



Facultad de
Ingeniería
UNAM



Universidad Nacional Autónoma de México

Proyecto Final

DCL y DML

M.I.A Martha López Pelcastre
Base de Datos

Ramírez García Diego Andrés
Morales Ortega Carlos
García López Erik
Fecha de entrega: 12/06/2023
Semestre 2023-2
Grupo 02

Universidad
Nacional
Autónoma de
México



1. Enunciado del problema

Se debe desarrollar una aplicación móvil para los usuarios de un servicio de guardería y veterinaria para perros y gatos, así como para los empleados que trabajan en los centros de cuidado.

Guardería

El dueño de una o varias mascotas (**usuario común**), puede dejarlas encargadas en alguno de los centros ubicados en diferentes estados del país, con un costo de \$300.00 por día de cuidado.

Para mantener un control sobre el estado de la mascota, a cada una de ellas se les coloca un brazalete, el cual monitorea los signos vitales del animal, como ritmo cardiaco, temperatura, nivel de oxígeno, etc. en tiempo real. Al mismo tiempo, el brazalete almacena información sobre el estatus de su mascota en sus cuidados diarios, ej.: si ya comió, cuanto comió y qué comió, si ya le fue administrado su medicamento (en caso de estar bajo tratamiento), o algún cuidado especial requerido rutinariamente. Esta información se le mostrará al usuario a través de la aplicación móvil.

Cuando el usuario deje en un centro a su(s) mascotas, se les asignará un **personal de ayuda** como cuidador, dependiendo de la especie. Éste personal será el responsable de hasta 5 mascotas de determinada especie por día. Administrará los alimentos, medicinas y cuidados especiales a cada mascota bajo su cuidado. Por esto mismo, este tipo de empleado será quien actualice la información respectiva en el brazalete de las mascotas bajo su cuidado.

De este personal se requerirá su nombre, CURP, edad, teléfonos, domicilio, tipo de mascotas que cuida (perros o gatos), y para modificar la información de las mascotas en su cuidado utilizará la aplicación, ingresando con su nombre de usuario y contraseña.

Para poder colocar el brazalete y dejar en un centro a una mascota, es necesario que el dueño esté registrado previamente. Esto lo puede hacer con algún **administrativo** en las oficinas de la empresa. De cada dueño se solicitará su nombre completo, su CURP, género, un número de teléfono, correo electrónico, domicilio y una tarjeta de crédito (número de tarjeta y vigencia). También se le pedirá que cree un nombre de usuario (único) y contraseña, para acceder a la aplicación móvil. Una vez registrado, el usuario podrá registrar a sus mascotas para que se le coloque un brazalete de monitoreo. Al hacer esto, se le solicitará la siguiente información de la mascota: nombre, especie, raza, edad, sexo y rasgos característicos.

Reportes:

A partir del registro de esta información, interesa poder obtener estadísticas de:

1. El tipo de mascota que más se deja en la guardería para su cuidado.
2. En qué época se deja un mayor número de mascotas en la guardería
3. Listado de las mascotas y su dueño datos generales, número de días en la guardería, datos generales de ambos. EL reporte se obtiene en un periodo de fechas
4. Enfermedades más frecuentes (5), total de animales con dicha enfermedad



Veterinaria

En el mismo centro de cuidado, se encuentra un espacio que funge como clínica veterinaria para las mascotas que usan el servicio de la guardería.

En cada centro se encuentran de 1 a 3 veterinarios responsables tanto de perros y gatos. De ellos se requiere su nombre, curp, cédula, especialidad y domicilio. Cuando se lleva a una mascota con el veterinario de un centro, éste genera reportes sobre la consulta, incluyendo la fecha y hora, el veterinario que realizó la examinación, los detalles de la examinación, así como el diagnóstico y los medicamentos recetados.

Al realizar una consulta médica, al dueño se le generará un recibo por los gastos de examinación (\$150.00 por consulta), así como por el tratamiento inmediato que se le administró a la mascota (ej.: inyecciones únicas, férulas, collar isabelino, anestesia, vendajes etc.). para ello se tiene inventario de medicamentos y costo.

Reportes

A partir del registro de esta información, interesa poder obtener estadísticas de:

1. Gastos de cada mascota
2. Listado con datos generales de las consultas por periodos de tiempo
3. Inventario de medicamentos, con su costo
4. Enfermedades más frecuentes

Ventas

Además del servicio de veterinaria, el centro cuenta con una tienda que vende diversos artículos para mascotas, como son juguetes, premios, alimentos, utensilios varios, etc.

Se pueden realizar compras personalmente con uno de los encargados de la tienda de souvenirs del centro o a través de la aplicación móvil.

Al momento de realizar compras a través de la aplicación, el cliente podrá ver los detalles de cada producto en cada momento. Si tiene intención de comprarlo, podrá añadirlo a una canasta de compras virtual, para posteriormente proceder a realizar una compra.

Cuando el usuario decida confirmar una compra en línea a partir de su canasta, se le cobrará una tarifa de \$100.00 de envío y los productos se le harán llegar a su domicilio en los siguientes 1 a 3 días. El usuario podrá cancelar su compra siempre y cuando los productos no se hayan entregado aún, sin embargo, se le cobrará una tarifa de cancelación de un 20% del costo de la compra.

Cuando se realiza una compra física, puesto a que el encargado es quien se encarga de efectuarlas, éste se lleva una comisión del 15% del total de la venta, además de su sueldo base de \$5000.00 quincenales



A algunos de los productos se les habilitarán ofertas para incentivar al cliente a realizar compras. Las ofertas tienen una fecha de inicio y fecha de fin (no más de 40 días), descripción y tipo de oferta (normal o limitada).

Cada centro cuenta con cierta disponibilidad de productos, pero para el catálogo en línea se usa el del centro principal (oficina regional).

Reportes

A partir del registro de esta información, interesa poder obtener la siguiente información:

1. El centro con mayor número de ventas en un periodo de tiempo. (Separar las ventas en línea de las ventas físicas de cada estado).
2. Los artículos más vendidos y los menos vendidos por categoría.
3. Época en la que más se vende.
4. Los 5 empleados con mayor comisión, este reporte se obtiene de manera mensual

Administración

Uno de los centros de cuidado de cada estado es también la oficina regional, donde además de los servicios de guardería, veterinaria y tienda, se encuentra el personal administrativo, encargado de manejar los productos de cada región, así como registrar usuarios nuevos.

En cada estado se contará con un **gerente**, quien será el encargado de contratar a todo el personal del estado.

Desglose de usuarios:

Usuario común (registrado)

Usuario común que puede realizar compras, o utilizar al servicio de guardería y veterinaria: Puede navegar el catálogo de productos disponibles para cierto estado.

- Buscar productos en el buscador destinado para tal fin.
- Visualiza los detalles de los productos del estado del catálogo consultado.
- Almacena productos en una canasta virtual. Para concluir la venta requerirá iniciar sesión como usuario registrado o la canasta se borrará automáticamente
- Vaciar la canasta de compra.
- Guardar la canasta de compra para concluirla/modificarla/cancelarla posteriormente.
- Efectuar compras en línea (de una canasta). o Cancelar compras (dentro del periodo de envío. Se cobrará una tarifa de cancelación del 15%)
- Registrar a sus mascotas para hacer uso de los servicios de la guardería o Consultar el estado de su(s) mascotas
- Dejar su mascota en la guardería • Acceso al servicio de veterinaria



Personal de ayuda (Cuidadores)

Personal de cuidado en cada centro de guardería. Es quien se hace responsable del bienestar de la mascota durante la estancia en la guardería. Hay hasta 8 cuidadores por centro.

- Hay un personal líder, que se encarga de asignar las mascotas a los otros miembros del personal durante la jornada.
- Sólo puede cuidar hasta 5 mascotas de la misma especie (gatos o perros) a la vez.
- Actualiza la información de cuidado diario de las mascotas en su cuidado para su consulta en la aplicación.

Veterinario

Especialista encargado de examinar, tratar y recetar a las mascotas de los dueños suscritos al servicio de guardería.

- Consultas médicas para las mascotas
- Proveer tratamientos

Encargado de tienda

Encargado de la tienda del centro de cuidado,

- Administración de las ventas personales o Realiza ventas.
- Actualiza el inventario de la clínica donde trabaja.

Personal administrativo

Encargado de la gestión de los productos de la tienda en línea de cada estado. Adicionalmente agrega usuarios comunes.

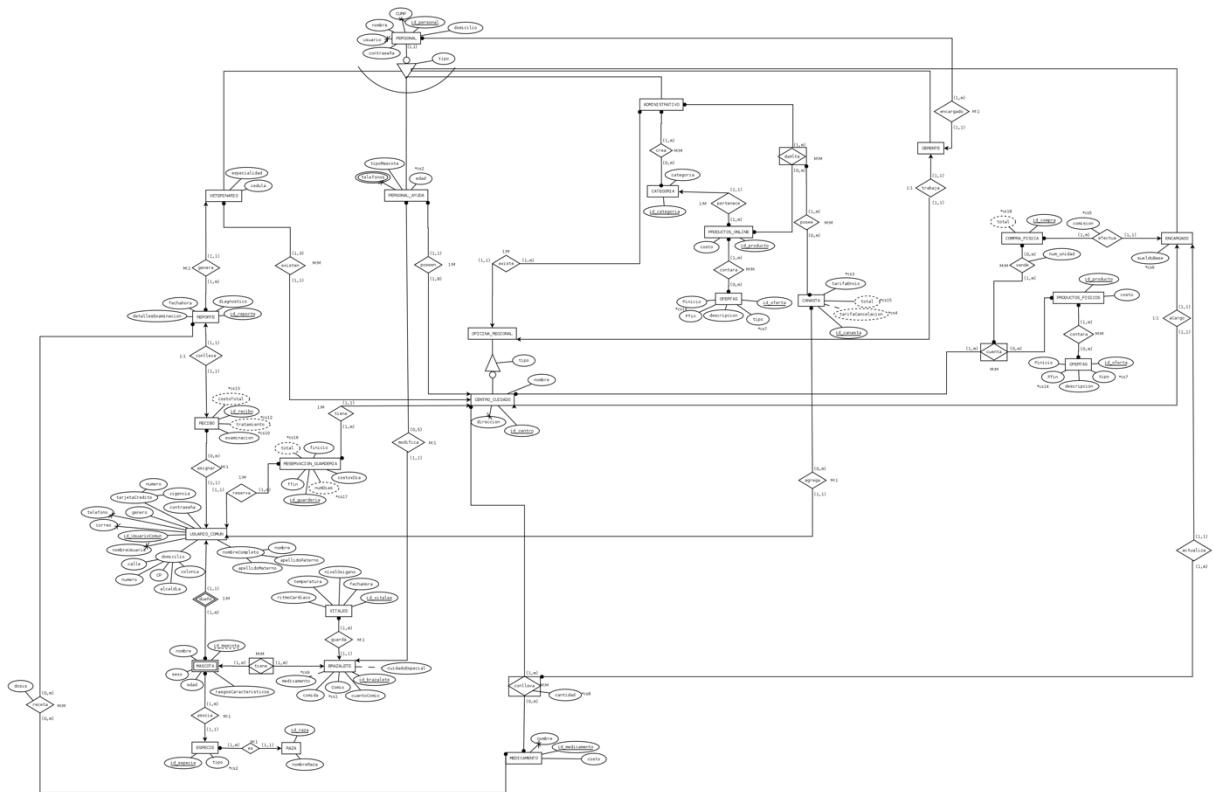
- Registrar usuarios nuevos.
- Gestionar las ofertas de los productos del estado.
- Gestionar las categorías de los productos: dar de alta nuevas categorías, modificarlas y eliminarlas
- Gestionar los productos: Agregar productos, modificarlos y eliminarlos

Gerente

- Administra el personal de cada centro en su respectivo estado.
 - Contratación
 - Agrega usuarios tipo personal de ayuda, veterinario y administrativo.
 - Asigna a un centro de cuidado al personal de ayuda, y veterinario.
- Modificación, reasigna al personal de ayuda y veterinario a otro centro en el mismo estado
- Despedir, elimina usuarios tipo personal de ayuda, veterinario, y administrativo.

2. Diseño Conceptual (MER en DIA)

Guarderia de Mascotas



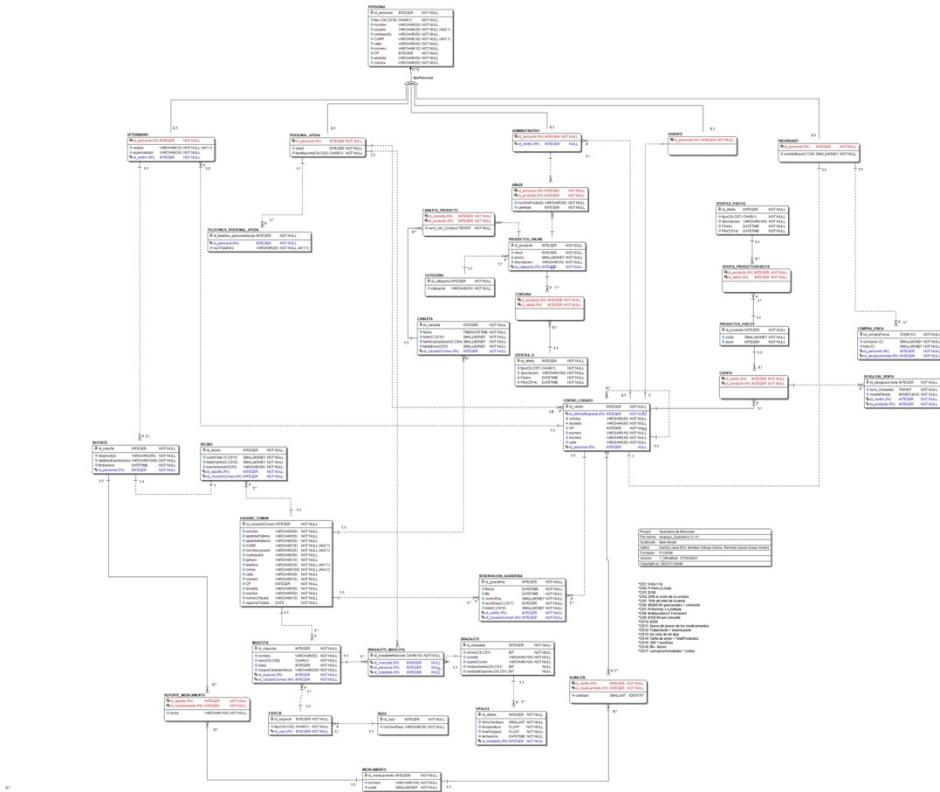
- *nro 3: S-S, No Rte
- *nro 4: Cerrito, G-Cafe
- *nro 5: \$200
- *nro 6: 20% el costo de la compra
- *nro 7: 10% del total de la venta
- *nro 8: 0.05% de la compra + comisión
- *nro 9: N-Normal, L-Comida
- *nro 10: M-Masculino, F-Femenino
- *nro 11: Si(S) Su administrador, no ha sido administrador de otra farmacia
- *nro 12: No(S) No consta
- *nro 13: 300 pesos
- *nro 14: Suministro del precio de los medicamentos
- *nro 15: Tratamiento y examenación
- *nro 16: Tarifa de envío de paquetes
- *nro 17: Tarifa de envío = totalProductos * 300/millones
- *nro 18: Tasa de envío
- *nro 19: Tarifa de envío = 0.001 unidades * costo

(Bases de Datos Grupo: B2
Integrantes
García López Erick
Morales Ortega Carlos
Ramírez García Diego Andrés



3. Diseño Lógico

a) Modelo Relacional (entregar archivo en ERStudio),





b) Diccionario de datos, debe contener:

ADMINISTRATIVO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_personal	int	Identificador único del personal administrativo.
id_centro	int	Identificador del centro al que está asignado el personal administrativo. Puede ser nulo.

ALMACEN

Nombre del atributo	Tipo de dato	Descripción del atributo
id_centro	int	Identificador único del centro de la guardería.
id_medicamento	int	Identificador único del medicamento en el almacén.
cantidad	smallint	Cantidad de medicamentos disponibles en el almacén.

AÑADE

Nombre del atributo	Tipo de dato	Descripción del atributo
id_personal	int	Identificador único del personal que añade un producto.
id_producto	int	Identificador único del producto añadido.
nombreProducto	varchar(50)	Nombre del producto añadido.
cantidad	int	Cantidad del producto añadido.

BRAZALETE

Nombre del atributo	Tipo de dato	Descripción del atributo
id_brazalete	int	Identificador único del brazalete asignado a un animal.
comio	bit	Indica si el animal ha comido o no.
comida	varchar(100)	Nombre o descripción de la comida proporcionada al animal.
cuantoComio	varchar(100)	Cantidad o información adicional sobre la cantidad de comida que consumió el animal.
medicamento	bit	Indica si al animal se le administró algún medicamento o no.
cuidadoEspecial	bit	Indica si el animal requiere cuidados especiales o no.



BRAZALETE_MASCOTA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_brazaleteMascota	char(10)	Identificador único del brazalete asignado a una mascota.
id_mascota	int	Identificador de la mascota asociada al brazalete. Puede ser nulo.
id_personal	int	Identificador del personal asociado al brazalete. Puede ser nulo.
id_brazalete	int	Identificador del brazalete asociado a la mascota. Puede ser nulo.

CANASTA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_canasta	int	Identificador único de la canasta.
fecha	time(7)	Hora en la que se realizó la canasta.
tarifaEnvio	smallmoney	Tarifa de envío de la canasta. Valor predeterminado: 100.
total	smallmoney	Total de la canasta.
tarifaCancelacion	smallmoney	Tarifa de cancelación de la canasta.
id_UsuarioComun	int	Identificador del usuario común asociado a la canasta.

CANASTA_PRODUCTO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_canasta	int	Identificador de la canasta a la que pertenece el producto.
id_producto	int	Identificador del producto en la canasta.
num_Uni_Compra	tinyint	Número de unidades del producto compradas en la canasta.

CATEGORIA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_categoria	int	Identificador único de la categoría.
categoria	varchar(50)	Nombre de la categoría.



BASES DE DATOS UNAM

Grupo 02
Sem. 2023-2

CENTRO_CUIDADO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_centro	int	Identificador único del centro de cuidado.
id_oficinaRegional	int	Identificador de la oficina regional asociada al centro. Puede ser nulo.
colonia	varchar(50)	Nombre de la colonia donde se encuentra el centro.
alcaldia	varchar(50)	Nombre de la alcaldía donde se encuentra el centro.
CP	int	Código postal del área donde se encuentra el centro.
numero	varchar(10)	Número de contacto del centro.
nombre	varchar(30)	Nombre del centro de cuidado.
calle	varchar(50)	Nombre de la calle donde se encuentra el centro.
id_personal	int	Identificador del personal asociado al centro. Puede ser nulo.
id_encargado	int	Identificador del encargado del centro.

COMPRA_FISICA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_compraFisica	int	Identificador único de la compra física.
total	smallmoney	Total de la compra física.
id_personal	int	Identificador del personal asociado a la compra física.
id_desgloseVenta	int	Identificador del desglose de venta asociado a la compra física.
comision	AS (total * 0.15) PERSISTED	Comisión calculada como el 15% del total de la compra física.

