Proyecto Final SQL: Sistema de Gestión de Tienda Online

Curso: SQL Flex

Desarrollado por: Bautista Arias Ameijde

Repositorio: https://github.com/bautya10/Proyecto_Final_BBDD_Arias.git

Contenido

1. INTRODUCCIÓN 3
2. MODELO RELACIONAL 4
3. TECNOLOGÍAS UTILIZADAS 4
4. COMPONENTES TÉCNICOS 5
4.1 SCRIPTS DE CREACIÓN DE LA BASE DE DATOS Y TABLAS 5
4.2 SCRIPTS DE INSERCIÓN DE DATOS 5
4.3 SCRIPTS DE CREACIÓN DE OBJETOS (VISTAS, STORED PROCEDURES, TRIGGERS Y FUNCIONES) . 5
5. ANÁLISIS DE DATOS 10
6. CONCLUSIONES Y REFLEXIONES 12

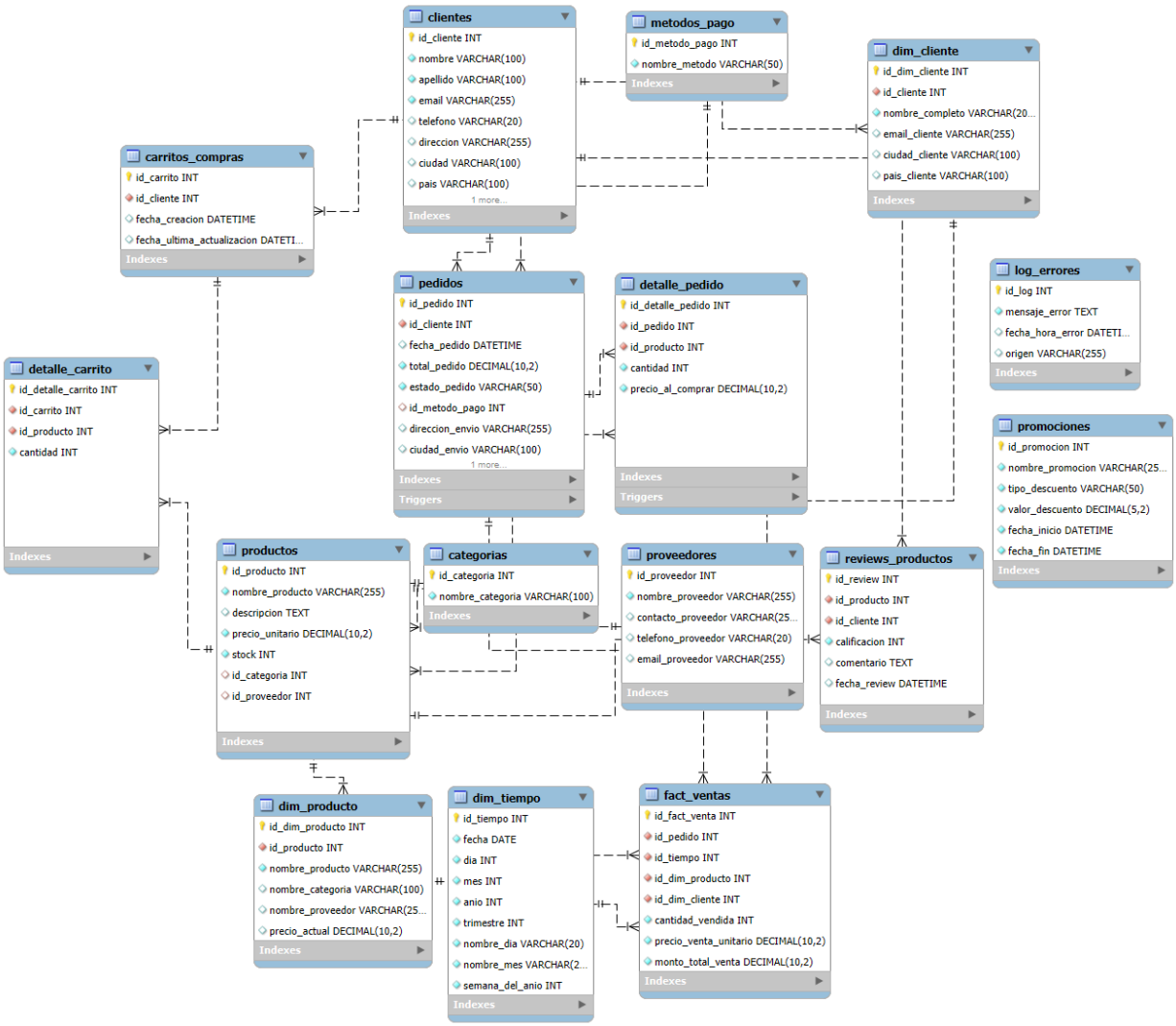
1. Introducción

En el actual panorama del comercio digital, contar con una base de datos eficiente es fundamental para la gestión de operaciones y el análisis de la información. El presente proyecto final aborda esta necesidad mediante el diseño y la implementación de una base de datos relacional para una **tienda online ficticia**, desarrollada desde cero, tomando como punto de partida un modelo de negocio. La propuesta fue trabajada de forma estructurada utilizando **MySQL Workbench** como entorno principal para la creación y gestión de la base de datos. La estructura final incluye más de una docena de tablas, entre operativas y analíticas, con una lógica de relaciones que asegura la integridad de los datos y una base sólida para futuras ampliaciones o integraciones.

El objetivo fue representar de manera organizada todos los elementos clave del funcionamiento de una tienda online: desde productos, clientes y proveedores, hasta aspectos más específicos como métodos de pago, carritos de compra, promociones, pedidos y sus respectivos detalles. Además, se incluyeron tablas dimensionales y una tabla de hechos (fact_ventas), lo que permitió simular un esquema orientado al análisis de datos, integrando también información temporal y de comportamiento de los clientes.

Para complementar el desarrollo, se utilizó **Microsoft Excel** en etapas específicas del diseño y documentación, facilitando la planificación y revisión de los datos. En conjunto, este proyecto busca reflejar no solo el dominio de conceptos fundamentales de bases de datos, sino también una aplicación práctica pensada para escenarios reales y escalables.

