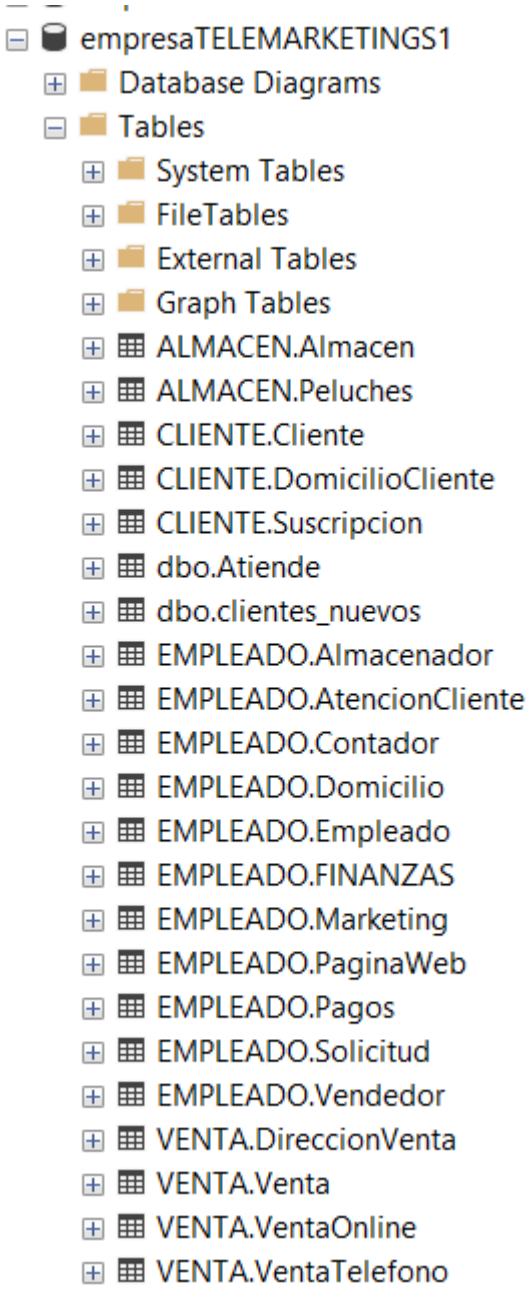


Diseño lógico

Modelo relacional



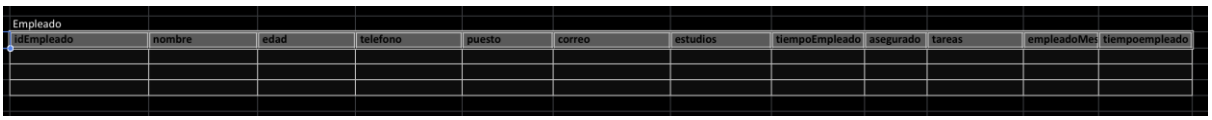
Tablas



Normalización

Normalización de el proyecto

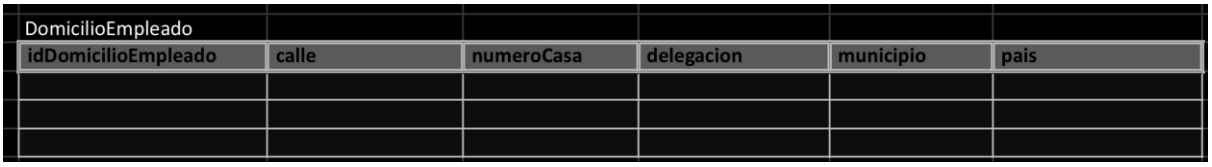
Empleado



3 FN

EMPLEADO = { idEmpleado (PK), nombre, edad, teléfono, puesto, correo, estudios, tiempoEmpleado, asegurado, tareas, empleadoMes }

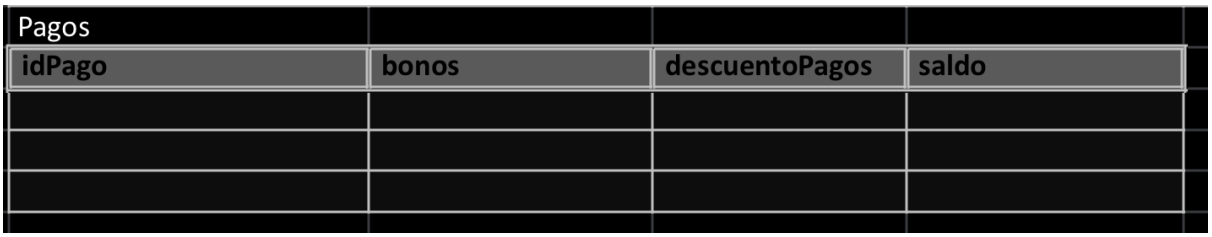
Domicilio Empleado



3 FN

DOMICILIOEMPLEADO = { idDomicilioEmpleado (PK), calle, numeroCasa, delegación, municipio, pais }

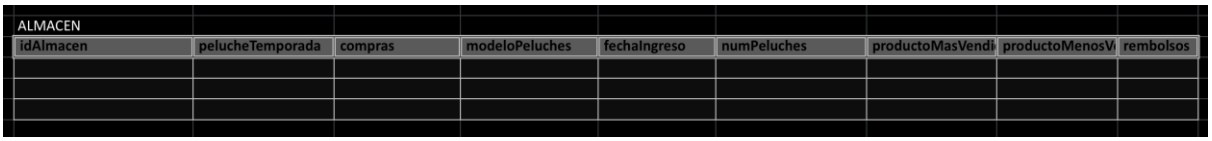
Pagos



3 FN

PAGOS = { idPagos (PK), bonos, descuentosPagos, saldo }

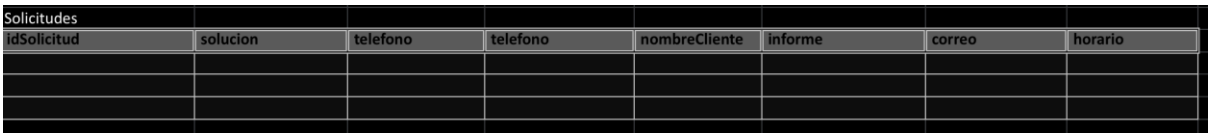
Almacén



3 FN

ALMACÉN { idAlmacen (PK), pelucheTemporada, compras, modeloPeluches, fechaIngreso, numPeluches, productoMasVendido, productoMenosVendido, rembolsos}

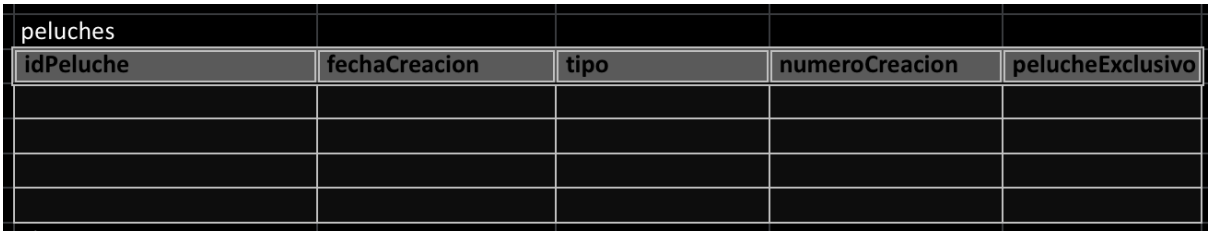
Solicitudes



3 FN

SOLICITUDES = { idSolicitudes (PK), solución, teléfono, nombreCliente, informe, correo, horario}

Peluches



3 FN

PELUCHES = { idPeluche (PK), fechaCreacion, tipo, numeroCreacion, pelucheExclusivo}

Finanzas

Finanzas				
idFinanzas	balanceos	descripcionPagos	costos	fechaCorte

3 FN

FINANZAS = { idFinanzas (PK), balanceos, descripcionPagos, costos, fechaCorte}

Página Web

paginaweb				
idPaginaweb	producto	verificacion	estadoPagina	url

3 FN

PAGINAWEB = { idPaginaWeb (PK), producto, verificación, estadoPagina, url}

Cliente

Cliente										
tarjetaPago	correo	nombre	telefono	apellidoMaterno	apellidoPaterno	fechaPedido	productoFavortio	edad	ventasRealizadas	ventasNoReal

3 FN

CLIENTE = { idCliente (PK), tarjetaPago, correo, teléfono, nombre, apellidoPaterno, apellidoMaterno, fechaPedido, productoFavorito, edad, ventasRealizadas, ventasNoRealizadas}