CONTARA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_producto	int	Identificador del producto en la relación.
id_oferta	int	Identificador de la oferta en la relación.



CUENTA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_centro	int	Identificador del centro asociado a la cuenta.
id_producto	int	Identificador del producto asociado a la cuenta.

DESGLOSE_VENTA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_desgloseVenta	int	Identificador único del desglose de venta.
num_Unidades	tinyint	Número de unidades en el desglose de venta.
montoParcial	money	Monto parcial del desglose de venta.
id_centro	int	Identificador del centro asociado al desglose de venta.
id_producto	int	Identificador del producto asociado al desglose de venta.

ENCARGADO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_personal	int	Identificador único del encargado.
sueldoBase	smallmoney	Sueldo base del encargado.

ESPECIE

Nombre del atributo	Tipo de dato	Descripción del atributo
id_especie	int	Identificador único de la especie.
tipo	char(1)	Tipo de mascota (perro o gato).
id_raza	int	Identificador de la raza asociada a la especie.

GERENTE

Nombre del atributo	Tipo de dato	Descripción del atributo
id_personal	int	Identificador único del gerente.



MASCOTA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_mascota	int	Identificador único de la mascota.
nombre	varchar(50)	Nombre de la mascota.
sexo	char(1)	Sexo de la mascota ('M' para masculino, 'F' para femenino).
edad	int	Edad de la mascota.
rasgosCaracteristicos	varchar(200)	Rasgos característicos de la mascota.
id_especie	int	Identificador de la especie asociada a la mascota.
id_UsuarioComun	int	Identificador del usuario común asociado a la mascota.

MEDICAMENTO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_medicamento	int	Identificador único del medicamento.
nombre	varchar(100)	Nombre del medicamento.
costo	smallmoney	Costo del medicamento.

OFERTA_PRODUCTOSFISICOS

Nombre del atributo	Tipo de dato	Descripción del atributo
id_producto	int	Identificador del producto asociado a la oferta.
id_oferta	int	Identificador único de la oferta de productos físicos.

OFERTAS_FISICOS

Nombre del atributo	Tipo de dato	Descripción del atributo
id_oferta	int	Identificador único de la oferta.
tipo	char(1)	Tipo de oferta ('N' para normal, 'L' para limitada).
descripcion	varchar(100)	Descripción de la oferta.
Finicio	datetime	Fecha y hora de inicio de la oferta.
Ffin	datetime	Fecha y hora de finalización de la oferta.



OFERTAS_O

Nombre del atributo	Tipo de dato	Descripción del atributo
id_oferta	int	Identificador único de la oferta.
tipo	char(1)	Tipo de oferta ('N' para normal, 'L' para limitada).
descripcion	varchar(100)	Descripción de la oferta.
Finicio	datetime	Fecha y hora de inicio de la oferta.
Ffin	datetime	Fecha y hora de finalización de la oferta.

PERSONAL

Nombre del atributo	Tipo de dato	Descripción del atributo
id_personal	int	Identificador único del personal.
tipo	char(1)	Tipo de personal ('V' para veterinario, 'H' para ayudante, 'A' para administrativo, 'G' para gerente, 'E' para encargado).
nombre	varchar(50)	Nombre del personal.
usuario	varchar(20)	Nombre de usuario del personal.
contrasena	varchar(50)	Contraseña del personal.
CURP	varchar(18)	CURP (Clave Única de Registro de Población) del personal.
calle	varchar(50)	Nombre de la calle de residencia del personal.
numero	varchar(10)	Número de la residencia del personal.
CP	int	Código Postal de la residencia del personal.
alcaldia	varchar(50)	Alcaldía o municipio de la residencia del personal.
colonia	varchar(50)	Colonia o barrio de la residencia del personal.

PERSONAL_AYUDA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_personal	int	Identificador único del personal.
edad	int	Edad del personal.
tipoMascota	char(1)	Tipo de mascota preferida por el personal ('P' para perro, 'G' para gato).
id_centro	int	Identificador del centro de cuidado donde trabaja el personal.



PRODUCTOS_FISICOS

Nombre del atributo	Tipo de dato	Descripción del atributo
id_producto	int	Identificador único del producto físico.
costo	smallmoney	Costo del producto físico.
stock	int	Cantidad disponible en stock del producto físico.

PRODUCTOS_ONLINE

Nombre del atributo	Tipo de dato	Descripción del atributo
id_producto	int	Identificador único del producto en línea.
stock	int	Cantidad disponible en stock del producto en línea.
precio	smallmoney	Precio del producto en línea.
descripcion	varchar(70)	Descripción del producto en línea.
id_categoria	int	Identificador de la categoría a la que pertenece el producto.

RAZA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_raza	int	Identificador único de la raza.
nombreRaza	varchar(30)	Nombre de la raza de la mascota.

RECIBO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_recibo	int	Identificador único del recibo.
costoTotal	smallmoney	Costo total del recibo.
tratamiento	smallmoney	Costo del tratamiento en el recibo.
examinacion	varchar(50)	Descripción de la examinación realizada en el recibo.
id_reporte	int	Identificador del reporte asociado al recibo.
id_UsuarioComun	int	Identificador del usuario común asociado al recibo.



REPORTE

Nombre del atributo	Tipo de dato	Descripción del atributo
id_reporte	int	Identificador único del reporte.
diagnostico	varchar(255)	Diagnóstico asociado al reporte.
detallesExaminacion	varchar(1000)	Detalles de la examinación realizada en el reporte.
fechaHora	datetime	Fecha y hora de creación del reporte.
id_personal	int	Identificador del personal responsable del reporte.

REPORTE_MEDICAMENTO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_reporte	int	Identificador del reporte asociado al medicamento.
id_medicamento	int	Identificador del medicamento asociado al reporte.
dosis	varchar(100)	Dosis del medicamento en el reporte.

RESERVACION_GUARDERIA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_guarderia	int	Identificador único de la reservación de guardería.
finicio	datetime	Fecha y hora de inicio de la reservación.
ffin	datetime	Fecha y hora de fin de la reservación.
costoXDia	smallmoney	Costo por día de la reservación.
numDias	int	Número de días de la reservación (calculado automáticamente).
total	smallmoney	Total a pagar por la reservación (calculado automáticamente).
id_centro	int	Identificador del centro de cuidado asociado a la reservación.
id_UsuarioComun	int	Identificador del usuario común asociado a la reservación.



TELEFONOS_PERSONAL_AYUDA

Nombre del atributo	Tipo de dato	Descripción del atributo
id_telefono_personalAyuda	int	Identificador único del teléfono del personal de ayuda.
id_personal	int	Identificador del personal de ayuda asociado al teléfono.
numTelefono	varchar(20)	Número de teléfono del personal de ayuda.

USUARIO_COMUN

Nombre del atributo	Tipo de dato	Descripción del atributo
id_UsuarioComun	int	Identificador único del usuario común.
nombre	varchar(50)	Nombre del usuario común.
apellidoPaterno	varchar(50)	Apellido paterno del usuario común.
apellidoMaterno	varchar(50)	Apellido materno del usuario común.
CURP	varchar(18)	CURP (Clave Única de Registro de Población) del usuario común.
nombreUsuario	varchar(20)	Nombre de usuario del usuario común.
contrasena	varchar(50)	Contraseña del usuario común.
genero	varchar(10)	Género del usuario común.
telefono	varchar(20)	Número de teléfono del usuario común.
correo	varchar(100)	Correo electrónico del usuario común.
calle	varchar(50)	Nombre de la calle de residencia del usuario común.
numero	varchar(10)	Número de la dirección de residencia del usuario común.
CP	int	Código Postal de la dirección de residencia del usuario común.
alcaldia	varchar(50)	Alcaldía o municipio de la dirección de residencia del usuario común.
colonia	varchar(50)	Colonia o localidad de la dirección de residencia del usuario común.
numeroTarjeta	varchar(16)	Número de tarjeta de crédito/débito del usuario común.
vigenciaTarjeta	date	Fecha de vigencia de la tarjeta de crédito/débito del usuario común.



VETERINARIO

Nombre del atributo	Tipo de dato	Descripción del atributo
id_personal	int	Identificador único del veterinario
cedula	varchar(10)	Cédula del veterinario
especialidad	varchar(50)	Especialidad del veterinario
id_centro	int	Identificador del centro de atención

VITALES

Nombre del atributo	Tipo de dato	Descripción del atributo
id_vitales	int	Identificador único de las mediciones vitales
ritmoCardiaco	smallint	Ritmo cardíaco del paciente
temperatura	float	Temperatura corporal del paciente
nivelOxigeno	float	Nivel de oxígeno en la sangre del paciente
fechaHora	datetime	Fecha y hora de la medición de las variables vitales
id_brazalete	int	Identificador del brazalete utilizado para la medición



4. Normalización hasta la 3era forma normal

La base de datos ha sido normalizada siguiendo un enfoque riguroso que involucra la construcción de un modelo entidad-relación inicial y luego su posterior transformación en un modelo relacional. Este proceso garantiza que la base de datos cumpla con los principios de la normalización hasta la tercera forma normal.

En primer lugar, el modelo entidad-relación nos permitió identificar todas las entidades relevantes en el dominio de la base de datos, así como las relaciones entre ellas. Mediante la identificación de estas entidades y sus atributos, pudimos representar de manera adecuada la estructura lógica de la información que se almacenará en la base de datos.

Una vez establecido el modelo entidad-relación, procedimos a transformarlo en un modelo relacional. Este proceso implicó la conversión de las entidades en tablas y las relaciones en restricciones de clave primaria y clave foránea. Al hacerlo, se logró una representación más precisa y coherente de los datos, al tiempo que se aseguró la integridad referencial.

Además, al aplicar las reglas de la normalización a las tablas resultantes, garantizamos que la base de datos cumpla con los requisitos de la primera, segunda y tercera forma normal. La primera forma normal nos asegura que cada atributo contenga un único valor, evitando la redundancia de datos. La segunda forma normal establece la dependencia funcional completa entre las claves primarias y los atributos no clave, eliminando las dependencias parciales. Finalmente, la tercera forma normal evita las dependencias transitivas, asegurando la eliminación de redundancias adicionales en la base de datos.



5. Diseño Físico

- a) DCL – entregar el script para la generación de usuarios, el cual deberá llamarse seguridad.sql**

```
use GUARDERIA
```

```
go
```

```
--Creando login Consulta
```

```
CREATE LOGIN usuarioConsulta WITH PASSWORD='1234zaq*',  
DEFAULT_DATABASE=GUARDERIA, CHECK_EXPIRATION=OFF,  
CHECK_POLICY=OFF
```

```
GO
```

```
--Creando user Consulta
```

```
create user usuarioConsulta for login usuarioConsulta
```

```
go
```

```
--ROL db_datareader
```

```
ALTER ROLE db_datareader ADD MEMBER usuarioConsulta
```

```
--Creando login Gestor
```

```
CREATE LOGIN usuarioGestor WITH PASSWORD='1234zaq*',  
DEFAULT_DATABASE=GUARDERIA, CHECK_EXPIRATION=OFF,  
CHECK_POLICY=OFF
```

```
GO
```

```
--Creando user Gestor
```

```
create user usuarioGestor for login usuarioGestor
```

```
go
```

```
--ROL db_writer
```

```
ALTER ROLE db_datawriter ADD MEMBER usuarioGestor
```

```
--Creando login Administrador
```

```
CREATE LOGIN usuarioAdministrador WITH PASSWORD='1234zaq*',  
DEFAULT_DATABASE=GUARDERIA, CHECK_EXPIRATION=OFF,  
CHECK_POLICY=OFF
```

```
GO
```

```
--Creando user Administrador
```

```
create user usuarioAdministrador for login usuarioAdministrador
```

```
go
```

```
--Otorgamos permisos para poder crear y eliminar
```

```
--usuarios a usuarioAdministrador a nivel servidor
```

```
ALTER SERVER ROLE securityadmin ADD MEMBER usuarioAdministrador
```

```
GO
```

```
--Creando login Administrador Chiquito
```



```
CREATE LOGIN usuarioAdministradorChiquito WITH PASSWORD='1234zaq*',  
DEFAULT_DATABASE=GUARDERIA, CHECK_EXPIRATION=OFF,  
CHECK_POLICY=OFF
```

```
GO
```

```
--Creando user Administrador
```

```
create user usuarioAdministradorChiquito for login usuarioAdministradorChiquito  
go
```

```
--Otorgamos permisos para poder crear y eliminar
```

```
--usuarios a usuarioAdministrador a nivel servidor
```

```
ALTER ROLE db_securityadmin ADD MEMBER usuarioAdministradorChiquito  
GO
```

```
-- Creando el rol Usuario Común
```

```
CREATE ROLE usuarioComun
```

```
GO
```

```
GRANT SELECT ON PRODUCTOS_ONLINE TO usuarioComun
```

```
GO
```

```
GRANT SELECT ON OFERTAS_O TO usuarioComun
```

```
GO
```

```
GRANT SELECT ON CANASTA TO usuarioComun
```

```
GO
```

```
GRANT UPDATE ON CANASTA TO usuarioComun
```

```
GO
```

```
GRANT INSERT ON MASCOTA TO usuarioComun
```

```
GO
```

```
GRANT SELECT ON MASCOTA TO usuarioComun
```

```
GO
```

```
GRANT SELECT ON VITALES TO usuarioComun
```

```
GO
```

```
GRANT SELECT ON BRAZALETE TO usuarioComun
```

```
GO
```

```
GRANT SELECT ON TELEFONOS_PERSONAL_AYUDA TO usuarioComun
```

```
GO
```

```
--Creando el rol Cuidadores
```

```
CREATE ROLE cuidador
```

```
GO
```

```
GRANT UPDATE ON BRAZALETE TO cuidador
```

```
GO
```

```
GRANT UPDATE ON VITALES TO cuidador
```



GO

--Creando el rol Cuidador Lider

CREATE ROLE cuidadorLider

GO

GRANT UPDATE ON BRAZALETE TO cuidadorLider

GO

GRANT UPDATE ON VITALES TO cuidadorLider

GO

GRANT UPDATE ON PERSONAL_AYUDA TO cuidadorLider

GO

--Creando el rol Veterinario

CREATE ROLE veterinario

GO

GRANT UPDATE ON REPORTE TO veterinario

GO

GRANT UPDATE ON REPORTE_MEDICAMENTO TO veterinario

GO

GRANT SELECT ON MEDICAMENTO TO veterinario

GO

--Creando el rol ENCARGADO TIENDA

CREATE ROLE encargadoTienda

GO

select * from PRODUCTOS_FISICOS

GRANT UPDATE ON PRODUCTOS_FISICOS TO encargadoTienda

GO

GRANT SELECT ON OFERTAS_FISICOS TO veterinario

GO

**b) DDL- el script para crear la base de datos debe llamarse
CreaBase.sql**

```
/*
* Company : FI-UNAM
* Project : Guarderia de Mascotas
* Author : Garcia Lopez Erik, Morales Ortega Carlos, Ramirez Garcia Diego Andres
*
* Date Created : Saturday, May 27, 2023 16:42:37
```



```
* Target DBMS : Microsoft SQL Server 2008  
*/
```

```
/*  
* TABLE: ADMINISTRATIVO  
*/
```

```
create database GUARDERIA  
go
```

```
use GUARDERIA  
go
```

```
CREATE TABLE ADMINISTRATIVO(  
    id_personal int NOT NULL,  
    id_centro int NULL,  
    CONSTRAINT pk_ADMINISTRATIVO PRIMARY KEY CLUSTERED (id_personal)  
)  
go
```

```
IF OBJECT_ID('ADMINISTRATIVO') IS NOT NULL  
    PRINT '<<< CREATED TABLE ADMINISTRATIVO >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE ADMINISTRATIVO >>>'  
go
```

```
/*  
* TABLE: ALMACEN  
*/
```

```
CREATE TABLE ALMACEN(  
    id_centro int NOT NULL,  
    id_medicamento int NOT NULL,  
    cantidad smallint NOT NULL,  
    CONSTRAINT pk_ALMACEN PRIMARY KEY CLUSTERED (id_centro,  
    id_medicamento)  
)  
go
```

```
IF OBJECT_ID('ALMACEN') IS NOT NULL  
    PRINT '<<< CREATED TABLE ALMACEN >>>'
```



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

/*
* TABLE: AÑADE
*/
```

```
CREATE TABLE AÑADE(
    id_personal    int      NOT NULL,
    id_producto   int      NOT NULL,
    nombreProducto varchar(50) NOT NULL,
    cantidad       int      NOT NULL,
    CONSTRAINT pk_añade PRIMARY KEY CLUSTERED (id_personal, id_producto)
)
go
```

```
IF OBJECT_ID('AÑADE') IS NOT NULL
PRINT '<<< CREATED TABLE AÑADE >>>'
ELSE
PRINT '<<< FAILED CREATING TABLE AÑADE >>>'
go

/*
* TABLE: BRAZALETE
*/
```

```
CREATE TABLE BRAZALETE(
    id_brazalete  int      NOT NULL identity(1,1),
    comio         bit      NOT NULL,
    comida        varchar(100) NOT NULL,
    cuantoComio   varchar(100) NOT NULL,
    medicamento   bit      NULL,
    cuidadoEspecial bit     NULL,
    CONSTRAINT pk_brazalete PRIMARY KEY CLUSTERED (id_brazalete),
    CONSTRAINT ck_comio check (comio in (0,1)),
    CONSTRAINT ck_medicamento check(medicamento in (0,1)),
    CONSTRAINT ck_cuidadoEspecial check(cuidadoEspecial in (0,1))
)
go
```



```
IF OBJECT_ID('BRAZALETE') IS NOT NULL
    PRINT '<<< CREATED TABLE BRAZALETE >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE BRAZALETE >>>'
go

/*
* TABLE: BRAZALETE_MASCOTA
*/
CREATE TABLE BRAZALETE_MASCOTA(
    id_brazaleteMascota int      NOT NULL IDENTITY(1,1),
    id_mascota          int      NULL,
    id_personal         int      NULL,
    id_brazalete        int      NULL,
    CONSTRAINT pk_BRAZALETE_MASCOTA PRIMARY KEY CLUSTERED
(id_brazaleteMascota)
)
go
/* Modificación */
ALTER TABLE BRAZALETE_MASCOTA
ADD estatus      CHAR(1)  NULL CONSTRAINT df_estatus DEFAULT 'A'
ALTER TABLE BRAZALETE_MASCOTA
ADD CONSTRAINT ck_estatus CHECK (estatus IN ('A','T'))
-----
```

```
IF OBJECT_ID('BRAZALETE_MASCOTA') IS NOT NULL
    PRINT '<<< CREATED TABLE BRAZALETE_MASCOTA >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE BRAZALETE_MASCOTA >>>'
go

