



---

# Universidad Nacional Autónoma de México

## Facultad de Ingeniería

# Propuesta de Diseño e Implementación del Sistema de Información para Papelería

*“Soluciones Inteligentes para la Gestión de Datos”*

**Equipo desarrollador: PapyrusDB Solutions**

**Integrantes:**

- Hernández Iríneo Jorge Manuel
- Hernández Martínez Cecilia Sayuri
- Zamora Ayala Antonio Manuel
- Zuñiga Arreguin Emilio

**Profesor:** Ing. Fernando Arreola Franco

**Semestre:** 2026-1

**Lugar:** Ciudad Universitaria, CDMX

**Fecha de entrega:** 6 de diciembre de 2025

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Plan de trabajo</b>	<b>3</b>
2.1. Actividades planeadas . . . . .	3
2.2. Distribución del trabajo por integrante . . . . .	4
2.3. Cronograma . . . . .	6
<b>3. Diseño</b>	<b>7</b>
3.1. Análisis de requerimientos . . . . .	7
3.2. Modelo conceptual (MER) . . . . .	7
3.3. Modelo lógico (MR) . . . . .	9
<b>4. Implementación</b>	<b>14</b>
4.1. Creación de tablas . . . . .	14
4.2. Triggers . . . . .	40
4.3. Funciones . . . . .	50
4.4. Vistas . . . . .	52
<b>5. Aplicación</b>	<b>54</b>
5.1. Dashboard 1: Ingresos Totales Mensuales por producto . . . . .	54
5.2. Dashboard 2: Rendimiento de los Vendedores . . . . .	55
5.3. Procedimiento para la creación del dashboard . . . . .	56
5.4. Dashboard: Ingresos totales mensuales por producto . . . . .	57
5.5. Dashboard: Rendimiento de vendedores . . . . .	59
<b>6. Conclusiones</b>	<b>60</b>

# 1. Introducción

La presente propuesta surge ante la necesidad de una papelería para modernizar la administración de la información y transacciones relacionadas con proveedores, clientes, productos, inventario y ventas. Hasta el momento, estos procesos se realizan de forma manual, lo cual, dificulta tener el control del inventario, mantener el control de ventas, generación de facturas y garantizar un control adecuado de las transacciones diarias.

Para solucionar estas problemáticas, se desarrolló un sistema basado en una base de datos relacional que permite almacenar, organizar y procesar la información de forma eficiente y automatizada. El proyecto se estructura en las siguientes fases:

1. **Análisis de requerimientos:** Identificación de las necesidades dentro de las reglas de negocio, compuestas por las relaciones entre almacenamiento de datos de proveedores, clientes, productos y ventas.
2. **Diseño de la base de datos:** Elaboración del modelo conceptual y lógico, asegurando el cumplimiento de los niveles de normalización y consistencia de la información.
3. **Implementación en PostgreSQL:** Creación del modelo físico mediante scripts, para la generación de tablas, relaciones y restricciones, junto con la inserción de datos de prueba.
4. **Automatización y funcionalidades avanzadas:** Desarrollo de procedimientos almacenados, triggers, vistas y funciones en PostgreSQL para gestionar la actualización automática del stock, la generación de facturas, el cálculo de utilidades, las alertas de inventario bajo, las consultas de productos con bajo stock de inventario y las consultas de ventas y ganancias por período.
5. **Dashboard opcional:** Construcción de una interfaz visual que permita a los usuarios de la base, monitorear datos esenciales para el negocio, como ingresos mensuales por artículo.

En este documento se presenta de manera detallada cada una de estas etapas, incluyendo los modelos MER y MR generados, el código implementado y las decisiones tomadas para la creación del diseño justificadas. Con esto, se ofrece una solución integral, escalable y alineada con los objetivos de la empresa, garantizando el control y la eficiencia en sus operaciones diarias.

## 2. Plan de trabajo

El plan de trabajo del proyecto se gestionó utilizando la plataforma Jira, una herramienta ampliamente utilizada en la industria del desarrollo de software para la administración de proyectos bajo metodologías ágiles. La elección de Jira se basó en sus múltiples ventajas, entre las que destacan su capacidad para organizar tareas mediante tableros Kanban o Scrum, asignar responsables, establecer prioridades y dar seguimiento al avance en tiempo real. Esto permitió que el equipo mantuviera una comunicación clara, una distribución equilibrada de actividades y una supervisión precisa del progreso de cada fase del proyecto. Gracias a Jira, fue posible estructurar adecuadamente las actividades planeadas, documentar evidencias y asegurar que cada integrante colaborara de manera coordinada y eficiente.

### 2.1. Actividades planeadas

Para la planificación del proyecto se definió una estructura de trabajo basada en EPICs, la cual permitió organizar las actividades en fases claras y manejables. Cada EPIC agrupa un conjunto de tareas relacionadas que facilitan la trazabilidad, la asignación de responsabilidades y el seguimiento dentro de Jira. Esta metodología aseguró un flujo de trabajo ordenado, desde el análisis inicial hasta la entrega del producto final.

A continuación, se describen las actividades planeadas según cada EPIC establecido:

- **EPIC 1: Análisis de Requerimientos y Diseño de la Base de Datos.** Incluyó la identificación de entidades, atributos y relaciones del sistema, la elaboración del Modelo Entidad–Relación (MER) y su transformación al Modelo Relacional (MR). También se consideraron procesos de validación y depuración del diseño conceptual y lógico.
- **EPIC 2: Implementación de la Base de Datos (DDL).** Comprendió la creación de tablas, llaves primarias y foráneas, restricciones y dominios. Se definieron índices estratégicos y se realizó la ejecución completa del script para asegurar su funcionamiento sin errores.
- **EPIC 3: Carga de Datos y Pruebas Iniciales (DML).** Contempló la preparación de los datos de prueba, la inserción en tablas principales y dependientes, la validación de integridad referencial y la corrección de inconsistencias detectadas durante el proceso de carga.

- **EPIC 4: Programación de la Lógica de Negocio.** Incluyó la creación de triggers, funciones almacenadas y vistas para cubrir los requerimientos del sistema. Se realizaron pruebas exhaustivas y ajustes derivados de los resultados obtenidos.
- **EPIC 5: Dashboard y Visualización de Datos.** Consistió en la definición de métricas, elaboración de consultas SQL, creación de gráficas y construcción del tablero analítico final. Se verificó la consistencia entre los datos presentados y la información almacenada en la base de datos.
- **EPIC 6: Documento Final del Proyecto.** Comprendió la redacción del documento en L<sup>A</sup>T<sub>E</sub>X, la integración de diagramas, evidencias, scripts, gráficas y conclusiones. Su objetivo fue generar un informe formal, claro y técnicamente completo.
- **EPIC 7: Presentación Final para el Cliente.** Englobó la elaboración del guion de presentación, preparación de diapositivas, diseño de la demostración del sistema y la definición de mensajes clave orientados al cliente.

Estas actividades fueron organizadas y monitoreadas dentro de Jira mediante tableros Scrum, permitiendo visualizar el avance por sprint, las tareas pendientes, en proceso y completadas, así como facilitar la colaboración entre los integrantes del equipo.

## 2.2. Distribución del trabajo por integrante

La distribución del trabajo se realizó de manera equitativa entre los cuatro integrantes del equipo, procurando que cada uno participara en todas las fases del proyecto. Aunque las actividades específicas fueron asignadas individualmente mediante Jira, la colaboración se mantuvo transversal: todos contribuyeron en el análisis, diseño, implementación, pruebas y documentación. De acuerdo con el panel de carga de trabajo de Jira, cada miembro asumió aproximadamente el 25 % de las actividades, garantizando un reparto equilibrado y acorde con la metodología Scrum utilizada en el proyecto.



Figura 1: Carga de trabajo para cada integrante del equipo

A continuación, se presenta una descripción general del aporte realizado por cada integrante:

- **Hernández Irineo Jorge Manuel.** Participó activamente en la organización y documentación del proyecto, contribuyendo al análisis inicial del sistema, al diseño de consultas para el dashboard y a la elaboración de la estructura base del documento en L<sup>A</sup>T<sub>E</sub>X. Asimismo, apoyó en la integración de contenidos, generación de la versión final del documento y definición de los objetivos de la presentación al cliente. Su participación fue clave para mantener la coherencia técnica y la coordinación general del proyecto.
- **Hernández Martínez Cecilia Sayuri.** Desempeñó un papel fundamental en las actividades relacionadas con la carga de datos y la manipulación de información, incluyendo la preparación del dataset inicial, la inserción de datos y la validación de su integridad. También desarrolló funciones dentro de la lógica de negocio, colaboró en la redacción de secciones del documento y participó en la creación de materiales para la presentación final. Su labor fortaleció la consistencia de los datos y la claridad del proyecto.
- **Zamora Ayala Antonio Manuel.** Colaboró ampliamente en la programación y validación de la lógica de negocio del sistema, realizando pruebas, ajustes y correcciones necesarios para asegurar su correcto funcionamiento. También participó en la construcción y validación del dashboard, integrando las diversas gráficas y métricas del proyecto. Adicionalmente, contribuyó al desarrollo de la documentación técnica

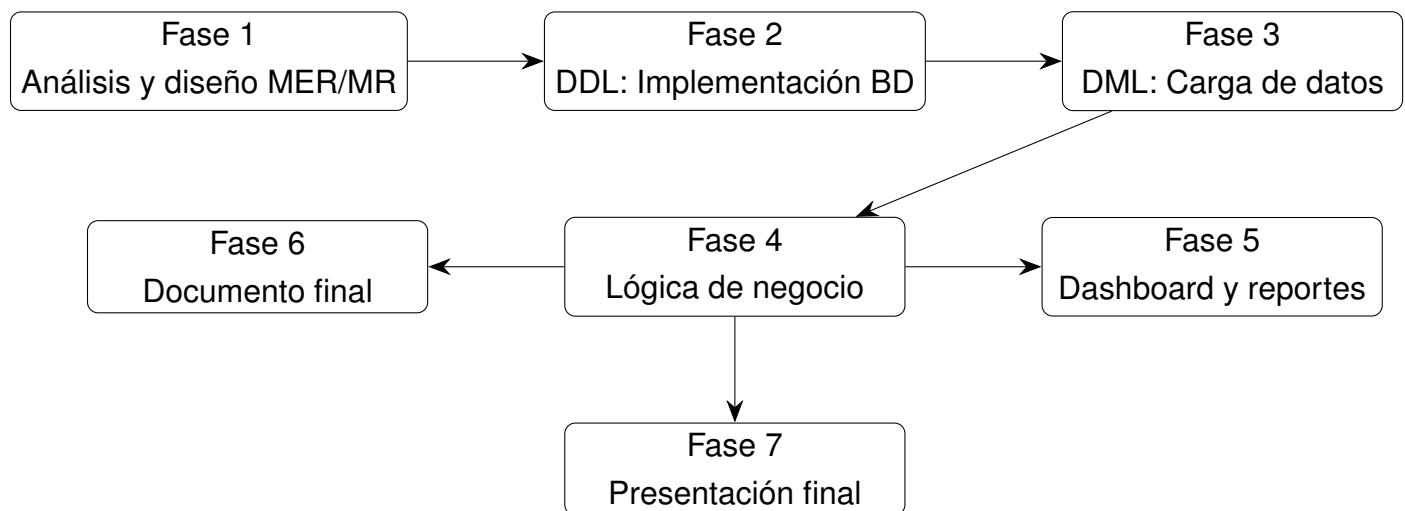
y a la redacción de materiales para la presentación. Su aporte fue esencial para la solidez técnica del sistema.

