*// creacion de tablas*

CREATE TABLE PROVEEDOR(

razon\_proveedor varchar(60) NOT **NULL**,

apellido\_p VARCHAR(40) NOT **NULL**,

apellido\_m VARCHAR(40) **NULL**,

nombre\_pila VARCHAR(60) NOT **NULL**,

colonia varchar(40) NOT **NULL**,

estado varchar(40) NOT **NULL**,

calle varchar(40) NOT **NULL**,

cp varchar(5) NOT **NULL**,

numero **int** NOT **NULL** check(numero>0),

CONSTRAINT razon\_proveedor\_PK PRIMARY KEY (razon\_proveedor)

);

CREATE TABLE TELEFONOS(

razon\_proveedor varchar(60) NOT **NULL**,

telefono varchar(10) NOT **NULL** ,

CONSTRAINT TELEFONO\_PK PRIMARY KEY(razon\_proveedor,telefono),

CONSTRAINT PROVEEDOR\_FK FOREIGN KEY (razon\_proveedor) REFERENCES PROVEEDOR (razon\_proveedor)

);

CREATE TABLE INVENTARIO(

codigo\_barras varchar(13) NOT **NULL** ,

fecha\_compra date NOT **NULL** **default** now(),

precio\_compra DECIMAL(6,2) NOT **NULL** CHECK (precio\_compra>0),

unidades\_stock **int** NOT **NULL** CHECK(unidades\_stock>=0) ,

CONSTRAINT codigo\_barras\_PK PRIMARY KEY (codigo\_barras)

);

CREATE TABLE PRODUCTO (

id\_producto **int** GENERATED ALWAYS AS IDENTITY NOT **NULL**,

nombre VARCHAR(40) NOT **NULL**,

precio\_venta DECIMAL(6,2) NOT **NULL** CHECK(precio\_venta>0),

marca VARCHAR(40) NOT **NULL**,

descripcion varchar(50) **NULL**,

codigo\_barras varchar(13) NOT **NULL**,

utilidad DECIMAL(6,2) NOT **NULL** ,

razon\_proveedor varchar(60) NOT **NULL**,

CONSTRAINT id\_producto\_PK PRIMARY KEY (id\_producto),

CONSTRAINT INVENTARIO\_FK FOREIGN KEY (codigo\_barras) REFERENCES INVENTARIO(codigo\_barras) ON DELETE CASCADE,

CONSTRAINT PROVEEDOR\_PRODUCTO\_FK FOREIGN KEY (razon\_proveedor) REFERENCES PROVEEDOR (razon\_proveedor)

);

CREATE TABLE CLIENTE(

razon\_cliente varchar(13) NOT **NULL**,

apellido\_p VARCHAR(40) NOT **NULL**,

apellido\_m VARCHAR(40) **NULL**,

nombre\_pila VARCHAR(60) NOT **NULL**,

colonia varchar(40) NOT **NULL**,

estado varchar(40) NOT **NULL**,

calle varchar(40) NOT **NULL**,

cp varchar(5) NOT **NULL**,

numero **int** CHECK (numero>0),

CONSTRAINT razon\_cliente\_PK PRIMARY KEY (razon\_cliente)

);

CREATE TABLE EMAIL(

razon\_cliente varchar(13) NOT **NULL**,

email varchar(100) NOT **NULL**,

CONSTRAINT EMAIL\_PK PRIMARY KEY(razon\_cliente,email),

CONSTRAINT CLIENTE\_FK FOREIGN KEY (razon\_cliente) REFERENCES CLIENTE (razon\_cliente)

);

CREATE TABLE ORDEN\_VENTA(

No\_venta **int** GENERATED ALWAYS AS IDENTITY NOT **NULL**,

fecha\_venta date NOT **NULL**,

razon\_cliente varchar(13) NOT **NULL**,

nota\_venta VARCHAR(10) null **default** 'VENT-',

CONSTRAINT ORDEN\_VENTA\_PK PRIMARY KEY (No\_venta),

CONSTRAINT CLIENTE\_FK\_VENTA FOREIGN KEY (razon\_cliente) REFERENCES CLIENTE(razon\_cliente)

);