/*
* TABLE: CANASTA
*/
CREATE TABLE CANASTA(
    id_canasta          int      NOT NULL identity(1,1),
    fecha                time(7)  NOT NULL,
    tarifaEnvio          smallmoney NOT NULL CONSTRAINT
df_tarifaEnvio DEFAULT (100),
    total                smallmoney NOT NULL ,
    tarifaCancelacion   smallmoney NOT NULL,
    id_UsuarioComun     int      NOT NULL,
    CONSTRAINT pk_CANASTA PRIMARY KEY CLUSTERED (id_canasta),
```



```
CONSTRAINT ck_tarifaCancelacion CHECK (tarifaCancelacion = total * 0.15),  
)  
go
```

```
IF OBJECT_ID('CANASTA') IS NOT NULL  
    PRINT '<<< CREATED TABLE CANASTA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE CANASTA >>>'  
go
```

```
/*  
* TABLE: CANASTA_PRODUCTO  
*/
```

```
CREATE TABLE CANASTA_PRODUCTO(  
    id_canasta      int      NOT NULL,  
    id_producto     int      NOT NULL,  
    num_Uni_Compra tinyint  NOT NULL,  
    CONSTRAINT pk_CANASTA_PRODUCTO PRIMARY KEY CLUSTERED (id_canasta,  
    id_producto)  
)  
go
```

```
IF OBJECT_ID('CANASTA_PRODUCTO') IS NOT NULL  
    PRINT '<<< CREATED TABLE CANASTA_PRODUCTO >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE CANASTA_PRODUCTO >>>'  
go
```

```
/*  
* TABLE: CATEGORIA  
*/
```

```
CREATE TABLE CATEGORIA(  
    id_categoria    int      NOT NULL identity(1,1),  
    categoria       varchar(50) NOT NULL,  
    CONSTRAINT pk_CATEGORIA PRIMARY KEY CLUSTERED (id_categoria)  
)  
go
```



```
IF OBJECT_ID('CATEGORIA') IS NOT NULL
    PRINT '<<< CREATED TABLE CATEGORIA >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE CATEGORIA >>>'
go

/*
* TABLE: CENTRO_CUIDADO
*/

CREATE TABLE CENTRO_CUIDADO(
    id_centro      int      NOT NULL identity(1,1),
    id_oficinaRegional int      Null,
    colonia        varchar(50) NOT NULL,
    alcaldia        varchar(50) NOT NULL,
    CP              int      NOT NULL,
    numero          varchar(10) NOT NULL,
    nombre          varchar(30) NOT NULL,
    calle            varchar(50) NOT NULL,
    id_personal     int      NULL,
    id_encargado    int      NOT NULL,
    CONSTRAINT pk_CENTRO_CUIDADO PRIMARY KEY CLUSTERED (id_centro)
)
go
```

```
IF OBJECT_ID('CENTRO_CUIDADO') IS NOT NULL
    PRINT '<<< CREATED TABLE CENTRO_CUIDADO >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE CENTRO_CUIDADO >>>'
go

/*
* TABLE: COMPRA_FISICA
*/


```

```
CREATE TABLE COMPRA_FISICA(
    id_compraFisica   int                  NOT NULL identity(1,1),
    total             smallmoney           NOT NULL,
    id_personal       int                  NOT NULL,
    id_desgloseVenta int                  NOT NULL,
    comision          AS (total * 0.15) PERSISTED,
    CONSTRAINT pk_COMPRA_FISICA PRIMARY KEY CLUSTERED (id_compraFisica),
```



```
)  
go
```

```
IF OBJECT_ID('COMPRA_FISICA') IS NOT NULL  
    PRINT '<<< CREATED TABLE COMPRA_FISICA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE COMPRA_FISICA >>>'  
go
```

```
/*  
* TABLE: CONTARA  
*/
```

```
CREATE TABLE CONTARA(  
    id_producto int NOT NULL,  
    id_oferta int NOT NULL,  
    CONSTRAINT PK31 PRIMARY KEY CLUSTERED (id_producto, id_oferta)  
)  
go
```

```
IF OBJECT_ID('CONTARA') IS NOT NULL  
    PRINT '<<< CREATED TABLE CONTARA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE CONTARA >>>'  
go
```

```
/*  
* TABLE: CUENTA  
*/
```

```
CREATE TABLE CUENTA(  
    id_centro int NOT NULL,  
    id_producto int NOT NULL,  
    CONSTRAINT PK38 PRIMARY KEY CLUSTERED (id_centro, id_producto)  
)  
go
```

```
IF OBJECT_ID('CUENTA') IS NOT NULL  
    PRINT '<<< CREATED TABLE CUENTA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE CUENTA >>>'
```



go

```
/*
* TABLE: DESGLOSE_VENTA
*/
```

```
CREATE TABLE DESGLOSE_VENTA(
    id_desgloseVenta int NOT NULL IDENTITY(1,1),
    num_Unidades tinyint NOT NULL,
    montoParcial money NOT NULL,
    id_centro int NOT NULL,
    id_producto int NOT NULL,
    CONSTRAINT pk_desgloseVenta PRIMARY KEY (id_desgloseVenta)
)
go
```

```
IF OBJECT_ID('DESGLOSE_VENTA') IS NOT NULL
    PRINT '<<< CREATED TABLE DESGLOSE_VENTA >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE DESGLOSE_VENTA >>>'
go
```

```
/*
* TABLE: ENCARGADO
*/
```

```
CREATE TABLE ENCARGADO(
    id_personal int NOT NULL,
    sueldoBase smallmoney NOT NULL,
    CONSTRAINT pk_encargado PRIMARY KEY (id_personal)
)
go
```

```
IF OBJECT_ID('ENCARGADO') IS NOT NULL
    PRINT '<<< CREATED TABLE ENCARGADO >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE ENCARGADO >>>'
go
```

```
/*
* TABLE: ESPECIE
```



*/

```
CREATE TABLE ESPECIE(
    id_especie int      NOT NULL IDENTITY(1,1),
    tipo       char(1)  NOT NULL,
    id_raza   int      NOT NULL,
    CONSTRAINT pk_especie PRIMARY KEY (id_especie),
    CONSTRAINT ck_tipoMacota check (tipo in ('P','G')),
)
go
```

```
IF OBJECT_ID('ESPECIE') IS NOT NULL
    PRINT '<<< CREATED TABLE ESPECIE >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE ESPECIE >>>'
go
```

```
/*
* TABLE: GERENTE
*/
```

```
CREATE TABLE GERENTE(
    id_personal int  NOT null,
    CONSTRAINT pk_gerente PRIMARY KEY (id_personal)
)
go
```

```
IF OBJECT_ID('GERENTE') IS NOT NULL
    PRINT '<<< CREATED TABLE GERENTE >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE GERENTE >>>'
go
```

```
/*
* TABLE: MASCOTA
*/
```

```
CREATE TABLE MASCOTA(
    id_mascota      int      NOT NULL IDENTITY(1,1),
    nombre          varchar(50) NOT NULL,
    sexo            char(1)   NOT NULL,
```



```
edad          int      NOT NULL,  
rasgosCaracteristicos  varchar(200) NOT NULL,  
id_especie      int      NOT NULL,  
id_UsuarioComun   int      NOT NULL,  
CONSTRAINT pk_mascota PRIMARY KEY (id_mascota),  
CONSTRAINT ck_sexo check (sexo in ('M','F'))  
)  
go
```

```
IF OBJECT_ID('MASCOTA') IS NOT NULL  
    PRINT '<<< CREATED TABLE MASCOTA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE MASCOTA >>>'  
go
```

```
/*  
* TABLE: MEDICAMENTO  
*/
```

```
CREATE TABLE MEDICAMENTO(  
    id_medicamento  int      NOT NULL IDENTITY(1,1),  
    nombre          varchar(100) NOT NULL,  
    costo           smallmoney NOT NULL,  
    CONSTRAINT pk_medicamento PRIMARY KEY (id_medicamento)  
)  
go
```

```
IF OBJECT_ID('MEDICAMENTO') IS NOT NULL  
    PRINT '<<< CREATED TABLE MEDICAMENTO >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE MEDICAMENTO >>>'  
go
```

```
/*  
* TABLE: OFERTA_PRODUCTOSFISICOS  
*/
```

```
CREATE TABLE OFERTA_PRODUCTOSFISICOS(  
    id_producto    int      NOT NULL,  
    id_oferta      int      NOT NULL,  
    CONSTRAINT pk_of_producto PRIMARY KEY (id_producto, id_oferta)
```



```
)  
go
```

```
IF OBJECT_ID('OFERTA_PRODUCTOSFISICOS') IS NOT NULL  
    PRINT '<<< CREATED TABLE OFERTA_PRODUCTOSFISICOS >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE OFERTA_PRODUCTOSFISICOS >>>'  
go
```

```
/*  
* TABLE: OFERTAS_FISICOS  
*/
```

```
CREATE TABLE OFERTAS_FISICOS(  
    id_oferta    int      NOT NULL IDENTITY(1,1),  
    tipo         char(1)   NOT NULL,  
    descripcion  varchar(100) NOT NULL,  
    Finicio      datetime  NOT NULL,  
    Ffin        datetime  NOT NULL,  
    CONSTRAINT pk_of_fisicas PRIMARY KEY (id_oferta),  
    CONSTRAINT ck_tipoOF CHECK (tipo in ('N','L')),  
    CONSTRAINT ck_MaxFfin CHECK (Ffin >= Finicio AND Ffin <= DATEADD(day, 40,  
    Finicio))  
)  
go
```

```
IF OBJECT_ID('OFERTAS_FISICOS') IS NOT NULL  
    PRINT '<<< CREATED TABLE OFERTAS_FISICOS >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE OFERTAS_FISICOS >>>'  
go
```

```
/*  
* TABLE: OFERTAS_O  
*/
```

```
CREATE TABLE OFERTAS_O(  
    id_oferta    int      NOT NULL IDENTITY(1,1),  
    tipo         char(1)   NOT NULL,  
    descripcion  varchar(100) NOT NULL,  
    Finicio      datetime  NOT NULL,
```



```
Ffin      datetime      NOT NULL,  
CONSTRAINT pk_oferta PRIMARY KEY (id_oferta),  
CONSTRAINT ck_tipoO CHECK (tipo in ('N','L')),  
CONSTRAINT ck_MaxFfinO CHECK (Ffin >= Finicio AND Ffin <= DATEADD(day, 40,  
Finicio))  
)  
go
```

```
IF OBJECT_ID('OFERTAS_O') IS NOT NULL  
    PRINT '<<< CREATED TABLE OFERTAS_O >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE OFERTAS_O >>>'  
go
```

```
/*  
* TABLE: PERSONAL  
*/
```

```
CREATE TABLE PERSONAL(  
    id_personal      int      NOT NULL IDENTITY(1,1),  
    tipo            char(1)   NOT NULL,  
    nombre          varchar(50) NOT NULL,  
    usuario         varchar(20) NOT NULL,  
    contrasena      varchar(50) NOT NULL,  
    CURP           varchar(18) NOT NULL,  
    calle           varchar(50) NOT NULL,  
    numero          varchar(10) NOT NULL,  
    CP              int      NOT NULL,  
    alcaldia        varchar(50) NOT NULL,  
    colonia         varchar(50) NOT NULL,  
    CONSTRAINT pk_personal PRIMARY KEY CLUSTERED (id_personal),  
    CONSTRAINT ck_tipoPersonal CHECK (tipo in ('V','H','A','G','E'))  
)  
go
```

```
IF OBJECT_ID('PERSONAL') IS NOT NULL  
    PRINT '<<< CREATED TABLE PERSONAL >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE PERSONAL >>>'  
go
```



```
/*
* TABLE: PERSONAL_AYUDA
*/
CREATE TABLE PERSONAL_AYUDA(
    id_personal      int      NOT NULL,
    edad            int      NOT NULL,
    tipoMascota     char(1)  NOT NULL,
    id_centro       int      NOT NULL,
    CONSTRAINT pk_personalAyuda PRIMARY KEY (id_personal)
)
go

IF OBJECT_ID('PERSONAL_AYUDA') IS NOT NULL
    PRINT '<<< CREATED TABLE PERSONAL_AYUDA >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE PERSONAL_AYUDA >>>'
go

/*
* TABLE: PRODUCTOS_FISICOS
*/
CREATE TABLE PRODUCTOS_FISICOS(
    id_producto    int      NOT NULL IDENTITY(1,1),
    costo          smallmoney NOT NULL,
    stock          int      NOT NULL,
    CONSTRAINT pk_producto PRIMARY KEY (id_producto),
)
go

IF OBJECT_ID('PRODUCTOS_FISICOS') IS NOT NULL
    PRINT '<<< CREATED TABLE PRODUCTOS_FISICOS >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE PRODUCTOS_FISICOS >>>'
go

/*
* TABLE: PRODUCTOS_ONLINE
*/
CREATE TABLE PRODUCTOS_ONLINE(
    id_producto    int      NOT NULL IDENTITY (1,1),
```



```
stock      int      NOT NULL,  
precio     smallmoney NOT NULL,  
descripcion varchar(70) NOT NULL,  
id_categoria int      NOT NULL,  
CONSTRAINT pk_producto_online PRIMARY KEY CLUSTERED (id_producto)  
)  
go
```

```
IF OBJECT_ID('PRODUCTOS_ONLINE') IS NOT NULL  
    PRINT '<<< CREATED TABLE PRODUCTOS_ONLINE >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE PRODUCTOS_ONLINE >>>'  
go
```

```
/*  
* TABLE: RAZA  
*/
```

```
CREATE TABLE RAZA(  
    id_raza    int      NOT NULL IDENTITY(1,1),  
    nombreRaza varchar(30) NOT NULL,  
    CONSTRAINT pk_raza PRIMARY KEY (id_raza)  
)  
go
```

```
IF OBJECT_ID('RAZA') IS NOT NULL  
    PRINT '<<< CREATED TABLE RAZA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE RAZA >>>'  
go
```

```
/*  
* TABLE: RECIBO  
*/
```

```
CREATE TABLE RECIBO(  
    id_recibo   int      NOT NULL IDENTITY(1,1),  
    costoTotal  smallmoney NOT NULL,  
    tratamiento smallmoney NOT NULL,  
    examinacion varchar(50) NOT NULL,  
    id_reporte  int      NOT NULL,
```



```
    id_UsuarioComun  int      NOT NULL,  
    CONSTRAINT pk_recibo PRIMARY KEY (id_recibo)  
)  
go
```

```
IF OBJECT_ID('RECIBO') IS NOT NULL  
    PRINT '<<< CREATED TABLE RECIBO >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE RECIBO >>>'  
go  
  
/*  
* TABLE: REPORTE  
*/
```

```
CREATE TABLE REPORTE(  
    id_reporte      int      NOT NULL IDENTITY(1,1),  
    diagnostico    varchar(255) NOT NULL,  
    detallesExaminacion  varchar(1000) NOT NULL,  
    fechaHora       datetime   NOT NULL,  
    id_personal     int      NOT NULL,  
    CONSTRAINT pk_reporte PRIMARY KEY CLUSTERED (id_reporte)  
)  
go
```

```
IF OBJECT_ID('REPORTE') IS NOT NULL  
    PRINT '<<< CREATED TABLE REPORTE >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE REPORTE >>>'  
go  
  
/*  
* TABLE: REPORTE_MEDICAMENTO  
*/
```

```
CREATE TABLE REPORTE_MEDICAMENTO(  
    id_reporte      int      NOT NULL, -- no IDENTITY  
    id_medicamento int      NOT NULL,  
    dosis          varchar(100) NOT NULL,  
    CONSTRAINT pk_rep_medicamento PRIMARY KEY (id_reporte, id_medicamento)
```



```
)  
go
```

```
IF OBJECT_ID('REPORTE_MEDICAMENTO') IS NOT NULL  
    PRINT '<<< CREATED TABLE REPORTE_MEDICAMENTO >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE REPORTE_MEDICAMENTO >>>'  
go
```

```
/*  
* TABLE: RESERVACION_GUARDERIA  
*/
```

```
CREATE TABLE RESERVACION_GUARDERIA(  
    id_guarderia int NOT NULL IDENTITY(1,1),  
    finicio datetime NOT NULL,  
    ffin datetime NOT NULL,  
    costoXDia smallmoney NOT NULL,  
    numDias AS DATEDIFF(day, finicio, ffin) PERSISTED,  
    total AS (300 * costoXDia) PERSISTED,  
    id_centro int NOT NULL,  
    id_UsuarioComun int NOT NULL,  
    CONSTRAINT pk_guarderia PRIMARY KEY CLUSTERED (id_guarderia)  
)  
go
```

```
IF OBJECT_ID('RESERVACION_GUARDERIA') IS NOT NULL  
    PRINT '<<< CREATED TABLE RESERVACION_GUARDERIA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE RESERVACION_GUARDERIA >>>'  
go
```

```
/*  
* TABLE: TELEFONOS_PERSONAL_AYUDA  
*/
```

```
CREATE TABLE TELEFONOS_PERSONAL_AYUDA(  
    id_telefono_personalAyuda int NOT NULL IDENTITY(1,1),  
    id_personal int NOT NULL,  
    numTelefono varchar(20) NOT NULL CONSTRAINT uk_telefono UNIQUE,  
    CONSTRAINT pf_telefono_personalAyuda PRIMARY KEY (id_telefono_personalAyuda)
```



```
)  
go
```

```
IF OBJECT_ID('TELEFONOS_PERSONAL_AYUDA') IS NOT NULL  
    PRINT '<<< CREATED TABLE TELEFONOS_PERSONAL_AYUDA >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE TELEFONOS_PERSONAL_AYUDA >>>'  
go
```

```
/*  
* TABLE: USUARIO_COMUN  
*/
```

```
CREATE TABLE USUARIO_COMUN(  
    id_UsuarioComun int      NOT NULL identity(1,1),  
    nombre        varchar(50) NOT NULL,  
    apellidoPaterno varchar(50) NOT NULL,  
    apellidoMaterno varchar(50) NOT NULL,  
    CURP         varchar(18) NOT NULL,  
    nombreUsuario varchar(20) NOT NULL,  
    contrasena     varchar(50) NOT NULL,  
    genero        varchar(10) NOT NULL,  
    telefono      varchar(20) NOT NULL,  
    correo        varchar(100) NOT NULL,  
    calle          varchar(50) NOT NULL,  
    numero        varchar(10) NOT NULL,  
    CP            int      NOT NULL,  
    alcaldia       varchar(50) NOT NULL,  
    colonia        varchar(50) NOT NULL,  
    numeroTarjeta varchar(16) NOT NULL,  
    vigenciaTarjeta date      NOT NULL,  
    CONSTRAINT PK4 PRIMARY KEY CLUSTERED (id_UsuarioComun)  
)  
go
```

```
IF OBJECT_ID('USUARIO_COMUN') IS NOT NULL  
    PRINT '<<< CREATED TABLE USUARIO_COMUN >>>'  
ELSE  
    PRINT '<<< FAILED CREATING TABLE USUARIO_COMUN >>>'  
go
```



```
/*
* TABLE: VETERINARIO
*/
```

```
CREATE TABLE VETERINARIO(
    id_personal int NOT NULL,
    cedula varchar(10) NOT NULL,
    especialidad varchar(50) NOT NULL,
    id_centro int NOT NULL,
    CONSTRAINT pk_veterinario PRIMARY KEY CLUSTERED (id_personal)
)
go
```