Domicilio Cliente

Domicilio					
idDomicilioCliente	calle	numeroCasa	delegacion	municipio	pais

3 FN

DOMICILIOCLIENTE = { idDomicilioCliente (PK), calle, numeroCasa, delegación, municipio, pais}

Suscripción

suscripcion					
idSuscripcion	estadoSub	fechaSolicitud	historial	tipoSuscripcion	formaPago

3 FN

SUSCRIPCIÓN = { idSuscripcion (PK), estadoSub, fechaSolicitud, historial, tipoSuscripcion, formaPago}

Venta

Venta										
idVenta	codigoProducto	fechaPedido	fechaPedido	nombreProducto	tipoVenta	horario	fechaVenta	cancelacion	chat	linea

1 FN

VENTA = { (idVenta, codigoProducto) (PK), fechaPedido, nombreProducto, tipoVenta, horario, fechaVenta, cancelación, chat, linea}

3 FN

PRODUCTO = { codigoProducto (PK), nombreProducto}

VENTA = { idVenta (PK), codigoProducto (FK), fechaPedido, fechaVenta, tipoVenta, horario, cancelación, chat, linea}

Dirección Venta

3 FN

DIRECCIONVENTA = { idDireccionVenta (PK), país, municipio, delegación, numeroCasa, calle}

Diseño Fisico

--Fecha Creacion : 21/12/2022

--Descripcion : Script para crear la base de datos telemarketing

--semestre : 2023-1 Grupo : 2

--Equipo: telemarketing

--Creacion de la base de datos

```
CREATE DATABASE empresaTELEMARKETINGS1
go

USE empresaTELEMARKETINGS1
GO
```

--Creación de esquemas (subcarpetas)

--Contendra : EMPLEADO,DOMICILIOEMPLEADO,PAGOS,ALMACEDADOR,ATENCIONCLIENTE,CONTADOR,MARKETING,VENDEDOR,SOLICITUDES,FINANZAS,PAGINAWEB

```
CREATE SCHEMA EMPLEADO
GO
```

