UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA
Bases de Datos. Grupo 1

Profesor: Ing. Fernando Arreola Franco.

Fecha de entrega: 18 de NOVIEMBRE de 2024

PROYECTO FINAL

Integrantes.

Apellido paterno	Apellido materno	Nombre(s)
Aquino	Lozada	Gabriela
Blanco	Pulido	Gabriel Alonzo
Hernández	Pérez	Yair Edwin
Rosas	Cañada	Abraham
Sánchez	Gachuz	Jenyfer Estefanya

Índice

Sección: 1

Introducción.

El proyecto nos habla sobre una cadena de papelerías que desea mejorar cómo guarda su información, y nos contrató para diseñar un sistema que cumpla con ciertos requerimientos. Dentro del proyecto se desea tener almacenados datos como:

- Razón social, domicilio, nombre y teléfonos de los proveedores, RFC.
- Nombre, domicilio y al menos un email de los clientes.
- Inventario de productos que se venden, incluyendo código de barras, precio de compra, foto, fecha de compra y cantidad en bodega (stock).

Se desea guardar la marca, descripción y precio de los regalos, artículos de papelería, impresiones y recargas, siempre y cuando se tenga su correspondiente registro en el inventario. Adicionalmente, se deben incluir datos de las ventas, como:

- Número de venta, fecha de venta, cantidad total a pagar por la venta.
- Empleado que concreta la venta, cantidad de cada artículo y precio total a pagar por artículo.

De los empleados interesa conocer su clave de empleado, nombre, fecha de nacimiento y fecha de ingreso.

Situaciones para resolver:

1. Generar una muestra de información que asemeje una factura de compra.
2. Calcular la cantidad total vendida y la ganancia en un periodo dado.
3. Actualizar el stock tras una venta y manejar alertas por bajo inventario.
4. Obtener nombres de productos con menos de tres unidades en stock y almacenarlos en una tabla independiente.
5. Calcular la utilidad a partir del código de barras de un producto.
6. Crear al menos un índice justificando su tipo y ubicación.

Se contemplarán diversas soluciones al problema y se utilizarán herramientas como PgSQL para implementar los requerimientos mencionados.

Sección: 2

Objetivo.

El alumno analizará una serie de requerimientos específicos relacionados con el diseño, implementación y administración de bases de datos. Durante este proceso, evaluará diferentes alternativas para identificar la solución más eficiente, teniendo en cuenta aspectos como la optimización del rendimiento, la integridad de los datos, y la escalabilidad del sistema.
Además, aplicará los conceptos aprendidos en el curso, como la normalización, el diseño de esquemas relacionales, la creación de consultas avanzadas en SQL, y el uso de triggers, funciones y vistas para automatizar y simplificar procesos complejos. El objetivo principal será proponer una solución integral que cumpla con los requerimientos funcionales y no funcionales establecidos, garantizando la confiabilidad del sistema desarrollado.

Sección: 3

Plan de trabajo.

Para la realización del proyecto, fue de gran importancia la planificación de las actividades y el tiempo que conlleva la realización de cada una de estas actividades para la realización del proyecto. Después de leer todos los requerimientos y las necesidades del proyecto, así como las reglas de negocio decidimos cómo era que íbamos a organizarnos para terminarlo progresivamente. Utilizamos la plataforma de Notion para que cada integrante del equipo tuviera acceso a la planificación del proyecto y a su vez poder ver el progreso del proyecto. La planificación del proyecto fue semanal, así que para nosotros se nos fue mejor planificarlo de esta manera para ir trabajando con el tiempo y de cierta forma tener esa presión de terminar las tareas que teníamos planificadas para esa semana. Además se hizo una planificación aparte para la creación de Dashboard.

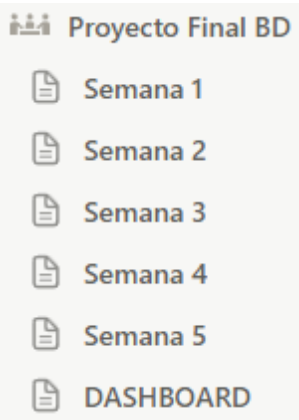
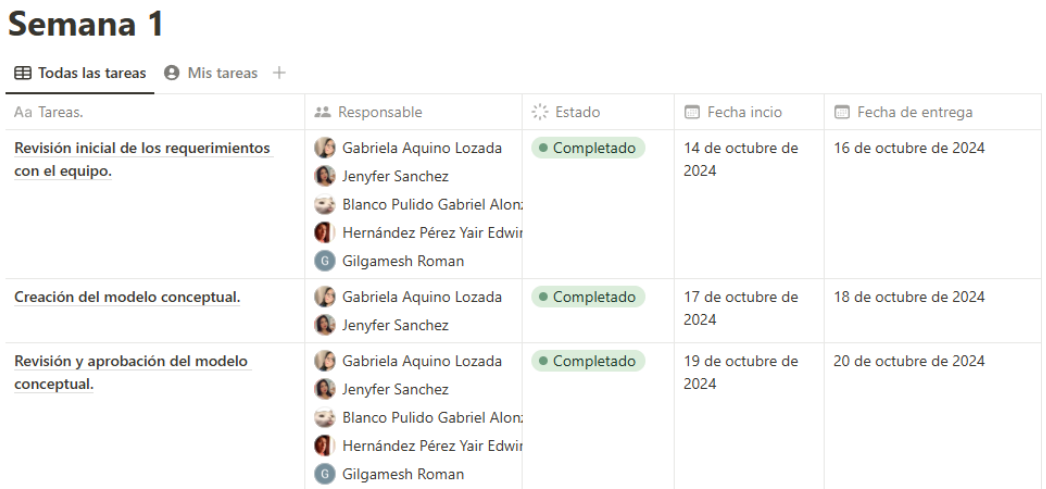


Figura 1: Planificación del proyecto.