```
IF OBJECT_ID('VETERINARIO') IS NOT NULL
    PRINT '<<< CREATED TABLE VETERINARIO >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE VETERINARIO >>>'
go
```

```
/*
* TABLE: VITALES
*/
```

```
CREATE TABLE VITALES(
    id_vitales int NOT NULL IDENTITY(1,1),
    ritmoCardiaco smallint NOT NULL,
    temperatura float NOT NULL,
    nivelOxigeno float NOT NULL,
    fechaHora datetime NOT NULL,
    id_brazalete int NOT NULL,
    CONSTRAINT pk_vitales PRIMARY KEY CLUSTERED (id_vitales)
)
go
```

```
IF OBJECT_ID('VITALES') IS NOT NULL
    PRINT '<<< CREATED TABLE VITALES >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE VITALES >>>'
go
```



```
/*
* TABLE: ADMINISTRATIVO
*/
```

```
ALTER TABLE ADMINISTRATIVO ADD CONSTRAINT fk_PERSONAL50
    FOREIGN KEY (id_personal)
        REFERENCES PERSONAL(id_personal)
go
```

```
ALTER TABLE ADMINISTRATIVO ADD CONSTRAINT fk_CENTRO_CUIDADO105
    FOREIGN KEY (id_centro)
        REFERENCES CENTRO_CUIDADO(id_centro)
go
```

```
/*
* TABLE: ALMACEN
*/
```

```
ALTER TABLE ALMACEN ADD CONSTRAINT fk_CENTRO_CUIDADO87
    FOREIGN KEY (id_centro)
        REFERENCES CENTRO_CUIDADO(id_centro)
go
```

```
ALTER TABLE ALMACEN ADD CONSTRAINT fk_MEDICAMENTO88
    FOREIGN KEY (id_medicamento)
        REFERENCES MEDICAMENTO(id_medicamento)
go
```

```
/*
* TABLE: AÑADE
*/
```

```
ALTER TABLE AÑADE ADD CONSTRAINT fk_ADMINISTRATIVO109
    FOREIGN KEY (id_personal)
        REFERENCES ADMINISTRATIVO(id_personal)
go
```

```
ALTER TABLE AÑADE ADD CONSTRAINT fk_PRODUCTOS_ONLINE110
    FOREIGN KEY (id_producto)
        REFERENCES PRODUCTOS_ONLINE(id_producto)
go
```



```
/*
* TABLE: BRAZALETE_MASCOTA
*/
```

```
ALTER TABLE BRAZALETE_MASCOTA ADD CONSTRAINT fk_MASCOTA9
    FOREIGN KEY (id_mascota)
        REFERENCES MASCOTA(id_mascota)
go
```

```
ALTER TABLE BRAZALETE_MASCOTA ADD CONSTRAINT
fk_PERSONAL_AYUDA101
    FOREIGN KEY (id_personal)
        REFERENCES PERSONAL_AYUDA(id_personal)
go
```

```
ALTER TABLE BRAZALETE_MASCOTA ADD CONSTRAINT fk_BRAZALETE117
    FOREIGN KEY (id_brazalete)
        REFERENCES BRAZALETE(id_brazalete)
go
```

```
/*
* TABLE: CANASTA
*/
```

```
ALTER TABLE CANASTA ADD CONSTRAINT fk_USUARIO_COMUN63
    FOREIGN KEY (id_UsuarioComun)
        REFERENCES USUARIO_COMUN(id_UsuarioComun)
go
```

```
/*
* TABLE: CANASTA_PRODUCTO
*/
```

```
ALTER TABLE CANASTA_PRODUCTO ADD CONSTRAINT fk_CANASTA97
    FOREIGN KEY (id_canasta)
        REFERENCES CANASTA(id_canasta)
go
```

```
ALTER TABLE CANASTA_PRODUCTO ADD CONSTRAINT
fk_PRODUCTOS_ONLINE108
    FOREIGN KEY (id_producto)
        REFERENCES PRODUCTOS_ONLINE(id_producto)
go
```



```
/*
* TABLE: CENTRO_CUIDADO
*/
ALTER TABLE CENTRO_CUIDADO ADD CONSTRAINT fk_encargadoC
    FOREIGN KEY (id_encargado)
        REFERENCES ENCARGADO(id_personal)
go

ALTER TABLE CENTRO_CUIDADO ADD CONSTRAINT fk_CENTRO_CUIDADO103
    FOREIGN KEY (id_oficinaRegional)
        REFERENCES CENTRO_CUIDADO(id_centro)
go

ALTER TABLE CENTRO_CUIDADO ADD CONSTRAINT fk_GERENTE104
    FOREIGN KEY (id_personal)
        REFERENCES GERENTE(id_personal)
go

/*
* TABLE: COMPRA_FISICA
*/
ALTER TABLE COMPRA_FISICA ADD CONSTRAINT fk_ENCARGADO93
    FOREIGN KEY (id_personal)
        REFERENCES ENCARGADO(id_personal)
go

ALTER TABLE COMPRA_FISICA ADD CONSTRAINT fk_DESGLOSE_VENTA114
    FOREIGN KEY (id_desgloseVenta)
        REFERENCES DESGLOSE_VENTA(id_desgloseVenta)
go

/*
* TABLE: CONTARA
*/
ALTER TABLE CONTARA ADD CONSTRAINT RefPRODUCTOS_ONLINE55
    FOREIGN KEY (id_producto)
        REFERENCES PRODUCTOS_ONLINE(id_producto)
go
```



```
ALTER TABLE CONTARA ADD CONSTRAINT RefOFERTAS_O56
    FOREIGN KEY (id_oferta)
        REFERENCES OFERTAS_O(id_oferta)
go
```

```
/*
* TABLE: CUENTA
*/
```

```
ALTER TABLE CUENTA ADD CONSTRAINT RefCENTRO_CUIDADO73
    FOREIGN KEY (id_centro)
        REFERENCES CENTRO_CUIDADO(id_centro)
go
```

```
ALTER TABLE CUENTA ADD CONSTRAINT RefPRODUCTOS_FISICOS74
    FOREIGN KEY (id_producto)
        REFERENCES PRODUCTOS_FISICOS(id_producto)
go
```

```
/*
* TABLE: DESGLOSE_VENTA
*/
```

```
ALTER TABLE DESGLOSE_VENTA ADD
    CONSTRAINT fk_codigo_producto FOREIGN KEY (id_centro, id_producto)
        REFERENCES CUENTA(id_centro, id_producto)
go
```

```
/*
* TABLE: ENCARGADO
*/
```

```
ALTER TABLE ENCARGADO ADD
    CONSTRAINT fk_personale FOREIGN KEY (id_personal) REFERENCES
PERSONAL(id_personal)
go
```

```
/*
* TABLE: ESPECIE
*/
```

```
ALTER TABLE ESPECIE ADD
```



```
CONSTRAINT fk_raza FOREIGN KEY (id_raza) REFERENCES RAZA(id_raza)
go
```

```
/*
* TABLE: GERENTE
*/
```

```
ALTER TABLE GERENTE ADD
CONSTRAINT fk_personalG FOREIGN KEY (id_personal) REFERENCES
PERSONAL(id_personal)
go
```

```
/*
* TABLE: MASCOTA
*/
```

```
ALTER TABLE MASCOTA ADD
CONSTRAINT fk_UsuarioComunM FOREIGN KEY (id_UsuarioComun) REFERENCES
USUARIO_COMUN(id_UsuarioComun)
go
```

```
ALTER TABLE MASCOTA ADD
CONSTRAINT fk_especieM FOREIGN KEY (id_especie) REFERENCES
ESPECIE(id_especie)
go
```

```
/*
* TABLE: OFERTA_PRODUCTOSFISICOS
*/
```

```
ALTER TABLE OFERTA_PRODUCTOSFISICOS ADD
CONSTRAINT fk_productoOF FOREIGN KEY (id_producto) REFERENCES
PRODUCTOS_FISICOS(id_producto)
go
```

```
ALTER TABLE OFERTA_PRODUCTOSFISICOS ADD
CONSTRAINT fk_ofertaOF FOREIGN KEY (id_oferta) REFERENCES
OFERTAS_FISICOS(id_oferta)
go
```

```
/*
```



* TABLE: PERSONAL_AYUDA
*/

ALTER TABLE PERSONAL_AYUDA ADD
CONSTRAINT ck_tipoMacotaPA check (tipoMascota in ('P','G')),
CONSTRAINT fk_personalAyuda FOREIGN KEY (id_personal) REFERENCES
PERSONAL(id_personal)
go

ALTER TABLE PERSONAL_AYUDA ADD
CONSTRAINT fk_centroPA FOREIGN KEY (id_centro) REFERENCES
CENTRO_CUIDADO(id_centro)
go

/*
* TABLE: PRODUCTOS_ONLINE
*/

ALTER TABLE PRODUCTOS_ONLINE ADD
CONSTRAINT fk_categoria_online FOREIGN KEY (id_categoria) REFERENCES
CATEGORIA(id_categoria)
go

/*
* TABLE: RECIBO
*/

ALTER TABLE RECIBO ADD
CONSTRAINT fk_reporte_rec FOREIGN KEY (id_reporte) REFERENCES
REPORTE(id_reporte)
go

ALTER TABLE RECIBO ADD
CONSTRAINT fk_usu_comun FOREIGN KEY (id_UsuarioComun) REFERENCES
USUARIO_COMUN(id_UsuarioComun)
go

/*
* TABLE: REPORTE
*/

ALTER TABLE REPORTE ADD



```
CONSTRAINT fk_personal_repo FOREIGN KEY (id_personal) REFERENCES
VETERINARIO(id_personal)
go
```

```
/*
* TABLE: REPORTE_MEDICAMENTO
*/
```

```
ALTER TABLE REPORTE_MEDICAMENTO ADD
CONSTRAINT fk_reporteR FOREIGN KEY (id_reporte) REFERENCES
REPORTE(id_reporte)
go
```

```
ALTER TABLE REPORTE_MEDICAMENTO ADD
CONSTRAINT fk_medicamentoR FOREIGN KEY (id_medicamento) REFERENCES
MEDICAMENTO(id_medicamento)
go
```

```
/*
* TABLE: RESERVACION_GUARDERIA
*/
```

```
ALTER TABLE RESERVACION_GUARDERIA ADD
CONSTRAINT fk_centroR FOREIGN KEY (id_centro) REFERENCES
CENTRO_CUIDADO(id_centro)
go
```

```
ALTER TABLE RESERVACION_GUARDERIA ADD
CONSTRAINT fk_UsuarioComunR FOREIGN KEY (id_UsuarioComun) REFERENCES
USUARIO_COMUN(id_UsuarioComun)
go
```

```
/*
* TABLE: TELEFONOS_PERSONAL_AYUDA
*/
```

```
ALTER TABLE TELEFONOS_PERSONAL_AYUDA ADD
CONSTRAINT fk_personalT FOREIGN KEY (id_personal) REFERENCES
PERSONAL_AYUDA(id_personal)
go
```



```
/*
 * TABLE: VETERINARIO
 */
```

```
ALTER TABLE VETERINARIO ADD  
    CONSTRAINT fk_centroV FOREIGN KEY (id_centro) REFERENCES  
CENTRO_CUIDADO(id_centro)  
go
```

```
ALTER TABLE VETERINARIO ADD  
CONSTRAINT fk_personalV FOREIGN KEY (id_personal) REFERENCES  
PERSONAL(id_personal)  
go
```

```
/*
 * TABLE: VITALES
 */
```

```
ALTER TABLE VITALES ADD  
    CONSTRAINT fk_brazalete FOREIGN KEY (id_brazalete) REFERENCES  
BRAZALETE(id_brazalete)  
go
```

c) DML -Transact-SQL: este script debe contener los procedimientos almacenados, funciones, disparadores y vistas (es obligatorio, el uso de cada uno). El script debe llamarse dml.sql

The image displays a sequence of four stages of a processing operation, indicated by the characters /* and */ at the top and bottom respectively. Each stage shows a central white rectangular area being processed by a black rectangular mask with a central circular hole. This mask is surrounded by a gray gradient. The background is black with a uniform dotted pattern.

/*

||\W||\W//\W // || \W || |\W|/\W|| \W|| \W|| \W|| //\W((\W
||/_||/_(())((== ||))||\V|| ||== |\W| || (()) \W
||\W\W//\W||_||/_||//||_|| \W||_|| \W|| \W//\W))



* /

- Realice un procedimiento almacenado para registrar una mascota y su brazalete

```

CREATE OR ALTER PROCEDURE pro_macota_brazalete
    @id_mascota INT,
    @id_brazalete INT ,
    @id_personal INT
AS
BEGIN
    IF EXISTS (SELECT * FROM MASCOTA WHERE id_mascota = @id_mascota)
        BEGIN
            IF EXISTS (SELECT * FROM PERSONAL_AYUDA WHERE id_personal =
@id_personal)
                BEGIN
                    IF EXISTS (SELECT * from BRAZALETE WHERE id_brazalete = @id_brazalete)
                        BEGIN
                            IF EXISTS (SELECT * FROM BRAZALETE_MASCOTA WHERE (id_mascota =
@id_mascota and estatus = 'A'))
                                SELECT 'La mascota ya tiene un registro activo' as WARNING
                                -- VALIDANDO la mascota no esté registrada ya
                            ELSE
                                BEGIN
                                    IF EXISTS (SELECT * FROM BRAZALETE_MASCOTA WHERE (id_brazalete =
@id_brazalete and estatus = 'A'))
                                        -- si el brazalete ya está usando otra mascota
                                        SELECT 'El brazalete ya tiene un registro activo' as WARNING
                                    ELSE
                                        BEGIN
                                            IF EXISTS (SELECT * FROM BRAZALETE_MASCOTA WHERE id_personal =
@id_personal)
                                                -- si ya está el personal ayuda con asignaciones verificamos las reglas de
integridad
                                                BEGIN
                                                    DECLARE @MaxAsignaciones int
                                                    SET @MaxAsignaciones = 5

                                                    IF ((SELECT COUNT(*) FROM BRAZALETE_MASCOTA
                                                        WHERE (id_personal = @id_personal and estatus = 'A')) <
@MaxAsignaciones)

```



```
-- Si el personal cuida a menos de 5 mascotas seguimos
BEGIN
    INSERT INTO BRAZALETE_MASCOTA
(id_mascota,id_brazalete,id_personal) VALUES
    (@id_mascota,@id_brazalete,@id_personal)
    SELECT 'Registro exitoso' as Resultado
END
ELSE
    SELECT 'El personal ya tiene 5 mascotas a su cargo'
END
ELSE
BEGIN
    INSERT INTO BRAZALETE_MASCOTA
(id_mascota,id_brazalete,id_personal) VALUES
    (@id_mascota,@id_brazalete,@id_personal)
    SELECT 'Registro exitoso' as Resultado
END
END
END
ELSE
    SELECT 'id_brazalete no valido' as ERROR
END
ELSE
    SELECT 'id_personalAyuda no valido' as ERROR
END
ELSE
    SELECT 'id_mascota no valido' as ERROR
END
GO
```

```
select id_personal, count(id_brazaleteMascota) from BRAZALETE_MASCOTA
Where estatus = 'A'
GROUP BY id_personal
```

```
-- marca error porque un encargado no puede estar a cargo de más de 5 mascotas
EXEC pro_macota_brazalete @id_mascota = 10,@id_brazalete = 9, @id_personal = 31
DBCC CHECKIDENT('BRAZALETE_MASCOTA',RESEED,20)
-- marca error porque id mascota no existe
EXEC pro_macota_brazalete @id_mascota = 100,@id_brazalete = 9, @id_personal = 26
DBCC CHECKIDENT('BRAZALETE_MASCOTA',RESEED,20)
-- marca error porque id BRAZALETE EN REGISTRO ACTIVO
EXEC pro_macota_brazalete @id_mascota = 10,@id_brazalete = 1, @id_personal = 26
DBCC CHECKIDENT('BRAZALETE_MASCOTA',RESEED,20)
-- si ejecuta
```



EXEC pro_macota_brazalete @id_mascota = 10,@id_brazalete = 9, @id_personal = 26
SELECT * FROM BRAZALETE_MASCOTA

select id_personal, count(id_brazaleteMascota) from BRAZALETE_MASCOTA
Where estatus = 'A'
GROUP BY id_personal

-- Realice un procedimiento almacenado para el actualizar
-- de las lecturas en los brazaletes

CREATE OR ALTER PROCEDURE pr_update_brazalte
 @id_brazalete int,
 @comio bit,
 @comida VARCHAR(100),
 @cuantoComio VARCHAR(100),
 @medicamento bit,
 @cuidadoEspecial bit
AS
BEGIN
 IF EXISTS (SELECT * FROM BRAZALETE WHERE id_brazalete = @id_brazalete)
 -- si existe el brazalete seguimos
 BEGIN
 IF EXISTS (SELECT * FROM BRAZALETE_MASCOTA WHERE (id_brazalete =
 @id_brazalete and estatus = 'A'))
 BEGIN
 UPDATE BRAZALETE
 SET comio = @comio,
 comida = @comida,
 cuantoComio = @cuantoComio,
 medicamento = @medicamento,
 cuidadoEspecial = @cuidadoEspecial
 WHERE id_brazalete = @id_brazalete
 SELECT 'UPDATED' as MENSAJE
 END
 END
 ELSE
 SELECT 'id_brazalete no esta en uso' as WARNING
 END
 ELSE
 SELECT 'id_brazalete no valido' as ERROR
 END



GO

```
SELECT * FROM BRAZALETE_MASCOTA  
DELETE FROM BRAZALETE_MASCOTA  
    WHERE id_brazaleteMascota = 21  
DELETE FROM BRAZALETE_MASCOTA  
    WHERE id_brazaleteMascota = 20  
DBCC CHECKIDENT('BRAZALETE_MASCOTA',RESEED,19)
```

```
-- No actualiza porque el id_brazalete no existe  
exec pr_update_brazalte 100, 1, 'Croquetas Pavo', 'Comió 1/4 de plato', 0, 0  
DBCC CHECKIDENT('BRAZALETE_MASCOTA',RESEED,19)  
-- No actualiza porque el id_brazalete no esta siendo utilizado en brazalete_mascota  
exec pr_update_brazalte 9, 1, 'Croquetas Pavo', 'Comió 1/4 de plato', 0, 0  
DBCC CHECKIDENT('BRAZALETE_MASCOTA',RESEED,19)  
-- si ejecuta  
exec pr_update_brazalte 8, 1, 'Croquetas Pavo', 'Comió 1/4 de plato', 0, 0
```

```
-----  
-----  
-- Realice un procedimiento almacenado para registrar una  
-- consulta  
-----  
-----
```