CREATE TABLE ORDEN\_DETALLE(

No\_orden\_detalle **int** GENERATED ALWAYS AS IDENTITY not null,

No\_venta **int** NOT **NULL**,

id\_producto **int** NOT **NULL**,

cantidad\_articulo **int** NOT **NULL**,

precio\_venta\_producto DECIMAL(6,2) not null,

total\_pagar DECIMAL(7,2) NOT **NULL**,

CONSTRAINT ORDEN\_VENTA\_\_DETALLE\_PK PRIMARY KEY (No\_venta,id\_producto),

CONSTRAINT PRODUCTO\_orden\_FK FOREIGN KEY (id\_producto) REFERENCES PRODUCTO(id\_producto) on delete

CASCADE,

CONSTRAINT Orden\_orden\_detalle\_FK FOREIGN KEY (No\_venta) REFERENCES ORDEN\_VENTA(No\_venta)on delete CASCADE

);

*//funcion si da codigo de barras devuelve el nombre del producto*

CREATE OR REPLACE FUNCTION nombre\_si\_da\_codigo(codigo varchar(13))

RETURNS SETOF PRODUCTO AS $$

BEGIN

RETURN QUERY SELECT \* FROM PRODUCTO where codigo\_barras=codigo;

END

$$ LANGUAGE plpgsql;

*//reincia2() es una funcion para reiniciar el campo id\_producto ya que es identity y si hay un error sigue incrementado por eso reiniciamos*

CREATE OR REPLACE FUNCTION reinicia2()

RETURNS **void** AS $$

declare maxid **int**;

BEGIN

begin

select max(id\_producto)+1 from PRODUCTO into maxid;

execute 'alter SEQUENCE producto\_id\_producto\_seq RESTART with '|| 1;

end;

END

$$ LANGUAGE plpgsql;

*//esta funcion codigo \_repetido es para producto en el inventario si inserta un codigo que ya estaba lo deja insertar luego lo borra*

*//pudimos haber hecho que se abortara la a transaccion de insertar pero debido a que al abortar sigue incrementando el campo*

*//id\_producto ya que es identity y se incrementa solo, y si le damos antes que se vuelva actualizar como se aborto*

*//todas las cosas que hicimos antes no las tomaba en cuenta por eso dejo que inserte y lo borro tomando el maximo o ultimo registro que se inserto*

*//deacuerdo al id\_producto*

CREATE OR REPLACE FUNCTION verifica\_codigo\_repetido()

RETURNS trigger AS

$$

declare maxid **int**;

begin

**if** exists((SELECT codigo\_barras,razon\_proveedor FROM PRODUCTO group by codigo\_barras,razon\_proveedor having count(\*)>1))then

raise notice 'tienes un producto con el mismo codigo y proveedor';

select (max(id\_producto)-1) from PRODUCTO into maxid;

execute 'alter SEQUENCE producto\_id\_producto\_seq RESTART with '|| maxid;

delete from PRODUCTO where id\_producto=(select (max(id\_producto)) from PRODUCTO);

**return** new;

**else**

select max(id\_producto)+1 from PRODUCTO into maxid;

execute 'alter SEQUENCE producto\_id\_producto\_seq RESTART with '|| maxid;

UPDATE PRODUCTO

set utilidad=(select precio\_venta from PRODUCTO group by precio\_venta having max(id\_producto)=(select max(id\_producto) from PRODUCTO))-

(select precio\_compra from INVENTARIO

where codigo\_barras=(select codigo\_barras from PRODUCTO group by codigo\_barras having max(id\_producto)=(select max(id\_producto) from PRODUCTO)))

where id\_producto=(select max(id\_producto) from PRODUCTO);

raise notice 'se inserto correctamente';

**return** new;

end **if**;

END;

$$

LANGUAGE plpgsql;

*//creamos el trigger sirve para ejecutar la funcion y decir cuando se hace el insert*

*//en este caso el insert after significaque vamos hacer algo cuando ya se inserto*

CREATE TRIGGER after\_trigger

after insert

ON PRODUCTO

FOR EACH ROW

EXECUTE PROCEDURE verifica\_codigo\_repetido();

*//verifica\_orden\_venta es una funcion para concatenar y dar formato al campo nota\_venta que es el formato que pidio el profesor 'VENT-001'*