Semana 1

Para la semana 1 tenemos registradas tres tareas que van relacionadas a la creación de todo lo que conlleva el diseño lógico de la base de datos. Para esto tenemos las tareas que son:

- Revisión inicial de los requerimientos con el equipo.
- Creación del modelo conceptual.
- Revisión y aprobación del modelo conceptual.



Semana 1

Todas las tareas | Mis tareas +

Aa Tareas.	Responsable	Estado	Fecha incio	Fecha de entrega
Revisión inicial de los requerimientos con el equipo.	Gabriela Aquino Lozada Jenyfer Sanchez Blanco Pulido Gabriel Alon Hernández Pérez Yair Edwi Gilgamesh Roman	Completado	14 de octubre de 2024	16 de octubre de 2024
Creación del modelo conceptual.	Gabriela Aquino Lozada Jenyfer Sanchez	Completado	17 de octubre de 2024	18 de octubre de 2024
Revisión y aprobación del modelo conceptual.	Gabriela Aquino Lozada Jenyfer Sanchez Blanco Pulido Gabriel Alon Hernández Pérez Yair Edwi Gilgamesh Roman	Completado	19 de octubre de 2024	20 de octubre de 2024

Figura 2: Actividades a realizar de la semana 1.

Semana 2

Para la semana 2 tenemos registradas tres tareas, que serían el fin de la parte del diseño lógico. Para esto tenemos las tareas que son:

- Convertir el modelo conceptual en el modelo lógico y definir claves primarias y foráneas.
- Aplicar el proceso de normalización.
- Validar el modelo lógico con el equipo.

Semana 2

Todas las tareas | Mis tareas

Aa Tarea	Responsable	Estado	Fecha incio	Fecha de entrega
Convertir el modelo conceptual en el modelo lógico y definir claves primarias y foráneas.	Hernández Pérez Yair Edwi	Completado	21 de octubre de 2024	23 de octubre de 2024
Aplicar el proceso de normalización.	Gilgamesh Roman Blanco Pulido Gabriel Alon	Completado	24 de octubre de 2024	25 de octubre de 2024
Validar el modelo lógico con el equipo.	Gabriela Aquino Lozada Gilgamesh Roman Blanco Pulido Gabriel Alon Jenyfer Sanchez Hernández Pérez Yair Edwi	Completado	26 de octubre de 2024	27 de octubre de 2024

Figura 3: Actividades a realizar de la semana 2.

Semana 3

Para la semana 3 tenemos registradas dos tareas, que serían el inicio de la implementación de nuestra base de datos. Para esto tenemos las tareas que son:

- Diseño físico y selección de tipos de datos.
- Implementar las tablas y restricciones en PostgreSQL.

Semana 3

Todas las tareas | Mis tareas

Aa Tarea	Responsable	Estado	Fecha incio	Fecha de entrega
Diseño físico y selección de tipos de datos.	Jenyfer Sanchez Gabriela Aquino Lozada	Completado	28 de octubre de 2024	30 de octubre de 2024
Implementar las tablas y restricciones en PostgreSQL.	Hernández Pérez Yair Edwi Blanco Pulido Gabriel Alon	Completado	31 de octubre de 2024	3 de noviembre de 2024

Figura 4: Actividades a realizar de la semana 3.

Semana 4

Para la semana 4 tenemos registradas dos tareas, que serían la implementación de nuestra base de datos, para la parte de pruebas. Para esto tenemos las tareas que son:

- Insertar datos de prueba en las tablas.
- Realizar pruebas de integridad y consistencia.

Semana 4

Todas las tareas | Mis tareas

Aa Tarea	Responsable	Estado	Fecha incio	Fecha de entrega
Insertar datos de prueba en las tablas.	Gilgamesh Roman	Completado	4 de noviembre de 2024	6 de noviembre de 2024
Realizar pruebas de integridad y consistencia.	Hernández Pérez Yair Edwi Gabriela Aquino Lozada	Completado	7 de noviembre de 2024	9 de noviembre de 2024

Figura 5: Actividades a realizar de la semana 4.

Semana 5

Para la semana 5 tenemos registradas tres tareas, que serían el final de la implementación de nuestra base de datos. Para esto tenemos las tareas que son:

- Optimización, creación de índices y pruebas de rendimiento.
- Pruebas finales de la funcionalidad de la base de datos.
- Documentación final.
- Revisión final y entrega formal.

Semana 5

Todas las tareas | Mis tareas

Aa Tarea	Responsable	Estado	Fecha incio	Fecha de entrega
Optimización, creación de índices y pruebas de rendimiento.	Blanco Pulido Gabriel Alon Jenyfer Sanchez	Completado	11 de noviembre de 2024	13 de noviembre de 2024
Pruebas finales de la funcionalidad de la base de datos.	Gabriela Aquino Lozada Gilgamesh Roman Blanco Pulido Gabriel Alon Jenyfer Sanchez Hernández Pérez Yair Edwi	Completado	13 de noviembre de 2024	15 de noviembre de 2024
Documentación final.	Gabriela Aquino Lozada Gilgamesh Roman Blanco Pulido Gabriel Alon Jenyfer Sanchez Hernández Pérez Yair Edwi	Completado	14 de noviembre de 2024	15 de noviembre de 2024
Revisión final y entrega formal.	Gabriela Aquino Lozada Jenyfer Sanchez Hernández Pérez Yair Edwi Blanco Pulido Gabriel Alon Gilgamesh Roman	Completado	16 de noviembre de 2024	17 de noviembre de 2024

Figura 6: Actividades a realizar de la semana 5.

