```
// 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 usuario(
                   usuario_pape varchar(10) NOT NULL,
                   password varchar (6) NOT NULL
                   );

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 table(
                        id_producto int,
                        nombre VARCHAR(40),
                        precio_venta DECIMAL(6,2) ,
                        marca VARCHAR(40),
                        descripcion varchar(50) ,
                        codigo_barras varchar(13),
                        utilidad DECIMAL(6,2) ,
                        razon_proveedor varchar(60) ,
                        codigo_barras1 varchar(13)  ,
                        fecha_compra date ,
                        precio_compra DECIMAL(6,2),
                        unidades_stock int )
AS $$
BEGIN
RETURN QUERY SELECT * FROM PRODUCTO JOIN INVENTARIO
        ON INVENTARIO.codigo_barras=PRODUCTO.codigo_barras WHERE
PRODUCTO.codigo_barras=codigo;
END
$$ 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();
```