



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería

Bases de datos
Creación de una base de datos

Proyecto Final
Equipo Glassy

Meza Nava Pablo
Rendon Hernandez Roberto
Rodriguez Estrella Mairol

Professor:
Ing. Fernando Arreola
Gpo.01, 2023-2

10.06.2023

Índice general

Introducción	1
Plan de trabajo	2
Diseño	2
Desarrollo	5

Índice de figuras

1.	Plan Trabajo	2
2.	MER	4
3.	MR	5
4.	Creacion de tablas	5
5.	Creacion de tablas	6
6.	Creacion de tablas	7
7.	Creacion de tablas	8
8.	Query creación de tablas	8
9.	Inserción artículo	9
10.	Inserción clientes	9
11.	Inserción email_clientes	9
12.	Inserción producto	9
13.	Inserción producto_artículo	10
14.	Inserción proveedor	10
15.	Inserción proveedor_producto	10
16.	Inserción telefonoproveedor	10
17.	Inserción venta	11
18.	Inserción venta_artículo	11

Resumen

El siguiente documento fue generado con la finalidad de dar a conocer el desarrollo que se llevo a cabo para la realización de nuestro proyecto final, así como mostrar la documentación generada y el como analizamos y desarrollamos el problema para la resolución exitosa de este.

Introducción

La necesidad de innovar en la forma de almacenar la información llevó a la contratación de un equipo especializado en sistemas informáticos para desarrollar una solución a medida. El objetivo principal fue diseñar una base de datos que permitiera almacenar y gestionar datos relacionados con proveedores, clientes, productos y ventas de manera eficiente y organizada.

El enunciado inicial proporcionó los requerimientos clave para la base de datos, incluyendo entidades como proveedores, clientes, productos y ventas, así como los atributos asociados a cada una. Además, se especificaron las relaciones y cardinalidades entre las entidades, lo que permitió establecer las conexiones necesarias para representar adecuadamente los vínculos entre los datos.

El proceso de diseño de la base de datos involucró la identificación de las entidades principales y sus atributos, la definición de las relaciones entre ellas y la creación de tablas que representaran estas entidades y relaciones en un modelo relacional. A partir de esta representación, se procedió a realizar la implementación práctica de la base de datos mediante la creación de tablas en un sistema de gestión de bases de datos.

A conitnuación se proporciona una descripción detallada del modelo relacional resultante, presentando las tablas y sus atributos, así como las conexiones entre ellas. Además, se muestra el código utilizado para crear las tablas en el sistema de gestión de bases de datos, lo que permitirá tener una visión completa de la estructura y funcionamiento de la base de datos implementada.

Plan de trabajo

La organización llevada a cabo por el equipo se presenta en la siguiente tabla, en donde a cada integrante se le asignó tareas de acuerdo a las capacidades desarrolladas a lo largo del curso mediante el uso de la plataforma Monday, la cual nos permitió llevar un mejor control de la gestión y asignación de tareas. Hay tareas que se desarrollaron en conjunto, y cada entrega parcial realizada llevó una revisión en juntas llevadas a cabo mediante la plataforma meet.


Proyecto Bases		Powered by 			
Este mes					
Nombre	Persona	Estado	Fecha	Archivo	Comentarios/Notas
Análisis y diseño inicial	Todos	Listo	2023-05-19	https://mairols-team.monday.com/protected_status/17318815/resources/912300983/REQUERIMIENTOS.docx	Reunión de equipo en meet, revisar archivo generado en Drive
MER	Rodriguez Estrella Mairol Elizabeth	Listo	2023-05-20	https://mairols-team.monday.com/protected_status/17318815/resources/912301040/MER%20PROYECTO.drawio%20(1).png	Realizado en draw.io
MR	Meza Nava Pablo	Listo	2023-05-24	https://mairols-team.monday.com/protected_status/17318815/resources/912301052/MER.docx	Documentación en drive
Tablas	Rendon Hernandez Roberto	Listo	2023-05-27	https://mairols-team.monday.com/protected_status/17318815/resources/912301472/Entregable.docx	Realizadas en PGModeler, revisar imagen
Código consultas	Principales: Meza Nava Pablo, Rendon Hernandez Roberto. Apoyo: Rodriguez Estrella Mairol Elizabeth	Listo	2023-06-03	https://mairols-team.monday.com/protected_status/17318815/resources/912301441/REQUERIMIENTOS.docx	Documentado en drive
Documentación Latex	Rodriguez Estrella Mairol Elizabeth	En curso	2023-06-08		Iniciada
			Desde el 2023-05-19 hasta el 2023-06-08		

Figura 1: Plan Trabajo

Diseño

MER

Partiendo de los requerimientos del enunciado se elaboró el MER. Se crearon las entidades y relaciones necesarias para el requerimiento de la base de datos, quedando de la siguiente forma:

ENTIDADES:

Proveedor:

Razón social (atributo) PK Domicilio → estado, código postal, colonia, calle y número. (atributo) COMPUESTO Nombre → nombre_pila, ap_paterno, ap_materno (atributo) COMPUESTO Teléfonos (atributo) MULTIVALUADO

Cliente:

Nombre \rightarrow nombre_pila, ap_paterno, ap_materno (atributo) COMPUESTO Domicilio \rightarrow estado, código_postal, colonia, calle y número. (atributo) COMPUESTO Email (atributo) MULTIVALUADO RFC (atributo) PK

Producto:

Código de barras (atributo) PK Precio de compra (atributo) Foto (atributo) Fecha de compra (atributo) Cantidad en bodega (atributo)

Artículo:

Podemos añadir un id para esta entidad, ya que ninguno de los otros atributos identifica de forma única e irrepetible el artículo Id_artículo PK Marca (atributo) Descripción (atributo) Precio (atributo)

Venta:

Número de venta (atributo) PK Fecha de venta (atributo) Cantidad total a pagar (atributo)

RELACIONES:

Proveedor_Producto(suministrar)

Producto_Artículo(asociado)

Venta_Artículo(asociado)

Cliente_Venta(tener):

En este modelo, se considera que la entidad “Producto” representa los productos generales en inventario, mientras que la entidad “Artículo” representa los diferentes tipos de productos que se pueden vender, como regalos, artículos de papelería, impresiones y recargas. La relación “Proveedor_Producto” vincula los proveedores con los productos que suministran, y la relación “Producto_Artículo” vincula los productos en el inventario con los diferentes tipos de artículos que pueden ser vendidos.

La relación “Venta_Artículo” se utiliza para asociar los artículos vendidos con una venta específica, incluyendo la cantidad de cada artículo y el precio total a pagar por artículo (atributos de la relación).

CARDINALIDADES

Proveedor_Producto (M,M)(suministrar)

Proveedor: (1, M) Un proveedor puede suministrar muchos productos. Producto: (0, M) Un producto puede ser suministrado por uno o varios proveedores.

Producto_Artículo (M,M)(asociado):

Producto: (0, M) Un producto puede estar asociado a uno o varios artículos. Artículo: (0, M) Un artículo puede estar asociado a uno o varios productos.

Venta_Artículo (M,M)(asociado):

Venta: (1, M) Una venta puede incluir uno o varios artículos. Artículo: (0, M) Un artículo puede estar presente en una o varias ventas.