DASHBOARD

Para la parte de la implementación del Dashaboard, para la parte de la presentación para los resultados de nuestra base de datos. Para esto tenemos las tareas que son:

- Importación de las tablas de la base de datos.

- Creación de filtros en DAX.
- Pruebas finales de la funcionalidad del DASHBOARD.

DASHBOARD

Todas las tareas | Mis tareas | + | Nuevo

Aa Tarea	Responsable	Estado	Fecha incio	Fecha de entrega
Importación de las tablas de la base de datos.	Blanco Pulido Gabriel Alon:	Completado	11 de noviembre de 2024	13 de noviembre de 2024
Creación de filtros en DAX.	Blanco Pulido Gabriel Alon:	Completado	13 de noviembre de 2024	15 de noviembre de 2024
Pruebas finales de la funcionalidad del DASHBOARD.	Blanco Pulido Gabriel Alon:	Completado	15 de noviembre de 2024	15 de noviembre de 2024

Figura 7: Actividades a realizar para la creación del dashboard.

Sección: 4

Diseño.

Modelo Conceptual

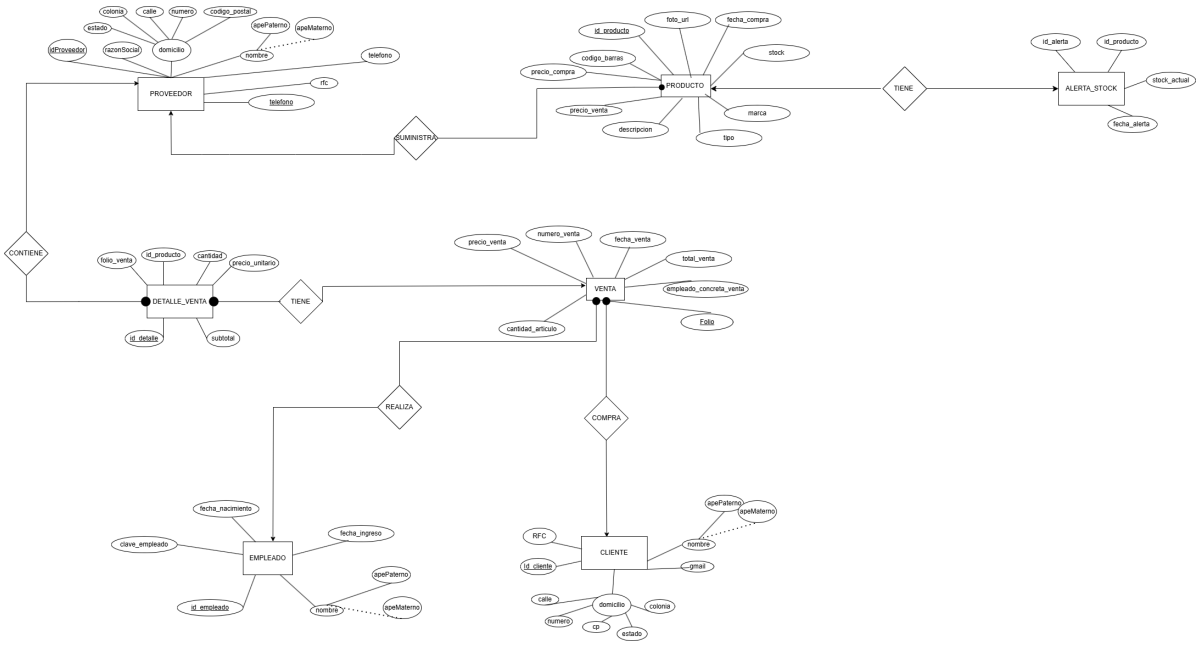


Figura 8: Modelo Conceptual de la base de datos.

Para la realización del modelo conceptual se utilizó la herramienta de Draw.io, una plataforma ampliamente reconocida para el diseño y diagramación de estructuras de datos. Esta herramienta facilitó la representación gráfica de las entidades y sus relaciones, asegurando que el modelo conceptual reflejara de manera precisa los requerimientos del sistema. Nos guiamos a partir del código generado anteriormente, el cual sirvió como base para estructurar y organizar la información de forma lógica y comprensible.

En el modelo conceptual, cada entidad cuenta con los atributos necesarios para describir su función y propósito dentro del sistema. Algunos de estos atributos son simples, como nombres o identificadores únicos, mientras que otros son compuestos. Por ejemplo, el atributo "domicilio"se descompone en subatributos como estado, colonia, calle, número y código postal. Esta descomposición permite manejar la información de forma más detallada y facilita consultas específicas sobre partes del domicilio. Además, cada entidad tiene asignada una llave primaria, la cual garantiza la unicidad de los registros y sirve como identificador principal para su relación con otras entidades.
Se definieron relaciones de cardinalidad que especifican cómo interactúan las entidades, tales como relaciones uno a uno, uno a muchos, o muchos a muchos, dependiendo de los requerimientos específicos del proyecto.

Modelo Relacional Intermedio

Este modelo es eficiente debido a que permite consultas complejas y facilita el mantenimiento, garantizando la confiabilidad y funcionalidad del sistema para un análisis estratégico.