- **Zuñiga Arreguín Emilio.** Se encargó principalmente de las actividades relacionadas con la implementación de la base de datos, incluyendo la creación de tablas, definición de llaves, restricciones e índices. También desarrolló vistas y consultas necesarias para distintos apartados del sistema, además de colaborar en la integración de modelos dentro del documento final. Participó en el diseño de consultas para el dashboard y en la generación de evidencia del funcionamiento del sistema. Su trabajo fue crucial para la estructura y funcionamiento del proyecto.

### 2.3. Cronograma

El desarrollo del proyecto se organizó en una secuencia de etapas que permitieron avanzar de manera estructurada y progresiva. El cronograma general del trabajo realizado fue el siguiente:

- Fase 1** → Análisis de requerimientos y diseño del MER/MR
- Fase 2** → Implementación de la base de datos (DDL)
- Fase 3** → Carga de datos y verificaciones iniciales
- Fase 4** → Programación de la lógica de negocio
- Fase 5** → Dashboard y reportes
- Fase 6** → Documento final del proyecto
- Fase 7** → Presentación final



Este cronograma resume el avance del proyecto en orden cronológico, permitiendo visualizar la evolución de cada etapa desde la conceptualización hasta la entrega final.

### **3. Diseño**

#### **3.1. Análisis de requerimientos**

En esta fase se identificó toda la información que el sistema necesita manejar y se analizaron los procesos principales de la papelería (entrega y venta de productos, almacenamiento en el inventario, detalles de las ventas). A partir de este análisis se determinaron las entidades, atributos y relaciones fundamentales para representar el funcionamiento del negocio.

#### **3.2. Modelo conceptual (MER)**

Ya identificadas las entidades, los atributos y las relaciones, tenemos lo siguiente:

##### **Entidades principales.**

- **Proveedor.** Almacena a los proveedores de la papelería.
- **Producto.** Representa los artículos que se venden y los tipos que existen (papelaría, recargas, impresiones y regalos).
- **Inventario.** Controla existencias, costos de compra y disponibilidad de cada producto.
- **Empleado.** Trabajadores responsables de realizar las ventas.
- **Cliente.** Personas que realizan compras.
- **Venta.** Registro general de una transacción efectuada de un cliente con uno o varios productos.

Además, se identificaron atributos multivaluados que requieren tratamiento especial:

- **telProveedor.** Un proveedor puede tener uno o varios números telefónicos.
- **emailCliente.** Un cliente puede contar con múltiples correos electrónicos.

##### **Relaciones unificadas.**

- Un proveedor puede tener varios teléfonos.
- Varios proveedores puede entregar varios productos (relación M:M).

- Un producto puede aparecer en varios registros de inventario a lo largo del tiempo.
- Un cliente puede tener varios correos electrónicos, pero siempre deberá tener al menos uno.
- Un cliente realiza ventas que deben ser atendidas por un empleado.
- Una venta está compuesta por uno o varios detalles de venta.

La fase de requerimientos permitió establecer con claridad qué información se debe gestionar y cómo interactúan las diferentes entidades, así como construir el Modelo Entidad/Relación que representa de mejor manera el sistema de la papelería.

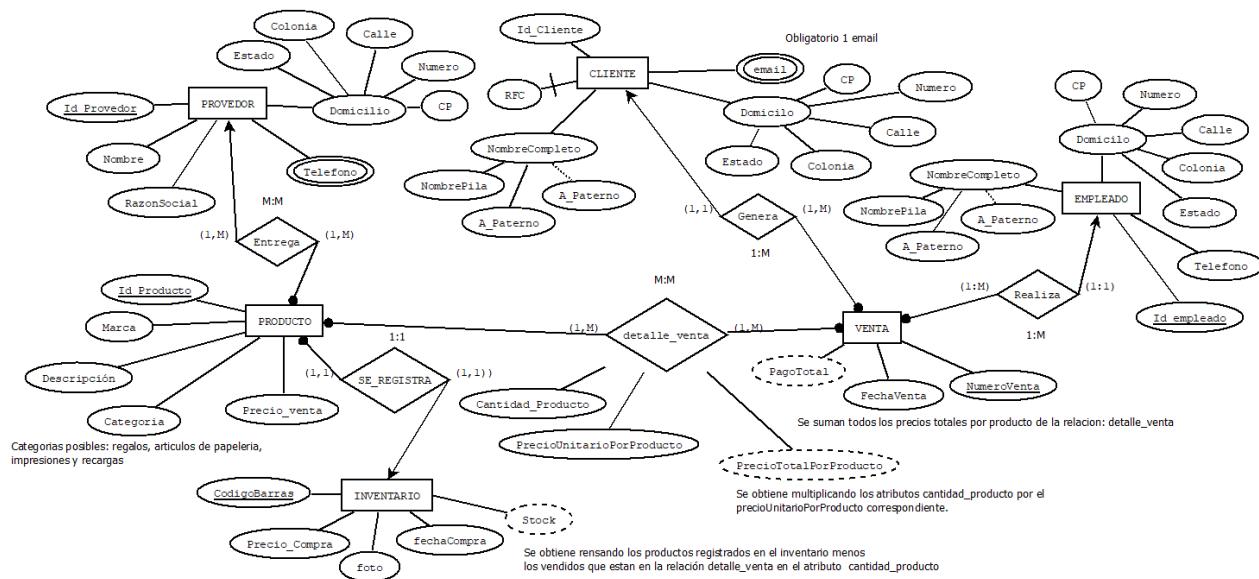


Figura 2: Modelo Entidad/Relación (MER)

### 3.3. Modelo lógico (MR)

Con base en el Modelo Entidad/Relación de la etapa anterior, dimos inicio a la construcción de nuestro Modelo Relacional (MR), el cual, se dividió en 4 etapas: identificación de PK's y FK's, tratamiento de atributos multivaluados y creación de nuevas tablas, descomposición de atributos compuestos, elección de los tipos de datos y restricciones.

#### 1. Identificación de PK's y FK's.

- **Proveedor.** PK —> *idProveedor*
- **Producto.** PK —> *idProducto*
- **Inventario.** PK —> *codigoBarras*  
FK —> *idProducto*
- **Empleado.** PK —> *idEmpleado*
- **Cliente.** PK —> *idCliente*
- **Venta.** PK —> *idVenta*  
FK's —> *idCliente, idEmpleado*

#### 2. Tratamiento de atributos multivaluados y creación de nuevas tablas.

Dentro de nuestro MER existen 2 relaciones M:M y 2 atributos multivaluados, por lo que es necesario crear 4 tablas nuevas.

- **DetalleVenta.** Especificación de los productos incluidos en cada venta (cantidad del producto, precio unitario y precio total).  
PK —> *idProducto (FK), idVenta (FK)*
- **Entrega.** Registro de productos entregados por proveedores.  
PK —> *idProveedor (FK), idProducto (FK)*
- **TelProveedor.**  
PK —> *idTelefono*  
FK —> *idProveedor*
- **EmailCliente.**  
PK —> *idEmail*  
FK —> *idCliente*

#### 3. Descomposición de atributos compuestos.

- **Nombre.** Tanto el nombre del cliente como el del empleado están compuestos por: nombre, apellido paterno y apellido materno (siendo opcional este último).
- **Domicilio.** El domicilio del proveedor y del cliente están compuestos por: estado, colonia, calle, número y CP.

#### 4. Elección de los tipos de datos y restricciones.

Una vez teníamos todos los atributos atómicos, seleccionamos el tipo de dato que iba a tener, de igual manera, verificamos si existían algunas restricciones (CHECK, NOT NULL, NULL, etc...) para dichos atributos.

Campo	Tipo de dato
ID_PROVEEDOR	VARCHAR(15) NOT NULL
NOMBRE	VARCHAR(50) NOT NULL
RAZON_SOCIAL	VARCHAR(25) NOT NULL
ESTADO	VARCHAR(50) NOT NULL
COLONIA	VARCHAR(30) NOT NULL
CALLE	VARCHAR(75) NOT NULL
NUMERO	INTEGER NOT NULL
CP	INTEGER NOT NULL

Cuadro 1: Tabla PROVEEDOR

Campo	Tipo de dato
ID_TELEFONO	VARCHAR(15) NOT NULL
TELEFONO	INTEGER NOT NULL
ID_PROVEEDOR (FK)	VARCHAR(15) NOT NULL

Cuadro 2: Tabla TEL\_PROVEEDOR

Campo	Tipo de dato
ID_PROVEEDOR (FK)	VARCHAR(15) NOT NULL
ID_PRODUCTO (FK)	VARCHAR(15) NOT NULL

Cuadro 3: Tabla ENTREGA

Campo	Tipo de dato
ID_PRODUCTO	VARCHAR(15) NOT NULL
MARCA	VARCHAR(40) NOT NULL
DESCRIPCION	TEXT NOT NULL
PRECIO_VENTA	MONEY(18,0) NOT NULL
CATEGORIA	VARCHAR(30) NOT NULL

Cuadro 4: Tabla PRODUCTO

Campo	Tipo de dato
CODIGO_BARRAS	VARCHAR(25) NOT NULL
PRECIO_COMPRA	MONEY(40,0) NOT NULL
FOTO	PICTURE NOT NULL
FECHA_COMPRA	DATE NOT NULL
STOCK	INTEGER NOT NULL
ID_PRODUCTO (FK)	VARCHAR(15) NOT NULL

Cuadro 5: Tabla INVENTARIO

Campo	Tipo de dato
ID_CLIENTE	VARCHAR(15) NOT NULL
RFC	VARCHAR(15) NOT NULL
NOMBRE	VARCHAR(50) NOT NULL
AP_PAT	VARCHAR(30) NOT NULL
AP_MAT	VARCHAR(30) NULL
ESTADO	VARCHAR(30) NOT NULL
COLONIA	VARCHAR(30) NOT NULL
CALLE	VARCHAR(75) NOT NULL
NUMERO	INTEGER NOT NULL
CP	INTEGER NOT NULL