Cliente_Venta(M,1)(tener):

Cliente: (1, N) Un cliente puede tener ninguna o varias ventas asociadas. Venta: (1, 1) Una venta está asociada a un único cliente.

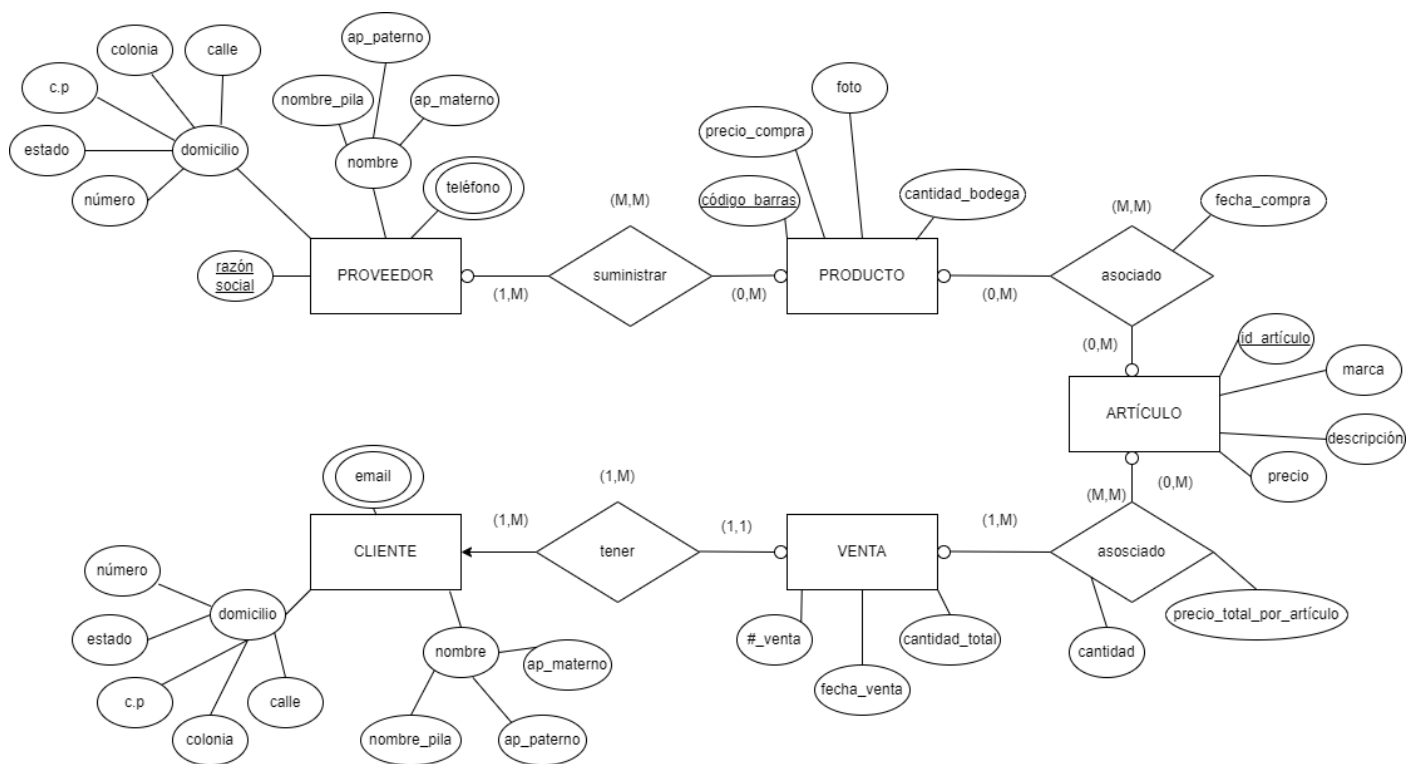


Figura 2: MER

MR

A partir del modelo entidad relacion creado en el punto anterior procedimos a generar el MR apoyandonos de la herramienta pgModeler, creando de esta forma las tablas correspondientes a un sistema de gestión de bases de datos.

Cada tabla representaba una entidad y sus atributos; además asignamos los tipos de datos apropiados a cada atributo según su naturaleza estableciendo las claves primarias y foráneas en las tablas relacionales. Definimos las restricciones necesarias para garantizar la integridad y consistencia de los datos incluyendo la especificación de claves primarias, claves foráneas y restricciones de integridad referencial para mantener la coherencia entre las tablas