```
CREATE OR ALTER PROCEDURE pr_consulta  
    @diagnostico VARCHAR(255),  
    @detallesExaminacion varchar(1000),  
    @fechaHora datetime,  
    @id_personal int  
AS  
BEGIN  
    if exists (select * from VETERINARIO where id_personal = @id_personal)  
        -- verificamos exista el personal  
        BEGIN  
            INSERT INTO REPORTE  
                VALUES(@diagnostico,@detallesExaminacion,@fechaHora,@id_personal)  
                SELECT 'REPORTE GENERADO' as RESULTADO  
        END  
        ELSE  
            SELECT 'no existe el id_personal del veterinario' as ERROR  
    END  
    GO
```

```
SELECT * FROM REPORTE  
SELECT * FROM VETERINARIO -- SOLO TENEMOS 15
```



```
INSERT INTO REPORTE (diagnostico, detallesExaminacion, fechaHora, id_personal)
VALUES
-- marca error porque no existe el veterinario
EXEC pr_consulta 'Gato LOCO', 'El gato presenta SINDROMES', '2023-06-20 12:30:00', 20
-- si ejecuta
EXEC pr_consulta 'Gato LOCO', 'El gato presenta SINDROMES', '2023-06-20 12:30:00', 15
```

```
-- Realice un procedimiento almacenado para el registro y
-- venta de medicamentos incluyendo la actualización del
-- stock
```

```
CREATE OR ALTER PROCEDURE pk_add_medicamento
    @id_centro INT,
    @id_medicamento INT,
    @cantidad SMALLINT
AS
BEGIN
    IF EXISTS (SELECT * FROM CENTRO_CUIDADO WHERE id_centro = @id_centro)
        -- existe el centro
        BEGIN
            IF EXISTS (SELECT * FROM MEDICAMENTO WHERE id_medicamento =
@id_medicamento)
                -- verificamos exista un medicamento con el id dado
                BEGIN
                    DECLARE @nombre VARCHAR(100)
                    IF EXISTS (SELECT id_medicamento FROM ALMACEN WHERE (id_centro =
@id_centro AND id_medicamento = @id_medicamento))
                        -- Actualizamos
                        BEGIN
                            DECLARE @cantidad_previa SMALLINT
                            SELECT @nombre = nombre FROM MEDICAMENTO WHERE id_medicamento =
@id_medicamento
                            SELECT @cantidad_previa = cantidad FROM ALMACEN WHERE (id_centro =
@id_centro AND id_medicamento = @id_medicamento)
                            SET @cantidad = @cantidad_previa + @cantidad
                            UPDATE ALMACEN
                            SET cantidad = @cantidad WHERE (id_centro = @id_centro AND id_medicamento =
@id_medicamento)
                            SELECT 'Update --> ' + @nombre + ':' + CAST(@cantidad as varchar)
                        END
                    ELSE
                        -- Insertamos
                END
            END
        END
    END
```



```
BEGIN
    SELECT @nombre = nombre FROM MEDICAMENTO WHERE id_medicamento =
@id_medicamento
    INSERT INTO ALMACEN VALUES
        (@id_centro, @id_medicamento, @cantidad)
    SELECT 'Agregado --> ' + @nombre + ':' + CAST(@cantidad as varchar)
END
END
ELSE
    SELECT 'id_medicamento no valido' as ERROR
END
ELSE
    SELECT 'id_CENTRO no valido' as ERROR
END
```

```
SELECT * FROM ALMACEN
select * from CENTRO_CUIDADO -- solo tenemos 7
SELECT * FROM MEDICAMENTO -- SOLO TENEMOS 12
-- agrega porque no existia stock para el centro 5
EXEC pk_add_medicamento 5, 1, 50-- Paracetamol
-- actualiza porque ya existia stock
EXEC pk_add_medicamento 2, 6, 60 -- Loratadina
-- ERROR PORQUE NO TENEMOS MEDICAMENTO CON ESE ID
EXEC pk_add_medicamento 2, 50, 60 -- Loratadina
-- ERROR PORQUE NO TENEMOS CENTRO CON ESE ID
EXEC pk_add_medicamento 100, 5, 60 -- Loratadina
```

```
CREATE OR ALTER PROCEDURE pk_quit_medicamento
    @id_reporte INT,
    @id_medicamento INT,
    @dosis VARCHAR(100)
AS
BEGIN
IF EXISTS (SELECT * FROM REPORTE WHERE id_reporte = @id_reporte)
    -- existe el reporte
    BEGIN
        IF EXISTS (SELECT * FROM MEDICAMENTO WHERE id_medicamento =
@id_medicamento)
            -- verificamos exista un medicamento con el id dado
            BEGIN
                -- obtenemos el id_centro del reporte
                DECLARE @id_centro int
                DECLARE @cantidad smallint
                SELECT @id_centro = v.id_centro from VETERINARIO v
                WHERE v.id_personal = (SELECT r.id_personal FROM REPORTE r
```



```
        WHERE r.id_reporte = @id_reporte)
DECLARE @nombre VARCHAR(100)
IF EXISTS (SELECT id_medicamento FROM ALMACEN WHERE (id_centro =
@id_centro AND id_medicamento = @id_medicamento))
    -- Actualizamos e insertamos en medicamento_registro (receta)
BEGIN
    DECLARE @cantidad_previa SMALLINT
    SELECT @nombre = nombre FROM MEDICAMENTO WHERE id_medicamento =
@id_medicamento
    SELECT @cantidad_previa = cantidad FROM ALMACEN WHERE (id_centro =
@id_centro AND id_medicamento = @id_medicamento)
    SET @cantidad = @cantidad_previa - 1
    UPDATE ALMACEN
        SET cantidad = @cantidad WHERE (id_centro = @id_centro AND
id_medicamento = @id_medicamento)
    SELECT 'Update --> ' + @nombre + ':' + CAST(@cantidad as varchar)

    INSERT INTO REPORTE_MEDICAMENTO VALUES
        (@id_reporte,@id_medicamento,@dosis)
    SELECT 'receta insertada' as MENSAJE
END
ELSE
    SELECT 'No hay en existencia' as MENSAJE
END
ELSE
    SELECT 'id_medicamento no valido' as ERROR
END
ELSE
    SELECT 'is_reporte no valido' as ERROR
END

SELECT * FROM REPORTE_MEDICAMENTO rm
    INNER JOIN REPORTE r on rm.id_reporte = r.id_reporte
    INNER JOIN VETERINARIO v on r.id_personal = v.id_personal
    Inner JOIN CENTRO_CUIDADO c on v.id_centro = c.id_centro
select * from ALMACEN
SELECT * FROM MEDICAMENTO

-- no debe ejecutarse porque solo tengo 7 reportes
exec pk_quit_medicamento 10, 1, '1 comprimido cada 8 horas'
-- no debe ejecutarse porque solo tengo 12 medicamentos
exec pk_quit_medicamento 7, 1000, 'PRUEBA 2'
-- no debe ejecutarse porque el centro 1 done pertenece la consulta, no tiene ese medicamento
exec pk_quit_medicamento 7, 12, 'PRUEBA 3'
-- si jala
```



-- actualizamos el stock del centro considerando por cada receta se da una entidad del medicamento (caja, botella, etc)

```
exec pk_quit_medicamento 5, 1, 'PRUEBA 4'  
/*
```

```
|| ||||| // || //\ | | | | ((  
|| == || ||\\| (( ||(( )) ||\\| ||== \\  
|| \\\// || \\| \\_ || \\// || \\| ||__ \_)
```

```
*/
```

-- Funcion para ver el inventario de medicamentos, con su costo por centro.

```
CREATE FUNCTION ObtenerDetalleAlmacen()  
RETURNS TABLE  
AS  
RETURN  
(  
    SELECT A.id_centro, M.nombre AS medicamento, A.cantidad, M.costo  
    FROM ALMACEN A  
    JOIN MEDICAMENTO M ON A.id_medicamento = M.id_medicamento  
    JOIN CENTRO_CUIDADO C ON A.id_centro = C.id_centro  
);  
  
SELECT *  
FROM dbo.ObtenerDetalleAlmacen();
```

-- Funcion para reporte de ventas e medicamentos en un periodo de tiempo, incluyendo cantidad y monto total

```
CREATE FUNCTION ObtenerDetalleMedicamentos()  
RETURNS TABLE  
AS  
RETURN  
(  
    SELECT M.nombre AS medicamento, COUNT(RM.id_medicamento) AS cantidad,  
    SUM(M.costo) AS monto_total  
    FROM MEDICAMENTO M  
    INNER JOIN REPORTE_MEDICAMENTO RM ON M.id_medicamento =  
    RM.id_medicamento  
    INNER JOIN REPORTE R ON RM.id_reporte = R.id_reporte  
    INNER JOIN RECIBO RC ON R.id_reporte = RC.id_reporte  
    WHERE R.fechaHora >= '2023-01-01' AND R.fechaHora <= '2023-06-30'  
    GROUP BY M.nombre
```



);

SELECT * FROM dbo.ObtenerDetalleMedicamentos();

/*

||| ||((\| |\| // \\\(|
\\|\| \\ \| \| = \| \\
\V/ \|_) \| \| \| _)

*/

-- 3. Listado de las mascotas y su dueño datos generales, número de días en la guardería, datos generales de ambos.

-- El reporte se obtiene en un periodo de fechas

CREATE VIEW listadoMascotas AS

SELECT

M.nombre AS nombre_mascota, M.edad AS edad_mascota, M.sexo AS sexo_mascota,
M.rasgosCaracteristicos AS rasgos_mascota,

U.nombre AS nombre_dueño, U.apellidoPaterno AS apellido_paterno_dueño,
U.apellidoMaterno AS apellido_materno_dueño, U.CURP AS curp_dueño,
U.telefono AS telefono_dueño,

R.finicio AS fecha_inicio_guarderia, R.ffin AS fecha_fin_guarderia, R.numDias AS
numero_dias_guarderia

FROM

MASCOTA AS M

INNER JOIN USUARIO_COMUN AS U ON M.id_UsuarioComun = U.id_UsuarioComun
INNER JOIN RESERVACION_GUARDERIA AS R ON M.id_mascota =

R.id_UsuarioComun

--WHERE

--R.finicio >= '2023-01-01' AND R.ffin <= '2023-12-31'

select * from listadoMascotas

-- 1. Gastos de cada mascota, datos de la mascota, fechas y días de estancia, descripción de gatos y costo.

CREATE VIEW desgloseMascotas AS

SELECT M.nombre AS NombreMascota, M.rasgosCaracteristicos, R.nombreRaza, E.tipo AS
Especie,

RG.finicio, RG.ffin, RG.numDias, RG.total
FROM MASCOTA AS M



```
JOIN USUARIO_COMUN AS UC ON M.id_UsuarioComun = UC.id_UsuarioComun
JOIN RESERVACION_GUARDERIA AS RG ON UC.id_UsuarioComun =
RG.id_UsuarioComun
JOIN BRAZALETE_MASCOTA AS BM ON M.id_mascota = BM.id_mascota
JOIN ESPECIE AS E ON M.id_especie = E.id_especie
JOIN RAZA AS R ON E.id_raza = R.id_raza
```

```
select * from desgloseMascotas
```

```
-- 2. Listado con datos generales de las consultas , incluyendo veterinario que lo atendió,
diagnóstico y medicamento.
```

```
CREATE VIEW datosGeneralesConsultas as
SELECT
    R.id_reporte AS NumeroReporte,
    V.cedula AS CedulaVeterinario,
    V.especialidad AS EspecialidadVeterinario,
    R.diagnostico AS Diagnostico,
    M.nombre AS Medicamento
FROM
    REPORTE AS R
JOIN VETERINARIO AS V ON R.id_personal = V.id_personal
JOIN REPORTE_MEDICAMENTO AS RM ON R.id_reporte = RM.id_reporte
JOIN MEDICAMENTO AS M ON RM.id_medicamento = M.id_medicamento
```

```
select * from datosGeneralesConsultas
```

```
/*
```

```
||||||\|| //\|| /\|| \|/\(\| ----- --
|| \|/\|(( __ (( __ \|== \|/\| \|
|| \|/\| \|/\| \|/\| \|/\| \|/\| \|/\|)
```

```
*/
```

```
--- $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  
--- SUELDO MAYOR A 5000  
--- $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
Create or alter trigger tr_verificasueldoI on ENCARGADO
instead of insert
as
begin
if exists(select sueldoBase from inserted)
```



```
BEGIN
DECLARE @sueldo INT
Select @sueldo = sueldoBase from inserted
if(@sueldo < 5000)
BEGIN
    SELECT 'No se puede poner un sueldo menor a 5000'
END
ELSE
BEGIN
    INSERT INTO ENCARGADO
    select id_personal, sueldoBase from inserted
    SELECT 'Se ha insertado el encargado :)'
END
END
end
go

---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
--- VALIDA JERARQUIA TIPOS
---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
CREATE OR ALTER TRIGGER tr_jerarquia on PERSONAL
instead of INSERT
AS
BEGIN
IF EXISTS (select * from PERSONAL
            WHERE CURP = (select CURP from INSERTED))
begin
    select 'Ya existe personal con este CURP'
end

-- ya no se puede seguir con la inserción
ELSE IF ((SELECT tipo FROM INSERTED) NOT IN ('A','H','G','E','V'))
BEGIN
    SELECT 'Tipo incorrecto, verifique las restricciones'
end
-- si el tipo es incorrecto n insertamos
ELSE IF EXISTS (
    SELECT *
    FROM inserted
    WHERE CURP LIKE '%[^A-Za-z0-9]%' or nombre LIKE '%[^A-Za-z0-9]%' or usuario
    LIKE '%[^A-Za-z0-9]%'
    or calle LIKE '%[^A-Za-z0-9]%' or colonia LIKE '%[^A-Za-z0-9]%'
)
BEGIN
    Select 'Haz introducido un dato erronéo, revisa bien' as mensaje
```



```
RETURN
END
ELSE
begin
    INSERT INTO PERSONAL
        SELECT tipo, nombre, usuario, contrasena, CURP, calle,
        numero, CP, alcaldia, colonia FROM INSERTED
        SELECT 'Inserción hecha correctamente :)'
    end
END
GO
```

```
CREATE OR ALTER TRIGGER tr_jerarquiaU on PERSONAL
instead of UPDATE
```

```
AS
BEGIN
    -- ya no se puede seguir con la inserción
    IF ((SELECT tipo FROM INSERTED) NOT IN ('A','H','G','E','V'))
        BEGIN
            SELECT 'Tipo incorrecto, verifique las restricciones'
        end
    -- si el tipo es incorrecto n insertamos
    ELSE
        begin
            UPDATE PERSONAL SET tipo = (SELECT tipo FROM inserted)
            WHERE id_personal = (select id_personal from inserted)
            SELECT 'Actualización hecha correctamente :)'
        end
END
GO
```

```
--- $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
---      OFERTAS NO MAYOR A 40 DIAS
```

```
--- $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
CREATE OR ALTER TRIGGER tr_ofertas on OFERTAS_FISICOS
instead of INSERT
```

```
AS
BEGIN
    -- ya no se puede seguir con la inserción
    IF ((SELECT Ffin FROM INSERTED) - (SELECT Finicio FROM INSERTED)>40)
        BEGIN
            SELECT 'LA OFERTAS NO PUEDES DURAR MAS DE 40 DIAS'
        end
    -- si el tipo es incorrecto n insertamos
    ELSE
```



```
begin
    insert into OFERTAS_FISICOS
        SELECT tipo, descripcion, Finicio, Ffin FROM inserted
        SELECT 'OFERTA ESTABLECIDA CORRECTAMENTE'
    end
END
GO

CREATE OR ALTER TRIGGER tr_ofertasU on OFERTAS_FISICOS
instead of UPDATE
AS
BEGIN
    -- ya no se puede seguir con la inserción
    IF ((SELECT Ffin FROM INSERTED) - (SELECT Finicio FROM DELETED)>40)
        BEGIN
            SELECT 'LA OFERTAS NO PUEDES DURAR MAS DE 40 DIAS'
        end
    -- si el tipo es incorrecto n insertamos
    ELSE
        begin
            UPDATE OFERTAS_FISICOS SET Ffin=(SELECT Ffin FROM INSERTED)
            WHERE id_oferta=(select id_oferta from inserted)
            SELECT 'OFERTA ACTUALIZADA CORRECTAMENTE'
        end
END
GO
```

--- \$

--- COMISION POR CADA VENTA

--- \$

--CALCULO DE LA COMISION SE HACE DESDE EL CHECK DE LA TABLA
--POR CADA VENTA ACTUALIZA EL SALARIO DEL ENCARGADO

```
CREATE OR ALTER TRIGGER tr_comision on COMPRA_FISICA
instead of insert
AS
BEGIN
    DECLARE @comision SMALLMONEY
    select @comision = comision from inserted
    UPDATE ENCARGADO
    SET sueldoBase = sueldoBase + @comision
    SELECT 'Se ha insertado la compra, el encargado ha recibido su comision'
```



END
GO

--- \$
--- ESTANDARES Y NOMENCLATURA
--- \$

```
CREATE or ALTER TRIGGER tr_nomenclaturaUser
on USUARIO_COMUN
INSTEAD OF INSERT
AS
BEGIN
    if exists (select correo from inserted
               where correo like '%@%')
        BEGIN
            INSERT INTO USUARIO_COMUN SELECT nombre, apellidoPaterno, apellidoMaterno,
CURP,
            nombreUsuario, contrasena, genero, telefono, correo, calle, numero, CP, alcaldia,
            colonia, numeroTarjeta, vigenciaTarjeta
            FROM inserted
            SELECT 'Se inserta el usuario correctamente' as mensaje
        END
    ELSE
        BEGIN
            SELECT 'Ingresa un correo válido con carácter @' as mensaje
        END
    END
GO
```

--- \$
--- CANCELACION 15%
--- \$

--VALIDADO EN EL CHECK DE LA TABLA

**d) Elaborar un script para las estadísticas, cuyo nombre debe ser
Informes.sql**

```
use GUARDERIA
go
```

-- GUARDERIA-----



-- 1. El tipo de mascota que más se deja en la guardería para su cuidado (tipo y cantidad).