Cuadro 6: Tabla CLIENTE

Campo	Tipo de dato
ID_EMAIL	VARCHAR(15) NOT NULL
EMAIL	VARCHAR(75) NOT NULL
ID_CLIENTE (FK)	VARCHAR(15) NOT NULL

Cuadro 7: Tabla EMAIL\_CLIENTE

Campo	Tipo de dato
ID_EMPLEADO	VARCHAR(15) NOT NULL
NOMBRE	VARCHAR(50) NOT NULL
AP_PAT	VARCHAR(30) NOT NULL
AP_MAT	VARCHAR(30) NULL
ESTADO	VARCHAR(50) NOT NULL
COLONIA	VARCHAR(30) NOT NULL
CALLE	VARCHAR(75) NOT NULL
NUMERO	INTEGER NOT NULL
CP	INTEGER NOT NULL
TELEFONO	INTEGER NOT NULL

Cuadro 8: Tabla EMPLEADO

Campo	Tipo de dato
ID_VENTA	VARCHAR(15) NOT NULL
FECHA_VENTA	DATE NOT NULL
PAGO_TOTAL	MONEY(40,0) NOT NULL
ID_CLIENTE (FK)	VARCHAR(15) NOT NULL
ID_EMPLEADO (FK)	VARCHAR(15) NULL

Cuadro 9: Tabla VENTA

<b>Campo</b>	<b>Tipo de dato</b>
ID_PRODUCTO (FK)	VARCHAR(15) NOT NULL
ID_VENTA (FK)	VARCHAR(15) NOT NULL
CANTIDAD_PRODUCTO	INTEGER NOT NULL
PRECIO_UNITARIO_POR_PRODUCTO	MONEY(40,0) NOT NULL
PRECIO_TOTAL_POR_PRODUCTO	MONEY(40,0) NOT NULL

Cuadro 10: Tabla DETALLE\_VENTA

El Modelo Relacional (MR) final para la papelería lo realizamos en *ERStudio* y es el siguiente:

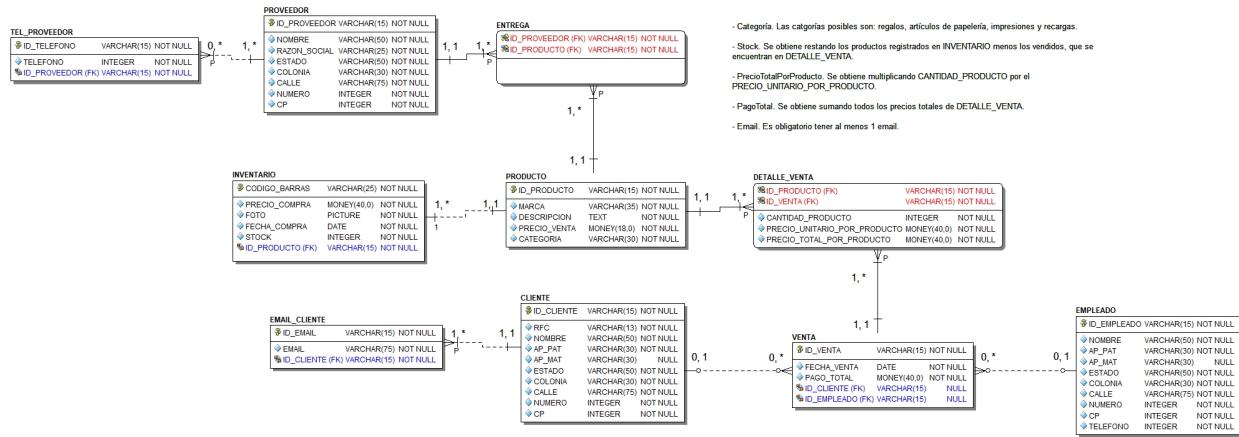


Figura 3: Modelo Relacional (MR)

## 4. Implementación

Utilizamos PostgreSQL como manejador de base de datos (DBMS) para este proyecto, por lo que todo el código SQL tiene esa sintaxis. A continuación mostraremos y describiremos los scripts que usamos para la construcción y funcionamiento óptimo de la papelería. Como primer paso creamos la base de datos llamada *PapyrusDB Solutions*.

```
CREATE DATABASE PapyrusDBSolutions
WITH OWNER = postgres
ENCODING = 'UTF8'
TEMPLATE = template1;
```

### 4.1. Creación de tablas

Basándonos en el Modelo Relacional (MR) hecho con anterioridad, creamos las tablas correspondientes, haciendo uso de los tipos de datos y de las restricciones puestas con anterioridad.

Para la tabla **proveedor**, tenemos lo siguiente:

```
CREATE TABLE proveedor (
    idProveedor VARCHAR(15) PRIMARY KEY,
    nombre VARCHAR(45) NOT NULL,
    razonSocial VARCHAR(80) NOT NULL,
    estado VARCHAR(30) NOT NULL,
    colonia VARCHAR(45) NOT NULL,
    calle VARCHAR(45) NOT NULL,
    numero INTEGER NOT NULL,
    cp INTEGER NOT NULL
);
```

Para la tabla **telProveedor**, tenemos lo siguiente:

```
CREATE TABLE telProveedor (
    idTelefono VARCHAR(15) PRIMARY KEY,
    idProveedor VARCHAR(15) NOT NULL,
    telefono VARCHAR(15) NOT NULL,
    CONSTRAINT telProveedorFkProv
    FOREIGN KEY (idProveedor)
    REFERENCES proveedor(idProveedor)
);
```

Para la tabla **producto**, tenemos lo siguiente:

```
CREATE TABLE producto (
    idProducto VARCHAR(15) PRIMARY KEY,
    marca VARCHAR(30) NOT NULL,
    descripcion TEXT NOT NULL,
    precioVenta NUMERIC(10,2) NOT NULL,
    categoria VARCHAR(30) NOT NULL
);
```

Para la tabla **inventario**, tenemos lo siguiente:

```
CREATE TABLE inventario (
    codigoBarras VARCHAR(15) PRIMARY KEY,
    idProducto VARCHAR(15) NOT NULL,
    precioCompra NUMERIC(10,2) NOT NULL,
    foto BYTEA,
    fechaCompra DATE NOT NULL,
    stock INTEGER NOT NULL,
    CONSTRAINT inventarioFkProd
    FOREIGN KEY (idProducto)
    REFERENCES producto(idProducto)
);
```

Para la tabla **empleado**, tenemos lo siguiente:

```
CREATE TABLE empleado (
    idEmpleado VARCHAR(15) PRIMARY KEY,
    nombre VARCHAR(30) NOT NULL,
    apPat VARCHAR(30) NOT NULL,
    apMat VARCHAR(30),
    estado VARCHAR(30) NOT NULL,
    colonia VARCHAR(45) NOT NULL,
    calle VARCHAR(45) NOT NULL,
    numero INTEGER NOT NULL,
    cp INTEGER NOT NULL,
    telefono VARCHAR(15) NOT NULL
);
```

Para la tabla **cliente**, tenemos lo siguiente:

```
CREATE TABLE cliente (
    idCliente VARCHAR(15) PRIMARY KEY,
    rfc VARCHAR(13),
    nombre VARCHAR(30) NOT NULL,
    apPat VARCHAR(30) NOT NULL,
    apMat VARCHAR(30),
    estado VARCHAR(30) NOT NULL,
    colonia VARCHAR(45) NOT NULL,
    calle VARCHAR(45) NOT NULL,
    numero INTEGER NOT NULL,
    cp INTEGER NOT NULL
);
```

Para la tabla **emailCliente**, tenemos lo siguiente:

```
CREATE TABLE emailCliente (
    idEmail VARCHAR(18) PRIMARY KEY,
    idCliente VARCHAR(15) NOT NULL,
    email VARCHAR(75) NOT NULL,
    CONSTRAINT emailClienteFkCli
    FOREIGN KEY (idCliente)
    REFERENCES cliente(idCliente)
);
```

Para la tabla **venta**, tenemos lo siguiente:

```
CREATE TABLE venta (
    idVenta VARCHAR(15) PRIMARY KEY,
    idCliente VARCHAR(15) NOT NULL,
    idEmpleado VARCHAR(15) NOT NULL,
    fechaVenta DATE NOT NULL,
    pagoTotal NUMERIC(10,2) NOT NULL,
    CONSTRAINT ventaFkCli
        FOREIGN KEY (idCliente)
        REFERENCES cliente(idCliente),
    CONSTRAINT ventaFkEmp
        FOREIGN KEY (idEmpleado)
        REFERENCES empleado(idEmpleado)
);
```

Para la tabla **detalleVenta**, tenemos lo siguiente:

```
CREATE TABLE detalleVenta (
    idDetalleVenta VARCHAR(15) PRIMARY KEY,
    idProducto VARCHAR(15) NOT NULL,
    idVenta VARCHAR(15) NOT NULL,
    cantidadProducto INTEGER NOT NULL,
    precioUnitarioPorProd NUMERIC(10,2) NOT NULL,
    precioTotalPorProd NUMERIC(10,2) NOT NULL,
    CONSTRAINT detvFkProd
        FOREIGN KEY (idProducto)
        REFERENCES producto(idProducto),
    CONSTRAINT detvFkVenta
        FOREIGN KEY (idVenta)
        REFERENCES venta(idVenta)
);
```

Para la tabla **entrega**, tenemos lo siguiente:

```

CREATE TABLE entrega (
    idEntrega VARCHAR(15) PRIMARY KEY,
    idProveedor VARCHAR(15) NOT NULL,
    idProducto VARCHAR(15) NOT NULL,
    fechaEntrega DATE NOT NULL,
    CONSTRAINT entregaFkProv
        FOREIGN KEY (idProveedor)
        REFERENCES proveedor(idProveedor),
    CONSTRAINT entregaFkProd
        FOREIGN KEY (idProducto)
        REFERENCES producto(idProducto)
);

```

Posteriormente hicimos uso de la sentencia **ALTER** para agregar los constraints correspondientes. Los cuáles son los siguientes:

- Para hacer el código de 4 letras en **producto**:

```

– Código de 4 letras

ALTER TABLE producto
ALTER COLUMN categoria TYPE VARCHAR(4);

ALTER TABLE producto
ALTER COLUMN categoria SET NOT NULL;

ALTER TABLE producto
ADD CONSTRAINT productoCategoriaNew
CHECK (categoria IN ('REGA','PAPE','IMPR','RECA'));

```

- Para corroborar que en ningún momento haya números negativos (stock, precios, cantidades) en **producto**, **inventario**, **detalleVenta** y **venta**:

– Números no negativos

```

ALTER TABLE producto
ADD CONSTRAINT productoPrecioVentaPosNew
CHECK (precioVenta > 0);

ALTER TABLE inventario
ADD CONSTRAINT inventarioPrecioCompraPosNew
CHECK (precioCompra > 0);

ALTER TABLE inventario
ADD CONSTRAINT inventarioStockNoNegNew
CHECK (stock >= 0);

ALTER TABLE detalleVenta
ADD CONSTRAINT detVentaCantidadPosNew
CHECK (cantidadProducto > 0);

ALTER TABLE detalleVenta
ADD CONSTRAINT detVentaPrecioUnitPosNew
CHECK (precioUnitarioPorProd > 0);

ALTER TABLE detalleVenta
ADD CONSTRAINT detVentaPrecioTotalNoNegNew
CHECK (precioTotalPorProd >= 0);

ALTER TABLE venta
ADD CONSTRAINT ventaPagoTotalNoNegNew
CHECK (pagoTotal >= 0);

```

- Para que el formato de venta sea: *VENT-001*, *VENT-002*, etc... En **venta**:

```
– Formato de ventas  
ALTER TABLE venta  
ADD CONSTRAINT ventaldFormatoNew  
CHECK (idVenta ~'^VENT-[0-9]{3}$');
```

Lo anterior forma parte del *Data Definition Language (DDL)*.