*//Concatenamos 'VENT-' con el No\_venta o id\_venta que hay en ese momento el right para hacer 000 conel No\_venta si es 1 es 001 right hacia el no\_Venta*

*//si es 000 y el no\_venta es 20 entonces hace right y es 010*

CREATE OR REPLACE FUNCTION verifica\_orden\_venta()

RETURNS trigger AS

$$

begin

**if**(exists(select No\_venta from ORDEN\_VENTA)) then

new.nota\_venta=concat(new.nota\_venta,(RIGHT((concat('000' , CAST(new.No\_venta AS VARCHAR(3)))),3)));

**return** new;

**else**

new.nota\_venta=concat(new.nota\_venta,(RIGHT((concat('000' , CAST(new.No\_venta AS VARCHAR(3)))),3)));

**return** new;

end **if**;

END;

$$

LANGUAGE plpgsql;

*// creamos el trriger antes de que haga la insersion en la tabla orden para que le de el formato antes de*

*//insertar*

CREATE TRIGGER before\_trigger\_orden

before insert

ON ORDEN\_VENTA

FOR EACH ROW

EXECUTE PROCEDURE verifica\_orden\_venta();

*//funcion verifica codigo repetido solo es para pruebas, por si se llegara a necesitar*

*// si existe un id\_porducto pues empieza ya la insersion y en caso contrario igual*

CREATE OR REPLACE FUNCTION verifica\_codigo\_repetido\_before()

RETURNS trigger AS

$$

begin

**if**(exists(select id\_producto from PRODUCTO)) then

**return** new;

**else**

**return** new;

end **if**;

END;

$$

LANGUAGE plpgsql;

*//se crea el trriger para antes del insert before y se llama a la funcion paraque haga su correspondiente proceso*

CREATE TRIGGER before\_trigger

before insert

ON PRODUCTO

FOR EACH ROW

EXECUTE PROCEDURE verifica\_codigo\_repetido\_before();

*//fucnion borrado producto cuando borramos la tabla producto es necesario actualizar el id\_producto*

*//ya que es identity si teniamos datos y borramos la tabla se queda en 6 entonces la reiniciamos la tabala producto*

CREATE OR REPLACE FUNCTION verifica\_borrado\_producto()

RETURNS trigger AS

$$

declare maxid **int**;

begin

**if** exists(select id\_producto from PRODUCTO) then

select (max(id\_producto)+1) from PRODUCTO into maxid;

execute 'alter SEQUENCE producto\_id\_producto\_seq RESTART with '|| maxid;

**return** new;

**else**

perform reinicia2();

**return** new;

end **if**;

END;

$$

LANGUAGE plpgsql;

*//hacemos el trigger por si hace el delete from producto y llame a la funcion verfica borrado producto*

CREATE TRIGGER trigger\_borrado\_producto\_actualiza\_id

after delete

ON PRODUCTO

FOR EACH ROW

EXECUTE PROCEDURE verifica\_borrado\_producto();

*//igual verficala la tabla orde\_venta si borra vuelve actualizar el id*

CREATE OR REPLACE FUNCTION verifica\_borrado\_orden\_func()

RETURNS trigger AS

$$

declare maxid **int**;

begin

**if**(exists(select (No\_venta) from ORDEN\_VENTA)) then

select (max(No\_venta)+1) from ORDEN\_VENTA into maxid;

execute 'alter SEQUENCE orden\_venta\_no\_venta\_seq RESTART with '|| maxid;

**return** new;

**else**

execute 'alter SEQUENCE orden\_venta\_no\_venta\_seq RESTART with '|| 1;

**return** new;

end **if**;

END;

$$

LANGUAGE plpgsql;

*//se crea el trigger por si haceel borrado*

CREATE TRIGGER trigger\_borrado\_orden

after delete

ON ORDEN\_VENTA

FOR EACH ROW

EXECUTE PROCEDURE verifica\_borrado\_orden\_func();

*// igual verificamos la tabla orden\_detalle como su primarykey es identity volvemos actualizar su id*

CREATE OR REPLACE FUNCTION verifica\_borrado\_orden\_detalle\_func()