--Contendra : ALMACEN,PELUCHES

```
CREATE SCHEMA ALMACEN
GO
```

--Contendra: CLIENTE,DOICILIOCLIENTE,SUSCRIPCION

```
CREATE SCHEMA CLIENTE
GO
```

--Contendra : VENTA,DIRECCIONVENTA,VENTAONLINE,VENTATELEFONO

```
CREATE SCHEMA VENTA
GO



--Creacion de tablas


--Tabla Empleado
--Tendremos idEmpleado, tipoEmpleado,nombre,telefono,estudio,tiempoEmpleado,asegurado,tarea,empleadoMes,puesto
CREATE TABLE EMPLEADO.Empleado(
	idEmpleado int not null,
	tipoEmpleado varchar(20) NOT NULL,
	nombre varchar(20) NOT NULL,
	telefono integer NOT NULL,
	estudios varchar(20) NOT NULL,
	tiempoEmpleado numeric(15,0) NOT NULL,
	asegurado BIT NOT NULL,
	tareas varchar(10) NOT NULL,
	empleadoMes varchar(10) NOT NULL,
	puesto varchar(15) NOT NULL,
	edad numeric(10,0) NOT NULL,
	CONSTRAINT pk_idEmpleado PRIMARY KEY CLUSTERED (idEmpleado)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Empleado') IS NOT NULL
```

```
        PRINT '<<< CREATED TABLE Empleado >>>'
ELSE
        PRINT '<<< FAILED CREATING TABLE Empleado >>>'
go



--Tabla domicilio
--Tendremos idDomicilio,pais,municipio,delegacion,numeroCasa,calle,idEmpleado
CREATE TABLE EMPLEADO.Domicilio(
        idDomicilioEmpleado int not null,
        pais varchar(20) NOT NULL,
        municipio varchar(20) NOT NULL,
        delegacion varchar(20) NOT NULL,
        numeroCasa integer NOT NULL,
        calle varchar(40) NOT NULL,
        idEmpleado int not null,
        CONSTRAINT pk_idDomicilio PRIMARY KEY CLUSTERED (idDomicilioEmpleado),
        CONSTRAINT fk_idEmpleadoDomicilio FOREIGN KEY (idEmpleado) REFERENCES
EMPLEADO.Empleado(idEmpleado)
)
go



--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Domicilio') IS NOT NULL
        PRINT '<<< CREATED TABLE Domicilio >>>'
```

```
ELSE
	PRINT '<<< FAILED CREATING TABLE Domicilio >>>'
go

--Tabla pagos
--Tendremos idPagos,sueldo,descuentoPagis,bonos
CREATE TABLE EMPLEADO.Pagos(
	idPagos int not null,
	sueldo money NOT NULL,
	descuentoPagos money NOT NULL,
	bonos money NOT NULL,
	CONSTRAINT pk_idPagos PRIMARY KEY CLUSTERED (idPagos),
	CONSTRAINT fk_idPagos FOREIGN KEY (idPagos) REFERENCES EMPLEADO.Empleado(idEmpleado)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Pagos') IS NOT NULL
	PRINT '<<< CREATED TABLE Pagos >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Pagos >>>'
go

--Tablas Almacenador
--Tendremos idEmpleado,rembolso
```

```
CREATE TABLE EMPLEADO.Almacenador(
	idEmpleado int not null,
	rembolso money NOT NULL,
	CONSTRAINT pk_ideEmpleadoAlmacenador PRIMARY KEY CLUSTERED (idEmpleado),
	CONSTRAINT fk_idEmpleadoAlmacenador FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Empleado(idEmpleado)
)
go



--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Almacenador') IS NOT NULL
	PRINT '<<< CREATED TABLE Almacenador >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Almacenador >>>'
go

--Tabla Atencion Cliente
--idEmpleado,horario
CREATE TABLE EMPLEADO.AtencionCliente(
	idEmpleado int not null,
	horario datetime NOT NULL,
	CONSTRAINT pk_ideEmpleado PRIMARY KEY CLUSTERED (idEmpleado),
	CONSTRAINT fk_idEmpleadoAC FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Empleado(idEmpleado)

)
go
```

```
--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('AtencionCliente') IS NOT NULL
	PRINT '<<< CREATED TABLE AtencionCliente >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE AtencionCliente >>>'
go

--Creacion de tabla Empleado.contador
--tendremo idEmpleado,finanzas,idPagos
CREATE TABLE EMPLEADO.Contador(
	idEmpleado int not null,
	finanzas varchar(40) NOT NULL,
	idPagos int NOT NULL,
	CONSTRAINT pk_ideEmpleadoC PRIMARY KEY CLUSTERED (idEmpleado),
	CONSTRAINT fk_idEmpleadoC FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Empleado(idEmpleado),
	CONSTRAINT fk_idPagosC FOREIGN KEY (idPagos) REFERENCES EMPLEADO.Pagos(idPagos)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Contador') IS NOT NULL
	PRINT '<<< CREATED TABLE Contador >>>'
ELSE
```

```
        PRINT '<<< FAILED CREATING TABLE Contador >>>'
go

--Creacion de tabla Empleado.PaginaWeb
--tendremos idPaginaWeb,Producto,verificacion,estadoPagina,urlPagina
CREATE TABLE EMPLEADO.PaginaWeb(
        idPaginaWeb int not null,
        producto varchar(30) NOT NULL,
        verificacion BIT NOT NULL,
        estadoPagina varchar(20) NOT NULL,
        urlPagina varchar(100) NOT NULL,
        CONSTRAINT idFinanzasPW PRIMARY KEY CLUSTERED (idPaginaWeb)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('PaginaWeb') IS NOT NULL
        PRINT '<<< CREATED TABLE PaginaWeb >>>'
ELSE
        PRINT '<<< FAILED CREATING TABLE PaginaWeb >>>'
go

--Creacion de tabla Empleado.Marketing
--tendremos idEmpleado y idPaginaWeb
CREATE TABLE EMPLEADO.Marketing(
        idEmpleado int not null,
```

```
	idPaginaWeb int NOT NULL,
	CONSTRAINT pk_ideEmpleadoM PRIMARY KEY CLUSTERED (idEmpleado),
	CONSTRAINT fk_idEmpleadoM FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Empleado(idEmpleado),
	CONSTRAINT fk_idPaginaWebM FOREIGN KEY (idPaginaWeb) REFERENCES EMPLEADO.PaginaWeb(idPaginaWeb)
)
go



--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Marketing') IS NOT NULL
	PRINT '<<< CREATED TABLE Marketing >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Marketing >>>'
go

--Creacion de tabla Empleado.Venddor
--tendremos idEmpleado,ventasRealizadas,ventasNoRealizadas
CREATE TABLE EMPLEADO.Vendedor(
	idEmpleado int not null,
	ventasRealizadas integer NOT NULL,
	ventasNoRealizadas integer NOT NULL,
	CONSTRAINT pk_ideEmpleadoV PRIMARY KEY CLUSTERED (idEmpleado),
	CONSTRAINT fk_idEmpleadoV FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Empleado(idEmpleado)
)
go
```

```
--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Vendedor') IS NOT NULL
	PRINT '<<< CREATED TABLE Vendedor >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Vendedor >>>'
go

--Creacion de tabla Empleado.solicitud
--tendremos idSolicitud,fechaProblema,nomreCliente,correo,informe,solucion,telefono,idEmpleado
CREATE TABLE EMPLEADO.Solicitud(
	idSolicitud int identity(1,1),
	fechaProblema date NOT NULL,
	nombreCliente varchar(50) NOT NULL,
	correo varchar(100) NOT NULL,
	informe varchar(100) NOT NULL,
	solucion varchar(150) NOT NULL,
	telefono integer NOT NULL,
	idEmpleado int NOT NULL,

	CONSTRAINT pk_idSolicitudS PRIMARY KEY CLUSTERED (idSolicitud),
	CONSTRAINT fk_idEmpleadoS FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.AtencionCliente(idEmpleado),
)
go
```

```
--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Solicitud') IS NOT NULL
	PRINT '<<< CREATED TABLE Solicitud >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Solicitud >>>'
go

--cracion de tabla empleado.finanzas
--tendremos idFinanzas,balances,descripcionPagos,costos,fechaCorte,idEmpleado
CREATE TABLE EMPLEADO.FINANZAS (
	idFinanzas int not null,
	balances varchar(100) NOT NULL,
	descripcionPagos varchar(100) NOT NULL,
	costos money NOT NULL,
	fechaCorte date NOT NULL,
	idEmpleado int NOT NULL,
	CONSTRAINT idFinanzasF PRIMARY KEY CLUSTERED (idFinanzas),
	CONSTRAINT fk_idEmpleadoF FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Contador(idEmpleado)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('FINANZAS') IS NOT NULL
	PRINT '<<< CREATED TABLE FINANZAS >>>'
```

```
ELSE
	PRINT '<<< FAILED CREATING TABLE FINANZAS >>>'
go

--Creacion de tabla cliente.cliente
--tendremos tarjetaPago,nombre,correo,apellidoMaterno,apellidoPaterno,fechaPago,productoFavorito,edad
CREATE TABLE CLIENTE.Cliente(
	tarjetaPago numeric(16,0) NOT NULL,
	nombre varchar(40) NOT NULL,
	correro varchar(10) NOT NULL,
	apellidoMaterno varchar(20) NOT NULL,
	apellidoPaterno varchar(20) NOT NULL,
	fechaPago date NOT NULL,
	productoFavorito varchar(50) NOT NULL,
	edad integer NOT NULL,
	CONSTRAINT tarjetaPagoCL PRIMARY KEY CLUSTERED (tarjetaPago),
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Cliente') IS NOT NULL
	PRINT '<<< CREATED TABLE Cliente>>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Cliente >>>'
go
```

```
--Creacion de cliente.domicilio