Una vez que nuestras tablas están correctamente creadas (con los constraints correspondientes), pasamos con el *Data Manipulation Language (DML)*. Comenzamos agregando información mediante la sentencia **INSERT INTO** seguido del nombre de nuestra tabla y finalizando con la sentencia **VALUES**. Primero agregamos información a nuestras tablas principales:

- Para la tabla **proveedor**:

```
INSERT INTO proveedor VALUES  
(‘PROV-001’, ‘Exel del Norte’, ‘Exel del Norte S.A. de C.V.’, ‘Nuevo León’,  
‘Cumbres 1er Sector’, ‘Av. Lázaro Cárdenas’, 18, 64920),  
(‘PROV-002’, ‘CT Internacional’, ‘CT Internacional del Noroeste S.A. de C.V.’,  
‘CDMX’, ‘Coapa’, ‘Calz Acoxpa’, 40, 14390),  
(‘PROV-003’, ‘PCH Mayoreo’, ‘PCH Mayoreo S.A. de C.V.’, ‘CDMX’, ‘Doctores’,  
‘Dr. José María Vértiz’, 43, 06720),  
(‘PROV-004’, ‘DC Mayorista’, ‘Distribuidor de Cómputo Mayorista S.A. de C.V.’,  
‘Jalisco’, ‘Jardines del Bosque’, ‘Av. Mariano Otero’, 98, 44520);
```

- Para la tabla **producto**:

```

INSERT INTO producto VALUES

('PROD-001', 'Casio', 'Calculadora científica FX-95MS', 299.00, 'PAPE'),
('PROD-002', 'Casio', 'Calculadora científica FX-570MS', 450.00, 'PAPE'),
('PROD-003', 'HP', 'Calculadora científica HP10s', 380.00, 'PAPE'),
('PROD-004', 'Manhattan', 'Cable USB-C reforzado 1m', 120.00, 'PAPE'),
('PROD-005', 'Logitech', 'Mouse inalámbrico M185', 259.00, 'PAPE'),
('PROD-006', 'Canon', 'Impresión blanco y negro por hoja', 1.50, 'IMPR'),
('PROD-007', 'HP', 'Impresión color por hoja', 3.00, 'IMPR'),
('PROD-008', 'PlotterPro', 'Impresión tamaño tabloide', 15.00, 'IMPR'),
('PROD-009', 'LaserPrint', 'Escaneo de documentos por hoja', 2.00, 'IMPR'),
('PROD-010', 'FotoExpress', 'Impresión fotográfica 10x15', 6.00, 'IMPR'),
('PROD-011', 'Telcel', 'Recarga electrónica 50 pesos', 50.00, 'RECA'),
('PROD-012', 'Movistar', 'Recarga electrónica 100 pesos', 100.00, 'RECA'),
('PROD-013', 'AT&T', 'Recarga electrónica 150 pesos', 150.00, 'RECA'),
('PROD-014', 'RegalosMX', 'Taza personalizada de cerámica', 150.00, 'REGA'),
('PROD-015', 'PaperGift', 'Agenda 2025 edición premium', 220.00, 'REGA'),
('PROD-016', 'Aurora', 'Peluche de animal de 15 cm', 95.00, 'REGA'),
('PROD-017', 'Creativa', 'Llavero metálico grabado', 40.00, 'REGA'),
('PROD-018', 'DecoArt', 'Tarjeta de regalo decorativa', 25.00, 'REGA');

```

- Para la tabla **empleado**:

```

INSERT INTO empleado VALUES
('EMP-001', 'Alejandro', 'Campos', 'Rodriguez', 'CDMX', 'Narvarte Poniente',
'La Quemada', 120, 03020, '5556123001'),
('EMP-002', 'Jorge', 'Martínez', 'Huerta', 'CDMX', 'Real del Moral',
'Río Tlalpenco', 55, 06760, '5557348822'),
('EMP-003', 'Sandra', 'Hernández', 'Castro', 'CDMX', 'Del Valle',
'Pilares', 78, 03100, '5588447733'),
('EMP-004', 'Luis', 'Hernández', 'Mendoza', 'CDMX', 'Nápoles',
'Arizona', 86, 03810, '5578329911'),
('EMP-005', 'Ricardo', 'Mendoza', 'Gómez', 'CDMX', 'Condesa',
'Atlixco', 90, 06170, '5567329988');

```

■ Para la tabla **cliente**:

```

INSERT INTO cliente VALUES
('CLI-001', 'RALJ900215HMX', 'José', 'Ramírez', 'López', 'CDMX',
'Portales Norte', 'Filadelfia', 87, 03303),
('CLI-002', 'GOGG920101HMX', 'Gabriel', 'González', 'García', 'EdoMex',
'Las Américas', 'Av. Central', 76, 55070),
('CLI-003', 'CALL940522HMX', 'Luis', 'Cárdenas', 'López', 'CDMX',
'Iztacalco', 'Sur 16', 10, 08900),
('CLI-004', 'GUOA880311HMX', 'Alejandra', 'Gutiérrez', 'Ortiz', 'CDMX',
'Roma Norte', 'Zacatecas', 65, 06700),
('CLI-005', 'SERI950715HMX', 'Erick', 'Serrano', 'Ríos', 'EdoMex',
'Ciudad Azteca', 'Av. Hank González', 25, 55120);

```

Después agregamos la información a las tablas dependientes (aquellas que tienen FK's y PK's compuestas) siguiendo la misma sintaxis:

- Para la tabla **telProveedor**:

```
INSERT INTO telProveedor VALUES  
(‘TEL-001’, ‘PROV-001’, ‘5554889067’),  
(‘TEL-002’, ‘PROV-002’, ‘5522106060’),  
(‘TEL-003’, ‘PROV-003’, ‘5557896500’),  
(‘TEL-004’, ‘PROV-004’, ‘5584044408’);
```

- Para la tabla **emailCliente**:

```
INSERT INTO emailCliente VALUES  
(‘EMA-001’, ‘CLI-001’, ‘jose.ramirez@gmail.com’),  
(‘EMA-002’, ‘CLI-002’, ‘gab.gon.ga@hotmail.com’),  
(‘EMA-003’, ‘CLI-003’, ‘luis.cardenas@outlook.com’),  
(‘EMA-004’, ‘CLI-004’, ‘maria.gtz@gmail.com’),  
(‘EMA-005’, ‘CLI-005’, ‘erick.serrano@gmail.com’);
```

- Para la tabla **inventario**:

```
INSERT INTO inventario VALUES  
(‘INV-001’, ‘PROD-001’, 220.00, NULL, ‘2025-01-10’, 40),  
(‘INV-002’, ‘PROD-002’, 330.00, NULL, ‘2025-01-11’, 25),  
(‘INV-003’, ‘PROD-003’, 300.00, NULL, ‘2025-01-12’, 30),  
(‘INV-004’, ‘PROD-004’, 80.00, NULL, ‘2025-01-13’, 60),  
(‘INV-005’, ‘PROD-005’, 180.00, NULL, ‘2025-01-14’, 50);
```

- Para la tabla **entrega**:

```
INSERT INTO entrega (idProveedor, idProducto, fechaEntrega) VALUES  
('PROV-001', 'PROD-001', '2025-11-10'),  
('PROV-001', 'PROD-002', '2025-11-11'),  
('PROV-002', 'PROD-003', '2025-11-12'),  
('PROV-003', 'PROD-004', '2025-11-13'),  
('PROV-004', 'PROD-005', '2025-11-14');
```

- Para la tabla **venta**:

```
INSERT INTO venta VALUES  
('VENT-001', 'CLI-001', 'EMP-001', '2025-12-01', 0),  
('VENT-002', 'CLI-002', 'EMP-002', '2025-12-03', 0),  
('VENT-003', 'CLI-003', 'EMP-003', '2025-12-04', 0),  
('VENT-004', 'CLI-004', 'EMP-004', '2025-12-05', 0),  
('VENT-005', 'CLI-005', 'EMP-005', '2025-12-06', 0),  
('VENT-006', 'CLI-002', 'EMP-005', '2025-12-03', 0),  
('VENT-007', 'CLI-001', 'EMP-003', '2025-12-03', 0);
```

- Para la tabla **detalleVenta**:

```
INSERT INTO detalleVenta (idProducto, idVenta, cantidadProducto) VALUES  
('PROD-001', 'VENT-001', 1),  
('PROD-004', 'VENT-002', 1),  
('PROD-002', 'VENT-003', 1),  
('PROD-003', 'VENT-004', 1),  
('PROD-005', 'VENT-005', 1),  
('PROD-001', 'VENT-006', 1),  
('PROD-004', 'VENT-007', 1);
```

Las inserciones pasadas no fueron las únicas que se realizaron, ya que con el propósito de darle un enfoque más realista a nuestra base de datos, hicimos nuevas inserciones en las tablas **principales** y en las **dependientes** para que al momento de enlazarla con el Dashboard, las gráficas tuvieran mayor sentido. Sin embargo, estas nuevas inserciones consideran los triggers de la sección **4.2 Triggers**, por lo que sería buena idea primero echarles un vistazo. También cabe destacar que algunas de las inserciones que se mostrarán a continuación son sólo un fragmento y no aparecen en su totalidad ya que son muchos datos, pero se pueden encontrar completos en el **SCRIPT\_Inserciones**

Para las tablas principales, las nuevas inserciones fueron las siguientes:

- Para la tabla **cliente**:

```
INSERT INTO cliente (rfc, nombre, apPat, apMat, estado, colonia, calle, numero, cp)  
VALUES  
('ROGL850715HM0','Roberto','Gómez','Lara','CDMX',  
'Roma Norte','Chiapas',123,06700),  
('MAAR920312HM1','María','Arcos','Ramírez','CDMX',  
'Del Valle','Pilares',88,03100),  
('HECJ880501HM2','Héctor','Cruz','Jiménez','CDMX',
```