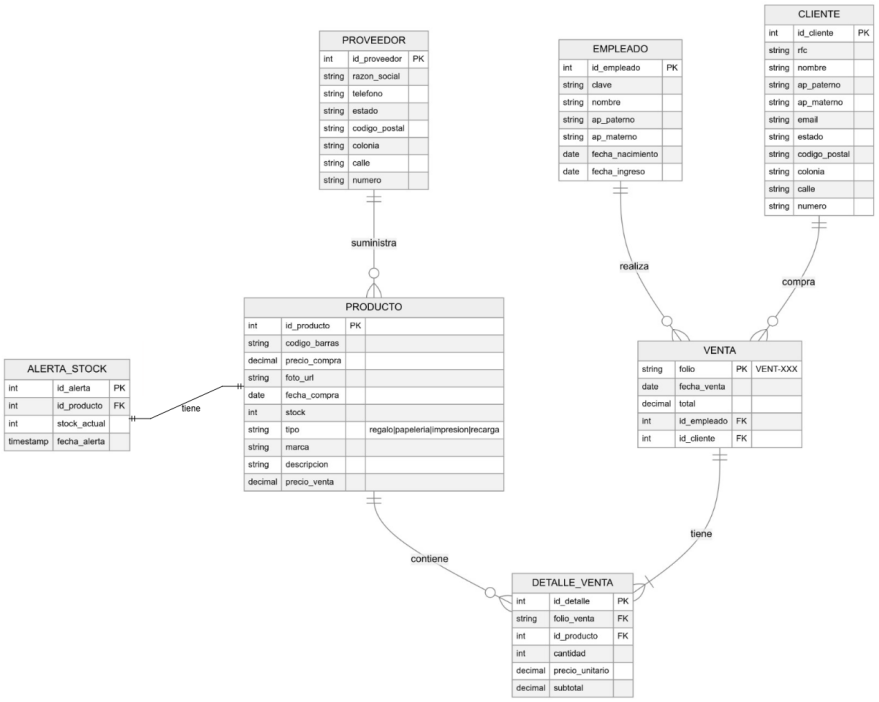


Figura 9: Modelo Relacional de la base de datos.

Dependencias

PROVEEDOR

$$A \to A, B, C, D, E, F, G, H, I$$

- A: `id_proveedor`

- B: `razon_social`
- C: `telefono`
- D: `estado`
- E: `codigo_postal`
- F: `colonia`
- G: `calle`
- H: `numero`

CLIENTE

$$A \rightarrow A, B, C, D, E, F, G, H, I, J, K, L$$

- A: `id_cliente`
- B: `rfc`
- C: `nombre`
- D: `ap_paterno`
- E: `ap_materno`
- F: `email`
- G: `estado`
- H: `codigo_postal`
- I: `colonia`
- J: `calle`
- K: `numero`

EMPLEADO

$$A \rightarrow A, B, C, D, E, F, G$$

- A: `id_empleado`
- B: `clave`

- C: `nombre`
- D: `ap_paterno`
- E: `ap_materno`
- F: `fecha_nacimiento`
- G: `fecha_ingreso`

PRODUCTO

$$A \rightarrow A, B, C, D, E, F, G, H, I, J, K, L$$

- A: `id_producto`
- B: `codigo_barras`
- C: `precio_compra`
- D: `foto_url`
- E: `fecha_compra`
- F: `stock`
- G: `tipo`
- H: `marca`
- I: `descripcion`
- J: `precio_venta`

VENTA

$$A \rightarrow A, B, C, D, E$$

- A: `folio`
- B: `fecha_venta`
- C: `total`
- D: `id_empleado`
- E: `id_cliente`

DETALLE_VENTA

$$A \rightarrow A, B, C, D, E, F$$

- A: `id_detalle`
- B: `folio_venta`
- C: `id_producto`
- D: `cantidad`
- E: `precio_unitario`
- F: `subtotal`

ALERTA_STOCK

$$A \rightarrow A, B, C, D$$

- A: `id_alerta`
- B: `id_producto`
- C: `stock_actual`
- D: `fecha_alerta`

Normalización

1FN

- **Eliminación de atributos multivalorados**: No existen atributos multivalorados en los esquemas.
- **Definición de clave primaria (PK)**: Todas las tablas tienen una PK identificada.
- **Eliminación de grupos repetidos**: No se observan grupos de repetición.

2FN

- **Eliminación de dependencias parciales**: Todas las dependencias funcionales en las tablas dependen completamente de la clave primaria.

3FN

- **Eliminación de dependencias transitivas**: Los atributos no clave dependen directamente de la clave primaria en todos los esquemas.

Sección: 5

Implementación.

Creación de las Tablas

Tabla proveedor: Almacena información de los proveedores de productos. Contiene detalles como nombre (razón social), teléfono, y dirección completa. Permite identificar proveedores únicos y asociar productos con ellos.

```
CREATE TABLE PROVEEDOR (
    id_proveedor INT DEFAULT nextval('seq_proveedor'),--id unico autoincremental
    razon_social VARCHAR(100) NOT NULL,--nombre proveedor
    telefono VARCHAR(15) NOT NULL,--telefono de contacto
    estado VARCHAR(50) NOT NULL,--estado del proveedor
    codigo_postal VARCHAR(5) NOT NULL,--codigo postal
    colonia VARCHAR(50) NOT NULL,--colonia
    calle VARCHAR(100) NOT NULL,--calle
    numero VARCHAR(10) NOT NULL,--numero exterior
    CONSTRAINT pk_proveedor PRIMARY KEY (id_proveedor),--llave primaria
    CONSTRAINT uk_proveedor_razon_social UNIQUE (razon_social)--razon social unica
);
```