--tendremos idDomicilio,tarjetaPago,pais,municipio,delegacion,numeroCasa,calle
Create table CLIENTE.DomicilioCliente(
	idDomicilioCliente int not null,
	tarjetaPago numeric(16,0) NOT NULL,
	pais varchar(50) NOT NULL,
	municipio varchar(10) NOT NULL,
	delegacion varchar(40) NOT NULL,
	numeroCasa integer NOT NULL,
	calle varchar(50) NOT NULL,
	CONSTRAINT pk_idDomicilioCliente PRIMARY KEY CLUSTERED (idDomicilioCliente),
	CONSTRAINT fk_tarjetaPagoCL FOREIGN KEY (tarjetaPago) REFERENCES CLIENTE.Cliente(tarjetaPago)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('DomicilioCliente') IS NOT NULL
	PRINT '<<< CREATED TABLE DomicilioCliente>>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE DomicilioCliente >>>'
go


--creacion de cliente.suscripcion
```

```
--tendremos
idSuscripcion,historial,tipoSuscripcion,formaPago,estadosub,fechaSolicitud,tarjetaPago

CREATE TABLE CLIENTE.Suscripcion(

	idSuscripcion INT PRIMARY KEY IDENTITY, ---llave artificial y se generará automáticamente con un valor incremental cada vez que se inserte una nueva fila en la tabla.

	historial varchar(50) NOT NULL,

	tipoSuscripcion varchar(20) NOT NULL,

	formaPago varchar(20) NOT NULL,

	estadoSub BIT NOT NULL,

	fechaSolicitud DATE NOT NULL,

	tarjetaPago numeric(16,0) NOT NULL,

	CONSTRAINT fk_tarjetaPagoS FOREIGN KEY (tarjetaPago) REFERENCES CLIENTE.Cliente(tarjetaPago)

)

go



--Verificamos si la creacion de la tabla fue realizada correctamente

IF OBJECT_ID('Suscripcion') IS NOT NULL

	PRINT '<<< CREATED TABLE Suscripcion>>>'

ELSE

	PRINT '<<< FAILED CREATING TABLE Suscripcion >>>'

go
```

```
--Tablas Almacen

--tendremos
idAlmacen,pelucheTemporada,compras,modeloPeluches,fechaIngreso,numPeluches,productoMa
Vendido,productoMenosVnedido,idEmpleado

CREATE TABLE ALMACEN.Almacen(

        idAlmacen int not null,

        pelucheTemporada varchar(40) NOT NULL,

        compras integer NOT NULL,

        modelosPeluches varchar(30) NOT NULL,

        fechaIngreso DATE NOT NULL,

        numPeluches integer NOT NULL,

        productoMaVendido VARCHAR(30) NOT NULL,

        productoMenosVendido varchar(40) NOT NULL,

        idEmpleado int NOT NULL,

        CONSTRAINT pk_idAlmacenA PRIMARY KEY CLUSTERED (idAlmacen),

        CONSTRAINT fk_idEmpleadoA FOREIGN KEY (idEmpleado) REFERENCES
EMPLEADO.Almacenador(idEmpleado)


)

go



--Verificamos si la creacion de la tabla fue realizada correctamente

IF OBJECT_ID('Almacen') IS NOT NULL

        PRINT '<<< CREATED TABLE Almacen >>>'

ELSE

        PRINT '<<< FAILED CREATING TABLE Almacen >>>'

go
```

```
--Creacion de almacen.peluches
--tendremos idPeluches,peluchesExclusivos,numeroColeccion,tipo,fehcaCreacion,idAlmacen
CREATE TABLE ALMACEN.Peluches(
	idPeluches int not null,
	peluchesExclusivos varchar(20) NOT NULL,
	numeroColeccion numeric(10,0) NOT NULL,
	tipo varchar(30) NOT NULL,
	fechaCreacion DATE NOT NULL,
	idAlmacen int NOT NULL,
	CONSTRAINT pk_idPeluchesP PRIMARY KEY CLUSTERED (idPeluches),
	CONSTRAINT fk_idAlmacenP FOREIGN KEY (idAlmacen) REFERENCES ALMACEN.Almacen(idAlmacen)

)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Peluches') IS NOT NULL
	PRINT '<<< CREATED TABLE Peluches >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Peluches >>>'
go

--creacion de venta.direccion
--tendremos idDireccionVenta,pais,municipio,delegacion,numeroCasa,calle
```

```
CREATE TABLE VENTA.DireccionVenta(
	idDireccionVenta int not null,
	pais varchar(50) NOT NULL,
	municipio varchar(30) NOT NULL,
	delegacion varchar(50) NOT NULL,
	numeroCasa integer NOT NULL,
	calle varchar(50) NOT NULL,
	CONSTRAINT pk_idDireccionVentaD PRIMARY KEY CLUSTERED (idDireccionVenta)
)
go



--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('DireccionVenta') IS NOT NULL
	PRINT '<<< CREATED TABLE DireccionVenta >>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE DireccionVenta >>>'
go


--creacion de venta.venta
--tendremos idVenta,tipoVeta,fechaSalidaProducto,horario,cancelacion,fechaPedido,nombreProducto,tarjetaPago,idAlmacen,idDireccionVneta
CREATE TABLE VENTA.Venta(
	idVenta int not null,
	tipoVenta varchar(100) NOT NULL,
	fechaSalidaProducto date NOT NULL,
	horario datetime NOT NULL,
```

```
	cancelacion bit NOT NULL,
	codigoProducto varchar(10) NOT NULL,
	fechaPedido date NOT NULL,
	nombreProducto varchar(30) NOT NULL,
	tarjetaPago numeric(16,0) NOT NULL,
	idAlmacen int NOT NULL,
	idDireccionVenta int NOT NULL,
	CONSTRAINT pk_idVentaVE PRIMARY KEY CLUSTERED (idVenta),
	CONSTRAINT fk_tarjetaPagoVE FOREIGN KEY (tarjetaPago) REFERENCES CLIENTE.Cliente(tarjetaPago),
	CONSTRAINT fk_idAlmacenVE FOREIGN KEY (idAlmacen) REFERENCES ALMACEN.Almacen(idAlmacen),
	CONSTRAINT fk_idDireccionVentaVE FOREIGN KEY (idDireccionVenta) REFERENCES VENTA.DireccionVenta(idDireccionVenta)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Venta') IS NOT NULL
	PRINT '<<< CREATED TABLE Venta>>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Venta >>>'
go


--creacion de atiende
--tendra tarjetaPago,idEmpleado,idVenta
CREATE TABLE Atiende (
```

```
	tarjetaPago numeric(16,0) NOT NULL,
	idEmpleado int NOT NULL,
	idVenta int NOT NULL,
	CONSTRAINT pk_tarjetaPago_idEmpleadoAT PRIMARY KEY CLUSTERED (tarjetaPago,idEmpleado),
	CONSTRAINT fk_tarjetaPagoAT FOREIGN KEY (tarjetaPago) REFERENCES CLIENTE.Cliente(tarjetaPago),
	CONSTRAINT fk_idEmpleadoAT FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Vendedor(idEmpleado),
	CONSTRAINT fk_idVentaAT FOREIGN KEY (idVenta) REFERENCES VENTA.Venta(idVenta)
)
go

--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('Atiende') IS NOT NULL
	PRINT '<<< CREATED TABLE Atiende>>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE Atiende >>>'
go

--creacion de ventaOnline
--tendremos idVenta,chat
CREATE TABLE VENTA.VentaOnline (
	idVenta int not null,
	chat varchar(50) NOT NULL,
	CONSTRAINT pk_idVentaO PRIMARY KEY CLUSTERED (idVenta),
	CONSTRAINT fk_idVentaO FOREIGN KEY (idVenta) REFERENCES VENTA.Venta(idVenta)
```

```
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('VentaOnline') IS NOT NULL
	PRINT '<<< CREATED TABLE VentaOnline>>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE VentaOnline >>>'
go

--creacion de venta.ventatelefono
--tendremos idVenta, linea
CREATE TABLE VENTA.VentaTelefono (
	idVenta int not null,
	linea varchar(50) NOT NULL,
	CONSTRAINT pk_idVentaVEN PRIMARY KEY CLUSTERED (idVenta),
	CONSTRAINT fk_idVentaVEN FOREIGN KEY (idVenta) REFERENCES VENTA.Venta(idVenta)
)
go


--Verificamos si la creacion de la tabla fue realizada correctamente
IF OBJECT_ID('VentaTelefono') IS NOT NULL
	PRINT '<<< CREATED TABLE VentaTelefono>>>'
ELSE
	PRINT '<<< FAILED CREATING TABLE VentaTelefono >>>'
```

```
go

------------------------------------------------------------------------
--Agregaciones para terminar la creacion de las tablas


ALTER TABLE ALMACEN.almacen
ADD correo VARCHAR(70) NOT NULL, pais VARCHAR(30) NOT NULL  ;
go

ALTER TABLE CLIENTE.Cliente ALTER COLUMN correro VARCHAR(30);
GO

ALTER TABLE Venta ALTER COLUMN correro VARCHAR(30);
GO

ALTER TABLE EMPLEADO.PaginaWeb DROP COLUMN urlpagina;
GO

ALTER TABLE [EMPLEADO].[Pagos] DROP COLUMN bonos;
GO

ALTER TABLE EMPLEADO.pagos ADD bono AS (sueldo-300);
go

ALTER TABLE [EMPLEADO].[Empleado]
ADD CONSTRAINT ck_tipoEmpleado check ([tipoEmpleado] in ('V','A','AC','M','C'))
```

```
go

ALTER TABLE [CLIENTE].[Suscripcion]
ADD CONSTRAINT ck_tipoSuscripcion check ([tipoSuscripcion] in ('Plata','Oro','Platino','M','C'))
go

UPDATE [EMPLEADO].[Almacenador]
set  [idEmpleado] = 6
where [idEmpleado] = 2
go

select * from [CLIENTE].[Cliente]
UPDATE [CLIENTE].[Cliente]
set  [SoliSub] = 0
where [tarjetaPago] = 4020086625254625
go


ALTER TABLE [EMPLEADO].[Empleado] DROP COLUMN [puesto];
GO

ALTER TABLE [EMPLEADO].[Empleado] ALTER COLUMN [telefono] numeric(10,0);
GO


ALTER TABLE [EMPLEADO].[Solicitud]
ALTER COLUMN [idSolicitud] INT IDENTITY(,1);
```

```
go


ALTER TABLE [EMPLEADO].[Domicilio] DROP COLUMN [municipio];
GO

ALTER TABLE [EMPLEADO].[Almacenador] ALTER COLUMN [rembolso] money not null;