2. Modelo Relacional



El modelo relacional se compone de 16 tablas principales que gestionan la información de la tienda online, incluyendo clientes, productos y su inventario, pedidos y sus detalles, métodos de pago, y categorías. Las relaciones entre estas tablas se establecen mediante claves primarias y foráneas para asegurar la integridad referencial de los datos.

3. Tecnologías Utilizadas

- **Sistema de Gestión de Bases de Datos:** MySQL Server (ej. Versión 8.0)
- **Entorno de Desarrollo de Bases de Datos:** MySQL Workbench (ej. Versión 8.0)
- **Herramienta de Análisis y Visualización:** Microsoft Excel
- **Control de Versiones (futuro):** GitHub

4. Componentes Técnicos

4.1 Scripts de Creación de la Base de Datos y Tablas

Este script SQL se encarga de crear el esquema (tienda_online) y todas las tablas que componen el modelo relacional del sistema. Incluye la definición de claves primarias y foráneas para asegurar la integridad referencial de los datos.

Ejemplo: CREATE TABLE IF NOT EXISTS Categorias (id_categoria INT PRIMARY KEY AUTO_INCREMENT, nombre_categoria VARCHAR(100) UNIQUE NOT NULL
);

https://github.com/bautya10/Proyecto_Final_BBDD_Arias/blob/main/Scripts_SQL/database_schema.sql

4.2 Scripts de Inserción de Datos

Este script SQL contiene los comandos necesarios para insertar datos de prueba en la base de datos. Estos datos simulan escenarios reales de una tienda online, incluyendo productos, clientes, pedidos, métodos de pago etc. Su objetivo es ocupar las tablas para validar el modelo, facilitar pruebas de funcióny permitir el análisis mediante consultas.

Ejemplo: INSERT INTO Categorias (nombre_categoria) VALUES

('Electrónica'),

('Ropa'),

('Libros'),

('Hogar'),

('Alimentos'),

('Deportes'),

('Belleza');

https://github.com/bautya10/Proyecto_Final_BBDD_Arias/blob/main/Scripts_SQL/sample_data.sql

4.3 Scripts de Creación de Objetos (Vistas, Stored Procedures, Triggers y Funciones)

Esta sección detalla los scripts SQL utilizados para la creación de objetos avanzados en la base de datos tienda_online. Estos objetos permiten implementar lógica de negocio, automatización de tareas, y facilitar el acceso a datos complejos, mejorando la eficiencia y la integridad del sistema.