'Narvarte','Zempoala',54,03020),  
('ALGR900914HM3','Alina','García','Rodríguez','CDMX',  
'Polanco','Homero',610,11550),  
('JOMM950227HM4','Jorge','Martínez','Mora','CDMX',  
'Portales','Necaxa',21,03300),  
('SAHN870903HM5','Sandra','Hernández','Nava','CDMX',  
'Iztacalco','Oriente 116',443,08200),  
('FELP990715HM6','Felipe','López','Paredes','CDMX',  
'Tlalpan','Insurgentes Sur',3855,14000),  
('VIRI991120HM8','Viridiana','Rivas','Ibarra','CDMX',  
'Centro','5 de Mayo',12,06000),  
('ANRS820430HM7','Ana','Ríos','Soto','CDMX',  
'Coyoacán','Miguel Ángel',77,04030),  
('LUGA930112HM8','Luis','García','Álvarez','CDMX',  
'Ajusco','Cumbres',52,04300),  
('BECR910629HM9','Berenice','Castro','Rivas','CDMX',  
'Roma Sur','Tonalá',216,06760),  
('DAVR960110HM1','David','Rangel','Ruiz','CDMX',  
'Obrera','Bolívar',235,06800),  
('SOMA880925HM2','Sofía','Maldonado','Arias','CDMX',  
'Doctores','Dr. Vértiz',94,06720),  
('JUGA900515HM3','Julia','García','Ahumada','CDMX',  
'Lindavista','Monte Albán',30,07300);

- Para la tabla **inventario**:

```

INSERT INTO inventario (codigoBarras, idProducto, precioCompra, foto, fechaCompra,
stock) VALUES

('INV-006', 'PROD-006', 120.00, NULL, '2025-01-01', 35),
('INV-007', 'PROD-007', 90.00, NULL, '2025-01-02', 42),
('INV-008', 'PROD-008', 150.00, NULL, '2025-01-03', 28),
('INV-009', 'PROD-009', 70.00, NULL, '2025-01-04', 55),
('INV-010', 'PROD-010', 45.00, NULL, '2025-01-05', 60),
('INV-011', 'PROD-011', 30.00, NULL, '2025-01-06', 48),
('INV-012', 'PROD-012', 55.00, NULL, '2025-01-07', 32),
('INV-013', 'PROD-013', 80.00, NULL, '2025-01-08', 40),
('INV-014', 'PROD-014', 100.00, NULL, '2025-01-09', 52),
('INV-015', 'PROD-015', 160.00, NULL, '2025-01-10', 33),
('INV-016', 'PROD-016', 85.00, NULL, '2025-01-11', 29),
('INV-017', 'PROD-017', 40.00, NULL, '2025-01-12', 61),
('INV-018', 'PROD-018', 15.00, NULL, '2025-01-14', 70);

```

■ Para la tabla **venta**:

```

INSERT INTO venta (idCliente, idEmpleado, fechaVenta, pagoTotal) VALUES

('CLI-012','EMP-003','2025-01-04',0),
('CLI-034','EMP-001','2025-01-15',0),
('CLI-009','EMP-005','2025-01-22',0),
('CLI-028','EMP-004','2025-01-28',0),
('CLI-017','EMP-002','2025-02-03',0),
('CLI-041','EMP-003','2025-02-10',0),

```

```
('CLI-006','EMP-001','2025-02-14',0),  
('CLI-023','EMP-005','2025-02-27',0),  
('CLI-048','EMP-004','2025-03-02',0),  
('CLI-015','EMP-005','2025-03-09',0),  
('CLI-031','EMP-002','2025-03-15',0),  
('CLI-052','EMP-003','2025-03-26',0),  
('CLI-024','EMP-005','2025-11-02',0),  
('CLI-001','EMP-005','2025-11-07',0),  
('CLI-030','EMP-002','2025-11-15',0),  
('CLI-043','EMP-004','2025-11-22',0),  
('CLI-019','EMP-003','2025-11-27',0),  
('CLI-054','EMP-001','2025-11-30',0);
```

Para las tablas dependientes, las nuevas inserciones fueron las siguientes:

■ Para la tabla **emailCliente**:

```
INSERT INTO emailCliente (idCliente, email) VALUES  
('CLI-006','roberto.gomez@gmail.com'),  
('CLI-007','maria.arcos@gmail.com'),  
('CLI-008','hector.cruz@hotmail.com'),  
('CLI-009','alina.garcia@gmail.com'),  
('CLI-010','jorge.martinez@yahoo.com'),  
('CLI-011','sandra.hernandez@gmail.com'),  
('CLI-012','felipe.lopez@hotmail.com'),  
('CLI-013','ana.rios@gmail.com'),
```

```
('CLI-014','luis.garcia@gmail.com'),  
('CLI-015','berenice.castro@hotmail.com'),  
('CLI-016','julian.paredes@gmail.com'),  
('CLI-017','carla.villarreal@gmail.com'),  
('CLI-018','montserrat.rios@hotmail.com'),  
('CLI-019','edgar.hernandez@gmail.com'),  
('CLI-020','arturo.fuentes@gmail.com');
```

■ Para la tabla **detalleVenta**:

```
– VENT-009 (4 productos)
```

```
INSERT INTO detalleVenta (idVenta, idProducto, cantidadProducto, precioUnitarioPorProd,  
precioTotalPorProd)
```

```
SELECT 'VENT-009','PROD-005',1, p.precioVenta, p.precioVenta*1 FROM producto p  
WHERE p.idProducto='PROD-005'
```

```
UNION ALL
```

```
SELECT 'VENT-009','PROD-006',2, p.precioVenta, p.precioVenta*2 FROM producto p  
WHERE p.idProducto='PROD-006'
```

```
UNION ALL
```

```
SELECT 'VENT-009','PROD-007',3, p.precioVenta, p.precioVenta*3 FROM producto p  
WHERE p.idProducto='PROD-007'
```

```
UNION ALL
```

```
SELECT 'VENT-009','PROD-008',1, p.precioVenta, p.precioVenta*1 FROM producto p  
WHERE p.idProducto='PROD-008';
```

– VENT-010 (venta 'extraordinaria': 7 productos)

```
INSERT INTO detalleVenta (idVenta, idProducto, cantidadProducto, precioUnitarioPorProd,  
precioTotalPorProd)
```

```
SELECT 'VENT-010','PROD-009',2, p.precioVenta, p.precioVenta*2 FROM producto p  
WHERE p.idProducto='PROD-009'
```

```
UNION ALL
```

```
SELECT 'VENT-010','PROD-010',1, p.precioVenta, p.precioVenta*1 FROM producto p  
WHERE p.idProducto='PROD-010'
```

```
UNION ALL
```

```
SELECT 'VENT-010','PROD-011',3, p.precioVenta, p.precioVenta*3 FROM producto p  
WHERE p.idProducto='PROD-011'
```

```
UNION ALL
```

```
SELECT 'VENT-010','PROD-012',1, p.precioVenta, p.precioVenta*1 FROM producto p  
WHERE p.idProducto='PROD-012'
```

```
UNION ALL
```

```
SELECT 'VENT-010','PROD-013',4, p.precioVenta, p.precioVenta*4 FROM producto p  
WHERE p.idProducto='PROD-013'
```

```
UNION ALL
```

```
SELECT 'VENT-010','PROD-014',2, p.precioVenta, p.precioVenta*2 FROM producto p  
WHERE p.idProducto='PROD-014'
```

```
UNION ALL
```

```
SELECT 'VENT-010','PROD-015',1, p.precioVenta, p.precioVenta*1 FROM producto p  
WHERE p.idProducto='PROD-015';
```

Finalizando con el *Data Manipulation Language (DML)*, hacemos uso de las sentencias **UPDATE** y **SELECT**. La sentencia **UPDATE** sirve para actualizar el campo *pagoTotal* de cada venta usando la suma de sus detalles, y si una venta no tiene detalles, entonces se le asigna un 0 en lugar de NULL.

```
UPDATE venta v  
SET pagoTotal = COALESCE((  
    SELECT SUM(d.precioTotalPorProd)  
    FROM detalleVenta d  
    WHERE d.idVenta = v.idVenta  
, 0);
```

Por último, con la sentencia **SELECT \* FROM** podemos observar el contenido de nuestras tablas.

SELECT \* FROM proveedor

Paprytus_Dashboard / Proveedor								Filtro	Resumir
Idproveedor	Nombre	Razonsocial	Estado	Colonia	Calle	Numero	Cp		
PROV-001	Exel del Norte	Exel del Norte S.A. de C.V.	Nuevo León	Cumbres 1er Sector	Av. Lázaro Cárdenas	18	64,920		
PROV-002	CT Internacional	CT Internacional del Noroeste S.A. de C.V.	CDMX	Coapa	Calz Acoxpa	40	14,390		
PROV-003	PCH Mayoreo	PCH Mayoreo S.A. de C.V.	CDMX	Doctores	Dr. José María Vértiz	43	6,720		
PROV-004	DC Mayorista	Distribuidor de Cómputo Mayorista S.A. de C.V.	Jalisco	Jardines del Bosque	Av. Mariano Otero	98	44,520		

SELECT \* FROM producto

Idproducto	Marca	Descripcion	Precioventa	Categoría	+
PROD-001	Casio	Calculadora científica FX-95MS	299	PAPE	
PROD-002	Casio	Calculadora científica FX-570MS	450	PAPE	
PROD-003	HP	Calculadora científica HP10s	380	PAPE	
PROD-004	Manhattan	Cable USB-C reforzado 1m	120	PAPE	
PROD-005	Logitech	Mouse inalámbrico M185	259	PAPE	
PROD-006	Canon	Impresión blanco y negro por hoja	1.5	IMPR	
PROD-007	HP	Impresión color por hoja	3	IMPR	
PROD-008	PlotterPro	Impresión tamaño tabloide	15	IMPR	
PROD-009	LaserPrint	Escaneo de documentos por hoja	2	IMPR	
PROD-010	FotoExpress	Impresión fotográfica 10x15	6	IMPR	
PROD-011	Telcel	Recarga electrónica \$50	50	RECA	
PROD-012	Movistar	Recarga electrónica \$100	100	RECA	

SELECT \* FROM empleado

Idempleado	Nombre	Appat	Apmat	Estado	Colonia	Calle	Numero	Cp	Telefono	Filtro
EMP-001	Alejandro	Campos	Rodriguez	CDMX	Narvarte Poniente	La Quemada	120	3,020	5556123001	
EMP-002	Jorge	Martínez	Huerta	CDMX	Real del Moral	Río Tlalpenco	55	6,760	5557348822	
EMP-003	Sandra	Hernández	Castro	CDMX	Del Valle	Pilares	78	3,100	5588447733	
EMP-004	Luis	Hernández	Mendoza	CDMX	Nápoles	Arizona	86	3,810	5578329911	
EMP-005	Ricardo	Mendoza	Gómez	CDMX	Condesa	Atlixco	90	6,170	5567329988	