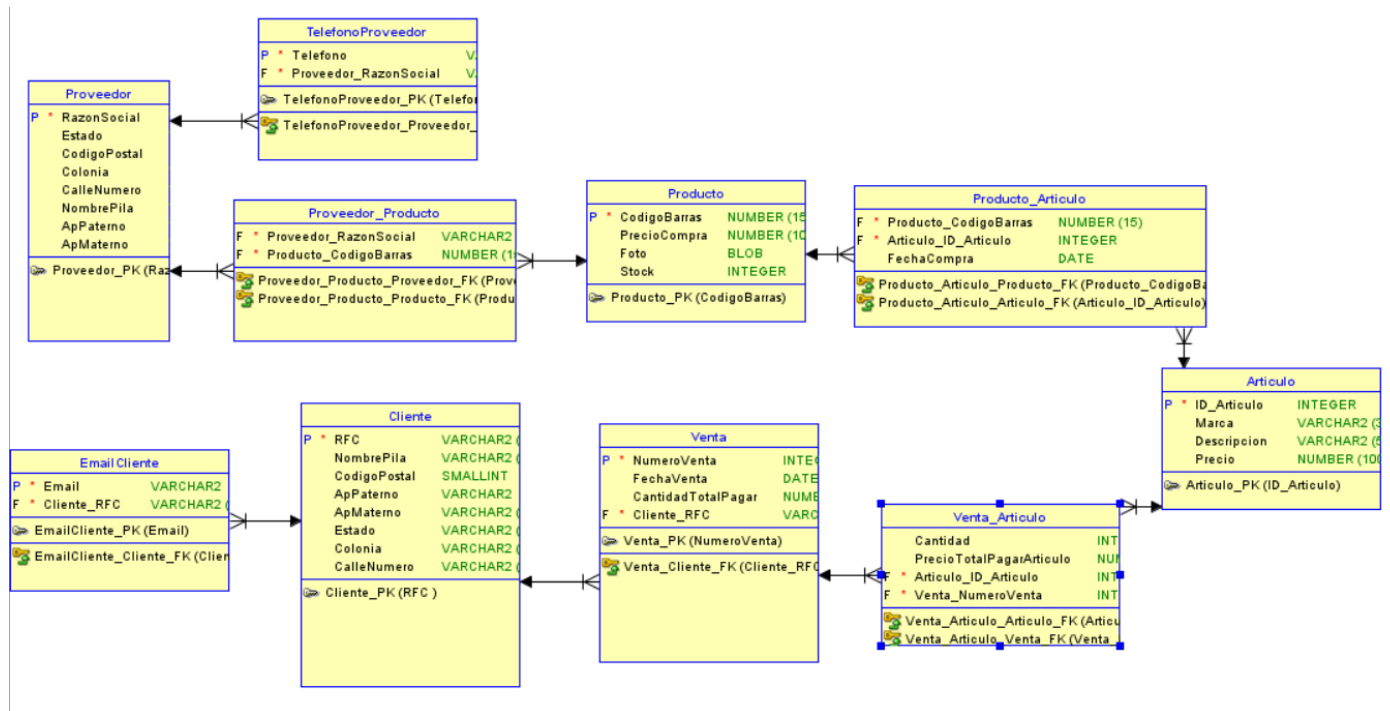


Figura 3: MR

Desarrollo

Creación, ejecución y pruebas de código

Se redactó el código PSQL necesario para crear las tablas en el sistema de gestión de bases de datos. Esto incluyó la declaración de las tablas, los atributos, las claves primarias y foráneas, así como las restricciones definidas previamente

```
CREATE TABLE articulo (
    id_articulo INT PRIMARY KEY,
    marca VARCHAR(30),
    descripcion VARCHAR(50),
    precio NUMERIC(1000)
);

CREATE TABLE cliente (
    rfc VARCHAR(30) NOT NULL PRIMARY KEY,
    nombrepila VARCHAR(50),
    codigopostal int,
    appaterno VARCHAR,
    apmaterno VARCHAR(30),
    estado VARCHAR(30),
    colonia VARCHAR(30),
    callennumero VARCHAR(30)
);

CREATE TABLE emailcliente (
    email VARCHAR NOT NULL,
    cliente_rfc VARCHAR(30) NOT NULL,
    PRIMARY KEY (email),
    FOREIGN KEY (cliente_rfc) REFERENCES cliente (rfc)
);
```

Figura 4: Creacion de tablas

CREATE TABLE artículo: Está instrucción crea la tabla “artículo” con las siguientes columnas: id_articulo de tipo INT y es la clave primaria de la tabla. marca de tipo VARCHAR(30), almacena la marca del artículo. descripcion de tipo VARCHAR(50), almacena la descripción del artículo. precio de tipo NUMERIC(1000), almacena el precio del artículo.

CREATE TABLE cliente:

Esta instrucción crea la tabla “cliente” con las siguientes columnas:

rfc de tipo VARCHAR(30) y es la clave primaria de la tabla. Almacena el RFC del cliente. nombrepila de tipo VARCHAR(50), que almacena el nombre del cliente. codigopostal de tipo INT, que almacena el código postal del cliente. appaterno de tipo VARCHAR, que almacena el apellido paterno del cliente. apmaterno de tipo VARCHAR(30), que almacena el apellido materno del cliente. estado de tipo VARCHAR(30), que almacena el estado del cliente. colonia de tipo VARCHAR(30), que almacena la colonia del cliente. callennumero de tipo VARCHAR(30), que almacena el número de calle del cliente.

CREATE TABLE emailcliente:

Esta instrucción crea la tabla “emailcliente” con las siguientes columnas:

email de tipo VARCHAR y es la clave primaria de la tabla. Almacena el correo electrónico del cliente. cliente_rfc de tipo VARCHAR(30), que almacena el RFC del cliente al que pertenece el correo electrónico. Esta columna tiene una restricción de clave externa que hace referencia a la columna rfc de la tabla “cliente”.

```
CREATE TABLE producto (
  codigobarras BIGINT NOT NULL PRIMARY KEY,
  preciocompra NUMERIC(1000),
  foto BYTEA,
  stock INTEGER
);

CREATE TABLE producto_articulo (
  producto_codigobarras BIGINT NOT NULL,
  articulo_id_articulo INTEGER NOT NULL,
  fechacompra DATE,
  FOREIGN KEY (producto_codigobarras) REFERENCES producto (codigobarras),
  FOREIGN KEY (articulo_id_articulo) REFERENCES articulo (id_articulo)
);

CREATE TABLE proveedor (
  razonsocial VARCHAR(50) NOT NULL PRIMARY KEY,
  estado VARCHAR(30),
  codigopostal int,
  colonia VARCHAR(30),
  callennumero VARCHAR,
  nombrepila VARCHAR(50),
  appaterno VARCHAR(30),
  apmaterno VARCHAR
);
```

Figura 5: Creacion de tablas

CREATE TABLE producto:

Esta instrucción crea la tabla “producto” con las siguientes columnas:

codigobarras de tipo BIGINT y es la clave primaria de la tabla. Almacena el código de barras del producto. preciocompra de tipo NUMERIC(1000), que almacena el precio de compra del producto. foto de tipo BYTEA, que almacena la foto del producto. stock de tipo INTEGER, que almacena la cantidad de stock disponible del producto.

CREATE TABLE producto_articulo:

Esta instrucción crea la tabla “producto_articulo” con las siguientes columnas:

producto_codigobarras de tipo BIGINT y es una clave externa que hace referencia a la columna codigobarras de la tabla “producto”. articulo_id.articulo de tipo INTEGER y es una clave externa que hace referencia a la columna id_articulo de la tabla “articulo”. fechacompra de tipo DATE, que almacena la fecha de compra del producto.

CREATE TABLE proveedor:

Esta instrucción crea la tabla “proveedor” con las siguientes columnas:

razonsocial de tipo VARCHAR(50) y es la clave primaria de la tabla. Almacena la razón social del proveedor. estado de tipo VARCHAR(30), que almacena el estado del proveedor. codigopostal de tipo INT, que almacena el código postal del proveedor. colonia de tipo VARCHAR(30), que almacena la colonia del proveedor. callennumero de tipo VARCHAR, que almacena el número de calle del proveedor. nombrepila de tipo VARCHAR(50), que almacena el nombre del proveedor. appaterno de tipo VARCHAR(30), que almacena el apellido paterno del proveedor. apmaterno de tipo VARCHAR, que almacena el apellido materno del proveedor.

```
CREATE TABLE proveedor_producto (
  proveedor_razonsocial VARCHAR(50) NOT NULL,
  producto_codigobarras BIGINT NOT NULL,
  FOREIGN KEY (proveedor_razonsocial) REFERENCES proveedor (razonsocial),
  FOREIGN KEY (producto_codigobarras) REFERENCES producto (codigobarras)
);

CREATE TABLE telefonoproveedor (
  telefono VARCHAR(50) NOT NULL,
  proveedor_razonsocial VARCHAR(50) NOT NULL,
  PRIMARY KEY (telefono),
  FOREIGN KEY (proveedor_razonsocial) REFERENCES proveedor (razonsocial)
);

CREATE TABLE venta (
  numeroventa VARCHAR(10) PRIMARY KEY,
  fechaventa DATE,
  cantidadtotalpagar NUMERIC,
  cliente_rfc VARCHAR(30) NOT NULL,
  FOREIGN KEY (cliente_rfc) REFERENCES cliente (rfc)
);
```

Figura 6: Creacion de tablas

CREATE TABLE proveedor_producto:

Esta instrucción crea la tabla “proveedor_producto” con las siguientes columnas:

proveedor_razonsocial de tipo VARCHAR(50) y es una clave externa que hace referencia a la columna razonsocial de la tabla ”proveedor”. producto_codigobarras de tipo BIGINT y es una clave externa que hace referencia a la columna codigobarras de la tabla ”producto”.

CREATE TABLE telefonoproveedor:

Esta instrucción crea la tabla “telefonoproveedor” con las siguientes columnas:

telefono de tipo VARCHAR(50) y es la clave primaria de la tabla. Almacena el número de teléfono del proveedor. proveedor_razonsocial de tipo VARCHAR(50), que almacena la razón social del proveedor al que pertenece el número de teléfono. Esta columna tiene una restricción de clave externa que hace referencia a la columna razonsocial de la tabla “proveedor”.