GO

ALTER TABLE [CLIENTE].[Cliente]
add SoliSub bit null
go

ALTER TABLE [ALMACEN].[Peluches]
add costo money not null
go

ALTER TABLE [EMPLEADO].[Pagos]
ADD CONSTRAINT fk_idEmpleadoPagos FOREIGN KEY (idEmpleado) REFERENCES EMPLEADO.Empleado(idEmpleado)
go


ALTER TABLE [EMPLEADO].[Solicitud] ALTER COLUMN [telefono] numeric(10,0);
GO

ALTER TABLE [VENTA].[DireccionVenta] DROP COLUMN [municipio];
```

```
GO

ALTER TABLE [CLIENTE].[DomicilioCliente] DROP COLUMN [municipio];
GO




------------------------------------------------------------


/*
 * INDEX: Almacen
 */
CREATE UNIQUE INDEX ak_Almacen ON ALMACEN.Almacen(idAlmacen)
go

/*
 * INDEX: Peluches
 */
 CREATE UNIQUE INDEX ak_Peluches ON ALMACEN.Peluches(idPeluches)
go

/*
 * INDEX: Cliente
 */
CREATE UNIQUE INDEX ak_cliente ON CLIENTE.Cliente(tarjetaPago)
go
```

```
/*
 * INDEX: Suscripcion
 */
CREATE UNIQUE INDEX ak_suscripcion ON [CLIENTE].[Suscripcion](idSuscripcion)
go

/*
 * INDEX: Atiende
 */
 CREATE UNIQUE INDEX ak_Atiende ON Atiende([tarjetaPago])
go

/*
 * INDEX: Almacenador
 */
 CREATE UNIQUE INDEX ak_Almacenador ON [EMPLEADO].[Almacenador]([idEmpleado])
go

/*
 * INDEX: AtencionCliente
 */
  CREATE UNIQUE INDEX ak_AtencionCliente ON [EMPLEADO].[AtencionCliente]([idEmpleado])
go

/*
 * INDEX: Contador
```

```
 */
  CREATE UNIQUE INDEX ak_Contador ON [EMPLEADO].[Contador]([idEmpleado])
go

/*
 * INDEX: Domicilio
 */
  CREATE UNIQUE INDEX ak_Domicilio ON [EMPLEADO].[Domicilio]([idDomicilioEmpleado])
go

/*
 * INDEX: Empleado
 */
  CREATE UNIQUE INDEX ak_Empleado ON [EMPLEADO].[Empleado]([idEmpleado])
go

/*
 * INDEX: FINANZAS
 */
  CREATE UNIQUE INDEX ak_FINANZAS ON [EMPLEADO].[FINANZAS]([idFinanzas])
go

/*
 * INDEX: Marketing
 */
  CREATE UNIQUE INDEX ak_Marketing ON [EMPLEADO].[Marketing]([idEmpleado])
go
```

```
/*
 * INDEX: PaginaWeb
 */
  CREATE UNIQUE INDEX ak_PaginaWeb ON [EMPLEADO].[PaginaWeb]([idPaginaWeb])
go

/*
 * INDEX: Pagos
 */
  CREATE UNIQUE INDEX ak_Pagos ON [EMPLEADO].[Pagos]([idPagos])
go

/*
 * INDEX: Solicitud
 */
  CREATE UNIQUE INDEX ak_Solicitud ON [EMPLEADO].[Solicitud]([idSolicitud])
go

/*
 * INDEX: Vendedor
 */
  CREATE UNIQUE INDEX ak_Vendedor ON [EMPLEADO].[Vendedor]([idEmpleado])
go

/*
 * INDEX: DireccionVenta
```

```
 */
  CREATE UNIQUE INDEX ak_DireccionVenta ON [VENTA].[DireccionVenta]([idDireccionVenta])
go

/*
 * INDEX: Venta
 */
  CREATE UNIQUE INDEX ak_Venta ON [VENTA].[Venta]([idVenta])
go

/*
 * INDEX: VentaOnline
 */
  CREATE UNIQUE INDEX ak_VentaOnline ON [VENTA].[VentaOnline]([idVenta])
go

/*
 * INDEX: VentaTelefono
 */
  CREATE UNIQUE INDEX ak_VentaTelefono ON [VENTA].[VentaTelefono]([idVenta])
go

-------------------------------------------------------------------------------------------------

-----       Hace de todo    -----

CREATE LOGIN propietario WITH PASSWORD = 'p123456'
```

```
GO

CREATE USER prop FOR LOGIN propietario
GO

ALTER ROLE db_ddladmin ADD MEMBER prop
GO

ALTER ROLE db_owner ADD MEMBER prop
GO


-----	No crea usuarios	-----


CREATE LOGIN administrador WITH PASSWORD = 'admin1'
GO

CREATE USER admi FOR LOGIN administrador
GO

ALTER ROLE db_ddladmin ADD MEMBER admi
GO

-----	Solo lectura	-----

CREATE LOGIN lector WITH PASSWORD = 'lector123'
```

```
GO

CREATE USER usuario FOR LOGIN lector
GO

ALTER ROLE db_datareader ADD MEMBER usuario
GO



Carga inicial

USE empresaTELEMARKETINGS1
go

/*
 *
 * Company :      Equipo de Bases de Datos
 * Project :      Proyecto.DM1
 * Author :       Rivera Morales Alan Adrian
 *
 * Date Created : 30/12/2022
  SD
 */

 /*
 select * from ALMACEN.Almacen
 begin tran
```

```
insert into ALMACEN.Almacen (pelucheTemporada,compras,modelosPeluches,fechaIngreso,numPeluches,productoMaVendido,productoMenosVendido,idEmpleado,correo,pais)
values
('Cariñoso', 9, 'cariñoso', '12-03-2022' ,1,1,0,18,'almacen1@','mexico')
rollback tran
GO
*/

select * from EMPLEADO.Empleado

begin tran
insert into [EMPLEADO].[Empleado] values
(1,'V', 'Aldo', 5534057290, 'Administracion', 5 , 1 , 'Venta', 'No',34 ),
(2,'A', 'Carlos', 5529461048, 'Ventas', 2, 1, 'Almacenar', 'No', 24),
(3,'AC', 'Gabriel', 5529103649, 'Administracion', 1, 0, 'RH', 'Si', 26),
(4,'M', 'Marcos', 5573629327, 'Administracion', 6,1,'Checar WEB', 'No', 36),
(5,'C', 'Sandra', 5537186189, 'Contador', 8,1,'Finanzas','Si',23)

commit tran
--rollback tran
select * from EMPLEADO.Empleado
go

-----------------------------------------------------------------------------------------
begin tran
insert into [EMPLEADO].[Empleado] values
(6,'V', 'Davinia', 5511245976, 'Administracion', 7 , 1 , 'Venta', 'No',23 ),
```

```
(7,'V', 'Dionisio', 556062909, 'Contaduria', 4 , 0 , 'Venta', 'Si',42 ),
(8,'V', 'Sabina', 5572578559, 'Ventas', 2 , 1 , 'Venta', 'No',22 )
commit tran

select * from EMPLEADO.Empleado
go
-----------------------------------------------------

select * from [EMPLEADO].[Domicilio]
begin tran
insert into [EMPLEADO].[Domicilio] values
(1,'Mexico','Azcapotzalco',12,'Calle 3',1),
(2,'Mexico','Iztapalapa',76,'Filipina',3),
(3,'Mexico', 'Tlalpan',45,'Reforma',5)

commit tran
--rollback tran

select * from [EMPLEADO].[Domicilio]
go
-----------------------------------------------------------

select * from [EMPLEADO].[Almacenador]
select * from EMPLEADO.Empleado
begin tran
insert into [EMPLEADO].[Almacenador]  values
(1,100),
```

```
(2,100 ),
(3,100)
commit tran
--rollback tran

 select * from [EMPLEADO].[Almacenador]
 go

 ------------------------------------------------------------

  select * from [EMPLEADO].[AtencionCliente]
 select * from EMPLEADO.Empleado
 begin tran
insert into [EMPLEADO].[AtencionCliente]  values
(3,'2022-01-01 23:59:59.997')
commit tran
--rollback tran
 select * from [EMPLEADO].[AtencionCliente]
go

---------------------------------------------------------------

 select * from [EMPLEADO].[Pagos]
 select * from EMPLEADO.Empleado
 begin tran
insert into [EMPLEADO].[Pagos]  values
(1,20000,100,1),
```

```
(2,30000,500,2),
(3,25000,300,3),
(4,20000,100,4),
(5,10000,100,5)
commit tran
--rollback tran
 select * from [EMPLEADO].[Pagos]
go
```

```
 select * from [EMPLEADO].[Contador]
 select * from EMPLEADO.Empleado
 begin tran
insert into [EMPLEADO].[Contador] values
(1,'Ventas realzadas',1),
(2,'Administracion de almacen',2),
(3,'Clientes atendidos',3),
(4,'Paginas web atendidas',4),
(5,'Finanzas atendidas',5)
commit tran
--rollback tran
 select * from [EMPLEADO].[Contador]
go
```

```
 select * from [EMPLEADO].[FINANZAS]
 select * from EMPLEADO.Empleado
 begin tran
insert into [EMPLEADO].[FINANZAS] values
(1,'Doc1','Pagos sobre el almacen',20000,'2022-06-08',1),
(2,'Doc 2','Pagos sobre atencion a clientes atedidos',10000,'1924-06-09',4),
(3,'Doc3','Pagos realizados a empleados',100000,'1924-06-09',5)
commit tran
--rollback tran
 select * from [EMPLEADO].[FINANZAS]
go

---------------------------------------------------------------

 select * from [EMPLEADO].[Marketing]
 select * from EMPLEADO.Empleado
 begin tran
insert into [EMPLEADO].[Marketing] values
(4,1)
commit tran
--rollback tran
 select * from [EMPLEADO].[Marketing]
go
---------------------------------------------------------------

 select * from [EMPLEADO].[PaginaWeb]
```

```
 select * from EMPLEADO.Empleado
 begin tran
insert into [EMPLEADO].[PaginaWeb] values
(1,'Peluches',1,'Activada'),
(2,'Suscripciones',1,'Activada'),
(3,'Atencion de compras',1,'Desactivada')
commit tran
--rollback tran
 select * from [EMPLEADO].[PaginaWeb]
go
------------------------------------------------------------
select * from [EMPLEADO].[Solicitud]
 select * from EMPLEADO.Empleado
 begin tran
insert into [EMPLEADO].[Solicitud] values
(1,'2022-03-02','Maximo','kodeulaffegei-9185@yopmail.com',
'Rembolso','cancelar',5574330077,3)
commit tran
--rollback tran
 select * from [EMPLEADO].[Solicitud]
go
------------------------------------------------------------

 select * from [VENTA].[DireccionVenta]
 select * from EMPLEADO.Empleado
 begin tran
insert into [VENTA].[DireccionVenta] values
```