```
SELECT E.tipo AS tipo_mascota, COUNT(*) AS cantidad
FROM MASCOTA AS M
JOIN ESPECIE AS E ON M.id_especie = E.id_especie
GROUP BY E.tipo
ORDER BY cantidad DESC
```

-- 2. En qué época del año se deja un mayor número de mascotas en la guardería

```
SELECT MONTH(finicio) AS Mes,
       COUNT(*) AS NumeroMascotas
  FROM RESERVACION_GUARDERIA
 WHERE YEAR(finicio) = YEAR(GETDATE()) -- Filtrar por el año actual
 GROUP BY MONTH(finicio)
 ORDER BY NumeroMascotas DESC;
```

-- 3. Listado de las mascotas y su dueño datos generales, número de días en la guardería, datos generales de ambos.

-- El reporte se obtiene en un periodo de fechas

```
CREATE VIEW listadoMascotas AS
SELECT
    M.nombre AS nombre_mascota, M.edad AS edad_mascota, M.sexo AS sexo_mascota,
    M.rasgosCaracteristicos AS rasgos_mascota,
    U.nombre AS nombre_dueño, U.apellidoPaterno AS apellido_paterno_dueño,
    U.apellidoMaterno AS apellido_materno_dueño, U.CURP AS curp_dueño,
    U.telefono AS telefono_dueño,
    R.finicio AS fecha_inicio_guarderia, R.ffin AS fecha_fin_guarderia, R.numDias AS
    numero_dias_guarderia
  FROM
    MASCOTA AS M
  INNER JOIN USUARIO_COMUN AS U ON M.id_UsuarioComun = U.id_UsuarioComun
  INNER JOIN RESERVACION_GUARDERIA AS R ON M.id_mascota =
    R.id_UsuarioComun
--WHERE
--  R.finicio >= '2023-01-01' AND R.ffin <= '2023-12-31'
select * from listadoMascotas
```

-- 4. Enfermedades más frecuentes (5), total de animales con dicha enfermedad

-- ordenados de mayor a menor



```
select * from REPORTE
```

```
SELECT TOP 5 diagnostico, COUNT(*) AS total_animales  
FROM REPORTE  
GROUP BY diagnostico  
ORDER BY total_animales DESC;
```

```
-----  
-- Veterinaria-----  
-----
```

```
-- 1. Gastos de cada mascota, datos de la mascota, fechas y días de estancia, descripción de gatos  
y costo.
```

```
CREATE VIEW desgloseMascotas AS  
SELECT M.nombre AS NombreMascota, M.rasgosCaracteristicos, R.nombreRaza, E.tipo AS  
Especie,  
RG.finicio, RG.ffin, RG.numDias, RG.total  
FROM MASCOTA AS M  
JOIN USUARIO_COMUN AS UC ON M.id_UsuarioComun = UC.id_UsuarioComun  
JOIN RESERVACION_GUARDERIA AS RG ON UC.id_UsuarioComun =  
RG.id_UsuarioComun  
JOIN BRAZALETE_MASCOTA AS BM ON M.id_mascota = BM.id_mascota  
JOIN ESPECIE AS E ON M.id_especie = E.id_especie  
JOIN RAZA AS R ON E.id_raza = R.id_raza
```

```
select * from desgloseMascotas
```

```
-- 2. Listado con datos generales de las consultas , incluyendo veterinario que lo atendió,  
diagnóstico y medicamento.
```

```
CREATE VIEW datosGeneralesConsultas as  
SELECT  
R.id_reporte AS NumeroReporte,  
V.cedula AS CedulaVeterinario,  
V.especialidad AS EspecialidadVeterinario,  
R.diagnostico AS Diagnostico,  
M.nombre AS Medicamento  
FROM  
REPORTE AS R  
JOIN VETERINARIO AS V ON R.id_personal = V.id_personal  
JOIN REPORTE_MEDICAMENTO AS RM ON R.id_reporte = RM.id_reporte  
JOIN MEDICAMENTO AS M ON RM.id_medicamento = M.id_medicamento
```



```
select * from datosGeneralesConsultas
```

-- 3. Inventario de medicamentos, con su costo por centro.

```
SELECT A.id_centro, M.nombre AS medicamento, A.cantidad, M.costo
FROM ALMACEN A
JOIN MEDICAMENTO M ON A.id_medicamento = M.id_medicamento
JOIN CENTRO_CUIDADO C ON A.id_centro = C.id_centro;
```

-- 4. Enfermedades más frecuentes, por centro, ordenadas de mayor a menor

```
SELECT C.nombre AS centro, R.diagnostico AS enfermedad, COUNT(*) AS frecuencia
FROM CENTRO_CUIDADO AS C
JOIN PERSONAL AS P ON C.id_personal = P.id_personal
JOIN VETERINARIO AS V ON P.id_personal = V.id_personal
JOIN REPORTE AS R ON V.id_personal = R.id_personal
GROUP BY C.nombre, R.diagnostico
ORDER BY frecuencia DESC;
```

-- 5. Reporte de ventas e medicamentos en un periodo de tiempo, incluyendo cantidad y monto total

```
SELECT M.nombre AS medicamento, COUNT(RM.id_medicamento) AS cantidad,
SUM(M.costo) AS monto_total
FROM MEDICAMENTO M
INNER JOIN REPORTE_MEDICAMENTO RM ON M.id_medicamento =
RM.id_medicamento
INNER JOIN REPORTE R ON RM.id_reporte = R.id_reporte
INNER JOIN RECIBO RC ON R.id_reporte = RC.id_reporte
WHERE R.fechaHora >= '2023-01-01' AND R.fechaHora <= '2023-06-30'
GROUP BY M.nombre
```

-- Ventas-----

-- 1. El centro con mayor número de ventas en un periodo de tiempo. (Separar las ventas en línea de las ventas físicas de cada estado).

-- 2. Los artículos más vendidos y los menos vendidos por categoría.

-- Mas



```
SELECT p.id_producto, p.stock, p.precio, p.descripcion, c.categoría
FROM (
    SELECT po.id_producto, po.stock, po.precio, po.descripcion, c.id_categoria,
           ROW_NUMBER() OVER (PARTITION BY c.id_categoria ORDER BY po.stock DESC)
    AS ranking
    FROM PRODUCTOS_ONLINE po
    INNER JOIN CATEGORIA c ON po.id_categoria = c.id_categoria
) p
INNER JOIN CATEGORIA c ON p.id_categoria = c.id_categoria
WHERE p.ranking = 1;
```

```
-- Menos
SELECT p.id_producto, p.stock, p.precio, p.descripcion, c.categoría
FROM (
    SELECT po.id_producto, po.stock, po.precio, po.descripcion, c.id_categoria,
           ROW_NUMBER() OVER (PARTITION BY c.id_categoria ORDER BY po.stock ASC)
    AS ranking
    FROM PRODUCTOS_ONLINE po
    INNER JOIN CATEGORIA c ON po.id_categoria = c.id_categoria
) p
INNER JOIN CATEGORIA c ON p.id_categoria = c.id_categoria
WHERE p.ranking = 1;
```

-- ROW_NUMBER(): Asigna un número de fila a cada registro dentro de cada partición.
-- OVER: Especifica cómo se deben dividir las filas en particiones.
-- PARTITION BY c.id_categoria: Define la partición por la columna id_categoria. Esto significa que los registros se dividirán en grupos separados según su categoría.

-- 3. Época en la que más se vende. Época del año y monto total

-- 4. Los 5 empleados con mayor comisión, este reporte se obtiene de manera mensual

```
SELECT TOP 5 p.nombre, cf.comision
FROM PERSONAL p
INNER JOIN COMPRA_FISICA cf ON p.id_personal = cf.id_personal
ORDER BY cf.comision DESC;
```

-- 5. Inventario de las tiendas de cada tienda

```
SELECT CC.nombre AS nombre_centro, PF.id_producto, PF.costo, PF.stock
FROM CENTRO_CUIDADO CC
JOIN CUENTA C ON CC.id_centro = C.id_centro
JOIN PRODUCTOS_FISICOS PF ON C.id_producto = PF.id_producto
```



order by CC.id_centro

e) Elaborar el script para la carga de información, el cual debe llamarse cargaInicial.sql. este contendrá los inserts necesarios para cargar al menos 20 ventas para usuarios registrados y 10 ventas en establecimiento, 10 usuarios registrados, 10 gestores, 5 vendedores, 4 categorías con 5 productos cada uno, 5 ofertas y 5 cestas pendientes de compra.

```
--DBCC CHECKIDENT('PERSONAL',RESEED,0)
```

```
-----  
--- PERSONAL  
-----
```

```
BEGIN TRANSACTION
```

```
-- Registros de tipo 'V' (19 registros)
```

```
INSERT INTO PERSONAL (tipo, nombre, usuario, contrasena, CURP, calle, numero, CP, alcaldia, colonia)
```

```
VALUES
```

```
    ('V', 'Juanita Sánchez', 'juanitasanchez', '987654', 'ASDFG987654ZXCVBNM', 'Calle21',  
     'Numero21', 10021, 'Álvaro Obregón', 'San Ángel'),
```

```
    ('V', 'Roberto Torres', 'robertotorres', '987654', 'ASDFG987654ZXCVBNM', 'Calle22',  
     'Numero22', 10022, 'Benito Juárez', 'Del Valle'),
```

```
    ('V', 'Patricia Ramírez', 'patriciaramirez', '987654', 'ASDFG987654ZXCVBNM', 'Calle23',  
     'Numero23', 10023, 'Cuautitlán Izcalli', 'Centro'),
```

```
    ('V', 'Daniel García', 'danielgarcia', '987654', 'ASDFG987654ZXCVBNM', 'Calle24',  
     'Numero24', 10024, 'Ecatepec', 'San Cristóbal Centro'),
```

```
    ('V', 'Silvia López', 'silvialopez', '987654', 'ASDFG987654ZXCVBNM', 'Calle25', 'Numero25',  
     10025, 'Toluca', 'Santa Ana Tlapaltitlán'),
```

```
    ('V', 'Rodrigo Martínez', 'rodrigomartinez', '987654', 'ASDFG987654ZXCVBNM', 'Calle26',  
     'Numero26', 10026, 'Naucalpan de Juárez', 'Satélite'),
```

```
    ('V', 'María Fernández', 'mariafernandez', '987654', 'ASDFG987654ZXCVBNM', 'Calle27',  
     'Numero27', 10027, 'Tlalnepantla de Baz', 'Lomas Verdes'),
```

```
    ('V', 'Jorge Sánchez', 'jorgesanchez', '987654', 'ASDFG987654ZXCVBNM', 'Calle28',  
     'Numero28', 10028, 'Coacalco de Berriozábal', 'San Lorenzo Tetlixtac'),
```

```
    ('V', 'Sara Rodríguez', 'sararodriguez', '987654', 'ASDFG987654ZXCVBNM', 'Calle29',  
     'Numero29', 10029, 'Texcoco', 'Santa Cruz de Arriba'),
```

```
    ('V', 'Fernanda González', 'fernandagonzalez', '987654', 'ASDFG987654ZXCVBNM',  
     'Calle30', 'Numero30', 10030, 'Metepec', 'San Miguel Totocuitlapilco'),
```

```
    ('V', 'Adrián Gómez', 'adriangomez', '987654', 'ASDFG987654ZXCVBNM', 'Calle31',  
     'Numero31', 10031, 'Nezahualcóyotl', 'Vicente Villada'),
```

```
    ('V', 'Laura Castro', 'lauracastro', '987654', 'ASDFG987654ZXCVBNM', 'Calle32',  
     'Numero32', 10032, 'Chalco', 'San Gregorio Cuautzingo'),
```



('V', 'Ricardo Herrera', 'ricardoherrera', '987654', 'ASDFG987654ZXCVBNM', 'Calle33',
'Numero33', 10033, 'Cuautitlán', 'Centro'),
('V', 'Diana Ramírez', 'dianaramirez', '987654', 'ASDFG987654ZXCVBNM', 'Calle34',
'Numero34', 10034, 'Atizapán de Zaragoza', 'Lomas de Atizapán'),
('V', 'Gabriel Martínez', 'gabrielmartinez', '987654', 'ASDFG987654ZXCVBNM', 'Calle35',
'Numero35', 10035, 'Tultitlán', 'Centro'),
('E', 'Carlos Martínez', 'empleado1', '987654', 'ASDFG987654ZXCVBNM', 'Calle36',
'Numero36', 10036, 'Álvaro Obregón', 'Colonia Roma'),
('E', 'Ana Rodríguez', 'empleado2', '987654', 'ASDFG987654ZXCVBNM', 'Calle37',
'Numero37', 10037, 'Benito Juárez', 'Colonia Del Valle'),
('E', 'Miguel Herrera', 'empleado3', '987654', 'ASDFG987654ZXCVBNM', 'Calle38',
'Numero38', 10038, 'Coyoacán', 'Colonia Coyoacán'),
('E', 'Isabel Gómez', 'empleado4', '987654', 'ASDFG987654ZXCVBNM', 'Calle39',
'Numero39', 10039, 'Cuauhtémoc', 'Colonia Condesa'),
('E', 'Javier López', 'empleado5', '987654', 'ASDFG987654ZXCVBNM', 'Calle40', 'Numero40',
10040, 'Gustavo A. Madero', 'Colonia Lindavista'),
('E', 'Martha Pérez', 'empleado6', '987654', 'ASDFG987654ZXCVBNM', 'Calle41',
'Numero41', 10041, 'Miguel Hidalgo', 'Colonia Polanco'),
('E', 'Roberto García', 'empleado7', '987654', 'ASDFG987654ZXCVBNM', 'Calle42',
'Numero42', 10042, 'Tlalpan', 'Colonia Pedregal'),
('H', 'Juan Pérez', 'juanperez', '987654', 'ASDFG987654ZXCVBNM', 'Calle43', 'Numero43',
10043, 'Iztapalapa', 'Colonia Leyes de Reforma'),
('H', 'María García', 'mariagarcia', '987654', 'ASDFG987654ZXCVBNM', 'Calle44',
'Numero44', 10044, 'Azcapotzalco', 'Colonia San Álvaro'),
('H', 'Pedro López', 'pedrolopez', '987654', 'ASDFG987654ZXCVBNM', 'Calle45',
'Numero45', 10045, 'Xochimilco', 'Colonia Nativitas'),
('H', 'Ana Martínez', 'anamartinez', '987654', 'ASDFG987654ZXCVBNM', 'Calle46',
'Numero46', 10046, 'Venustiano Carranza', 'Colonia Jardín Balbuena'),
('H', 'Luisa Torres', 'luisatorres', '987654', 'ASDFG987654ZXCVBNM', 'Calle47', 'Numero47',
10047, 'Milpa Alta', 'Colonia San Francisco Tlalnepantla'),
('H', 'Carlos Ramírez', 'carlosramirez', '987654', 'ASDFG987654ZXCVBNM', 'Calle48',
'Numero48', 10048, 'Tláhuac', 'Colonia Del Mar'),
('H', 'Laura Sánchez', 'laurasanchez', '987654', 'ASDFG987654ZXCVBNM', 'Calle49',
'Numero49', 10049, 'Cuajimalpa de Morelos', 'Colonia Cuajimalpa'),
('H', 'José Rodríguez', 'joserodriguez', '987654', 'ASDFG987654ZXCVBNM', 'Calle50',
'Numero50', 10050, 'Magdalena Contreras', 'Colonia San Jerónimo Lídice'),
('H', 'Sofía Morales', 'sofiamorales', '987654', 'ASDFG987654ZXCVBNM', 'Calle51',
'Numero51', 10051, 'Tlalpan', 'Colonia Fuentes del Pedregal'),
('H', 'Fernando González', 'fernandogonzalez', '987654', 'ASDFG987654ZXCVBNM',
'Calle52', 'Numero52', 10052, 'Iztacalco', 'Colonia Agrícola Oriental'),
('A', 'Elena Sánchez', 'esanchez', '987654', 'ASDFG987654ZXCVBNM', 'Calle53', 'Numero53',
10053, 'Tlalpan', 'Colonia Parques del Pedregal'),
('A', 'Roberto Rodríguez', 'rrodriguez', '987654', 'ASDFG987654ZXCVBNM', 'Calle54',
'Numero54', 10054, 'Magdalena Contreras', 'Colonia Las Águilas'),



('A', 'Adriana Torres', 'atorres', '987654', 'ASDFG987654ZXCVBNM', 'Calle55', 'Numero55', 10055, 'Álvaro Obregón', 'Colonia Santa Fe'),
('A', 'Mauricio Ríos', 'mrrios', '987654', 'ASDFG987654ZXCVBNM', 'Calle56', 'Numero56', 10056, 'Miguel Hidalgo', 'Colonia Polanco'),
('G', 'Alejandra Ramírez', 'aramirez', '987654', 'ASDFG987654ZXCVBNM', 'Calle57', 'Numero57', 10057, 'Benito Juárez', 'Colonia Del Valle'),
('G', 'Felipe Herrera', 'fherrera', '987654', 'ASDFG987654ZXCVBNM', 'Calle58', 'Numero58', 10058, 'Cuauhtémoc', 'Colonia Roma'),
('G', 'Verónica Vargas', 'vvargas', '987654', 'ASDFG987654ZXCVBNM', 'Calle59', 'Numero59', 10059, 'Gustavo A. Madero', 'Colonia Lindavista'),
('G', 'Julio López', 'jlopez', '987654', 'ASDFG987654ZXCVBNM', 'Calle60', 'Numero60', 10060, 'Coyoacán', 'Colonia Del Carmen'),
('A', 'Luisa Gómez', 'lgomez', '987654', 'ASDFG987654ZXCVBNM', 'Calle57', 'Numero57', 10057, 'Benito Juárez', 'Colonia Del Valle'),
('A', 'Carlos López', 'clopez', '987654', 'ASDFG987654ZXCVBNM', 'Calle58', 'Numero58', 10058, 'Cuauhtémoc', 'Colonia Roma'),
('A', 'Ana Ramírez', 'aramirez', '987654', 'ASDFG987654ZXCVBNM', 'Calle59', 'Numero59', 10059, 'Gustavo A. Madero', 'Colonia Lindavista');

```
SELECT * FROM PERSONAL  
SELECT count(*) FROM PERSONAL  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

--- ENCARGADO

```
BEGIN TRANSACTION  
INSERT INTO ENCARGADO  
    SELECT id_personal, 5000 from PERSONAL  
    WHERE tipo = 'E'  
    SELECT * from ENCARGADO  
    SELECT count(*) from ENCARGADO  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

--- GERENTE

```
BEGIN TRANSACTION  
INSERT INTO GERENTE
```



```
SELECT id_personal from PERSONAL
WHERE tipo = 'G'
SELECT * from GERENTE
SELECT count(*) from GERENTE
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- CENTRO

```
-- SELECT id_personal from PERSONAL
-- WHERE tipo = 'G' -- 37,38,39,40
-- SELECT id_personal from PERSONAL
-- WHERE tipo = 'E' -- 16,17,18,19,20,21,22
BEGIN TRANSACTION
INSERT INTO CENTRO_CUIDADO (id_oficinaRegional, colonia, alcaldia, CP, numero,
nombre, calle, id_personal,id_encargado)
VALUES
(NULL, 'Colonia1', 'Alcaldia1', 10001, 'Numero1', 'Centro1', 'Calle1', 37,16),
(1, 'Colonia2', 'Alcaldia2', 10002, 'Numero2', 'Centro2', 'Calle2', NULL,17),
(NULL, 'Colonia3', 'Alcaldia3', 10003, 'Numero3', 'Centro3', 'Calle3', 40,18),
(3, 'Colonia4', 'Alcaldia4', 10004, 'Numero4', 'Centro4', 'Calle4', NULL,18),
(5, 'Colonia5', 'Alcaldia5', 10005, 'Numero5', 'Centro5', 'Calle5', NULL,20),
(NULL, 'Colonia6', 'Alcaldia6', 10006, 'Numero6', 'Centro6', 'Calle6', 38,21),
(NULL, 'Colonia7', 'Alcaldia7', 10007, 'Numero7', 'Centro7', 'Calle7', 39,22);
SELECT * FROM CENTRO_CUIDADO
SELECT count(*) FROM CENTRO_CUIDADO
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- PERSONAL AYUDA