CREATE TABLE venta:

Esta instrucción crea la tabla “venta” con las siguientes columnas:

numeroventa de tipo VARCHAR(10) y es la clave primaria de la tabla. Almacena el número de venta. fechaventa de tipo DATE, que almacena la fecha de venta. cantidadtotalpagar de tipo NUMERIC, que almacena la cantidad total a pagar de la venta. cliente_rfc de tipo VARCHAR(30), que almacena el RFC del cliente. Esta columna tiene una restricción de clave externa que hace referencia a la columna rfc de la tabla “cliente”.

```
CREATE TABLE venta_articulo (  
    cantidad                INTEGER,  
    preciototalpagararticulo NUMERIC,  
    articulo_id_articulo    INTEGER NOT NULL,  
  
    venta_numeroventa       VARCHAR(10) NOT NULL,  
    FOREIGN KEY (articulo_id_articulo) REFERENCES articulo (id_articulo),  
    FOREIGN KEY (venta_numeroventa) REFERENCES venta (numeroventa)  
);
```

Figura 7: Creacion de tablas

CREATE TABLE venta_articulo:

Esta instrucción crea la tabla “venta_articulo” con las siguientes columnas:

cantidad de tipo INTEGER, que almacena la cantidad de artículos vendidos. preciototalpagararticulo de tipo NUMERIC, que almacena el precio total a pagar por el artículo. articulo_id_articulo de tipo INTEGER y es una clave externa que hace referencia a la columna id_articulo de la tabla “articulo”. venta_numeroventa de tipo VARCHAR(10) y es una clave externa que hace referencia a la columna numeroventa de la tabla “venta”.

```
CREATE TABLE  
  
Query returned successfully in 113 msec.
```

Figura 8: Query creación de tablas

Nos muestra que las tablas fueron creadas de manera exitosa y el tiempo que tardaron en ser creadas

Continuamos realizando las inserciones correspondientes a cada tabla para la implementación de nuestra base con los puntos solicitados

```

INSERT INTO articulo (id_articulo, marca, descripcion, precio)
VALUES
  (1,'Sigel', 'Cuaderno Profesional', 100.0),
  (2,'Pritt', 'Lápiz adhesivo', 50.0),
  (3,'OfficeMax', 'Corrector 5mm x 8 mm', 75.0),
  (4,'JeanBook', 'Carpeta tamaño carta', 120.0),
  (5,'Maped', 'Compás metálico', 80.0),
  (6,'OfficeMax', 'Regla aluminio 30cm', 90.0),
  (7,'Pilot', 'Grapa estándar', 60.0);

```

Figura 9: Inserción artículo

```

INSERT INTO cliente (rfc, nombrepila, codigopostal, appaterno, apmaterno, estado, colonia, callennumero)
VALUES
  ('MENP010223', 'Pablo', 10840, 'Meza', 'Nava', 'Cdmx', 'Las Calles', 'Juan Álvarez 203'),
  ('MELM830528', 'Mónica', 64108, 'Méndez', 'Luna', 'Monterrey', 'Plutarco Elías Calles', 'Agua Prieta 42'),
  ('LOMP820628', 'Pablo', 83570, 'López', 'Morales', 'Sonora', 'La Copa', 'Hidalgo 33'),
  ('VACS691110', 'Antonio', 34194, 'Rodríguez', 'Vázquez', 'Durango', 'Valle Verde', 'Margarita 12'),
  ('TOGM900603', 'Mónica', 86400, 'Torres', 'Guzmán', 'Tabasco', 'Centro', 'Ignacio Allende 84'),
  ('SAVL852312', 'Lorena', 87700, 'Saldaña', 'Velazquez', 'Tamaulipas', 'Zona Centro', 'Libertad 68'),
  ('PEGJ198305', 'José', 98619, 'Pérez', 'García', 'Zacatecas', 'Los Pirules', 'Cerro de Sombreretillo');

```

Figura 10: Inserción clientes

```

INSERT INTO emailcliente (email, cliente_rfc)
VALUES
  ('fraheyounnutu@hotmail.com', 'MENP010223'),
  ('luquateubegreu@gmail.com', 'MELM830528'),
  ('haprapeijicroi@gmail.com', 'LOMP820628'),
  ('gubumoutose@hotmail.com', 'VACS691110'),
  ('koicajotiso@gmail.com', 'TOGM900603'),
  ('tamerappeuna@hotmail.com', 'SAVL852312'),
  ('voreucenetra-7847@gmail.com', 'PEGJ198305');

```

Figura 11: Inserción email_clientes

```

INSERT INTO producto (codigobarras, preciocompra, foto, stock)
VALUES
  (123456789, 10.0, NULL, 50),
  (987654321, 20.0, NULL, 30),
  (456789123, 15.0, NULL, 20),
  (321654987, 8.0, NULL, 40),
  (789123456, 12.0, NULL, 15),
  (654987321, 25.0, NULL, 25),
  (987123456, 18.0, NULL, 35);

```

Figura 12: Inserción producto

```
INSERT INTO producto_articulo (producto_codigobarras, articulo_id_articulo, fechacompra)
VALUES
(123456789, 1, '2023-05-01'),
(987654321, 2, '2023-05-02'),
(456789123, 3, '2023-05-03'),
(321654987, 4, '2023-05-04'),
(789123456, 5, '2023-05-05'),
(654987321, 6, '2023-05-06'),
(987123456, 7, '2023-05-07');
```

Figura 13: Inserción producto_artículo

```
INSERT INTO proveedor (razonsocial, estado, codigopostal, colonia, callennumero, nombrepila, appaterno, apmaterno)
VALUES
('CRECE COMPUTACION, S.A. DE C.V.', 'Jalisco', 44100, 'Moderna', 'Jose Guadalupe Montenegro 2393', 'Pascual', 'Ruiz', 'López'),
('IMPREJAL, S.A. DE C.V.', 'Jalisco', 44290, 'Jardines del Bosque', 'Nicolas Romero 518', 'Lydia', 'Martinez', 'Chavez'),
('ALBE INTERNACIONAL SA DE CV', 'Jalisco', 45010, 'Colonia3', 'Calzada Norte 7336', 'Flora', 'Camara', 'Orozco'),
('PLASTICOS Y METALES MYC, S.A. DE C.V.', 'Estado de México', 52600, 'Santiago Tianguistengo', 'Santiago Chapultepec 709', 'Melanie'),
('GRUPO AVESTRUZ, S.A DE.C.V. ', 'Jalisco', 44900, 'Del Fresno', 'Pelicano 2135', 'Mónica', 'Cortes', 'Sánchez'),
('LONAS LORENZO, S.A. DE C.V.', 'Jalisco', 44190, 'Moderna', 'Av 8 de julio', 'Hugo', 'Fonseca', 'Alegre'),
('JOSÉ RICARDO NISHIMURA TORRES S.A. DE C.V.', 'Tamaulipas', 44200, 'Artesanos', 'Independencia 110', 'Noelia', 'Tovar', 'Periz');
```