(1,'Spain','A Jiménez',37848,'Travesía Blasco'),

(2,'South Korea','안산시',056,'역삼로'),

(3,'Italia','Primo ligure',96 ,'Strada Sala, Appartamento 57')

commit tran

--rollback tran

select * from [VENTA].[DireccionVenta]

go

---

select * from [CLIENTE].[Cliente]

begin tran

insert into [CLIENTE].[Cliente] values

(4011664960121897,'Patiño','qualesajoize-4912@yopmail.com','Sala','Muñiz','2022-12-25','Peluche de temporada 20',24),

(4020086625254625,'Arsenio','ceiceiyoubra-7272@yopmail.com','Carbonell','Piedad','2022-11-01','Peluche de temporada 4',37),

(4017575344866482,'Salvador','jiwegommuffu-8118@yopmail.com','Frutos','Mira','2022-11-25','Peluche de temporada 2',21)

commit tran

--rollback tran

select * from [CLIENTE].[Cliente]

go

---

select * from [CLIENTE].[DomicilioCliente]

select * from [CLIENTE].[Cliente]

begin tran

```
insert into [CLIENTE].[DomicilioCliente] values
(1,4011664960121897,'USA','North Tommouth',76312, 'Romaguera Landing'),
(2,4020086625254625,'USA','Terrellburgh',7570, 'Onie Divide'),
(3,4017575344866482,'Peru','Don Fernando Quezada',8,'Cl. Anthony Sosa' )
commit tran
--rollback tran
 select * from [CLIENTE].[DomicilioCliente]
go

---------------------------------------------------------------------

 select * from [CLIENTE].[Suscripcion]
 select * from [CLIENTE].[Cliente]
 begin tran
insert into [CLIENTE].[Suscripcion] values
('3 meses','Plata','Playpal',1,'2022-11-08',4011664960121897),
('1 año','Oro','Pago en efectivo',1,'2022-12-15',4020086625254625),
('2 meses','Platino','Pago en credito',0,'2022-11-21',4017575344866482)
commit tran
--rollback tran
 select * from [CLIENTE].[Suscripcion]
go

-------------------------------------------------------------------------

 select * from [CLIENTE].[Cliente]
 select * from [EMPLEADO].[Vendedor]
 select * from [VENTA].[Venta]
```

```
 select * from [dbo].[Atiende]
 begin tran
insert into [dbo].[Atiende] values
(4011664960121897,1,1),
(4017575344866482,6,2),
(4020086625254625,8,3)
commit tran
--rollback tran
 select * from [dbo].[Atiende]
go


--------------------------------------------------------------------------------
 select * from [CLIENTE].[Cliente]
 select * from [EMPLEADO].[Almacenador]
 select * from [VENTA].[Venta]
 begin tran
insert into [ALMACEN].[Almacen] values
(1,'Temporada 10',291728,'Peluches Cariñoso','2022-12-12',4,'Si','No',1,'mexusumeimme-2259@yopmail.com','Mexico'),
(2,'Temporada 20',214914,'Peluche de animalitos','2022-11-07',9,'Si','No',2,'traubozuzauzo-9443@yopmail.com','EU'),
(3,'Temporada 3',193,'Peluches oro','2022-11-25',3,'No','Si',3,'feubanneigemei-6458@yopmail.com','Mexico')
commit tran
--rollback tran
 select * from [ALMACEN].[Almacen]
go
```

```
commit tran


------------------------------------------------------------------------------
 select * from [ALMACEN].[Almacen]
 begin tran
insert into [ALMACEN].[Peluches] values
(1,'Si',2347923498,'Peluche De princesas','2022-11-05',1,100),
(2,'No',9914282184,'Peluches de Ositos','2022-11-17',2,200),
(3,'Si',0241827487,'Peluches de Mascotas','2022-11-19',3,500)
commit tran
--rollback tran
 select * from [ALMACEN].[Peluches]
go


------------------------------------------------------------------------
 select * from [CLIENTE].[Cliente]
 select * from [ALMACEN].[Almacen]
 select * from [VENTA].[DireccionVenta]
 select * from [VENTA].[Venta]
 begin tran
insert into [VENTA].[Venta] values
(1,'Venta de peluches','2022-11-29','01:23:02',0,0928352835,'2022-11-24','Osito cariñoso',4011664960121897,1,1),
(2,'Venta de peluches','2022-12-02','11:22:22',1,2187427428,'2022-11-17','Osito Rinoceronte',4017575344866482,2,2),
(3,'Venta de peluches','2022-11-22','12:34:23',1,8914272148,'2022-12-15','Osito Carro',4020086625254625,3,3)
commit tran
```

```
--rollback tran
 select * from [VENTA].[Venta]
go
-------------------------------------------------------------------------
 select * from [VENTA].[Venta]
 begin tran
insert into [VENTA].[VentaOnline] values
(1,'Chat con ip 565674541991981'),
(2,'Chat con ip 545462427272527'),
(3,'Chat con ip 865645456766867')
commit tran
--rollback tran
 select * from [VENTA].[VentaOnline]
go

-------------------------------------------------------------------------
 select * from [VENTA].[Venta]
 begin tran
insert into [VENTA].[VentaTelefono] values
(1,'Linea telefonica 526522'),
(2,'Linea telefonica 252652'),
(3,'Linea telefonica 145145')
commit tran
--rollback tran
 select * from [VENTA].[VentaTelefono]
go
-----------------------------------------------------------------------------
```

```
 select * from [EMPLEADO].[Vendedor]
 begin tran
insert into [EMPLEADO].[Vendedor] values
(1,12,2),
(6,26,12),
(7,17,1),
(8,12,2)
commit tran
--rollback tran
 select * from [EMPLEADO].[Vendedor]
go
```

```
select * from [ALMACEN].[Almacen]

select * from [ALMACEN].[Peluches]

select * from [CLIENTE].[Cliente]

select * from [CLIENTE].[DomicilioCliente]

select * from [CLIENTE].[Suscripcion]

select * from [dbo].[Atiende]

select * from [EMPLEADO].[Almacenador]
```

```
select * from [EMPLEADO].[AtencionCliente]

select * from [EMPLEADO].[Contador]

select * from [EMPLEADO].[Domicilio]

select * from [EMPLEADO].[Empleado]

select * from [EMPLEADO].[FINANZAS]

select * from [EMPLEADO].[Marketing]

select * from [EMPLEADO].[PaginaWeb]

select * from [EMPLEADO].[Pagos]

select * from [EMPLEADO].[Solicitud]

select * from [EMPLEADO].[Vendedor]

select * from [VENTA].[DireccionVenta]

select * from [VENTA].[Venta]

select * from [VENTA].[VentaOnline]
```

```
select * from [VENTA].[VentaTelefono]
```

```
use empresaTELEMARKETING

/*
 *
 * Company :      Equipo de Bases de Datos
 * Project :      Proyecto.DM1
 * Author :       Rivera Morales Alan Adrian
 *
 * Date Created : 30/12/2022
  SD
 */
---1---

--cracion de Procedure sobre cliente,peluche, destino, fecha salida, fecha llegada, numero de la coleccion y el total

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE SP_GENERAR_VENTA
        @p_idCliente INT
AS
BEGIN
        SET NOCOUNT ON;
```

```
	SELECT CONCAT(p.nombre,' ',p.apellidoPaterno,' ',ISNULL(p.apellidoMaterno, ' ')) AS Nombre,
			a.[numeroColeccion] AS Peluche,
			d.[pais] AS Destino,
			h.[fechaSalidaProducto] AS 'Fecha de Salida',
			h.[horario] AS 'Hora de Salida',
			h.[fechaPedido] AS 'Fecha de Llegada',
			a.[numeroColeccion] AS 'Numero de Coleccion',
			a.[costo] AS total

	FROM VENTA.Venta r
	LEFT JOIN CLIENTE.Cliente c ON c.tarjetaPago = r.tarjetaPago
	INNER JOIN CLIENTE.Cliente p ON p.tarjetaPago = c.tarjetaPago
	INNER JOIN VENTA.Venta v ON v.tarjetaPago = r.tarjetaPago
	INNER JOIN [VENTA].[DireccionVenta]d ON d.[idDireccionVenta] = v.[idDireccionVenta]
	INNER JOIN [VENTA].[Venta] h ON h.[idAlmacen]= v.[idAlmacen]
	INNER JOIN [ALMACEN].[Peluches] a ON a.[idAlmacen] = h.[idAlmacen]
	WHERE r.[idVenta] = @p_idCliente
END
GO

/*
select * from [VENTA].[Venta]
EXEC SP_GENERAR_VENTA   @p_idCliente = 2;
-----------------------------------------------------------------------------------------------
go
```

```
*/


--Creacion de procedure que muestra el destino de un producto que vendimos buscando el destino
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE SP_BUSQUEDA_DESTINIO
		@p_destino VARCHAR(50)
AS
BEGIN
	SET NOCOUNT ON;

	SELECT h.[pais] AS 'Destino',
		a.[modelosPeluches] AS 'Tipo de Peluches',
		v.[fechaSalidaProducto] AS 'Fecha de Salida',
		v.[horario] AS 'Hora de Salida',
		v.[fechaPedido] AS 'Fecha de Llegada'

		FROM [VENTA].[Venta] v
		RIGHT JOIN [ALMACEN].[Almacen] a ON v.[idAlmacen] = a.[idAlmacen]
		INNER JOIN [VENTA].[DireccionVenta] h ON v.[idDireccionVenta] = h.[idDireccionVenta]
		WHERE h.[pais] = @p_destino
```

```
END
GO

--EXEC SP_BUSQUEDA_DESTINIO  @p_destino = 'Spain';

------------------------------------------------------------------------------

---3---

--creacion de trigger que verifica si se realizo la cancelacion de un producto y manda el mensaje de ello
CREATE TRIGGER cancellation_check
ON venta.venta
AFTER INSERT
AS
BEGIN
   -- trigger body
   SET NOCOUNT ON;

   -- verificar si se ha ingresado un 1 en la columna "cancelación"
   IF EXISTS (SELECT 1 FROM inserted WHERE [cancelacion] = 1)
   BEGIN
      -- enviar mensaje de alerta
      PRINT 'Producto cancelado, Realizar una solicitud de cancelacion Por favor'
   END

         else
```

```
	begin
		Print 'Producto correctamente'
	end
END

go

/*
select * from [VENTA].[Venta]
select * from [CLIENTE].[Suscripcion]

begin tran

insert into venta.venta values
(6,'Venta de peluches', '2022-12-02', '2022-12-02 09:02:14.000',1,842231545,'2022-11-21','Peluche de Princesa',4011664960121897,1,1)

rollback tran
GO
*/
-----------------------------------------------------------------------

/*
 *
 * Company :      Equipo de Bases de Datos
 * Project :      Proyecto.DM1
 * Author :       Román Cruz Hugo
```

```
*
* Date Created : 1/1/2023
 SD
*/

 -- Obtener los tipos de los productos de la tienda.

    SELECT tipo FROM ALMACEN.PELUCHES


--2.  Obtener los productos mas vendidos y los menos vendidos de los productos de la tienda.

   SELECT [pelucheTemporada],[productoMaVendido], [productoMenosVendido] FROM [ALMACEN].[Almacen]


--3.  Obtener el nombre de los empleados cuyo sueldo sea mayor a 10000

    SELECT sueldo FROM EMPLEADO.PAGOS WHERE sueldo > 10000


--4.  Listar todos los clientes que tengan apellido paterno empiece con C

   SELECT * FROM [CLIENTE].[Cliente] WHERE apellidoPaterno LIKE 'M%'
```