Tabla cliente: Registra datos de los clientes, incluyendo RFC, nombre, apellidos, correo electrónico, y dirección. Es utilizada para asociar ventas con clientes y calcular ingresos por cliente.

```
CREATE TABLE CLIENTE (
    id_cliente INT DEFAULT nextval('seq_cliente'),--id unico autoincremental
    rfc VARCHAR(13) NOT NULL,--rfc del cliente
    nombre VARCHAR(50) NOT NULL,--nombre del cliente
    ap_paterno VARCHAR(50) NOT NULL,--apellido paterno
    ap_materno VARCHAR(50),--apellido materno opcional
    email VARCHAR(100) NOT NULL,--correo del cliente
    estado VARCHAR(50) NOT NULL,--estado del cliente
    codigo_postal VARCHAR(5) NOT NULL,-- código postal
    colonia VARCHAR(50) NOT NULL,--colonia
    calle VARCHAR(100) NOT NULL,--calle
    numero VARCHAR(10) NOT NULL,--numero exterior
    CONSTRAINT pk_cliente PRIMARY KEY (id_cliente),--llave primaria
    CONSTRAINT uk_cliente_rfc UNIQUE (rfc),--rfc unico
```

```
    CONSTRAINT uk_cliente_email UNIQUE (email)--email unico
);
```

Tabla empleado: Almacena información de los empleados que realizan ventas, asociando las ventas a cada empleado. Esto permite calcular las órdenes registradas por cada uno.

```
CREATE TABLE EMPLEADO (
    id_empleado INT DEFAULT nextval('seq_empleado'),--id unico auto incremental
    clave VARCHAR(10) NOT NULL,-- clave unica para identificar al empleado
    nombre VARCHAR(50) NOT NULL,--nombre del empleado
    ap_paterno VARCHAR(50) NOT NULL,--apellido
    ap_materno VARCHAR(50),--apellido materno opcional
    fecha_nacimiento DATE NOT NULL,--fecha de nacimiento del empleado
    fecha_ingreso DATE NOT NULL,-- fecha que ingreso al empleo
    CONSTRAINT pk_empleado PRIMARY KEY (id_empleado),--llave priamria
    CONSTRAINT uk_empleado_clave UNIQUE (clave),--clave unica
    CONSTRAINT chk_empleado_fecha_ingreso CHECK (fecha_ingreso >= fecha_nacimiento)
    --check para validar que la edad no tenga incoherencias
);
```

Tabla producto: Contiene datos del inventario, como el proveedor, código de barras, precios (compra y venta), stock disponible, tipo, y descripción.

```
CREATE TABLE PRODUCTO (
    id_producto INT DEFAULT nextval('seq_producto'),--id unico autocrimental
    id_proveedor INT NOT NULL,--relacion con la tabla proveedor
    codigo_barras VARCHAR(13) NOT NULL,--codigo de barras unico
    precio_compra DECIMAL(10,2) NOT NULL,--precio de compra del producto en decimal
    foto_url VARCHAR(255),-- foto del producto en varchar para no subir fotos deberia ser blob
    fecha_compra DATE NOT NULL,--fecha de adquisision
    stock INT NOT NULL,--cantidad disponible en el inventario
    tipo VARCHAR(20) NOT NULL,--tipo (papeleria, recarga, impresion , reagalo)
    marca VARCHAR(50) NOT NULL,--marca del producto
    descripcion VARCHAR(200) NOT NULL,--descripcion del producto
    precio_venta DECIMAL(10,2) NOT NULL,--precio al que se vende el producto
    CONSTRAINT pk_producto PRIMARY KEY (id_producto),--llave primaria
    CONSTRAINT fk_producto_proveedor FOREIGN KEY (id_proveedor)
    REFERENCES PROVEEDOR(id_proveedor),--relacion con el provedor
    CONSTRAINT uk_producto_codigo_barras UNIQUE (codigo_barras),-- código unico
    CONSTRAINT chk_producto_stock CHECK (stock >= 0),--check para el stock no sea negativo
    CONSTRAINT chk_producto_precios CHECK (precio_venta >= precio_compra),--para que el precio de venta
    --sea mayor o igual a la compra
    CONSTRAINT chk_producto_tipo CHECK (tipo IN ('regalo', 'papeleria', 'impresion', 'recarga'))
    --check para los tipos validos
);
```

Tabla venta: Registra cada transacción realizada, enlazando con los empleados responsables y los clientes. Proporciona datos para calcular ingresos mensuales y analizar las ventas.

```sql
CREATE TABLE VENTA (
    folio VARCHAR(8),--folio unico formato vent 001
    fecha_venta DATE NOT NULL DEFAULT CURRENT_DATE,--fecha en la que se realiza la venta
    total DECIMAL(10,2) NOT NULL,--total de la venta
    id_empleado INT NOT NULL,--empleado responsable de la venta
    id_cliente INT NOT NULL,--cliente que realizo la venta
    CONSTRAINT pk_venta PRIMARY KEY (folio),--llave primaria el folio
    CONSTRAINT fk_venta_empleado FOREIGN KEY (id_empleado)
    REFERENCES EMPLEADO(id_empleado),--relacion con empleado
    CONSTRAINT fk_venta_cliente FOREIGN KEY (id_cliente)
    REFERENCES CLIENTE(id_cliente),--relacion con cliente
    CONSTRAINT chk_venta_folio CHECK (folio ~ '^VENT-\d{3}$'),
    --check para la validacion del formato del folio 001
    CONSTRAINT chk_venta_total CHECK (total >= 0)
    --check para la validacion del total no puede ser negativo
);
```