SELECT \* FROM cliente

Paprytus_Dashboard / Cliente											
Idcliente	Rfc	Nombre	Appat	Apmat	Estado	Colonia	Calle	Numero	Cp		
CLI-001	RAIJ900215HMX	José	Ramírez	López	CDMX	Portales Norte	Filadelfia	87	3,303		
CLI-002	GOGG920101HMX	Gabriel	González	García	EdoMex	Las Américas	Av. Central	76	55,070		
CLI-003	CALL940522HMX	Luis	Cárdenas	López	CDMX	Iztacalco	Sur 16	10	8,900		
CLI-004	GUOA880311HMX	Alejandra	Gutiérrez	Ortiz	CDMX	Roma Norte	Zacatecas	65	6,700		
CLI-005	SERI950715HMX	Erick	Serrano	Ríos	EdoMex	Ciudad Azteca	Av. Hank González	25	55,120		
CLI-006	ROGL850715HM0	Roberto	Gómez	Lara	CDMX	Roma Norte	Chiapas	123	6,700		
CLI-007	MAAR920312HM1	Maria	Arcos	Ramírez	CDMX	Del Valle	Pilares	88	3,100		
CLI-008	HECJ880501HM2	Héctor	Cruz	Jiménez	CDMX	Narvarte	Zempoala	54	3,020		
CLI-009	ALGR900914HM3	Alina	García	Rodríguez	CDMX	Polanco	Homero	610	11,550		
CLI-010	JOMM950227HM4	Jorge	Martínez	Mora	CDMX	Portales	Necaxa	21	3,300		
CLI-011	SAHN870903HM5	Sandra	Hernán...	Navá	CDMX	Iztacalco	Oriente 116	443	8,200		
CLI-012	FELP990715HM6	Felipe	López	Paredes	CDMX	Tlalpan	Insurgentes Sur	3,855	14,000		

SELECT \* FROM telProveedor

Paprytus_Dashboard / Telproveedor		
Idtelefono	Idproveedor	Telefono
TEL-001	PROV-001	5554889067
TEL-002	PROV-002	5522106060
TEL-003	PROV-003	5557896500
TEL-004	PROV-004	5584044408

SELECT \* FROM emailCliente

Papyrus_Dashboard / Emailcliente		
Idemail	Idcliente	Email
EMA-001	CLI-001	jose.ramirez@gmail.com
EMA-002	CLI-002	gab.gon.ga@hotmail.com
EMA-003	CLI-003	luis.cardenas@outlook.com
EMA-004	CLI-004	maria.gtz@gmail.com
EMA-005	CLI-005	erick.serrano@gmail.com
EMA-006	CLI-006	roberto.gomez@gmail.com
EMA-007	CLI-007	maria.arcos@gmail.com
EMA-008	CLI-008	hector.cruz@hotmail.com
EMA-009	CLI-009	alina.garcia@gmail.com
EMA-010	CLI-010	jorge.martinez@yahoo.com
EMA-011	CLI-011	sandra.hernandez@gmail.com
EMA-012	CLI-012	felipe.lopez@hotmail.com

SELECT \* FROM inventario

Codigobarras	Idproducto	Preciocompra	Foto	Fechacompra	Stock	+
INV-008	PROD-008	150		enero 3, 2025	22	
INV-009	PROD-009	70		enero 4, 2025	46	
INV-010	PROD-010	45		enero 5, 2025	53	
INV-011	PROD-011	30		enero 6, 2025	41	
INV-012	PROD-012	55		enero 7, 2025	26	
INV-013	PROD-013	80		enero 8, 2025	30	
INV-014	PROD-014	100		enero 9, 2025	43	
INV-015	PROD-015	160		enero 10, 2025	25	
INV-016	PROD-016	85		enero 11, 2025	25	
INV-017	PROD-017	40		enero 12, 2025	55	
INV-018	PROD-018	15		enero 14, 2025	64	
INV-001	PROD-001	220		enero 10, 2025	30	

SELECT \* FROM entrega

Idproveedor	Idproducto	Fechaentrega	+
PROV-001	PROD-001	noviembre 10, 2025	
PROV-001	PROD-002	noviembre 11, 2025	
PROV-002	PROD-003	noviembre 12, 2025	
PROV-003	PROD-004	noviembre 13, 2025	
PROV-004	PROD-005	noviembre 14, 2025	

SELECT \* FROM venta

Paprytus_Dashboard / Venta					
	Idventa	Idcliente	Idempleado	Fechaventa	Pagototal
	VENT-001	CLI-001	EMP-001	diciembre 1, 2025	299
	VENT-002	CLI-002	EMP-002	diciembre 3, 2025	120
	VENT-003	CLI-003	EMP-003	diciembre 4, 2025	450
	VENT-004	CLI-004	EMP-004	diciembre 5, 2025	380
	VENT-005	CLI-005	EMP-005	diciembre 6, 2025	259
1	VENT-006	CLI-002	EMP-005	diciembre 3, 2025	299
	VENT-007	CLI-001	EMP-003	diciembre 3, 2025	120
	VENT-023	CLI-027	EMP-005	abril 29, 2025	0
	VENT-024	CLI-033	EMP-002	mayo 2, 2025	0
	VENT-025	CLI-014	EMP-005	mayo 8, 2025	0
	VENT-026	CLI-045	EMP-003	mayo 19, 2025	0
	VENT-027	CLI-011	EMP-001	mayo 28, 2025	0

```
SELECT * FROM detalleVenta
```

### Dashboard / Detalleventa

Idproducto	Idventa	Cantidadproducto	Preciounitarioporprod	Preciototalporprod	+
PROD-001	VENT-001	1	299	299	
PROD-004	VENT-002	1	120	120	
PROD-002	VENT-003	1	450	450	
PROD-003	VENT-004	1	380	380	
PROD-005	VENT-005	1	259	259	
PROD-001	VENT-006	1	299	299	
PROD-004	VENT-007	1	120	120	
PROD-001	VENT-008	2	299	598	
PROD-002	VENT-008	1	450	450	
PROD-003	VENT-008	3	380	1,140	
PROD-004	VENT-008	1	120	120	
PROD-005	VENT-009	1	259	259	

## 4.2. Triggers

Hicimos uso de diversos triggers, algunos de ellos se ejecutan antes de comenzar con la inserción de datos y otros después de insertar datos. Los triggers que se ejecutan antes de insertar datos son los siguientes:

- El trigger **trg\_email\_minimo** garantiza que cada cliente tenga al menos un email.

```
CREATE OR REPLACE FUNCTION validar_email_minimo()
```

```
RETURNS TRIGGER AS $$
```

```
DECLARE
```

```
totalEmails INT;
```

```

BEGIN

    – Contar cuántos emails tiene el cliente asociado

    SELECT COUNT(*) INTO totalEmails

    FROM emailCliente

    WHERE idCliente = OLD.idCliente;

    – Si solo existía un correo, no permitir eliminarlo

    IF totalEmails = 1 THEN

        RAISE EXCEPTION 'No se puede eliminar el único email del cliente %', OLD.idCliente;

    END IF;

    RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_email_minimo

BEFORE DELETE ON emailCliente

FOR EACH ROW

EXECUTE FUNCTION validar_email_minimo();

```

- El trigger **before\_insert\_detalleVenta** verifica si el producto existe en el inventario, checando y actualizando el stock, así como obtener el precio unitario del producto y calculando el precio total del mismo. También nos envía una alerta cuando hay stock bajo (debajo de 3 unidades).

```

CREATE OR REPLACE FUNCTION trg_detalleVenta_before_insert()

RETURNS TRIGGER AS $$

DECLARE

```

```

stock_actual INT;

precio_unitario NUMERIC(10,2);

nuevo_stock INT;

BEGIN

    – Obtener stock actual del inventario

    SELECT stock INTO stock_actual

    FROM inventario

    WHERE idProducto = NEW.idProducto;

    IF stock_actual IS NULL THEN

        RAISE EXCEPTION 'El producto % no existe en inventario.',

        NEW.idProducto;

    END IF;

    – Calcular el nuevo stock

    nuevo_stock := stock_actual - NEW.cantidadProducto;

    – Validar si queda sin stock o negativo

    IF nuevo_stock <= 0 THEN

        RAISE EXCEPTION 'Stock insuficiente para vender el producto %',

        NEW.idProducto;

    END IF;

    – Alertar si queda menos de 3

    IF nuevo_stock < 3 THEN

        RAISE NOTICE 'ALERTA: Poco stock restante para el producto %

        (quedan % unidades).',

        NEW.idProducto, nuevo_stock;

```

```

END IF;

– Actualizar stock en inventario

UPDATE inventario

SET stock = nuevo_stock

WHERE idProducto = NEW.idProducto;

– Obtener precio de venta del producto

SELECT precioVenta INTO precio_unitario

FROM producto

WHERE idProducto = NEW.idProducto;

– Forzar el precio unitario desde producto

NEW.precioUnitarioPorProd := precio_unitario;

– Calcular el precio total por producto

NEW.precioTotalPorProd := NEW.cantidadProducto * precio_unitario;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

```

```
CREATE TRIGGER before_insert_detalleVenta
```

```
BEFORE INSERT ON detalleVenta
```

```
FOR EACH ROW
```

```
EXECUTE FUNCTION trg_detalleVenta_before_insert();
```

- Los triggers **trg\_generar\_id\_proveedor**, **trg\_generar\_id\_producto**, **trg\_generar\_id\_empleado**, **trg\_generar\_id\_cliente**, **trg\_generar\_id\_telproveedor**, **trg\_generar\_id\_email**, **trg\_generar\_id\_inventario**, **trg\_generar\_id\_venta** garantizan que aunque se ingrese un ID NULL, automáticamente se generará un ID que siga la

secuencia de los ID's existentes.

```
CREATE SEQUENCE seq_proveedor START WITH 5;

CREATE OR REPLACE FUNCTION generar_id_proveedor()
RETURNS TRIGGER AS $$

BEGIN

IF NEW.idProveedor IS NULL THEN

NEW.idProveedor := 'PROV-' || LPAD(NEXTVAL('seq_proveedor')::TEXT, 3, '0');

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_generar_id_proveedor
BEFORE INSERT ON proveedor
FOR EACH ROW
EXECUTE FUNCTION generar_id_proveedor();
```

```
CREATE SEQUENCE seq_producto START WITH 19;

CREATE OR REPLACE FUNCTION generar_id_producto()
RETURNS TRIGGER AS $$

BEGIN

IF NEW.idProducto IS NULL THEN

NEW.idProducto := 'PROD-' || LPAD(NEXTVAL('seq_producto')::TEXT, 3, '0');

END IF;

RETURN NEW;
```

```
END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_generar_id_producto
BEFORE INSERT ON producto
FOR EACH ROW
EXECUTE FUNCTION generar_id_producto();
```

```
CREATE SEQUENCE seq_empleado START WITH 6;
CREATE OR REPLACE FUNCTION generar_id_empleado()
RETURNS TRIGGER AS $$

BEGIN

IF NEW.idEmpleado IS NULL THEN

NEW.idEmpleado := 'EMP-' || LPAD(NEXTVAL('seq_empleado')::TEXT, 3, '0');

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_generar_id_empleado
BEFORE INSERT ON empleado
FOR EACH ROW
EXECUTE FUNCTION generar_id_empleado();
```

```
CREATE SEQUENCE seq_cliente START WITH 6;
CREATE OR REPLACE FUNCTION generar_id_cliente()
```

```

RETURNS TRIGGER AS $$

BEGIN IF NEW.idCliente IS NULL THEN

NEW.idCliente := 'CLI-' || LPAD(NEXTVAL('seq_cliente')::TEXT, 3, '0');

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_generar_id_cliente
BEFORE INSERT ON cliente
FOR EACH ROW
EXECUTE FUNCTION generar_id_cliente();