--5. Obtener todos los datos de los datos de las compras que están entre los $600 y $1500 (dos formas para resolver)

```
SELECT * FROM [ALMACEN].[Almacen] WHERE compras >600 AND compras < 300000
```

```
SELECT * FROM [ALMACEN].[Almacen] WHERE compras BETWEEN 600 AND 300000
```

```
/*
 *
 * Company :      Equipo de Bases de Datos
 * Project :      Proyecto.DM1
 * Author :       Hernandez Solis Brandon
 *
 * Date Created : 05/01/2023
  SD
 */

USE empresaTELEMARKETINGS1

SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO


CREATE OR ALTER PROCEDURE SP_INSERTAR_CLIENTE

        ----    Cliente         ----
        @pTarjetaPago           NUMERIC(16,0),
        @pNombre                VARCHAR(40),
        @pParterno              VARCHAR(20),
        @pMaterno               VARCHAR(20),
        @pCorreo                VARCHAR(30),  -- NULL
        @pFechaPago             DATE,
        @pProductoFavorito      VARCHAR(50),
        @pEdad                  INT,
        ----    Domicilio       ----
        @pPais                  VARCHAR(50),
        @pDelegacion            VARCHAR(40),
        @pNumeroCasa            INT,
        @pCalle                 VARCHAR(50)

AS
BEGIN
        SET NOCOUNT ON;

        BEGIN TRAN
        INSERT INTO CLIENTE.Cliente VALUES(
```

```
            @pTarjetaPago,
            @pNombre,
            ISNULL(@pCorreo, ' '),
            @pMaterno,
            @pParterno,
            @pFechaPago,
            @pProductoFavorito,
            @pEdad
        )

        --      ID del domicilio
        DECLARE @pIDDomicilio INT;
        SET @pIDDomicilio = (SELECT MAX(idDomicilioCliente) FROM CLIENTE.DomicilioCliente);

        INSERT INTO CLIENTE.DomicilioCliente VALUES(
            @pIDDomicilio,
            @pTarjetaPago,
            @pPais,
            @pDelegacion,
            @pNumeroCasa,
            @pCalle
        )

        COMMIT TRAN
END
GO
```

```
CREATE OR ALTER PROCEDURE SP_INSERTAR_SOLICITUD

	@pFechaProblema		DATE,
	@pNombreCliente		VARCHAR(50),
	@pCorreo		VARCHAR(100),
	@pInforme		VARCHAR(100),
	@pSolucion		VARCHAR(150),
	@pTelefono		NUMERIC(10,0),
	@pIDEmpleado	INT

AS
BEGIN
	SET NOCOUNT ON;

	--	ID de la solicitud
	DECLARE @pIDSolicitud	INT;
	SET @pIDSolicitud = (SELECT MAX(idSolicitud) FROM EMPLEADO.Solicitud);

	BEGIN TRAN

	INSERT INTO EMPLEADO.Solicitud VALUES(
		@pIDSolicitud,
		@pFechaProblema,
		@pNombreCliente,
		@pCorreo,
		@pInforme,
```

```
            @pSolucion,
            ISNULL(@pTelefono, ' '),
            @pIDEmpleado
        )

        COMMIT TRAN
END
GO

CREATE OR ALTER PROCEDURE SP_INSERTAR_WEB

        @pProducto              VARCHAR(30),
        @pVerificacion          BIT,
        @pEstadoPagina          VARCHAR(20)

AS
BEGIN
        SET NOCOUNT ON;

        DECLARE @pIDPaginaWeb       INT;
        SET @pIDPaginaWeb = (SELECT MAX(idPaginaWeb) FROM EMPLEADO.PaginaWeb);

        BEGIN TRAN

        INSERT INTO EMPLEADO.PaginaWeb VALUES(
            @pIDPaginaWeb,
            @pProducto,
```

```
            @pVerificacion,
            @pEstadoPagina
        )

        COMMIT TRAN
END
GO

CREATE OR ALTER PROCEDURE SP_INSERTAR_Pagos

        @pSueldo                MONEY,
        @pDescuento             MONEY,
        @pIDEmpleado        INT

AS
BEGIN
        SET NOCOUNT ON;

        DECLARE @pIDPagos    INT;
        SET @pIDPagos = (SELECT MAX(idPagos) FROM EMPLEADO.Pagos);

        BEGIN TRAN

        INSERT INTO EMPLEADO.Pagos (idPagos,sueldo,descuentoPagos,idEmpleado) VALUES
        (@pIDPagos, @pSueldo, @pDescuento, @pIDEmpleado)

        COMMIT TRAN
```

```
END
GO

CREATE OR ALTER PROCEDURE SP_INSERTAR_Suscripcion

	@pHistorial			VARCHAR(50),
	@pTipoSuscripcion	VARCHAR(20),
	@pFormaPago			VARCHAR(20),
	@pEstadoSub			BIT,
	@pFechaSolicitud	DATE,
	@pTarjetaPago		NUMERIC(16,0)

AS
BEGIN
	SET NOCOUNT ON;

	DECLARE	@pIDSuscripcion		INT;
	SET @pIDSuscripcion = (SELECT MAX(idSuscripcion) FROM CLIENTE.Suscripcion);

	BEGIN TRAN

	INSERT INTO CLIENTE.Suscripcion VALUES(
	@pIDSuscripcion,
	@pHistorial,
	@pTipoSuscripcion,
	@pFormaPago,
	@pEstadoSub,
```

```
        @pFechaSolicitud,
        @pTarjetaPago
        )

        COMMIT TRAN
END
GO

CREATE OR ALTER PROCEDURE SP_MODIFICAR_CLIENTE_DOMICILIO

        @pIDTarjetaPago             NUMERIC(16,0),
        @pPais                      VARCHAR(50),
        @pDelegacion        VARCHAR(40),
        @pNumeroCasa                INT,
        @pCalle                             VARCHAR(50)

AS
BEGIN
        SET NOCOUNT ON;

        BEGIN TRAN

        UPDATE CLIENTE.DomicilioCliente
                SET
                pais = @pPais,
                delegacion = @pDelegacion,
                numeroCasa = @pNumeroCasa,
```

```
            calle = @pCalle

            WHERE tarjetaPago = @pIDTarjetaPago

        COMMIT TRAN
END
GO

CREATE OR ALTER PROCEDURE SP_MODIFICAR_EMPLEADO_DOMICILIO

        @pIDEmpleado        INT,
        @pPais                  VARCHAR(20),
        @pDelegacion        VARCHAR(20),
        @pNumeroCasa            INT,
        @pCalle                     VARCHAR(40)

AS
BEGIN
        SET NOCOUNT ON;

        BEGIN TRAN

        UPDATE EMPLEADO.Domicilio
            SET
            pais = @pPais,
            delegacion = @pDelegacion,
            numeroCasa = @pNumeroCasa,
```

```
                    calle = @pCalle

                    WHERE idEmpleado = @pIDEmpleado

          COMMIT TRAN
END



/*
 *
 * Company :      Equipo de Bases de Datos
 * Project :      Proyecto.DM1
 * Author :       Rivera Morales Alan Adrian
 *
 * Date Created : 9/1/2023
  SD
 */

--Este código crea una tabla artificial llamada clientes_nuevos que almacena todos los datos de la tabla clientes.
SELECT *
INTO clientes_nuevos
FROM [CLIENTE].[Cliente];
go

select * from clientes_nuevos
go
```

```
--Añadimos dos columnas mas a la tabla artificial clientes_nuevos

alter table clientes_nuevos

add telefono numeric(10,0) null,

hablar_con_cliente bit null

go


-- y agregamos valores nuevos a la tabla clientes_nuevos

UPDATE clientes_nuevos

set  telefono = 5539564295, hablar_con_cliente =1

where [tarjetaPago] = 4020086625254625

go


---------------------------------------------------------------------------


--Este código crea un cursor llamado cur_clientes que selecciona todos los clientes que deben ser contactados de la tabla clientes.

--Luego, el cursor se recorre fila por fila y se procesa cada fila imprimiendo un mens

begin tran

DECLARE @nombre VARCHAR(50);

DECLARE @telefono VARCHAR(50);


DECLARE cur_clientes CURSOR FOR

SELECT nombre, telefono FROM clientes_nuevos WHERE hablar_con_cliente = 1;


OPEN cur_clientes;
```

```
FETCH NEXT FROM cur_clientes INTO @nombre, @telefono;

WHILE @@FETCH_STATUS = 0
BEGIN
   -- Procesar la fila actual
   PRINT 'Llamando a: ' + @nombre + ' al número ' + @telefono;

   FETCH NEXT FROM cur_clientes INTO @nombre, @telefono;
END

CLOSE cur_clientes;
DEALLOCATE cur_clientes;
rollback tran

--------------------------------------------------------------------------------
--Este código crea una tabla temporal llamada clientes_temp y luego inserta algunos datos en ella.
--Luego, selecciona todos los datos de la tabla temporal y los muestra.
CREATE TABLE #Ventas_temp (
   id INT identity(1,1) PRIMARY KEY,
   cliente VARCHAR(50),
   producto VARCHAR(50),
   cantidad int,
         fecha date
);

INSERT INTO #Ventas_temp ( cliente, producto, cantidad,fecha)
VALUES ( 'Juan', 'Peluche de coleccion 23',20,'2022-11-25'),
```

```
    ( 'María', 'Peluche de coleccion 19',10,'2022-11-21'),
    ( 'Pedro', 'Peluches de coleccion 2',15,'2022-12-25');

SELECT * FROM #Ventas_temp;
```

```
--Este código crea un cursor llamado cur_ventas que selecciona todas las ventas que se realizaron y muestra el cliente que
--la realizo, el producto que compro y la cantiad que compro con la fecha del dia del pedido
begin tran
DECLARE @id INT;
DECLARE @cliente VARCHAR(50);
DECLARE @producto VARCHAR(50);
DECLARE @cantidad INT;
DECLARE @fecha DATE;

DECLARE cur_ventas CURSOR FOR
SELECT id, cliente, producto, cantidad, fecha FROM #Ventas_temp;

OPEN cur_ventas;

FETCH NEXT FROM cur_ventas INTO @id, @cliente, @producto, @cantidad, @fecha;

WHILE @@FETCH_STATUS = 0
BEGIN
```

```
    -- Procesar la fila actual
    PRINT 'Venta: ' + CAST(@id AS VARCHAR) + ', Cliente: ' + @cliente + ', Producto: ' + @producto +
', Cantidad: ' + CAST(@cantidad AS VARCHAR) + ', Fecha: ' + CAST(@fecha AS VARCHAR);


    FETCH NEXT FROM cur_ventas INTO @id, @cliente, @producto, @cantidad, @fecha;
END


CLOSE cur_ventas;
DEALLOCATE cur_ventas;
rollback tran


-----------------------------------------------------------------
select * from [ALMACEN].[Peluches]


BEGIN TRY
    -- Actualizar el precio de un producto
    UPDATE [ALMACEN].[Peluches]
    SET [costo] = 100
    WHERE [idPeluches] = 1;
END TRY
BEGIN CATCH
    -- Acciones a realizar en caso de error
    PRINT 'Ocurrió un error: ' + ERROR_MESSAGE();
END CATCH


-----------------------------------------------------------------------
```

```
BEGIN TRY
   -- Consulta que puede generar un error
   DELETE FROM [CLIENTE].[Cliente] WHERE [tarjetaPago] = 10;
END TRY
BEGIN CATCH
   -- Acciones a realizar en caso de error
   PRINT 'Ocurrió un error al eliminar el cliente con ID 100.';
END CATCH
```

```
/*
 *
 * Company :      Equipo de Bases de Datos
 * Project :      Proyecto.DM1
 * Author :       Román Cruz Hugo
 *
 * Date Created : 1/1/2023
  SD
 */

 --Creacion de usuarios

-----  Hace de todo  -----

CREATE LOGIN propietario WITH PASSWORD = 'p123456'
GO

CREATE USER prop FOR LOGIN propietario
GO

ALTER ROLE db_ddladmin ADD MEMBER prop
GO

ALTER ROLE db_owner ADD MEMBER prop
GO


-----  No crea usuarios    -----


CREATE LOGIN administrador WITH PASSWORD = 'admin1'
GO

CREATE USER admi FOR LOGIN administrador
GO

ALTER ROLE db_ddladmin ADD MEMBER admi
GO

-----  Solo lectura  -----

CREATE LOGIN lector WITH PASSWORD = 'lector123'
GO

CREATE USER usuario FOR LOGIN lector
GO

ALTER ROLE db_datareader ADD MEMBER usuario
GO
```