```
-- SELECT id_personal from PERSONAL
-- WHERE tipo = 'H' -- 23,24,25,26,27,28,29,30,31,32
-- select * from CENTRO_CUIDADO -- 1,...,7
BEGIN TRANSACTION
INSERT INTO PERSONAL_AYUDA (id_personal,edad,tipоМascota,id_centro)
VALUES(23,25,'G',1),
(24,32,'P',2),
```



(25,28,'G',3),
(26,40,'P',4),
(27,38,'G',5),
(28,23,'P',6),
(29,41,'P',7),
(30,21,'G',1),
(31,29,'P',2),
(32,30,'G',3)

```
SELECT count(*) FROM PERSONAL_AYUDA
SELECT * FROM PERSONAL_AYUDA
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- VETERINARIO

```
-- SELECT id_personal from PERSONAL
-- WHERE tipo = 'V' --1,...,15
-- select * from CENTRO_CUIDADO -- 1,,,7
BEGIN TRANSACTION
INSERT INTO VETERINARIO (id_personal, cedula, especialidad, id_centro)
VALUES
(1, 'CED001', 'Dermatología', 1),
(2, 'CED002', 'Oftalmología', 2),
(3, 'CED003', 'Ortopedia', 3),
(4, 'CED004', 'Cardiología', 4),
(5, 'CED005', 'Endocrinología', 5),
(6, 'CED006', 'Odontología', 6),
(7, 'CED007', 'Neuroología', 7),
(8, 'CED008', 'Radiología', 1),
(9, 'CED009', 'Anestesiología', 2),
(10, 'CED010', 'Patología', 3),
(11, 'CED011', 'Gastroenterología', 4),
(12, 'CED012', 'Uroología', 5),
(13, 'CED013', 'Hematología', 6),
(14, 'CED014', 'Nutrición', 7),
(15, 'CED015', 'Oncología', 7);
SELECT * FROM VETERINARIO
SELECT COUNT(*) FROM VETERINARIO
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```



--- ADMINISTRATIVOS

BEGIN TRANSACTION
INSERT INTO ADMINISTRATIVO
SELECT id_personal, NULL from PERSONAL
WHERE tipo = 'A'
--ROLLBACK TRANSACTION
COMMIT TRANSACTION

UPDATE ADMINISTRATIVO
SET id_centro = 1 where id_personal = 33
UPDATE ADMINISTRATIVO
SET id_centro = 2 where id_personal = 34
UPDATE ADMINISTRATIVO
SET id_centro = 3 where id_personal = 35
UPDATE ADMINISTRATIVO
SET id_centro = 4 where id_personal = 36
UPDATE ADMINISTRATIVO
SET id_centro = 5 where id_personal = 41
UPDATE ADMINISTRATIVO
SET id_centro = 6 where id_personal = 42
UPDATE ADMINISTRATIVO
SET id_centro = 7 where id_personal = 43
SELECT * from ADMINISTRATIVO
SELECT COUNT(*) from ADMINISTRATIVO

--- USUARIO COMUN

BEGIN TRANSACTION
INSERT INTO USUARIO_COMUN (nombre, apellidoPaterno, apellidoMaterno, CURP,
nombreUsuario, contrasena, genero, telefono, correo, calle, numero, CP, alcaldia, colonia,
numeroTarjeta, vigenciaTarjeta)
VALUES



('Juan', 'Pérez', 'García', 'PEJG890123HDFRNR01', 'juanperez', 'contraseña1', 'masculino', '55555555', 'juanperez@gmail.com', 'Calle 1', '10', 12345, 'Benito Juárez', 'Nápoles', '1234567890123456', '2024-01-01'),
('María', 'González', 'Hernández', 'GOHM890123MDFRRA02', 'mariagonzalez', 'contraseña2', 'femenino', '55555556', 'mariagonzalez@gmail.com', 'Calle 2', '20', 23456, 'Cuauhtémoc', 'Condesa', '2345678901234567', '2025-02-01'),
('Pedro', 'López', 'Sánchez', 'LOSP900123HDFRDR03', 'pedrolopez', 'contraseña3', 'masculino', '55555557', 'pedrolopez@gmail.com', 'Calle 3', '30', 34567, 'Miguel Hidalgo', 'Polanco', '3456789012345678', '2026-03-01'),
('Ana', 'Martínez', 'Gómez', 'MAGO890123MDFRMA04', 'anamartinez', 'contraseña4', 'femenino', '55555558', 'anamartinez@gmail.com', 'Calle 4', '40', 45678, 'Coyoacán', 'Del Carmen', '4567890123456789', '2027-04-01'),
('Jorge', 'Hernández', 'Flores', 'HEFJ870123HDFRJR05', 'jorgehernandez', 'contraseña5', 'masculino', '55555559', 'jorgehernandez@gmail.com', 'Calle 5', '50', 56789, 'Iztapalapa', 'Santa Martha', '5678901234567890', '2028-05-01'),
('Fernanda', 'Díaz', 'Ramírez', 'DIRF910123MDFRNN06', 'fernandadiaz', 'contraseña6', 'femenino', '55555560', 'fernandadiaz@gmail.com', 'Calle 6', '60', 67890, 'Alvaro Obregón', 'San Angel', '6789012345678901', '2029-06-01'),
('Ricardo', 'García', 'Martínez', 'GAMR840123HDFRRC07', 'ricardogarcia', 'contraseña7', 'masculino', '55555561', 'ricardogarcia@gmail.com', 'Calle 7', '70', 78901, 'Azcapotzalco', 'Clavería', '7890123456789012', '2030-07-01'),
('Laura', 'Sánchez', 'López', 'SALL900123MDFRLA08', 'laurasanchez', 'contraseña8', 'femenino', '55555562', 'laurasanchez@gmail.com', 'Calle 8', '80', 89012, 'Gustavo A. Madero', 'Lindavista', '8901234567890123', '2031-08-01'),
('Alejandro', 'Pérez', 'González', 'PEGJ750123HDFRNL09', 'alejandroperez', 'contraseña9', 'masculino', '55555563', 'alejandroperez@gmail.com', 'Calle 9', '90', 90123, 'Iztacalco', 'Agrícola Oriental', '9012345678901234', '2032-09-01'),
('Mariana', 'Hernández', 'Martínez', 'HEMM800123MDFRNR10', 'marianahernandez', 'contraseña10', 'femenino', '55555564', 'marianahernandez@gmail.com', 'Calle 10', '100', 12340, 'Venustiano Carranza', 'Morelos', '0123456789012345', '2033-10-01'),
('José', 'González', 'Pérez', 'GOPJ880123HDFRNS11', 'josegonzalez', 'contraseña11', 'masculino', '55555565', 'josegonzalez@gmail.com', 'Calle 11', '110', 23451, 'Tlalpan', 'San Pedro Mártir', '1234567890123456', '2034-11-01'),
('Sofía', 'López', 'García', 'LOGS920123MDFRFS12', 'sofialopez', 'contraseña12', 'femenino', '55555566', 'sofialopez@gmail.com', 'Calle 12', '120', 34562, 'Xochimilco', 'Barrio 18', '2345678901234567', '2035-12-01');
select * from USUARIO_COMUN
select count(*) from USUARIO_COMUN
--ROLLBACK TRANSACTION
COMMIT TRANSACTION

--- CATEGORIA



```
BEGIN TRANSACTION
INSERT INTO CATEGORIA (categoria)
VALUES
    ('Alimentos y Snacks'),
    ('Juguetes y Entretenimiento'),
    ('Camas y Descanso'),
    ('Higiene y Cuidado');
select * from CATEGORIA
select count(*) from CATEGORIA
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

```
--- PRODUCTO ONLINE
```

```
BEGIN TRANSACTION
INSERT INTO PRODUCTOS_ONLINE (stock, precio, descripcion, id_categoria)
VALUES
    (10, 9.99, 'Alimento seco para perros, 1 kg', 1),
    (20, 5.99, 'Snacks de pollo para gatos, 100 g', 1),
    (18, 12.99, 'Collar ajustable para perros, color rojo', 1),
    (14, 16.99, 'Alimento húmedo para perros, pack de 6 unidades', 1),
    (10, 6.99, 'Snacks naturales para perros, sin aditivos ni conservantes', 1),
    (5, 14.99, 'Pelota de tenis para perros, tamaño mediano', 2),
    (15, 7.99, 'Rascador de cartón para gatos, diseño moderno', 2),
    (9, 8.99, 'Juguete con catnip para gatos, forma de ratón', 2),
    (7, 11.99, 'Juguete interactivo para gatos, con plumas y luces', 2),
    (5, 9.99, 'Ratón de juguete con plumas para gatos, colores surtidos', 2),
    (8, 29.99, 'Cama ortopédica para perros mayores, tamaño grande', 3),
    (12, 19.99, 'Cuna suave para gatos, colores variados', 3),
    (6, 39.99, 'Cama redonda para perros y gatos, tamaño pequeño', 3),
    (3, 54.99, 'Cama elevada para perros, resistente al agua', 3),
    (3, 29.99, 'Hamaca para gatos, fácil de montar en ventanas', 3),
    (30, 3.99, 'Champú para perros, aroma a coco', 4),
    (25, 6.99, 'Arena absorbente para gatos, control de olores', 4),
    (20, 4.99, 'Cepillo para perros y gatos, de cerdas suaves', 4),
    (22, 9.99, 'Cepillo de dientes para perros y gatos, de doble cabezal', 4),
    (15, 8.99, 'Cepillo deshedding para perros y gatos, para eliminar el pelo suelto', 4);
select * from PRODUCTOS_ONLINE
select count(*) from PRODUCTOS_ONLINE
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```



--- OFERTAS

BEGIN TRANSACTION

INSERT INTO OFERTAS_O (tipo, descripcion, Finicio, Ffin)

VALUES

('N', 'Oferta especial: 20% de descuento en alimentos secos para perros', '2023-06-06', '2023-07-06'),

('L', 'Liquidación: Últimas unidades de juguetes para gatos', '2023-06-08', '2023-06-15'),

('N', 'Promoción: Compra 2 camas para perros y obtén un descuento del 15%', '2023-06-12', '2023-07-12'),

('N', 'Oferta relámpago: Snacks para gatos al 50% de descuento', '2023-06-15', '2023-06-16'),

('L', 'Gran liquidación: Todos los productos de higiene con descuentos increíbles', '2023-06-18', '2023-06-25'),

('N', 'Oferta destacada: Collares y correas para perros con envío gratis', '2023-06-20', '2023-07-20'),

('L', 'Remate final: Precios bajos en camas y cojines para gatos', '2023-06-22', '2023-06-29'),

('N', 'Promoción especial: Compra 3 juguetes para perros y lleva el cuarto gratis', '2023-06-25', '2023-07-25');

select * from OFERTAS_O

select count(*) from OFERTAS_O

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- RELACIÓN MUCHOS A MUCHOS (CONTARÁ)

BEGIN TRANSACTION

INSERT INTO CONTARA (id_producto, id_oferta)

VALUES

(1, 1),

(2, 3),

(5, 2),

(8, 4),

(10, 5);

select * from CONTARA

select count(*) from CONTARA

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- CANASTA



BEGIN TRANSACTION

INSERT INTO CANASTA (fecha, tarifaEnvio, total, tarifaCancelacion, id_UsuarioComun)
VALUES

('10:00:00.0000000', 100.00, 150.00, 22.50, 3),
(12:30:00.0000000', 100.00, 250.00, 37.50, 8),
(15:45:00.0000000', 100.00, 180.00, 27.00, 5),
(17:20:00.0000000', 100.00, 320.00, 48.00, 10),
(20:15:00.0000000', 100.00, 200.00, 30.00, 2);

select * from CANASTA

select count(*) from CANASTA

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- CANASTA PRODUCTO

BEGIN TRANSACTION

INSERT INTO CANASTA_PRODUCTO (id_canasta, id_producto, num_Uni_Compra)

VALUES

(1, 1, 2),
(1, 3, 1),
(2, 2, 3),
(2, 4, 2),
(2, 5, 1),
(3, 1, 1),
(3, 4, 4),
(4, 3, 2),
(4, 5, 3),
(5, 2, 1),
(5, 3, 1),
(5, 5, 2);

select * from CANASTA_PRODUCTO

select count(*) from CANASTA_PRODUCTO

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- PRODUCTOS FISICOS

BEGIN TRANSACTION

INSERT INTO PRODUCTOS_FISICOS (costo, stock)



VALUES

```
(29.99, 10),  
(19.99, 5),  
(39.99, 20),  
(9.99, 15),  
(14.99, 8),  
(24.99, 30),  
(12.99, 12),  
(34.99, 7),  
(8.99, 18),  
(17.99, 3);
```

```
select * from PRODUCTOS_FISICOS  
select count(*) from PRODUCTOS_FISICOS  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

-- OFERTAS FISICOS

BEGIN TRANSACTION

```
INSERT INTO OFERTAS_FISICOS (tipo, descripcion, Finicio, Ffin)  
VALUES
```

```
('N', 'Descuento del 20% en collares y correas para perros', '2023-06-13', '2023-07-23'),  
(L', 'Oferta 3x2 en productos de higiene para gatos', '2023-06-14', '2023-07-24'),  
(N', 'Promoción: Compra una cama para perros y llévate una manta gratis', '2023-06-15',  
'2023-07-25'),  
(L', 'Descuento del 10% en juguetes interactivos para gatos', '2023-06-16', '2023-07-26'),  
(N', 'Oferta especial: Compra un producto de salud para perros y recibe un regalo sorpresa',  
'2023-06-17', '2023-07-27'),  
(L', 'Envío gratis en todos los productos físicos para gatos', '2023-06-18', '2023-07-28'),  
(N', 'Descuento del 15% en camas ortopédicas para perros mayores', '2023-06-19', '2023-07-  
29');
```

```
select * from OFERTAS_FISICOS  
select count(*) from OFERTAS_FISICOS  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

-- OFERTA RODUCTOSFISICOS

BEGIN TRANSACTION

```
INSERT INTO OFERTA_PRODUCTOSFISICOS (id_producto, id_oferta)  
VALUES
```



(1, 1),
(3, 2),
(6, 3),
(9, 4),
(10, 5);

```
select * from OFERTA_PRODUCTOSFISICOS
select count(*) from OFERTA_PRODUCTOSFISICOS
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- RAZA

```
BEGIN TRANSACTION
INSERT INTO RAZA (nombreRaza)
VALUES ('Golden Retriever'),
('Siamese'),
('Labrador Retriever'),
('Persian'),
('Bulldog'),
('Maine Coon'),
('Beagle'),
('Sphynx'),
('Poodle'),
('Ragdoll'),
('German Shepherd'),
('British Shorthair'),
('French Bulldog'),
('Scottish Fold'),
('Chihuahua');
```

```
select * from RAZA
select count(*) from RAZA
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- ESPECIE

```
BEGIN TRANSACTION
INSERT INTO ESPECIE (tipo, id_raza)
VALUES
('P', 1), -- Golden Retriever (Perro)
('G', 2), -- Siamese (Gato)
```



('P', 3), -- Labrador Retriever (Perro)

('G', 3), -- Labrador Retriever (Gato)

('P', 5), -- Bulldog (Perro)

('G', 4), -- Persian (Gato)

('P', 6), -- Maine Coon (Perro)

('G', 6), -- Maine Coon (Gato)

('P', 8), -- Poodle (Perro)

('G', 7); -- Beagle (Gato)

select * from ESPECIE

select count(*) from ESPECIE

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- MASCOTA

BEGIN TRANSACTION

INSERT INTO MASCOTA (nombre, sexo, edad, rasgosCaracteristicos, id_especie, id_UsuarioComun)

VALUES

('Max', 'M', 3, 'Juguetón y cariñoso', 1, 1),

('Luna', 'F', 2, 'Curiosa y tranquila', 2, 2),

('Rocky', 'M', 5, 'Energético y protector', 1, 3),

('Coco', 'M', 4, 'Amigable y leal', 3, 4),

('Bella', 'F', 1, 'Juguetona y cariñosa', 2, 5),

('Charlie', 'M', 2, 'Aventurero y obediente', 4, 6),

('Lola', 'F', 3, 'Sociable y juguetona', 6, 7),

('Simba', 'M', 4, 'Curioso y valiente', 5, 8),

('Mia', 'F', 2, 'Cariñosa y tranquila', 7, 9),

('Rocky', 'M', 1, 'Divertido y activo', 4, 10);

SELECT * FROM MASCOTA

SELECT count(*) FROM MASCOTA

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- BRAZALETE

BEGIN TRANSACTION

INSERT INTO BRAZALETE (comio, comida, cuantoComio, medicamento, cuidadoEspecial)

VALUES (1, 'Croquetas', 'Comió todo el plato', 0, 1),

(1, 'Lata de atún', 'Comió la mitad', 0, 0),

(0, 'Pollo cocido', 'No comió nada', 1, 1),



```
(1, 'Pienso seco', 'Comió una porción', 0, 0),
(1, 'Filete de salmón', 'Comió todo', 1, 0),
(0, 'Comida enlatada', 'No comió nada', 0, 1),
(1, 'Pavo asado', 'Comió la mitad', 1, 0),
(1, 'Pollo a la parrilla', 'Comió todo', 0, 0),
(1, 'Hamburguesa', 'Comió una porción', 1, 1),
(0, 'Vegetales hervidos', 'No comió nada', 0, 0);
select * from BRAZALETE
select count(*) from BRAZALETE
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- VITALES

BEGIN TRANSACTION

```
INSERT INTO VITALES (ritmoCardiaco, temperatura, nivelOxigeno, fechaHora, id_brazalete)
VALUES (80, 37.5, 98.5, '2023-06-06 08:00:00', 1),
(72, 36.9, 97.8, '2023-06-06 09:15:00', 2),
(90, 38.2, 98.9, '2023-06-06 10:30:00', 3),
(76, 37.1, 97.2, '2023-06-06 11:45:00', 4),
(85, 37.8, 98.3, '2023-06-06 13:00:00', 5),
(78, 37.2, 97.5, '2023-06-06 14:15:00', 6),
(95, 38.5, 99.1, '2023-06-06 15:30:00', 7),
(82, 37.6, 98.0, '2023-06-06 16:45:00', 8),
(88, 38.0, 98.7, '2023-06-06 18:00:00', 9),
(70, 36.7, 97.9, '2023-06-06 19:15:00', 10);
```

select * from VITALES

select count(*) from VITALES

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- MEDICAMENTO

BEGIN TRANSACTION

```
INSERT INTO MEDICAMENTO (nombre, costo)
VALUES
('Paracetamol', 10.50),
('Ibuprofeno', 12.75),
('Amoxicilina', 15.20),
('Omeprazol', 8.90),
```



```
('Dipirona', 9.50),  
('Loratadina', 7.80),  
('Cetirizina', 11.30),  
('Aspirina', 6.50),  
('Metformina', 13.40),  
('Atorvastatina', 18.60),  
('Insulina', 21.80),  
('Dexametasona', 14.90);  
select * from MEDICAMENTO  
select count(*) from MEDICAMENTO  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