Figura 14: Inserción proveedor

```
INSERT INTO proveedor_producto (proveedor_razonsocial, producto_codigobarras)
VALUES
('CRECE COMPUTACION, S.A. DE C.V.', 123456789),
('IMPREJAL, S.A. DE C.V.', 987654321),
('ALBE INTERNACIONAL SA DE CV', 456789123),
('PLASTICOS Y METALES MYC, S.A. DE C.V.', 321654987),
('GRUPO AVESTRUZ, S.A DE.C.V. ', 789123456),
('LONAS LORENZO, S.A. DE C.V.', 654987321),
('JOSÉ RICARDO NISHIMURA TORRES S.A. DE C.V.', 987123456);
```

Figura 15: Inserción proveedor_producto

```
INSERT INTO telefonoproveedor (telefono, proveedor_razonsocial)
VALUES
('3336156444', 'CRECE COMPUTACION, S.A. DE C.V'),
('3338254769', 'IMPREJAL, S.A. DE C.V'),
('3312537000', 'ALBE INTERNACIONAL SA DE CV'),
('5558123402', 'PLASTICOS Y METALES MYC, S.A. DE C.V.'),
('3338106699', 'GRUPO AVESTRUZ, S.A DE.C.V. '),
('3336503541', 'LONAS LORENZO, S.A. DE C.V.'),
('3335630862', 'JOSÉ RICARDO NISHIMURA TORRES S.A. DE C.V.');
```

Figura 16: Inserción telefonoproveedor

```

INSERT INTO venta (numeroventa, fechaventa, cantidadtotalpagar, cliente_rfc)
VALUES
  ('VENT-001', '2023-05-01', 500.0, 'MENP010223'),
  ('VENT-002', '2023-05-02', 400.0, 'MELM830528'),
  ('VENT-003', '2023-05-03', 225.0, 'LOMP820628'),
  ('VENT-004', '2023-05-04', 720.0, 'VACS691110'),
  ('VENT-005', '2023-05-05', 320.0, 'TOGM900603'),
  ('VENT-006', '2023-05-06', 630.0, 'SAVL852312'),
  ('VENT-007', '2023-05-07', 120.0, 'PEGJ198305');

```

Figura 17: Inserción venta

```

INSERT INTO venta_articulo (cantidad, preciotalpagararticulo, articulo_id_articulo, venta_numeroventa)
VALUES
  (5, 500.0, 1, 'VENT-001'),
  (8, 400.0, 2, 'VENT-002'),
  (3, 225.0, 3, 'VENT-003'),
  (6, 720.0, 4, 'VENT-004'),
  (4, 320.0, 5, 'VENT-005'),
  (7, 630.0, 6, 'VENT-006'),
  (2, 120.0, 7, 'VENT-007');

```

Figura 18: Inserción venta_artículo

IMPLEMENTACIÓN

Inserciones tabla articulo

```
INSERT INTO articulo (id_articulo, marca, descripcion, precio)
VALUES
(1,'Sigel', 'Cuaderno Profesional', 100.0),
(2,'Pritt', 'Lápiz adhesivo', 50.0),
(3,'OfficeMax', 'Corrector 5mm x 8 mm', 75.0),
(4,'JeanBook', 'Carpeta tamaño carta', 120.0),
(5,'Maped', 'Compás metálico', 80.0),
(6,'OfficeMax', 'Regla aluminio 30cm', 90.0),
(7,'Pilot', 'Grapa estándar', 60.0);
```

INSERT 0 7

Query returned successfully in 76 msec.

Inserción de tabla cliente:

```
INSERT INTO cliente (rfc, nombrepila, codigopostal, appaterno, apmaterno, estado, colonia, callennumero)
VALUES
('MENP010223', 'Pablo', 10840, 'Meza', 'Nava', 'Cdmx', 'Las Calles', 'Juan Álvarez 203'),
('MELM830528', 'Mónica', 64108, 'Méndez', 'Luna', 'Monterrey', 'Plutarco Elias Calles', 'Agua Prieta 42'),
('LOMP820628', 'Pablo', 83570, 'López', 'Morales', 'Sonora', 'La Copa', 'Hidalgo 33'),
('VACS691110', 'Antonio', 34194, 'Rodríguez', 'Vázquez', 'Durango', 'Valle Verde', 'Margarita 12'),
('TOGM900603', 'Mónica', 86400, 'Torres', 'Guzmán', 'Tabasco', 'Centro', 'Ignacio Allende 84'),
('SAVL852312', 'Lorena', 87700, 'Saldaña', 'Velazquez', 'Tamaulipas', 'Zona Centro', 'Libertad 68'),
('PEGJ198305', 'José', 98619, 'Pérez', 'García', 'Zacatecas', 'Los Pirules', 'Cerro de Sombreretillo');
```

INSERT 0 7

Query returned successfully in 39 msec.

Inserción de nuestro campo multivaluado email:

```
INSERT INTO emailcliente (email, cliente_rfc)
VALUES
('fraheyounnutu@hotmail.com', 'MENP010223'),
('luquateubegreu@gmail.com', 'MELM830528'),
('haprapeijicroi@gmail.com', 'LOMP820628'),
('gubumoutose@hotmail.com', 'VACS691110'),
('koicajotiso@gmail.com', 'TOGM900603'),
('tamerappeuna@hotmail.com', 'SAVL852312'),
('voreucenetra-7847@gmail.com', 'PEGJ198305');
```

```
INSERT 0 7
```

```
Query returned successfully in 41 msec.
```

Inserción de tabla producto:

```
INSERT INTO producto (codigobarras, preciocompra, foto, stock)
VALUES
  (123456789, 10.0, NULL, 50),
  (987654321, 20.0, NULL, 30),
  (456789123, 15.0, NULL, 20),
  (321654987, 8.0, NULL, 40),
  (789123456, 12.0, NULL, 15),
  (654987321, 25.0, NULL, 25),
  (987123456, 18.0, NULL, 35);
```

```
INSERT 0 7
```

```
Query returned successfully in 59 msec.
```

Inserción de relación que existe entre producto y artículo

```
INSERT INTO producto_articulo (producto_codigobarras, articulo_id_articulo, fechacompra)
VALUES
  (123456789, 1, '2023-05-01'),
  (987654321, 2, '2023-05-02'),
  (456789123, 3, '2023-05-03'),
  (321654987, 4, '2023-05-04'),
  (789123456, 5, '2023-05-05'),
  (654987321, 6, '2023-05-06'),
  (987123456, 7, '2023-05-07');
```

```
INSERT 0 7
```

```
Query returned successfully in 48 msec.
```