RETURNS trigger AS

$$

declare maxid **int**;

begin

**if**(exists(select No\_orden\_detalle from ORDEN\_DETALLE)) then

select (max(No\_orden\_detalle)+1) from ORDEN\_DETALLE into maxid;

execute 'alter SEQUENCE orden\_detalle\_no\_orden\_detalle\_seq RESTART with '|| maxid;

**return** new;

**else**

execute 'alter SEQUENCE orden\_detalle\_no\_orden\_detalle\_seq RESTART with '|| 1;

**return** new;

end **if**;

END;

$$

LANGUAGE plpgsql;

*//creamos al trigger por si borra la tabla orde\_detalle*

CREATE TRIGGER trigger\_borrado\_orden\_detalle

after delete

ON ORDEN\_DETALLE

FOR EACH ROW

EXECUTE PROCEDURE verifica\_borrado\_orden\_detalle\_func();

*// funcion verifica orden es por sino hay stock suficiente, actualizar el precio\_venta\_del\_producto por lo metio mal,*

*//hace el calculo del total a pagar actualizando, actualiza el inventario, y aunque haga la insersion*

*// si llega a ver algun producto con stock menor a 3 nos envia un mensaje*

CREATE OR REPLACE FUNCTION verificar\_orden()

RETURNS trigger AS

$$

declare maxid **int**;

begin

*//if para saber si hay suficientes unidades parahacerel insert*

*//auqnue el insert ya sehizo*

**if**((select Unidades\_stock -(select cantidad\_articulo

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))

from INVENTARIO where codigo\_barras=(select codigo\_barras from PRODUCTO P where id\_producto=

(select id\_producto

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))))< 0) then

raise notice 'No se registrar la venta no hay inventario suficiente';*//envia mensaje*

delete from ORDEN\_DETALLE where No\_orden\_detalle=(select (max(No\_orden\_detalle)) from ORDEN\_DETALLE);*//como ya hizo el insert pues lo borramos si no hay inventario suficiente*

**return** null;

**else**

*//si hay inventario suficientehacemos el update dl campo precio\_por\_unidad del producto*

UPDATE ORDEN\_DETALLE

set precio\_venta\_producto=

(select precio\_venta from PRODUCTO P where id\_producto=

(select id\_producto

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE ))))

where No\_orden\_detalle=(select max(No\_orden\_detalle) from ORDEN\_DETALLE);

*//actualizamos el total a pagar*

UPDATE ORDEN\_DETALLE

set total\_pagar=

(select cantidad\_articulo\* (select precio\_venta\_producto from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle

from ORDEN\_DETALLE group by No\_orden\_detalle having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))

where No\_orden\_detalle=(select max(No\_orden\_detalle) from ORDEN\_DETALLE );

*//actualizamos el inventario*

UPDATE INVENTARIO

set unidades\_stock=

(select Unidades\_stock -(select cantidad\_articulo

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))

from INVENTARIO where codigo\_barras=(select codigo\_barras from PRODUCTO P where id\_producto=

(select id\_producto

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))))

where codigo\_barras=(select codigo\_barras from PRODUCTO P where id\_producto=

(select id\_producto

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE ))));

*//si quedan menor que 3 en el inventario*

**if**((select Unidades\_stock -(select cantidad\_articulo

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))

from INVENTARIO where codigo\_barras=(select codigo\_barras from PRODUCTO P where id\_producto=

(select id\_producto

from ORDEN\_DETALLE

where No\_orden\_detalle=(select No\_orden\_detalle from ORDEN\_DETALLE group by No\_orden\_detalle

having max(No\_orden\_detalle)=(select max(No\_orden\_detalle) from ORDEN\_DETALLE )))))<=3) then

raise notice 'El stock de tu producto es menor a 3';

end **if**;

raise notice 'se inserto correctamente';

**return** new;

end **if**;

END;

$$

LANGUAGE plpgsql;

*//hacemos el trigger del insert en orden\_detalle que es la venta del producto y aqui es after osea que ya hizo el insert*

CREATE TRIGGER test\_trigger

AFTER INSERT

ON ORDEN\_DETALLE

FOR EACH ROW

EXECUTE PROCEDURE verificar\_orden();

*//creamos un indice donde es en orden\_Detalle debido a que es donde hay más datas y se hacen más consultas a esta tabla y va tener*