4.3.1 Vistas

- Las **vistas** se utilizan para simplificar consultas complejas, restringir el acceso a datos y presentar la información de manera más accesible para los usuarios o herramientas de análisis.

Vista_Productos_Stock_Bajo

Objetivo/Función: Proporciona una lista de productos cuyo nivel de stock actual es inferior a un umbral predefinido (en este caso, 50 unidades). Es útil para la gestión de inventario y la identificación de productos que requieren reposición.
Fragmento de Código:
SQL

```
CREATE VIEW Vista_Productos_Stock_Bajo AS
SELECT
    p.id_producto,
    p.nombre_producto,
    p.stock,
    p.precio,
    c.nombre_categoria
FROM
    Productos p
JOIN
    Categorias c ON p.id_categoria = c.id_categoria
WHERE
    p.stock < 50;
```

Vista_Clientes_Activos

Objetivo/Función: Muestra los clientes que han realizado al menos un pedido y cuya cuenta está activa. Esta vista es útil para segmentar clientes y enfocar estrategias de marketing.
Fragmento de Código:
SQL

```
CREATE VIEW Vista_Clientes_Activos AS
SELECT DISTINCT
    c.id_cliente,
    c.nombre,
    c.apellido,
    c.email,
    c.fecha_registro
FROM
    Clientes c
JOIN
    Pedidos p ON c.id_cliente = p.id_cliente
WHERE
    c.estado = 'activo';
```

4.3.2 Stored Procedures (Procedimientos Almacenados)

Los procedimientos almacenados encapsulan lógica de negocio, permitiendo ejecutar una secuencia de operaciones SQL como una sola unidad transaccional, mejorando la seguridad, el rendimiento y la modularidad del código.

SP_RegistrarNuevoPedido

Objetivo/Función: Facilita el proceso de creación de un nuevo pedido. Inserta un nuevo registro en la tabla Pedidos y los detalles correspondientes en la tabla Detalle_Pedido. Además, actualiza el stock de los productos involucrados.
Fragmento de Código:
SQL

```
DELIMITER $$
CREATE PROCEDURE SP_RegistrarNuevoPedido (
    IN p_id_cliente INT,
    IN p_metodo_pago VARCHAR(50),
    IN p_estado_pedido VARCHAR(50),
    IN p_monto_total DECIMAL(10,2),
    IN p_productos_json JSON -- Formato: [{"id_producto": 1, "cantidad": 2,
"precio_unitario": 10.00}]
)
BEGIN
    DECLARE v_id_pedido INT;
    -- Insertar en Pedidos
    INSERT INTO Pedidos (id_cliente, fecha_pedido, metodo_pago,
total_pedido, estado_pedido)
    VALUES (p_id_cliente, NOW(), p_metodo_pago, p_monto_total,
p_estado_pedido);

    SET v_id_pedido = LAST_INSERT_ID ();

    -- Insertar en Detalle_Pedido y actualizar stock
    INSERT INTO Detalle_Pedido (id pedido, id_producto, cantidad,
precio_unitario)
    SELECT
        v_id_pedido,
        JSON_UNQUOTE(JSON_EXTRACT(p.value, '$.id_producto')),
        JSON_UNQUOTE(JSON_EXTRACT(p.value, '$.cantidad')),
        JSON_UNQUOTE(JSON_EXTRACT(p.value, '$.precio_unitario'))
    FROM
        JSON_TABLE(p_productos_json, '$[*]' COLUMNS (
            id_producto INT PATH '$.id_producto',
            cantidad INT PATH '$.cantidad',
            precio_unitario DECIMAL(10,2) PATH '$.precio_unitario'
        )) AS p;

    -- Actualizar stock (Este trigger se encarga de esto)
    -- UPDATE Productos
    -- SET stock = stock - JSON_UNQUOTE(JSON_EXTRACT(p.value,
'$.cantidad'))
```

```
    -- WHERE id_producto = JSON_UNQUOTE(JSON_EXTRACT(p.value,
'$.id_producto'));
END$$
DELIMITER ;
```