Inserción tabla proveedores:

```
INSERT INTO proveedor (razonsocial, estado, codigopostal, colonia, callennumero, nombrepila, appaterno, apmaterno)
VALUES
  ('CRECE COMPUTACION, S.A. DE C.V.', 'Jalisco', 44100, 'Moderna', 'Jose Guadalupe Montenegro 2393', 'Pascual', 'Ruiz', 'López'),
  ('IMPREJAL, S.A. DE C.V.', 'Jalisco', 44290, 'Jardines del Bosque', 'Nicolas Romero 518', 'Lydia', 'Martinez', 'Chavez'),
  ('ALBE INTERNACIONAL SA DE CV', 'Jalisco', 45010, 'Colonia3', 'Calzada Norte 7336', 'Flora', 'Camara', 'Orozco'),
  ('PLASTICOS Y METALES MYC, S.A. DE C.V.', 'Estado de México', 52600, 'Santiago Tianguistengo', 'Santiago Chapultepec 709', 'Melanie'),
  ('GRUPO AVESTRUZ, S.A. DE C.V.', 'Jalisco', 44900, 'Del Fresno', 'Pelicano 2135', 'Mónica', 'Cortes', 'Sánchez'),
  ('LONAS LORENZO, S.A. DE C.V.', 'Jalisco', 44190, 'Moderna', 'Av 8 de julio', 'Hugo', 'Fonseca', 'Alegre'),
  ('JOSÉ RICARDO NISHIMURA TORRES S.A. DE C.V.', 'Tamaulipas', 44200, 'Artesanos', 'Independencia 110', 'Noelia', 'Tovar', 'Periz');
```

```
INSERT 0 7
```

```
Query returned successfully in 37 msec.
```

Inserción de relación proveedor y producto

```
INSERT INTO proveedor_producto (proveedor_razonsocial, producto_codigobarras)
VALUES
('CRECE COMPUTACION, S.A. DE C.V.', 123456789),
('IMPREJAL, S.A. DE C.V.', 987654321),
('ALBE INTERNACIONAL SA DE CV', 456789123),
('PLASTICOS Y METALES MYC, S.A. DE C.V.', 321654987),
('GRUPO AVESTRUZ, S.A DE.C.V. ', 789123456),
('LONAS LORENZO, S.A. DE C.V.', 654987321),
('JOSÉ RICARDO NISHIMURA TORRES S.A. DE C.V.', 987123456);
```

INSERT 0 7

Query returned successfully in 38 msec.

Inserción campo multivaluado teléfono del proveedor:

```
INSERT INTO telefonoproveedor (telefono, proveedor_razonsocial)
VALUES
('3336156444', 'CRECE COMPUTACION, S.A. DE C.V'),
('3338254769', 'IMPREJAL, S.A. DE C.V'),
('3312537000', 'ALBE INTERNACIONAL SA DE CV'),
('5558123402', 'PLASTICOS Y METALES MYC, S.A. DE C.V.'),
('3338106699', 'GRUPO AVESTRUZ, S.A DE.C.V. '),
('3336503541', 'LONAS LORENZO, S.A. DE C.V.'),
('3335630862', 'JOSÉ RICARDO NISHIMURA TORRES S.A. DE C.V.');
```

INSERT 0 7

Query returned successfully in 38 msec.

Inserción tabla venta:

```
INSERT INTO venta (numeroventa, fechaventa, cantidadtotalpagar, cliente_rfc)
VALUES
('VENT-001', '2023-05-01', 500.0, 'MENP010223'),
('VENT-002', '2023-05-02', 400.0, 'MELM830528'),
('VENT-003', '2023-05-03', 225.0, 'LOMP820628'),
('VENT-004', '2023-05-04', 720.0, 'VACS691110'),
('VENT-005', '2023-05-05', 320.0, 'TOGM900603'),
('VENT-006', '2023-05-06', 630.0, 'SAVL852312'),
('VENT-007', '2023-05-07', 120.0, 'PEGJ198305');
```

INSERT 0 7

Query returned successfully in 38 msec.

Inserción relación de venta y artículo

```
INSERT INTO venta_articulo (cantidad, preciotalpagararticulo, articulo_id_articulo, venta_numeroventa)
VALUES
(5, 500.0, 1, 'VENT-001'),
(8, 400.0, 2, 'VENT-002'),
(3, 225.0, 3, 'VENT-003'),
(6, 720.0, 4, 'VENT-004'),
(4, 320.0, 5, 'VENT-005'),
(7, 630.0, 6, 'VENT-006'),
(2, 120.0, 7, 'VENT-007');
```

INSERT 0 7

Query returned successfully in 44 msec.

-De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una compra.

Esta vista contendrá los datos de la factura, incluyendo el número de venta, la fecha de venta, el nombre del cliente, el artículo vendido, la cantidad y el precio total del artículo.

```
CREATE VIEW factura_compra AS
SELECT v.numeroventa, v.fechaventa, c.nombrepila || ' ' || c.appaterno || ' ' || c.apmaterno AS cliente,
a.descripcion AS articulo, va.cantidad, va.preciototalpagararticulo
FROM venta v
JOIN cliente c ON v.cliente_rfc = c.rfc
JOIN venta_articulo va ON v.numeroventa = va.venta_numeroventa
JOIN articulo a ON va.articulo_id_articulo = a.id_articulo;
```

CREATE VIEW

Query returned successfully in 188 msec.

Hacemos la consulta de nuestra vista:

```
SELECT * FROM factura_compra;
```

numeroventa character varying (10)	fechaventa date	cliente text	articulo character varying (50)	cantidad integer	preciotalpagararticulo numeric
VENT-001	2023-05-01	Pablo Meza Nava	Cuaderno Profesional	5	500.0
VENT-002	2023-05-02	Mónica Méndez Luna	Lápiz adhesivo	8	400.0
VENT-003	2023-05-03	Pablo López Morales	Corrector 5mm x 8 mm	3	225.0
VENT-004	2023-05-04	Antonio Rodríguez Vázquez	Carpeta tamaño carta	6	720.0
VENT-005	2023-05-05	Mónica Torres Guzmán	Compás metálico	4	320.0
VENT-006	2023-05-06	Lorena Saldaña Velazquez	Regla aluminio 30cm	7	630.0
VENT-007	2023-05-07	José Pérez García	Grapa estándar	2	120.0

Se observa que se creó de forma correcta la vista la cual asemeja una factura de las compras realizadas.

-Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió y la ganancia correspondiente en esa fecha/periodo.

```
SELECT SUM(va.cantidad) AS cantidad_total, SUM(va.preciototalpagararticulo) AS ganancia_total
FROM venta v
JOIN venta_articulo va ON v.numeroventa = va.venta_numeroventa
WHERE v.fechaventa BETWEEN '2023-05-01' AND '2023-05-31';
```

	cantidad_total bigint	ganancia_total numeric
1	35	2915.0

Para verificar el resultado consultamos el precio de cada artículo

```
SELECT PRECIO FROM ARTICULO;
```

	precio numeric (1000)
1	100
2	50
3	75
4	120
5	80
6	90
7	60

Ahora consultamos la cantidad que cada uno compró de cada artículo

```
SELECT CANTIDAD, VENTA_NUMEROVENTA FROM VENTA_ARTICULO;
```

	cantidad integer	venta_numeroventa character varying (10)
1	5	VENT-001
2	8	VENT-002
3	3	VENT-003
4	6	VENT-004
5	4	VENT-005
6	7	VENT-006
7	2	VENT-007

Multiplicamos el precio de cada artículo por la cantidad que se compró y eso nos da el precio total a pagar

```
SELECT CANTIDAD, PRECIOTOTALPAGARARTICULO FROM VENTA_ARTICULO;
```

	cantidad integer	preciototalpagararticulo numeric
1	5	500.0
2	8	400.0
3	3	225.0
4	6	720.0
5	4	320.0
6	7	630.0
7	2	120.0

Podemos ver que las ventas se hicieron del 01-05-2023 al 07-05-2023 y en la consulta ponemos un rango del 01-05-2021 al 31-05-2023, por lo que entra dentro del rango que se realizaron las ventas.

```
SELECT NUMEROVENTA, FECHAVENTA FROM VENTA;
```

	numeroventa [PK] character varying (10)	fechaventa date
1	VENT-001	2023-05-01
2	VENT-002	2023-05-02
3	VENT-003	2023-05-03
4	VENT-004	2023-05-04
5	VENT-005	2023-05-05
6	VENT-006	2023-05-06
7	VENT-007	2023-05-07

Podemos observar que la cantidad total de artículos vendida fue de 35. Si hacemos la suma de las cantidades que se llevaron los clientes podemos confirmar que es correcto la cantidad de artículos vendida

$$5 + 8 + 3 + 6 + 4 + 7 + 2 =$$

$$35$$

Ahora hacemos la suma del total que se pagó entre todas las ventas.

$$500 + 400 + 225 + 720 + 320 + 630 + 120 =$$

$$2915$$

Se observa que la consulta fue hecha de forma correcta.