```

```

CREATE SEQUENCE seq_telprove START WITH 5;

CREATE OR REPLACE FUNCTION generar_id_telproveedor()
RETURNS TRIGGER AS $$

BEGIN

IF NEW.idTelefono IS NULL THEN

NEW.idTelefono := 'TEL-' || LPAD(NEXTVAL('seq_telprove')::TEXT, 3, '0');

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_generar_id_telproveedor
BEFORE INSERT ON telProveedor

```

```
FOR EACH ROW
```

```
EXECUTE FUNCTION generar_id_telproveedor();
```

```
CREATE SEQUENCE seq_email START WITH 6;  
  
CREATE OR REPLACE FUNCTION generar_id_email()  
RETURNS TRIGGER AS $$  
  
BEGIN IF NEW.idEmail IS NULL THEN  
  
NEW.idEmail := 'EMA-' || LPAD(NEXTVAL('seq_email')::TEXT, 3, '0');  
  
END IF;  
  
RETURN NEW;  
  
END;  
  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER trg_generar_id_email  
BEFORE INSERT ON emailCliente  
FOR EACH ROW  
EXECUTE FUNCTION generar_id_email();
```

```
CREATE SEQUENCE seq_inventario START WITH 6;
```

```
CREATE OR REPLACE FUNCTION generar_id_inventario()
```

```
RETURNS TRIGGER AS $$
```

```
BEGIN
```

```
IF NEW.codigoBarras IS NULL THEN
```

```
NEW.codigoBarras := 'INV-' || LPAD(NEXTVAL('seq_inventario')::TEXT, 3, '0');
```

```
END IF;
```

```

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_generar_id_inventario
BEFORE INSERT ON inventario
FOR EACH ROW
EXECUTE FUNCTION generar_id_inventario();

```

```

CREATE SEQUENCE seq_venta START WITH 8;

CREATE OR REPLACE FUNCTION generar_id_venta()
RETURNS TRIGGER AS $$

BEGIN

IF NEW.idVenta IS NULL THEN

NEW.idVenta := 'VENT-' || LPAD(NEXTVAL('seq_venta')::TEXT, 3, '0');

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_generar_id_venta
BEFORE INSERT ON venta
FOR EACH ROW
EXECUTE FUNCTION generar_id_venta();

```

Pasando con los triggers que se ejecutan después de la inserción de datos en las tablas, tenemos los siguientes:

- El trigger **after\_insert\_detalleVenta** actualiza el *pagoTotal* de una venta después de agregar un producto, recalculando el total de la venta.

```

CREATE OR REPLACE FUNCTION trg_detalleVenta_after_insert()
RETURNS TRIGGER AS $$

DECLARE
    suma NUMERIC(10,2);

BEGIN
    - - Sumar todos los detalles asociados a esta venta
    SELECT SUM(precioTotalPorProd)
    INTO suma
    FROM detalleVenta
    WHERE idVenta = NEW.idVenta;
    - - Actualizar pagoTotal en venta
    UPDATE venta
    SET pagoTotal = suma
    WHERE idVenta = NEW.idVenta;
    RETURN NEW;
END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER after_insert_detalleVenta
AFTER INSERT ON detalleVenta
FOR EACH ROW
EXECUTE FUNCTION trg_detalleVenta_after_insert();

```

### 4.3. Funciones

Dentro de las funciones, nosotros hicimos uso de 3, las cuales se mostrarán a continuación:

1. **periodo\_ventas\_ganancias** recibe dos parámetros: 'fechalinicio' y 'fechaFin'. Posteriormente nos regresa la cantidad total que se vendió y la ganancia correspondiente al periodo entre esas dos fechas.

```
CREATE FUNCTION periodo_ventas_ganancias(fechalnicio DATE, fechaFin DATE)
RETURNS TABLE(total_vendido NUMERIC, ganancia NUMERIC) AS $$

BEGIN

RETURN QUERY

SELECT

SUM(detalleVenta.precioTotalPorProd),

SUM((producto.precioVenta - inventario.precioCompra) * detalleVenta.cantidadProducto)

FROM venta

JOIN detalleVenta ON venta.idVenta = detalleVenta.idVenta

JOIN producto ON detalleVenta.idProducto = producto.idProducto

JOIN inventario ON producto.idProducto = inventario.idProducto

WHERE venta.fechaVenta BETWEEN fechalinicio AND fechaFin;

END;

$$ LANGUAGE plpgsql;

-- SELECT para ver las ventas en el periodo del mes de diciembre

SELECT * FROM periodo_ventas_ganancias('2025-12-01','2025-12-31')
```

2. **bajo\_stock** nos permite tener el control de los artículo que tienen un bajo stock, mostrando el id y el nombre de dichos artículos.

```

CREATE FUNCTION bajo_stock()

RETURNS TABLE(idProducto VARCHAR(15), Nombre TEXT, Stock INTEGER) AS $$

BEGIN

RETURN QUERY

SELECT producto.idProducto, producto.descripcion, inventario.stock

FROM inventario

JOIN producto ON inventario.idProducto = producto.idProducto

WHERE inventario.stock <3;

END;

$$ LANGUAGE plpgsql;

-- SELECT para ver todos los productos con bajo stock

SELECT * FROM bajo_stock()

```

3. **obtener\_utilidad** recibe como parámetro el código de barras correspondiente a un producto existente en el inventario y si no se encuentra ningún artículo, se manda una excepción indicando que el producto no existe.

```

CREATE FUNCTION obtener_utilidad(codBarras VARCHAR(15))

RETURNS NUMERIC(10,2) AS $$

DECLARE

utilidad NUMERIC(10,2);

BEGIN

SELECT (producto.precioVenta - inventario.precioCompra)

INTO utilidad

```

```

FROM inventario

JOIN producto ON inventario.idProducto = producto.idProducto

WHERE inventario.codigoBarras = codBarras;

IF utilidad IS NULL THEN

RAISE EXCEPTION 'No existe producto con código de barras %', codBarras;

END IF;

RETURN utilidad;

END;

$$ LANGUAGE plpgsql;

-- SELECT para obtener el producto que tiene un código de barras 'INV-001'

SELECT FROM obtener_utilidad('INV-001')

```

#### 4.4. Vistas

Creamos la vista **factura\_venta** para que a partir de un *idVenta* se simule una ticket de venta, el cual contiene: datos de la venta, datos del cliente, productos vendidos y total de la venta.

```

CREATE VIEW factura_venta AS

SELECT

— Datos de la venta

venta.idVenta,

venta.fechaVenta,

venta.pagoTotal,

— Datos del cliente

cliente.idCliente,

```

```

cliente.nombre AS clienteNombre,
cliente.apPat AS clienteApellidoPaterno,
cliente.apMat AS clienteApellidoMaterno,
cliente.estado AS clienteEstado,
cliente.colonia AS clienteColonia,
cliente.calle AS clienteCalle,
cliente.numero AS clienteNumero,
cliente.cp AS clienteCP,
— Productos vendidos
producto.idProducto,
producto.marca,
producto.descripcion,
detalleVenta.cantidadProducto,
detalleVenta.precioUnitarioPorProd,
detalleVenta.precioTotalPorProd
FROM venta
JOIN cliente ON venta.idCliente = cliente.idCliente
JOIN detalleVenta ON detalleVenta.idVenta = venta.idVenta
JOIN producto ON detalleVenta.idProducto = producto.idProducto;

SELECT * FROM factura_venta
WHERE idVenta = 'VENT-001';

```

## **5. Aplicación**

En el presente proyecto se definieron dos dashboards orientados al monitoreo y análisis del desempeño operativo de la papelería. Cada dashboard fue diseñado a partir de los requerimientos funcionales, considerando las necesidades de información del administrador. A continuación, se describen las métricas seleccionadas y se justifica su importancia dentro del sistema.

### **5.1. Dashboard 1: Ingresos Totales Mensuales por producto**

El primer dashboard está enfocado en el análisis financiero de los productos, permitiendo visualizar de manera clara el comportamiento de las ventas a lo largo de un año determinado. El objetivo de este dashboard es ofrecer una herramienta que facilite la toma de decisiones sobre la gestión de inventarios, el rendimiento de cada artículo y la rentabilidad de las categorías de productos.

Las métricas seleccionadas fueron elegidas porque permiten identificar patrones de demanda, detectar productos altamente rentables y evaluar el comportamiento general del negocio en distintos períodos. Asimismo, las métricas permiten visualizar tendencias mensuales, lo cual es esencial para planear compras, reposición de stock y estrategias comerciales.

## Resumen de Métricas del Dashboard 1

Métrica	Descripción	Tablas de Origen
Ingresos mensuales por producto	Monto total generado por cada artículo en cada mes del año seleccionado.	venta, detalle_venta, producto
Producto más rentable	Identificación del artículo con mayores ingresos acumulados durante el año.	detalle_venta, producto
Ingresos mensuales totales	Suma total de ingresos obtenidos por el negocio cada mes del año.	venta, detalle_venta
Categoría más rentable	Categoría de productos con mayor contribución financiera en el periodo.	producto, detalle_venta

Cuadro 42: Resumen de métricas del Dashboard 1

## 5.2. Dashboard 2: Rendimiento de los Vendedores

El segundo dashboard se centra en el análisis del desempeño de los vendedores, ofreciendo indicadores clave sobre la cantidad de ventas realizadas y la utilidad generada por cada empleado. Este dashboard permite evaluar el rendimiento individual, identificar al vendedor con mejor desempeño y detectar posibles áreas de mejora en el equipo de ventas.

Las métricas fueron seleccionadas con el fin de proporcionar visibilidad sobre el impacto directo de cada vendedor en los ingresos del negocio, permitiendo tomar decisiones relacionadas con asignación de turnos, bonificaciones, capacitación o estrategias comerciales personalizadas.