*//mas informaciony haci la consulta sea un poco mejor*

Create index indice\_ventas on ORDEN\_DETALLE(No\_orden\_detalle);

*//se hace una funcion vista para cada orden de producto como hay ordenes con diferentes producto aqui arroja todos los productos que se estan*

*//vendiendo*

CREATE OR REPLACE FUNCTION vista\_informacion(No\_orden\_cliente\_recibida VARCHAR(8))

RETURNS table(

No\_orden\_cliente VARCHAR(8),

fecha date,

Nombre text,

Producto varchar(40),

Marca varchar(40),

cantidad\_articulo **int**,

precio\_cada\_producto decimal(6,2),

total\_por\_cada\_prodcuto decimal(7,2)) AS

$$

begin

**return** query (select ov.nota\_venta,ov.fecha\_venta,concat(c.nombre\_pila,' ',c.apellido\_p,' ',c.apellido\_m),

p.nombre,p.marca,od.cantidad\_articulo,od.precio\_venta\_producto,od.total\_pagar

from ORDEN\_VENTA as ov inner join

ORDEN\_DETALLE as od on ov.No\_venta=od.No\_venta

inner join CLIENTE as c on c.razon\_cliente=ov.razon\_cliente

inner join producto as p on p.id\_producto=od.id\_producto

where ov.nota\_venta=No\_orden\_cliente\_recibida);

END;

$$

LANGUAGE plpgsql;

*//se crea otravista solo conel total de articulos más formal y con el total a pagar sin ver cada producto*

CREATE OR REPLACE FUNCTION vista\_informacion\_por\_orden(No\_orden\_cliente\_recibida varchar(8))

RETURNS table(

No\_orden\_cliente varchar(8),

fecha date,

Nombre text,

cantidad\_articulos bigint,

total\_pagar DECIMAL(7,2)) AS

$$

begin

**return** query (select ov.nota\_venta,ov.fecha\_venta,concat(c.nombre\_pila,' ',c.apellido\_p,' ',c.apellido\_m),

count(od.cantidad\_articulo),SUM(od.total\_pagar)

from ORDEN\_VENTA as ov inner join

ORDEN\_DETALLE as od on ov.No\_venta=od.No\_venta

inner join CLIENTE as c on c.razon\_cliente=ov.razon\_cliente

where ov.nota\_venta=No\_orden\_cliente\_recibida group by ov.No\_venta,ov.fecha\_venta,c.nombre\_pila,c.apellido\_p,c.apellido\_m);

END;

$$

LANGUAGE plpgsql;

*//esta fucnion nos devuelve el campo que tengaestock menor a 3 en tabla inventario*

CREATE OR REPLACE FUNCTION stock\_menor\_3()

RETURNS table(nombre varchar(40))

as

$$

begin

**return** query (select p.nombre from inventario i inner join producto p on i.codigo\_barras=p.codigo\_barras where i.unidades\_stock<3);

END;

$$

LANGUAGE plpgsql;

*//se crea la funcion cantidad vendidapor fecha si da una fechadeinicio y fin nos devuelve el total vendido en esas fechas*

CREATE OR REPLACE FUNCTION CANTIDAD\_VENDIDA\_POR\_FECHA(FECHA\_INICIO DATE, FECHA\_FIN DATE)

RETURNS table(cantidad\_vendida numeric)

as

$$

begin

**return** query (select sum(od.total\_pagar)

from ORDEN\_VENTA as ov inner join

ORDEN\_DETALLE as od on ov.No\_venta=od.No\_venta

where ov.No\_venta=od.No\_venta and ov.fecha\_venta between FECHA\_INICIO and FECHA\_FIN);

END;

$$

LANGUAGE plpgsql;