```
-- TELEFONOS PERSONAL AYUDA
```

```
-- SELECT *from PERSONAL_AYUDA
```

```
BEGIN TRANSACTION  
INSERT INTO TELEFONOS_PERSONAL_AYUDA (id_personal, numTelefono)  
VALUES  
(23, '555-111-1111'),  
(24, '555-222-2222'),  
(25, '555-333-3333'),  
(26, '555-444-4444'),  
(27, '555-555-5555'),  
(28, '555-666-6666'),  
(29, '555-777-7777'),  
(30, '555-888-8888'),  
(31, '555-999-9999'),  
(32, '555-000-0000');  
select * from TELEFONOS_PERSONAL_AYUDA  
select count(*) from TELEFONOS_PERSONAL_AYUDA  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```



---BRAZALETE MASCOTA

```
-- exec sp_help[BRAZALETE_MASCOTA]
-- select * from MASCOTA
-- SELECT * FROM BRAZALETE
-- select * from PERSONAL_AYUDA
DBCC CHECKIDENT('BRAZALETE_MASCOTA',reseed,0)
BEGIN TRANSACTION
    INSERT INTO BRAZALETE_MASCOTA (id_mascota, id_personal, id_brazalete, estatus)
    VALUES
        ( 1, 23, 1,'I'),
        ( 2, 23, 2,'I'),
        ( 3, 24, 3,'I'),
        ( 4, 23, 4,'I'),
        ( 5, 23, 5,'I'),
        ( 6, 23, 6,'I'),
        ( 7, 24, 7,'I'),
        ( 8, 25, 8,'I'),
        ( 9, 32, 9,'I'),
        ( 10, 31, 10,'I'),
        ( 1, 31, 1,'A'),
        ( 2, 31, 2,'A'),
        ( 3, 31, 3,'A'),
        ( 4, 25, 4,'A'),
        ( 5, 28, 5,'A'),
        ( 6, 31, 6,'A'),
        ( 7, 31, 7,'A'),
        ( 8, 26, 8,'A'),
        ( 9, 24, 9,'I'),
        ( 9, 26, 10,'A');
    select * from BRAZALETE_MASCOTA
    select count(*) from BRAZALETE_MASCOTA
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
DELETE FROM BRAZALETE_MASCOTA
```


---RESERVACION_GUARDERIA

```
-- exec sp_help[RESERVACION_GUARDERIA]
-- SELECT * FROM USUARIO_COMUN --12 USUARIOS
```



```
-- SELECT * FROM CENTRO_CUIDADO -- 7
BEGIN TRANSACTION
    INSERT INTO RESERVACION_GUARDERIA (finicio, ffin, costoXDia, id_centro,
    id_UsuarioComun)
    VALUES
        ('2023-06-01', '2023-06-03', 100, 1, 1),
        ('2023-06-02', '2023-06-05', 150, 2, 2),
        ('2023-06-03', '2023-06-06', 120, 1, 3),
        ('2023-06-04', '2023-06-07', 110, 3, 4),
        ('2023-06-05', '2023-06-08', 130, 2, 5),
        ('2023-06-06', '2023-06-09', 140, 4, 6),
        ('2023-06-07', '2023-06-10', 125, 5, 7),
        ('2023-06-08', '2023-06-11', 135, 2, 8),
        ('2023-06-09', '2023-06-12', 105, 4, 9),
        ('2023-06-10', '2023-06-13', 115, 3, 10),
        ('2023-06-11', '2023-06-14', 145, 6, 11),
        ('2023-06-12', '2023-06-15', 125, 7, 12),
        ('2023-06-13', '2023-06-16', 135, 2, 1),
        ('2023-06-14', '2023-06-17', 130, 7, 2),
        ('2023-06-15', '2023-06-18', 140, 7, 3),
        ('2023-06-16', '2023-06-19', 120, 2, 4),
        ('2023-06-17', '2023-06-20', 110, 6, 5),
        ('2023-06-18', '2023-06-21', 150, 5, 5),
        ('2023-06-19', '2023-06-22', 125, 2, 7),
        ('2023-06-20', '2023-06-23', 130, 4, 8);
    SELECT * FROM RESERVACION_GUARDERIA
    SELECT count(*) FROM RESERVACION_GUARDERIA
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- CUENTA

```
--exec sp_help[CUENTA]
--SELECT * FROM CENTRO_CUIDADO -- 7 centros
--SELECT * FROM PRODUCTOS_FISICOS -- 10 productos
BEGIN TRANSACTION
    INSERT INTO CUENTA (id_centro, id_producto)
    VALUES
        (1, 1),
        (2, 2),
        (3, 3),
```



(4, 4),
(5, 5),
(6, 6),
(7, 7),
(1, 8),
(5, 9),
(7, 10);

```
SELECT * FROM CUENTA  
SELECT count(*) FROM CUENTA  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

--- DESGLOSE VENTA

```
--exec sp_help[DESGLOSE_VENTA]  
BEGIN TRANSACTION  
    INSERT INTO DESGLOSE_VENTA (num_Unidades, montoParcial, id_centro, id_producto)  
    VALUES  
        (5, 500.00, 1, 1),  
        (3, 300.00, 2, 2),  
        (2, 200.00, 3, 3),  
        (4, 400.00, 4, 4),  
        (6, 600.00, 5, 5),  
        (1, 100.00, 6, 6),  
        (7, 700.00, 7, 7),  
        (9, 900.00, 1, 8),  
        (8, 800.00, 5, 9),  
        (10, 1000.00, 7, 10);  
    SELECT * FROM DESGLOSE_VENTA  
    SELECT count(*) FROM DESGLOSE_VENTA  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

--- COMPRA FISICA

```
--exec sp_help[COMPRA_FISICA]  
--SELECT * FROM DESGLOSE_VENTA -- Hay máximo 10  
--SELECT * FROM ENCARGADO -- del 16 al 22
```



BEGIN TRANSACTION

INSERT INTO COMPRA_FISICA (total, id_personal, id_desgloseVenta)
VALUES

(1000, 16, 1),
(1500, 17, 2),
(500, 18, 3),
(800, 19, 4),
(2000, 20, 5),
(300, 21, 6),
(600, 22, 7),
(1200, 16, 8),
(2600, 22, 9),
(900, 17, 10),
(400, 19, 1),
(200, 20, 2),
(700, 21, 3),
(1100, 17, 4);

SELECT * FROM COMPRA_FISICA

SELECT count(*) FROM COMPRA_FISICA

--ROLLBACK TRANSACTION

COMMIT TRANSACTION

--- REPORTE

BEGIN TRANSACTION

INSERT INTO REPORTE (diagnostico, detallesExaminacion, fechaHora, id_personal)
VALUES

('Gato con fiebre', 'El gato presenta fiebre alta y falta de apetito.', '2023-06-01 10:30:00', 1),
('Perro con fractura', 'El perro tiene una fractura en la pata trasera derecha.', '2023-06-02
15:45:00', 2),

('Gato con problemas urinarios', 'El gato muestra dificultad para orinar y se lame
constantemente.', '2023-06-03 11:20:00', 3),

('Perro con infección en el oído', 'El perro presenta secreción y mal olor en el oído derecho.',
'2023-06-04 14:10:00', 4),

('Gato con vómitos', 'El gato ha vomitado varias veces en las últimas horas.', '2023-06-05
09:55:00', 5),

('Perro con picadura de insecto', 'El perro tiene una reacción alérgica a una picadura de insecto
en el hocico.', '2023-06-06 13:00:00', 6),

('Gato con herida en la pata', 'El gato tiene una herida profunda en la pata delantera izquierda.',
'2023-06-06 16:30:00', 7);

SELECT * FROM REPORTE



```
SELECT count(*) FROM REPORTE  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

```
--- REPORTE MEDICAMENTO
```

```
BEGIN TRANSACTION  
INSERT INTO REPORTE_MEDICAMENTO (id_reporte, id_medicamento, dosis)  
VALUES  
(1, 1, '1 comprimido cada 8 horas'),  
(2, 3, '1 cápsula cada 12 horas'),  
(3, 2, '2 tabletas al día'),  
(4, 4, '5 gotas en el oído afectado cada 6 horas'),  
(5, 6, '1 tableta al día'),  
(6, 5, '1 inyección intravenosa cada 24 horas'),  
(7, 8, '1 tableta cada 6 horas');  
SELECT * FROM REPORTE  
SELECT count(*) FROM REPORTE  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

```
--- RECIBO
```

```
BEGIN TRANSACTION  
INSERT INTO RECIBO (costoTotal, tratamiento, examinacion, id_reporte, id_UsuarioComun)  
VALUES  
(150.00, 100.00, 'Vacunación', 1, 1),  
(500.00, 300.00, 'Radiografía', 2, 2),  
(200.00, 150.00, 'Desparasitación', 3, 3),  
(1000.00, 800.00, 'Limpieza dental', 4, 4),  
(250.00, 200.00, 'Análisis de sangre', 5, 5),  
(80.00, 50.00, 'Corte de uñas', 6, 6),  
(300.00, 250.00, 'Baño y cepillado', 7, 7);  
SELECT * FROM RECIBO  
SELECT count(*) FROM RECIBO  
--ROLLBACK TRANSACTION  
COMMIT TRANSACTION
```

```
--- ALMACEN
```



```
BEGIN TRANSACTION
INSERT INTO ALMACEN (id_centro, id_medicamento, cantidad)
VALUES
    (1, 1, 50), -- Paracetamol
    (1, 2, 30), -- Ibuprofeno
    (1, 3, 40), -- Amoxicilina
    (1, 4, 25), -- Omeprazol
    (1, 5, 20), -- Dipirona
    (2, 6, 35), -- Loratadina
    (2, 3, 25), -- Loratadina***
    (2, 7, 45), -- Cetirizina
    (2, 8, 60), -- Aspirina
    (2, 9, 55), -- Metformina
    (3, 10, 15), -- Atorvastatina
    (3, 2, 5), -- Atorvastatina**
    (3, 11, 10), -- Insulina
    (3, 12, 5), -- Dexametasona
    (4, 4, 35), -- Dexametasona**
    (5, 6, 35), -- Dexametasona**
    (6, 5, 55), -- Dexametasona**
    (7, 8, 20); -- Dipirona**
SELECT * FROM ALMACEN
SELECT count(*) FROM ALMACEN
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

--- AÑADE

```
BEGIN TRANSACTION
INSERT INTO AÑADE (id_personal, id_producto, nombreProducto, cantidad)
VALUES
    (33, 1, 'Alimento seco para perros, 1 kg', 5),
    (33, 2, 'Snacks de pollo para gatos, 100 g', 10),
    (34, 3, 'Collar ajustable para perros, color rojo', 3),
    (34, 4, 'Alimento húmedo para perros, pack de 6 unidades', 8),
    (35, 5, 'Snacks naturales para perros', 4),
    (35, 6, 'Pelota de tenis para perros, tamaño mediano', 2),
    (36, 7, 'Rascador de cartón para gatos, diseño moderno', 5),
    (36, 8, 'Juguete con catnip para gatos, forma de ratón', 3),
    (41, 9, 'Juguete interactivo para gatos, con plumas y luces', 2),
    (41, 10, 'Ratón de juguete con plumas para gatos', 1),
```



```
(42, 11, 'Cama ortopédica para perros mayores, tamaño grande', 3),
(42, 12, 'Cuna suave para gatos, colores variados', 5),
(43, 13, 'Cama redonda para perros y gatos, tamaño pequeño', 2),
(43, 14, 'Cama elevada para perros, resistente al agua', 1),
(33, 15, 'Hamaca para gatos, fácil de montar en ventanas', 2),
(33, 16, 'Champú para perros, aroma a coco', 6),
(34, 17, 'Arena absorbente para gatos, control de olores', 4),
(34, 18, 'Cepillo para perros y gatos, de cerdas suaves', 3),
(35, 19, 'Cepillo de dientes para perros y gatos', 2),
(35, 20, 'Cepillo deshedding para perros y gatos', 5);
SELECT * FROM AÑADE
SELECT count(*) FROM AÑADE
--ROLLBACK TRANSACTION
COMMIT TRANSACTION
```

f) Para cada uno de los triggers creados se deberá realizar un script SQL (validaTriggers.sql) que ejecute un escenario para validar el correcto funcionamiento del trigger. El script deberá ser lo más automatizado posible y evitar la intervención manual.

```
/*
SCRIPT PARA LA VALIDACION DE TRIGGERS
*/
---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
---          SUELDO MAYOR A 5000
---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

--$$$$$$$$$$$$$ VERIFICACION $$$$$$$
-- ERROR SUELDO MENOR A 5000
BEGIN TRAN
select 'Marca error sueldo menor a 3000' as mensaje
INSERT INTO ENCARGADO
VALUES (1,3000)
ROLLBACK

--SIN ERROR SUELDO MAYOR A 5000
BEGIN TRAN
select 'No marca error sueldo mayor a 5000' as mensaje
INSERT INTO ENCARGADO
VALUES (1,8000)
ROLLBACK
```



```
---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  
---VALIDA JERARQUIA TIPOS  
---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
--$$$$$$$$$$ PRUEBAS$$$$$  
--no nos permite porque ya existe ese CURP  
select 'Marca error curp existente' as mensaje  
INSERT INTO PERSONAL (tipo, nombre, usuario, contrasena, CURP, calle, numero, CP,  
alcaldia, colonia)  
VALUES  
('V', 'Juanita Sánchez', 'juanitasanchez', '987654', 'ASDFG987654ZXCVBNM', 'Calle21',  
'Numero21', 10021, 'Álvaro Obregón', 'San Ángel')  
--No permite porque no existe ese tipo  
select 'Marca error tipo S no valido' as mensaje  
INSERT INTO PERSONAL (tipo, nombre, usuario, contrasena, CURP, calle, numero, CP,  
alcaldia, colonia)  
VALUES  
('S', 'Juanita Sánchez', 'juanitasanchez', '987654', 'ASDMMM87654ZXCVBNM', 'Calle21',  
'Numero21', 10021, 'Álvaro Obregón', 'San Ángel')
```

```
--ERRRO TIPO S NO EXISTE  
select 'Marca error tipo S no valido' as mensaje  
UPDATE PERSONAL  
SET tipo = 'S' where id_personal= 4  
--SI DEJA HACER EL CAMBIO  
BEGIN TRAN  
select 'no error tipo valido' as mensaje  
UPDATE PERSONAL  
SET CURP = 'H' where id_personal= 4  
ROLLBACK
```

```
---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  
---OFERTAS NO MAYOR A 40 DIAS  
---$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
--$$$$$ PRUEBAS$$$$$  
--ERROR OFERTA DURA MÁS DE 40 DIAS  
BEGIN TRAN  
select 'Marca erroroferta mas de 40' as mensaje  
INSERT INTO OFERTAS_FISICOS (tipo, descripcion, Finicio, Ffin)
```



VALUES

('N', 'Descuento del 20% en collares y correas para perros', '2023-06-13', '2023-10-23')
ROLLBACK

--NO HAY ERROR OFERTA NO DURA MAS DE 40

BEGIN TRAN

select 'no marca error oferta menos de 40' as mensaje
INSERT INTO OFERTAS_FISICOS (tipo, descripcion, Finicio, Ffin)
VALUES

('N', 'Descuento del 20% en collares y correas para perros', '2023-06-13', '2023-7-23')
ROLLBACK

--ERROR LA OFERTA DURARÁ MÁS DE 40 DÍAS

BEGIN TRAN

select 'Marca error si actualiza oferta mas de 40' as mensaje
UPDATE OFERTAS_FISICOS
SET Ffin = '2023-10-23'
WHERE id_oferta=1
ROLLBACK

--SIN ERROR LA OFERTA NO DURA MÁS DE 40 DÍAS

BEGIN TRAN

select 'no marca error oferta no dura mas de 40' as mensaje
UPDATE OFERTAS_FISICOS
SET Ffin = '2023-07-21'
WHERE id_oferta=1
ROLLBACK

---\$

--- COMISION POR CADA VENTA

---\$

--CALCULO DE LA COMISION SE HACE DESDE EL CHECK DE LA TABLA

--POR CADA VENTA ACTUALIZA EL SALARIO DEL ENCARGADO

--\$\$\$\$\$ PRUEBAS\$\$\$\$\$

-- Al insertar una venta al encargado id = 16, se actualiza su sueldo base
BEGIN TRAN

select 'Venta para encargado 16 -> modifica sueldo' as mensaje
select * from ENCARGADO where id_personal=16
INSERT INTO COMPRA_FISICA (total, id_personal, id_desgloseVenta)
VALUES (1000, 16, 10)



```
select * from ENCARGADO where id_personal=16
ROLLBACK
```

```
--- $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  
--- ESTANDARES Y NOMENCLATURA  
--- $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
--ERROR PORQUE EL RFC TIENE CARACTER ESPECIAL TRIGGER JERARQUIA
select 'Marca error caracter invalido en curp' as mensaje
```

```
BEGIN TRAN
```

```
INSERT INTO PERSONAL (tipo, nombre, usuario, contrasena, CURP, calle, numero, CP,
alcaldia, colonia)
```

```
VALUES
```

```
('V', 'Juanita Sánchez', 'juanitasanchez', '987654', 'ASDFG98%654ZXCVBNM', 'Calle21',
'Número21', 10021, 'Álvaro Obregón', 'San Ángel')
```

```
ROLLBACK
```

```
--ERROR PORQUE EL CORREO ES INVALIDO
```

```
BEGIN TRAN
```

```
select 'Marca error correo invalido' as mensaje
```

```
INSERT INTO USUARIO_COMUN (nombre, apellidoPaterno, apellidoMaterno, CURP,
nombreUsuario, contrasena,
```

```
genero, telefono, correo, calle, numero, CP, alcaldia, colonia, numeroTarjeta, vigenciaTarjeta)
VALUES
```

```
('Carlos', 'Pérez', 'Magnum', 'MEHH890123HDFRNR01', 'carlos', 'contraseña1', 'masculino',
'55555555',
```

```
'carlosp@gmail.com', 'Calle 1', '10', 12345, 'Benito Juárez', 'Nápoles', '1234567890123456', '2024-
01-01')
```

```
ROLLBACK
```

```
--NO HAY ERROR CORREO VALIDO
```

```
BEGIN TRAN
```

```
select 'Sin error' as mensaje
```

```
INSERT INTO USUARIO_COMUN (nombre, apellidoPaterno, apellidoMaterno, CURP,
nombreUsuario, contrasena,
```

```
genero, telefono, correo, calle, numero, CP, alcaldia, colonia, numeroTarjeta, vigenciaTarjeta)
VALUES
```

```
('Carlos', 'Pérez', 'Magnum', 'MEHH890123HDFRNR01', 'carlos', 'contraseña1', 'masculino',
'55555555',
```

```
'carlosp@gmail.com', 'Calle 1', '10', 12345, 'Benito Juárez', 'Nápoles', '1234567890123456',
'2024-01-01')
```

```
ROLLBACK
```

```
--- $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```



--- CANCELACION 15%

--- \$

--VALIDADO EN EL CHECK DE LA TABLA