-Cada que haya la venta de un artículo, deberá decrementarse el stock por la cantidad vendida de ese artículo. Si el valor llega a cero, abortar la transacción. Si el pedido se completa pero quedan menos de 3 en stock, se deberá emitir una alerta. Debe actualizarse el total a pagar por artículo y el total a pagar por la venta.

```
***CREATE TABLE alerta (  
  id SERIAL PRIMARY KEY,  
  descripcion TEXT,  
  fecha TIMESTAMP DEFAULT NOW()  
);
```

```
CREATE OR REPLACE FUNCTION decrementar_stock_emitir_alerta()  
RETURNS TRIGGER AS $$  
BEGIN  
  -- Decrementar el stock del producto vendido  
  UPDATE producto  
  SET stock = stock - NEW.cantidad  
  WHERE id_articulo = NEW.articulo_id_articulo;  
  
  -- Emitir una alerta si el stock es menor o igual al umbral  
  IF (SELECT stock FROM producto WHERE id_articulo = NEW.articulo_id_articulo) <= 5 THEN  
    INSERT INTO alerta (descripcion) VALUES ('Stock bajo: ' || NEW.articulo_id_articulo);  
  END IF;  
  
  -- Actualizar el total a pagar en la tabla "venta"  
  UPDATE venta  
  SET total_pagar = total_pagar + NEW.preciototalpagararticulo  
  WHERE numeroventa = NEW.venta_numeroventa;  
  
  RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER decrementar_stock_trigger  
AFTER INSERT OR UPDATE ON venta_articulo  
FOR EACH ROW  
EXECUTE FUNCTION decrementar_stock_emitir_alerta();  
***
```

Con esta configuración, cada vez que se inserte o actualice un registro en la tabla "venta_articulo", el disparador se activará y realizará las siguientes acciones:

Decrementará el stock del producto vendido en la tabla "producto".

Emitirá una alerta en la tabla "alerta" si el stock del producto es igual o inferior a 5.

Actualizará el total a pagar en la tabla "venta" sumando el precio total a pagar por el artículo vendido.

-Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock.

Ya que no tenemos productos con un stock menor a 3, modificamos la tabla y nuestro producto con id=7 ya solo tiene 2 en stock.

```
UPDATE producto SET stock = 2 WHERE stock=35;
```

Checamos cual es nuestro artículo con id = 7

```
264 SELECT DESCRIPCION FROM ARTICULO WHERE ID_ARTICULO = 7;
265
```

Data Output Messages Notifications

	descripcion character varying (50) 🔒
1	Grapa estándar

Ahora realizamos nuestra consulta y podemos observar que si nos regresa el nombre de los productos de los cuales hay menos de 3 en stock

```
SELECT descripcion FROM articulo JOIN producto_articulo ON articulo.id_articulo = producto_articulo.articulo_id_articulo
JOIN producto ON producto_articulo.producto_codigobarras = producto.codigobarras
WHERE producto.stock < 3;
```

	descripcion character varying (50) 🔒
1	Grapa estándar

Se comprueba que la consulta ha sido hecha de la manera correcta

-Al recibir el código de barras de un producto, regrese la utilidad.

Fórmula para calcular la utilidad bruta

$$\text{Utilidad bruta} = \text{Ventas totales} - \text{Costo de la mercancía vendida}$$

www.dripcapital.com



Vamos a sacar la utilidad bruta para nuestro producto con código de barras = **123456789**

```
SELECT (va.preciototalpagararticulo - (pr.preciocompra * 5)) AS utilidad
FROM venta_articulo va
JOIN articulo a ON va.articulo_id_articulo = a.id_articulo
JOIN producto_articulo pa ON a.id_articulo = pa.articulo_id_articulo
JOIN producto pr ON pa.producto_codigobarras = pr.codigobarras
WHERE pr.codigobarras = 123456789;
```

	utilidad numeric
1	450.0

Consultamos el id de dicho código de barras

```
SELECT producto_codigobarras, articulo_id_articulo FROM producto_articulo;
```

	producto_codigobarras bigint	articulo_id_articulo integer
1	123456789	1
2	987654321	2
3	456789123	3
4	321654987	4
5	789123456	5
6	654987321	6
7	987123456	7

Se observa que el artículo con código de barras = **123456789** es el artículo con id = **1**

Ahora vemos el precio de compra de dicho artículo con el mismo código de barras

	codigobarras [PK] bigint	preciocompra numeric (1000)
1	123456789	10
2	987654321	20
3	456789123	15
4	321654987	8
5	789123456	12
6	654987321	25
7	987123456	18

Se observa que tuvo un precio de compra = **10**

Ahora también consultamos la cantidad, el precio total y el id del artículo

```
SELECT cantidad, preciototalpagararticulo, articulo_id_articulo FROM venta_articulo;
```

cantidad integer	preciototalpagararticulo numeric	articulo_id_articulo integer
5	500.0	1
8	400.0	2
3	225.0	3
6	720.0	4
4	320.0	5
7	630.0	6
2	120.0	7

Se observa que del artículo con id = 1, qué es el artículo con código de barras = 123456789. El precio total a pagar fue de 500 por que se compraron 5.

Ya teniendo estos datos aplicamos la fórmula para sacar la Utilidad Bruta de dicho producto.

UTILIDAD BRUTA = VENTAS TOTALES – COSTO DE LA MERCANCÍA VENDIDA

El producto tuvo de ventas totales un valor de **500** y el costo de la mercancía fue de 10, pero como fue una cantidad = 5 se multiplica y nuestro costo de la mercancía será de **50**

UTILIDAD BRUTA = 500 – 50 = 450

```
SELECT (va.preciototalpagararticulo - (pr.preciocompra * 5)) AS utilidad
FROM venta_articulo va
JOIN articulo a ON va.articulo_id_articulo = a.id_articulo
JOIN producto_articulo pa ON a.id_articulo = pa.articulo_id_articulo
JOIN producto pr ON pa.producto_codigobarras = pr.codigobarras
WHERE pr.codigobarras = 123456789;
```

utilidad	🔒
numeric	
450.0	

Se comprueba que la consulta fue hecha de la manera correcta

NOTA: Esto nos sirve para cualquier código de barras de nuestra base, lo único que deberíamos cambiar es la cantidad del producto con dicho código de barras que se vendió y modificar el valor por el que se multiplica *pr.preciocompra*.