Conclusión.

El hecho de que el desarrollo de la practica se diera desde una etapa temprana del semestre ayudó a conceptualizar mejor la base de datos. Un ejemplo es el cómo fueron cambiando los esquemas antes de realizar la base de datos. Los últimos temas nos resultaron la parte más dificil de completar, ocasionado por la falta de tiempo que implicó el paro. Por lo tanto, no pudimos profundizar en eso eso temas y aplicarlos como nos hubiera gustado, nos limitamos unicamente a cumplir los requerimientos de una manera sencilla.

El resto del proyecto, si bien implicaba entender y aplicar de una manera útil los comocimientos que se revisaron a lo largo del semestre; no representaron tareas complicada en cuanto a encontrar una manera eficaz y buena de ser útilizados. Sin embargo, también coincidimos que en esta parte se encuentran las tareas que conllevan un tiempo de realización largo y pesado, como lo fue la carga de información para inicializar la base de datos.

La tarea más complicada fue la de el uso de los disparadores o triggers, ya que en un inicio no eramos capaces de comprender cómo o donde utilizarlos. Una vez supera esa parte; tuvimos una situacion confusa, ya que no sabíamos diferenciar si lo que estabamos utilizando era un proceso o un trigger. Sin embargo se que se cumplió correctamente con el uso de estos.

El resto de tareas finales implicaban más un ejercicio de identificarnos con el rol del usuario para encontrar qué necesidades podría tener este. Un ejemplo fue el encontrar consultas que arrojaran información que el usuario encontrara útil.