*//inserts*

insert into PROVEEDOR values('MRJE','Martinez','Rojas','Jose Eduardo','Venustiano Carranza','CDMX','Av. Iglesias Calderon','15900',10);

insert into PROVEEDOR values('AGM','Alvarado','Garcia','Manuel','Izrtacalco','CDMX','Resina','08720',5);

insert into PROVEEDOR values('SMJ','Sanchez','Maldonado','Juan','Izrtacalco','CDMX','Resina','08720',8);

insert into INVENTARIO values('7503028607006','2021/01/31',5,20);

insert into INVENTARIO(codigo\_barras,precio\_compra,unidades\_stock) values('7503028601231',4,50);

insert into INVENTARIO values('7503028604332','2021/01/31',8,50);

insert into INVENTARIO values('7503028605334','2021/01/31',50,1000);

insert into INVENTARIO values('7503028605335','2021/01/31',5,1000);

insert into INVENTARIO values('7503028605337','2021/01/31',3,2);

insert into PRODUCTO(nombre,precio\_venta,marca,codigo\_barras,utilidad,razon\_proveedor) values('monografia',20,'cielito','7503028607006',10,'AGM');

insert into PRODUCTO(nombre,precio\_venta,marca,codigo\_barras,utilidad,razon\_proveedor) values('monografia',20,'cielito','7503028607006',10,'MRJE');

insert into PRODUCTO(nombre,precio\_venta,marca,codigo\_barras,utilidad,razon\_proveedor) values('Lapiz',4.5,'Maped','7503028601231',4,'SMJ');

insert into PRODUCTO(nombre,precio\_venta,marca,codigo\_barras,utilidad,razon\_proveedor) values('Goma',9,'Maped','7503028604332',4,'MRJE');

insert into PRODUCTO(nombre,precio\_venta,marca,codigo\_barras,utilidad,razon\_proveedor) values('recarga',50,'movistar','7503028605334',2,'MRJE');

insert into PRODUCTO(nombre,precio\_venta,marca,codigo\_barras,utilidad,razon\_proveedor) values('sacapuntas',6,'maped','7503028605335',1,'MRJE');

insert into PRODUCTO(nombre,precio\_venta,marca,codigo\_barras,utilidad,razon\_proveedor) values('pluma',8,'vic','7503028605337',1,'MRJE');

insert into CLIENTE values('MRP','Martinez','Rojas','Pedro','Venustiano Carranza','CDMX','Av. Iglesias Calderon','15900',10);

insert into CLIENTE values('MGM','Martel','Garcia','Manuel','Izrtacalco','CDMX','Resina','08720',5);

insert into ORDEN\_VENTA(fecha\_venta,razon\_cliente) values('2021/01/21','MRP');

insert into ORDEN\_VENTA(fecha\_venta,razon\_cliente) values('2021/01/22','MGM');

insert into ORDEN\_VENTA(fecha\_venta,razon\_cliente) values('2021/02/21','MGM');

insert into ORDEN\_VENTA(fecha\_venta,razon\_cliente)values('2023/02/21','MRP');

insert into ORDEN\_DETALLE(No\_venta,id\_producto,cantidad\_articulo,precio\_venta\_producto,total\_pagar) values(1,1,5,1,100);

insert into ORDEN\_DETALLE(No\_venta,id\_producto,cantidad\_articulo,precio\_venta\_producto,total\_pagar) values(2,5,100,1,100);

insert into ORDEN\_DETALLE(No\_venta,id\_producto,cantidad\_articulo,precio\_venta\_producto,total\_pagar) values(2,3,2,1,100);

insert into ORDEN\_DETALLE(No\_venta,id\_producto,cantidad\_articulo,precio\_venta\_producto,total\_pagar) values(2,4,2,1,100);

insert into ORDEN\_DETALLE(No\_venta,id\_producto,cantidad\_articulo,precio\_venta\_producto,total\_pagar) values(3,5,2,1,100);

insert into ORDEN\_DETALLE(No\_venta,id\_producto,cantidad\_articulo,precio\_venta\_producto,total\_pagar) values(2,1,16,1,100);

insert into ORDEN\_DETALLE(No\_venta,id\_producto,cantidad\_articulo,precio\_venta\_producto,total\_pagar) values(4,1,1,1,100);

*//ver las funcione vista para como veria el usuario la informacion si se proporciona el nota\_venta*

select \* from vista\_informacion('VENT-001');

select \* from vista\_informacion\_por\_orden('VENT-001');

*//pruebas para el stock y cantdad vendida por fecha*

select \* from CANTIDAD\_VENDIDA\_POR\_FECHA ('2021/01/21','2021/01/23') ;

select \* from stock\_menor\_3();