Tabla detalle venta: Desglosa cada transacción, indicando los productos vendidos, sus cantidades, precios unitarios y subtotales. Permite analizar qué productos son más vendidos y en qué volúmenes.

```sql
CREATE TABLE DETALLE_VENTA (
    id_detalle INT DEFAULT nextval('seq_detalle_venta'),--id unico autoincremental
    folio_venta VARCHAR(8) NOT NULL,--folio de venta asociada
    id_producto INT NOT NULL,--producto vendido
    cantidad INT NOT NULL,--cantidad del producto
    precio_unitario DECIMAL(10,2) NOT NULL,--precio unitario del producto
    subtotal DECIMAL(10,2) NOT NULL,--subtotal (cantidad * producto unitario)
    CONSTRAINT pk_detalle_venta PRIMARY KEY (id_detalle),--llave primaria id detalle
    CONSTRAINT fk_detalle_venta_venta FOREIGN KEY (folio_venta)
    REFERENCES VENTA(folio),--relacion con venta fk
    CONSTRAINT fk_detalle_venta_producto FOREIGN KEY (id_producto)
    REFERENCES PRODUCTO(id_producto),--relacion con producto
    CONSTRAINT chk_detalle_venta_cantidad CHECK (cantidad > 0),--check para cantidad positiva
    CONSTRAINT chk_detalle_venta_precios CHECK (precio_unitario > 0 AND subtotal > 0)
    --check para precios sean positivos
);
```

Tabla alerta stock: Monitorea productos con bajo stock en inventario. Genera alertas automáticas con información sobre el producto y la cantidad actual, para facilitar la gestión del inventario.

```
CREATE TABLE ALERTA_STOCK (
    id_alerta INT DEFAULT nextval('seq_alerta_stock'),--id unico autocrimental
    id_producto INT NOT NULL,--producto con bajo stock
    stock_actual INT NOT NULL,--cantidad actual en inventario
    fecha_alerta TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,--fecha de generacion de la alerta
    CONSTRAINT pk_alerta_stock PRIMARY KEY (id_alerta),--llave primaria
    CONSTRAINT fk_alerta_stock_producto FOREIGN KEY (id_producto)
    REFERENCES PRODUCTO(id_producto)--relacion con el producto
);
```

Creación de Funciones y Triggers

Función generar factura.

```
CREATE OR REPLACE FUNCTION generar_factura(folio_venta_param VARCHAR)
RETURNS TABLE(
    folio VARCHAR,
    fecha DATE,
    cliente_nombre VARCHAR,
    empleado_nombre VARCHAR,
    productos JSON,
    total DECIMAL
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        v.folio::VARCHAR,
        v.fecha_venta,
        CONCAT(c.nombre, ' ', c.ap_paterno, ' ', COALESCE(c.ap_materno, ''))::VARCHAR AS cliente_nombre,
        CONCAT(e.nombre, ' ', e.ap_paterno, ' ', COALESCE(e.ap_materno, ''))::VARCHAR AS empleado_nombre,
        json_agg(json_build_object(
            'producto', p.descripcion,
            'cantidad', dv.cantidad,
            'subtotal', dv.subtotal))::JSON AS productos,
        v.total
    FROM VENTA v
    JOIN CLIENTE c ON v.id_cliente = c.id_cliente
    JOIN EMPLEADO e ON v.id_empleado = e.id_empleado
    JOIN DETALLE_VENTA dv ON v.folio = dv.folio_venta
    JOIN PRODUCTO p ON dv.id_producto = p.id_producto
    WHERE v.folio = folio_venta_param
    GROUP BY v.folio, v.fecha_venta, cliente_nombre, empleado_nombre, v.total;
END;
```

```
$$ LANGUAGE plpgsql;
```

Función reporte ganancias.

```
CREATE OR REPLACE FUNCTION reporte_ganancias(fecha_inicio DATE, fecha_fin DATE)
RETURNS TABLE(
    fecha DATE,
    total_vendido DECIMAL,
    ganancia DECIMAL
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        v.fecha_venta,
        SUM(dv.subtotal) AS total_vendido,
        SUM(dv.subtotal - (p.precio_compra * dv.cantidad)) AS ganancia
    FROM VENTA v
    JOIN DETALLE_VENTA dv ON v.folio = dv.folio_venta
    JOIN PRODUCTO p ON dv.id_producto = p.id_producto
    WHERE v.fecha_venta BETWEEN fecha_inicio AND fecha_fin
    GROUP BY v.fecha_venta;
END;
$$ LANGUAGE plpgsql;
```

Función actualizar stock.

```
CREATE OR REPLACE FUNCTION actualizar_stock()
RETURNS TRIGGER AS $$
BEGIN
    -- Restar la cantidad vendida al stock
    UPDATE PRODUCTO
    SET stock = stock - NEW.cantidad
    WHERE id_producto = NEW.id_producto;

    -- Si el stock llega a cero, se aborta la transacción
    IF (SELECT stock FROM PRODUCTO WHERE id_producto = NEW.id_producto) = 0 THEN
        RAISE EXCEPTION 'Stock agotado para el producto con id %', NEW.id_producto;
    END IF;

    -- Si el stock es menor a 3, se inserta una alerta
    IF (SELECT stock FROM PRODUCTO WHERE id_producto = NEW.id_producto) < 3 THEN
        INSERT INTO ALERTA_STOCK (id_producto, stock_actual)
        VALUES (NEW.id_producto, (SELECT stock FROM PRODUCTO WHERE id_producto = NEW.id_producto));
    END IF;

    RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;
```

Trigger actualizar stock.

```
CREATE TRIGGER trg_actualizar_stock
AFTER INSERT ON DETALLE_VENTA
FOR EACH ROW
EXECUTE FUNCTION actualizar_stock();
```