-Crear al menos, un índice, del tipo que se prefiera y donde se prefiera.

Justificar el porqué de la elección en ambos aspectos

Tipo de índice en PostgreSQL,

Existen varios tipos de índices disponibles, como B-tree, Hash, GIN, GiST, etc. La elección del tipo de índice depende de la naturaleza de los datos y del tipo de consultas que se realizan con mayor frecuencia en la tabla.

-Si los datos son únicos y se utilizan operaciones de igualdad, mayor que o menor que, se puede utilizar un índice B-tree. Este tipo de índice es eficiente para búsquedas exactas y rangos de búsqueda.

-Si los datos son no únicos y se realizan consultas de igualdad o verificación de membresía, se puede considerar el uso de un índice Hash.

-Si los datos son complejos o de texto completo, se pueden utilizar índices GIN o GiST.

En el caso de la columna "rfc" de la tabla "cliente", si se espera realizar búsquedas de igualdad en esta columna, un índice B-tree sería una elección adecuada.


Se creó el índice en la columna "rfc" de la tabla "cliente". Esto significa que el índice se utilizará para acelerar las consultas que involucren búsquedas o comparaciones en la columna "rfc".

Para comprobar la eficiencia del índice elegido realizaremos 2 consultas utilizando la cláusula EXPLAIN ANALYZE, la cual nos dará la información detallada sobre el plan de ejecución de la consulta y los tiempos de ejecución reales antes de crear al índice y después de haber creado el índice.

CONSULTAS ANTES DE LA CREACIÓN DEL ÍNDICE


1.-Esta consulta ordenará todos los registros de la tabla “cliente” según su rfc

```
EXPLAIN ANALYZE SELECT * FROM cliente ORDER BY rfc;
```

QUERY PLAN	
text	
Sort (cost=16.39..16.74 rows=140 width=544) (actual time=0.033..0.034 rows=7 loops=1)	
Sort Key: rfc	
Sort Method: quicksort Memory: 25kB	
-> Seq Scan on cliente (cost=0.00..11.40 rows=140 width=544) (actual time=0.009..0.009 rows=7 loops=1)	
Planning Time: 1.484 ms	
Execution Time: 0.054 ms	

2.-Ésta consulta buscará el registro específico de nuestra tabla cliente con rfc = “MENP010223”

```
EXPLAIN ANALYZE SELECT * FROM cliente WHERE rfc = 'MENP010223';
```

QUERY PLAN	
text	
Index Scan using cliente_pkey on cliente (cost=0.14..8.16 rows=1 width=544) (actual time=0.019..0.020 rows=1 loops=1)	
Index Cond: ((rfc)::text = 'MENP010223'::text)	
Planning Time: 0.060 ms	
Execution Time: 0.033 ms	

-Creamos el índice:

```
CREATE INDEX idx_cliente_rfc ON cliente (rfc);
```

```
CREATE INDEX
```

```
Query returned successfully in 133 msec.
```

CONSULTAS DESPUÉS DE LA CREACIÓN DEL ÍNDICE

```
EXPLAIN ANALYZE SELECT * FROM cliente ORDER BY rfc;
```

Output Messages Notifications



QUERY PLAN

text

Sort (cost=1.17..1.19 rows=7 width=544) (actual time=0.033..0.034 rows=7 loops=1)

Sort Key: rfc

Sort Method: quicksort Memory: 25kB

-> Seq Scan on cliente (cost=0.00..1.07 rows=7 width=544) (actual time=0.010..0.011 rows=7 loops=1)

Planning Time: 1.461 ms

Execution Time: 0.045 ms

```
EXPLAIN ANALYZE SELECT * FROM cliente WHERE rfc = 'MENP010223';
```

Output Messages Notifications



QUERY PLAN

text

Seq Scan on cliente (cost=0.00..1.09 rows=1 width=544) (actual time=0.015..0.016 rows=1 loops=1)

Filter: ((rfc)::text = 'MENP010223'::text)

Rows Removed by Filter: 6

Planning Time: 0.086 ms

Execution Time: 0.030 ms

Tabla comparativa de tiempos de ejecución

Tiempo de ejecución en Ms	Consulta 1	Consulta 2
Antes de índice	0.054	0.033
Después de índice	0.045	0.030

Podemos observar que al crear el índice, el tiempo de ejecución de las consultas es más rápido y podemos concluir que se cumple con el punto requerido.

Conclusiones

Meza Nava Pablo:

A lo largo de la realización del proyecto me fui dando cuenta del reto que fue lograr terminar el mismo. En lo personal, lo primero que realicé fue un análisis de los requerimientos solicitados en el documento que se encuentra en el GitHub. Una vez realizado este análisis, logré identificar entidades, atributos, tipos de atributos, relaciones y por último cardinalidades. Lo más difícil para mi fueron las cardinalidades pero al final logré idéntico cardinalidades mínimas y máximas para las relaciones. Lo siguiente era realizar el MR para así poder generar un código que iba a ser la base de nuestro proyecto hecho en Postgresql. Este punto lo realizó otro compañero del equipo. Ya que se logró el MR empecé a realizar inserciones de registros para cada tabla, (7 registros para cada tabla) cabe mencionar que la mayoría de los registros son generados de manera aleatoria, pero, todos cumplen con uno de los puntos más importantes de las bases de datos que es la congruencia. Cuando logré meter registros a cada tabla , lo siguiente era realizar cada punto que nos pedían los requerimientos. Esta parte fue todo un reto para mi , ya que tuve que aplicar todo lo aprendido tanto en el laboratorio como en la clase de teoría, pero al final logré todos los puntos menos uno. El punto que no logré fue el punto se debe a que no tuve mi práctica de PL SQL en el laboratorio y por más que investigue no pude encontrar una solución al mismo. Fuera de eso muchos de los requerimientos que venían en el documento de especificaciones se parecían un poco a ejercicios de prácticas que hice en el laboratorio. Y esto me ayudó mucho para poder realizar cada punto Puedo concluir que la realización de este proyecto fue aplicar todo lo aprendido a lo largo del curso tanto en teoría como en laboratorio, y hasta cierto punto fue divertido y satisfactorio el hecho de saber que aprendí bien los fundamentos de las bases de datos.

Rendon Hernandez Roberto:

El diseño e implementación de la base de datos ha permitido a la cadena de papelerías mejorar la manera en que almacena y gestiona su información. Mediante la creación de tablas, definición de relaciones y uso de elementos como stored procedures, triggers y funciones, se logra un almacenamiento eficiente, mantenimiento de la integridad de los datos y automatización de tareas clave. La base de datos proporciona una estructura sólida para gestionar proveedores, clientes, productos y ventas, cumpliendo con los objetivos establecidos. Esto brinda a la cadena de papelerías una herramienta confiable y escalable para tomar decisiones informadas y llevar a cabo sus operaciones de manera efectiva.

Rodriguez Estrella Mairol Elizabeth: La creación de este proyecto representó un reto muy grande para mi, pues desde la organización hasta la creación del documento latex intente cosas nuevas que jamas habia hecho antes, creo que lo más complicado de todo fue aprender a usar latex, pues es una herramienta nueva que nunca antes había logrado utilizar, además todo esto me permitió profundizar los conocimientos obtenidos en la materia a lo largo del semestre así como todo lo aprendido en el laboratorio.