## Resumen de Métricas del Dashboard 2

Métrica	Descripción	Tablas de Origen
Número de ventas por vendedor	Cantidad total de ventas realizadas por cada empleado en el periodo.	venta, empleado
Ingresos generados por vendedor	Suma total de ingresos obtenidos por cada vendedor según las ventas registradas.	venta, detalle_venta, empleado
Mejor vendedor del periodo	Empleado con mayor monto total generado durante el periodo seleccionado.	venta, empleado

Cuadro 43: Resumen de métricas del Dashboard 2

### 5.3. Procedimiento para la creación del dashboard

Para la construcción del dashboard del proyecto se eligió Metabase como herramienta de análisis y visualización de datos, debido a su integración nativa con bases de datos PostgreSQL, su facilidad de uso y su capacidad para generar reportes sin necesidad de programación adicional. Metabase permite conectarse directamente al motor de base de datos, ejecutar consultas, interpretar resultados y generar visualizaciones de manera intuitiva. Esta elección resultó adecuada para el proyecto, ya que agiliza la creación de indicadores clave y facilita la interpretación de la información obtenida a partir del sistema desarrollado.

El proceso seguido para la elaboración del dashboard consistió en las siguientes etapas:

- **Preparación de vistas y consultas SQL en PostgreSQL.** Se construyeron vistas especializadas que reúnen y procesan información proveniente de distintas tablas. Dichas vistas permiten obtener métricas consolidadas como ventas totales, ingresos por vendedor, productos más vendidos o variaciones mensuales. Su uso garantiza coherencia, reduce la complejidad de las consultas y facilita la integración con Metabase.
- **Conexión de Metabase con la base de datos.** Se estableció la conexión entre Metabase y PostgreSQL ingresando las credenciales correspondientes. Esto habi-

litó el acceso directo a las tablas y vistas del sistema, permitiendo que Metabase reconociera automáticamente la estructura y los datos disponibles.

- **Creación de consultas dentro de Metabase.** Una vez conectada la base de datos, se generaron las consultas necesarias para alimentar cada visualización. Algunas se apoyaron directamente en las vistas creadas en PostgreSQL, mientras que otras fueron construidas mediante consultas personalizadas dentro de Metabase, de acuerdo con los indicadores definidos en la etapa de diseño del dashboard.
- **Generación de gráficas y visualizaciones.** Para cada consulta, se seleccionó el tipo de gráfica más adecuado (barras, líneas, pastel, tablas dinámicas, etc.), ajustando ejes, filtros y parámetros de visualización. Este proceso permitió crear representaciones claras e interpretables de las métricas más relevantes para el negocio.
- **Integración y diseño del dashboard final.** Finalmente, todas las visualizaciones se organizaron dentro de un único dashboard en Metabase. Se estructuraron por secciones según su propósito, se ajustaron tamaños y posiciones, y se añadieron títulos y descripciones para facilitar la comprensión del usuario final. El resultado fue un tablero funcional, intuitivo y capaz de mostrar en tiempo real el desempeño y comportamiento de las ventas.

Este procedimiento permitió transformar los datos almacenados en la base de datos en un conjunto de indicadores visuales útiles para la toma de decisiones, fortaleciendo la interpretación y el análisis del sistema desarrollado.

#### **5.4. Dashboard: Ingresos totales mensuales por producto**

A continuación se presentan capturas de las visualizaciones generadas en Metabase como parte del dashboard del sistema. Estas imágenes muestran la correcta ejecución de las consultas SQL, la integración de las vistas y la presentación final de los indicadores definidos para el análisis del negocio.

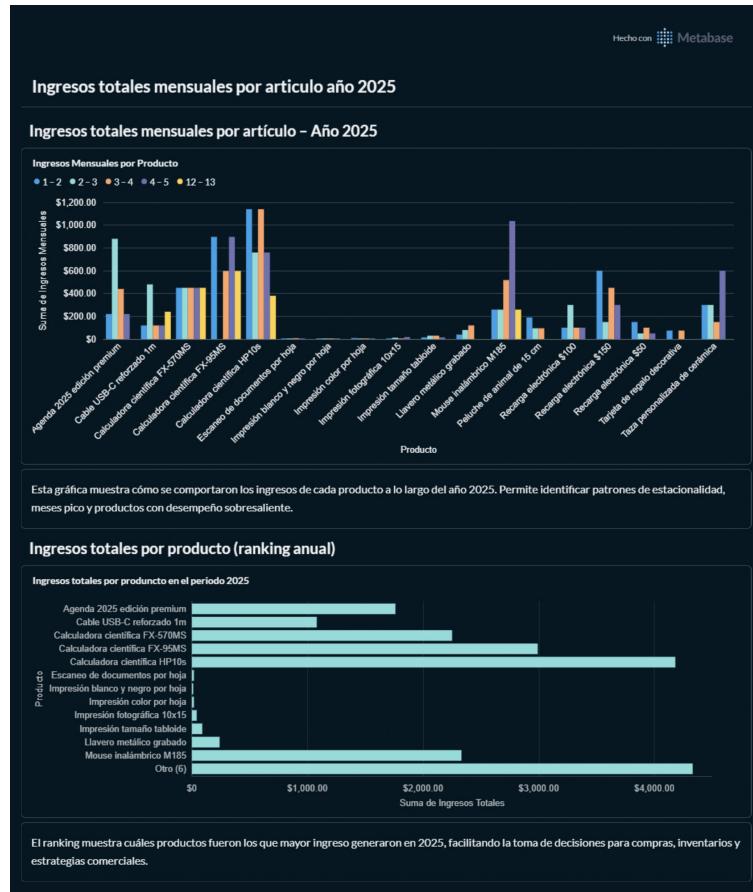


Figura 4: Ingresos totales mensuales por producto



Figura 5: Análisis de los ingresos por categoría y el ingreso total de la tienda en el presente año

## 5.5. Dashboard: Rendimiento de vendedores

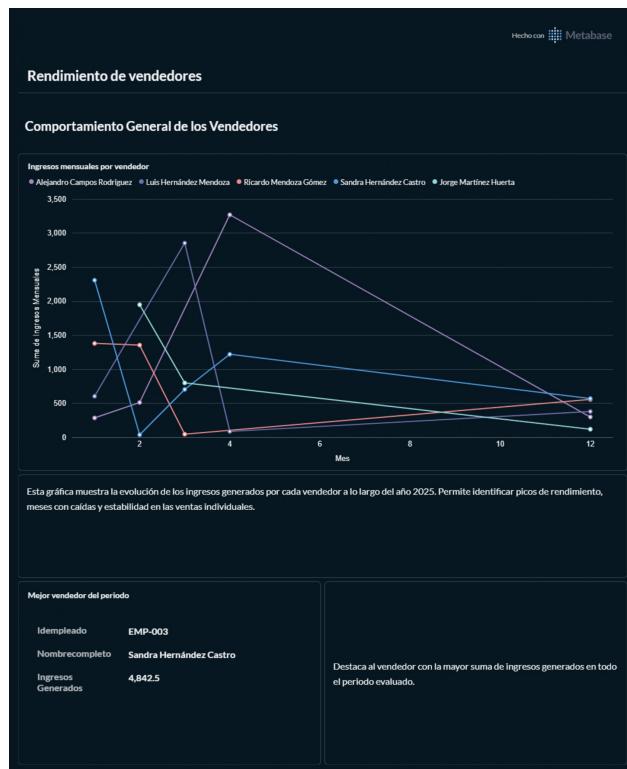


Figura 6: Comportamiento general de los vendedores



Figura 7: Mejores vendedores de cada mes

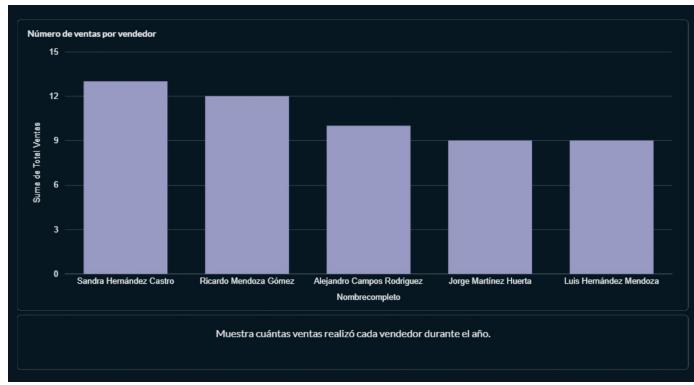


Figura 8: Número de ventas por cada vendedor

## 6. Conclusiones

El desarrollo de este proyecto permitió la construcción de una solución de base de datos robusta, técnicamente fundamentada y plenamente alineada con las necesidades operativas de la papelería, conforme a los requerimientos establecidos. A lo largo del proceso se efectuó un análisis riguroso de los requerimientos del negocio, lo que facilitó la elaboración de un modelo conceptual y lógico consistente, sustentado en dependencias funcionales correctamente identificadas y en la aplicación sistemática de las formas normales. Este enfoque metodológico aseguró que la estructura obtenida evitara redundancias, preservara la integridad referencial y ofreciera un desempeño eficiente en las operaciones de consulta y actualización.

La transición del Modelo Entidad–Relación hacia el Modelo Relacional, y posteriormente su implementación física en PostgreSQL, constituyó un ejercicio integral de diseño orientado a la calidad y estabilidad del sistema. La definición de restricciones CHECK, el establecimiento de claves primarias y foráneas, la incorporación de reglas de validación y el uso de formatos y dominios controlados permitieron consolidar una arquitectura confiable capaz de prevenir inconsistencias desde el nivel estructural. De igual manera, la integración de *triggers*, funciones almacenadas y vistas especializadas incrementó el nivel de automatización del sistema, optimizando tareas críticas como la actualización dinámica del inventario, el cálculo automático de totales y utilidades, y la generación de facturas de manera directa desde la base de datos. Tales mecanismos no solo elevaron la precisión de los procesos, sino que mejoraron significativamente la eficiencia operativa y la capacidad de adaptación del sistema a escenarios reales de uso.

Desde una perspectiva evolutiva, el proyecto avanzó desde un contexto caracterizado por la gestión manual y fragmentada de proveedores, productos, clientes e inventarios, hacia una plataforma estructurada que centraliza, organiza y relaciona la información de manera coherente. Cada etapa del desarrollo permitió profundizar en la comprensión del flujo de datos y de las interacciones internas del negocio, lo que favoreció la construcción de una solución no solo funcional, sino también escalable y preparada para futuras ampliaciones.

Asimismo, se identificó que el sistema podría fortalecerse mediante la incorporación de módulos orientados al control de usuarios, manejo de roles y seguridad avanzada, lo cual habilitaría su operación en entornos multiusuario con altos estándares de confiabilidad. Adicionalmente, la creación de índices especializados o mecanismos de auditoría representaría mejoras relevantes para su rendimiento y trazabilidad.

En síntesis, este proyecto demuestra que un diseño cuidadoso, acompañado de la aplicación disciplinada de principios de normalización y técnicas de automatización, puede transformar procesos manuales en un sistema integral, eficiente y escalable. La base de datos desarrollada no solo satisface los requerimientos actuales, sino que establece una infraestructura sólida sobre la cual pueden implementarse futuras capacidades para alguna papelería.