Función obtener utilidad.

```
CREATE OR REPLACE FUNCTION obtener_utilidad(codigo_barras_param VARCHAR)
RETURNS TABLE(
    producto_nombre VARCHAR,--para que salga el nombre del producto
    utilidad DECIMAL--para la utilidad
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        descripcion AS producto_nombre, -- Descripción del producto
        (precio_venta - precio_compra) AS utilidad -- Cálculo de la utilidad
    FROM PRODUCTO
    WHERE codigo_barras = codigo_barras_param;

    -- Si no se encuentra el producto, lanzar excepción
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Producto con código de barras % no encontrado', codigo_barras_param;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Sección: 6

Presentación.

Para complementar la implementación del sistema, se diseñó un dashboard interactivo utilizando Power BI. Este dashboard tiene como objetivo proporcionar una herramienta visual que facilite la comprensión de los datos almacenados en la base de datos, transformándolos en información útil para la toma de decisiones.
El desarrollo del dashboard se enfocó en representar gráficamente los indicadores clave del negocio, abarcando los diferentes requerimientos establecidos. Para lograrlo, se realizó la extracción de datos desde la base de datos en PostgreSQL y se transformaron los mismos con Power Query para su utilización en Power BI. Las métricas visualizadas incluyen:

- **Ingresos del mes**: Muestra las cifras de inversión, ingresos totales y ganancias obtenidas durante el período seleccionado.
- **Top 3 de artículos más vendidos**: Presenta los productos con mayor volumen de ventas.
- **Empleados que registran más órdenes**: Indica cuáles empleados han generado la mayor cantidad de ventas.
- **Artículos con mayor stock**: Resalta los productos con mayor cantidad disponible en inventario.
- **Ganancias**: Proporciona una vista clara del beneficio neto generado por las ventas de productos.
- **Total de ventas**: Resume el número total de transacciones realizadas.
- **Stock total**: Ofrece una visión general del inventario disponible.

Fue necesario implementar diferentes medidas en DAX, las cuales se declararon de la siguiente manera:

```
Ganancias = [TotalIngresos] - [Total Invertido]
Total Invertido = SUMX(DETALLE_VENTA, DETALLE_VENTA[cantidad] * RELATED(PRODUCTO[precio_compra]))
TotalIngresos = SUM(DETALLE_VENTA[subtotal])
TotalOrdenes = COUNT(VENTA[folio])
TotalVentas = SUM(VENTA[total])
Total Vendido = SUM(DETALLE_VENTA[cantidad])
```

Dando por resultado lo siguiente:

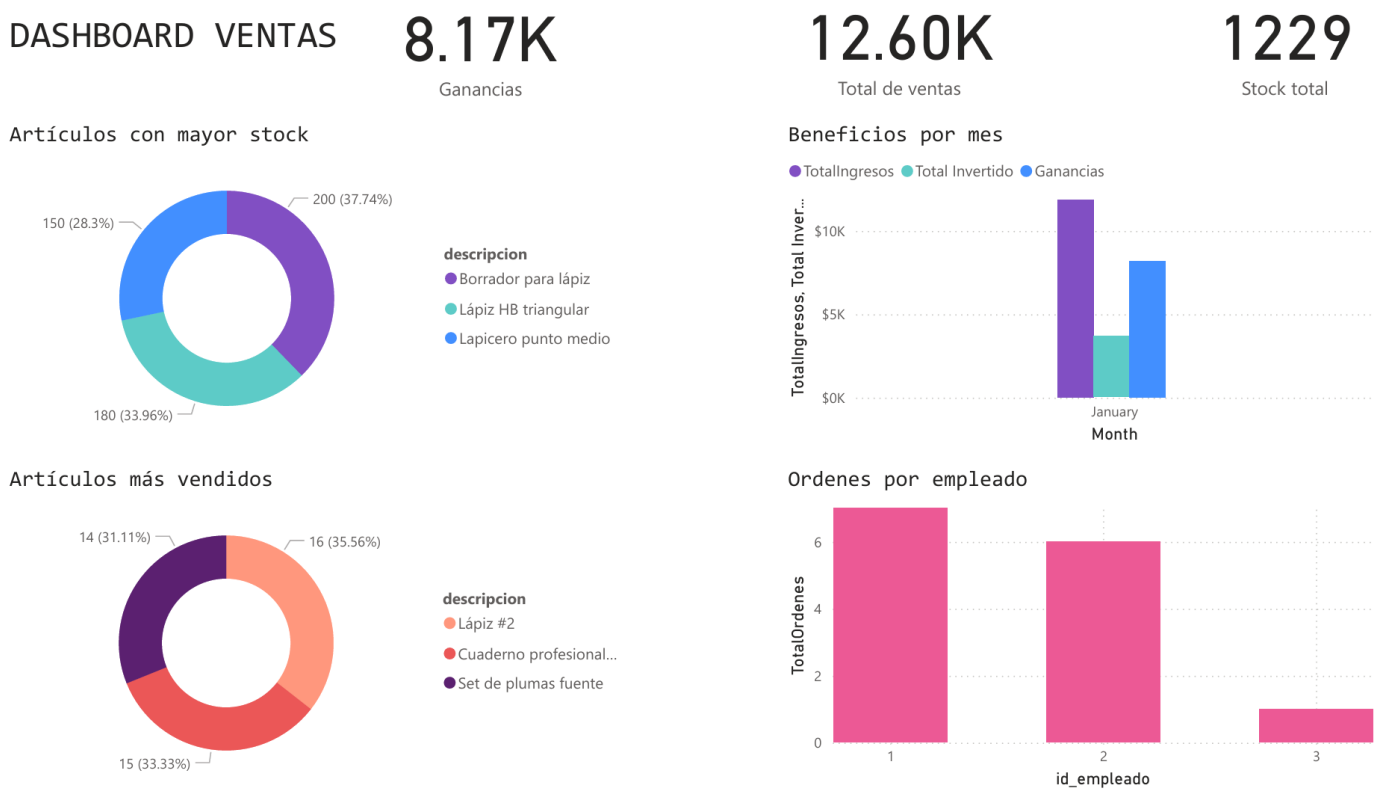


Figura 10: Dashboard ventas.

Sección: 7

Conclusiones.

Aquino Lozada Gabriela: Durante la elaboración del proyecto,se pusieron en practica los conocimientos adquiridos durante todo el curso de Bases de datos, además de la implementación que a simple vista podria llegar a ser facil a la hora de crear las tablas, se tiene que tener en mente la importancia de que exista una coherencia y una estructura correcta de la información para garantizar la consistencia de los datos. Lo que aún inicio me cuestp un poco de comprensión es en el uso de los tiggers, pero con la ayuda de mi equipo de trabajo y fuentes de consult, fue un poco más ameno la comoprensión del uso de los tiggers en la automatización de tareas. Ademas de poner en practica nuestros conocimiento de la materia, es de importancia reconocer el trabajo en equipo, si bien muchas veces no lo tomamos en cuenta. De lo que llevo cursado de la materia, seria la primera vez que hago un proyecto que fácilmente podríamos encontrarnos en la vida laborar.

Blanco Pulido Gabriel Alonzo: A nivel personal, uno de los mayores retos fue garantizar la integridad y coherencia de los datos mientras se automatizaban procesos. Integrar triggers y funciones, al ser un área en la que tenía poco conocimiento, me resultó particularmente complicado, obteniendo muchos errores durante su implementación inicial. Sin embargo, después de varias pruebas y con ayuda, logré superar este punto. En general, aprendí la importancia de prever y abordar problemas desde las etapas iniciales de diseño, así como la necesidad de ser flexible cuando las soluciones iniciales no funcionaban como se esperaba. Aunque al

principio tuve ciertas dificultades para interpretar los requerimientos del proyecto, pude consolidar muchos de los conocimientos adquiridos durante el curso.

Hernández Pérez Yair Edwin: Al realizar el proyecto, aprendí a aplicar de manera práctica los conceptos fundamentales de bases de datos en PostgreSQL, lo que me permitió comprender la importancia de estructurar correctamente la información mediante el uso de claves primarias, claves foráneas y restricciones de integridad para garantizar la consistencia de los datos. Además, trabajar con triggers y funciones me ayudó a reforzar mis conocimientos sobre cómo automatizar tareas importantes, como la actualización del inventario y la generación de alertas cuando el stock es bajo. Implementar estas funcionalidades me hizo más consciente de cómo traducir reglas de negocio en lógica SQL. Finalmente, este proyecto no solo me permitió afianzar mis conocimientos, también me ayudó a analizar problemas y proponer soluciones.

Rosas Cañada Abraham: A lo largo del desarrollo de este proyecto, nuestro equipo logró diseñar e implementar una base de datos que responde a los requerimientos de una cadena de papelerías, con el objetivo de optimizar la gestión de información clave, como proveedores, clientes, inventarios, ventas y empleados. Gracias a una planificación detallada y un enfoque estructurado, abordamos cada fase del proyecto de manera progresiva, desde el diseño conceptual hasta la implementación final.
Uno de los desafíos principales fue asegurar la correcta actualización de los inventarios y la implementación de las alertas para productos con bajo stock, así como el manejo de las transacciones y la generación de reportes. Sin embargo, con un trabajo colaborativo y una adecuada distribución de tareas, logramos superar estos obstáculos y cumplir con las expectativas del proyecto.
El uso de herramientas como Notion nos permitió organizar y coordinar el trabajo de manera eficiente, asegurando que cada miembro del equipo cumpliera con sus responsabilidades y mantuviera un seguimiento constante del progreso. Al final, la base de datos fue optimizada para información y garantizar el rendimiento en consultas y transacciones.
Este proyecto no solo nos permitió aplicar los conceptos aprendidos en el curso, sino que también fortaleció nuestras habilidades en la gestión de proyectos, la resolución de problemas complejos y el trabajo en equipo. El sistema desarrollado cumple con los objetivos establecidos y está preparado para ser escalado y mantenido en el futuro. La documentación final garantiza una comprensión clara del funcionamiento del sistema y su implementación.

Sánchez Gachuz Jenyfer Estefanya: Podemos ver como los objetivos de este proyecto fueron cumplidos de manera eficiente, debido a que además de utilizar distintas fuentes de consulta pudimos aplicar lo visto en clase con anterioridad, desde el tema uno hasta el tema siete, durante este proyecto nos encontramos ocn varios obtaculos el principal y más importante fue la organización en equipo, posterior a esto fue buscar el camino más simple y completo para llegar a la mejor solución al problema propuesto, con forme el equipo iba avanzando teniamos nuevos retos, algunos un poco más complejos que otros como lo fueron desde aprender como usar una herramienta de diseño hasta aprender a hacer consultas o funciones un poco más avanzadas.

Referencias

[1] W3Schools. (s.f). *PostgreSQL SELECT DISTINCT* W3Schools. https://www.w3schools.com/postgresql/postgresql_select_distinct.php

[2] *Row and Array Comparisons.* (s.f). PostgreSQL Documentation. https://www.postgresql.org/docs/current/functions-comparisons.html

[3] *Aggregate Functions.* (s.f). PostgreSQL Documentation. https://www.postgresql.org/docs/current/functions-aggregate.html

[4] *Joins Between Tables.* (s.f). PostgreSQL Documentation. https://www.postgresql.org/docs/current/tutorial-join.html