SP_ActualizarPrecioProducto

Objetivo/Función: Permite actualizar el precio de un producto específico, asegurando que los precios se mantengan al día.
Fragmento de Código:
SQL

```
DELIMITER $$
CREATE PROCEDURE SP_ActualizarPrecioProducto(
    IN p_id_producto INT,
    IN p_nuevo_precio DECIMAL(10,2)
)
BEGIN
    UPDATE Productos
    SET precio = p_nuevo_precio
    WHERE id_producto = p_id_producto;
END$$
DELIMITER ;
```

4.3.3 Triggers

Los triggers son rutinas SQL que se ejecutan automáticamente en respuesta a eventos específicos (INSERT, UPDATE, DELETE) en una tabla, lo que permite la automatización de la lógica de negocio y el mantenimiento de la integridad de los datos.

TR_ActualizarStock_AFTER_INSERT_DetallePedido

Objetivo/Función: Este trigger se activa automáticamente después de que se inserta un nuevo registro en la tabla Detalle_Pedido. Su función es restar la cantidad del producto comprado del stock disponible en la tabla Productos, asegurando la consistencia del inventario.

Fragmento de Código:

SQL

```
DELIMITER $$

CREATE TRIGGER
TR_ActualizarStock_AFTER_INSERT_DetallePedido

AFTER INSERT ON Detalle_Pedido

FOR EACH ROW

BEGIN
```

```
    UPDATE Productos
    SET stock = stock - NEW.cantidad
    WHERE id_producto  = NEW.id_producto;
END$$
DELIMITER ;
```

TR_ActualizarFechaModificacion_BEFORE_UPDATE_Producto

Objetivo/Función: Este trigger se activa antes de que se realice una actualización en la tabla Productos. Su propósito es actualizar automáticamente la columna fecha_ultima_modificacion con la fecha y hora actuales, registrando así cuándo fue la última vez que un producto fue modificado.

Fragmento de Código:

SQL

```
DELIMITER $$
CREATE TRIGGER TR_ActualizarFechaModificacion_BEFORE_UPDATE_Producto
BEFORE UPDATE ON Productos
FOR EACH ROW
BEGIN
    SET NEW.fecha_ultima_modificacion = NOW();
END$$
DELIMITER ;
```

4.3.4 Funciones

Las funciones SQL son rutinas que realizan un cálculo y devuelven un único valor. Son útiles para operaciones repetitivas que no modifican datos.

FN_CalcularMontoIVA

Objetivo/Función: Calcula el monto del Impuesto al Valor Agregado (IVA) para un precio base dado. Esta función permite estandarizar el cálculo del IVA en diferentes contextos.

Fragmento de Código:

SQL

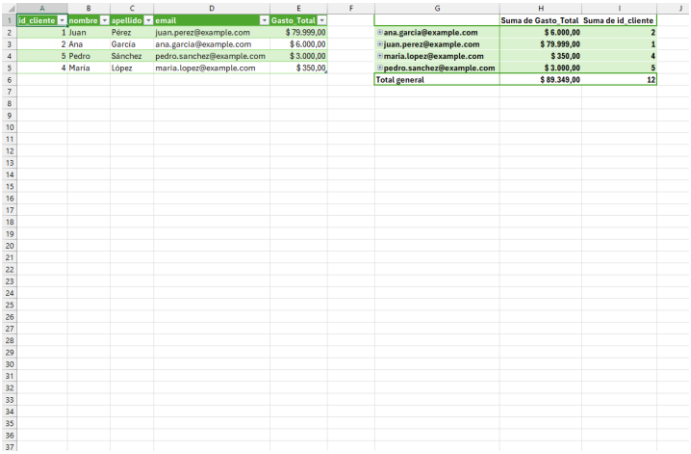
```
DELIMITER $$

CREATE FUNCTION FN_CalcularMontoIVA(precio  DECIMAL(10,2),
tasa_iva DECIMAL(5,2))

RETURNS  DECIMAL(10,2)

DETERMINISTIC

BEGIN

    RETURN precio * tasa_iva;

END$$

DELIMITER ;
```

https://github.com/bautya10/Proyecto_Final_BBDD_Arias/blob/main/Scripts_SQL/database_objects.sql

5. Análisis de Datos

En esta sección se presentan los análisis realizados a partir de los datos extraídos de la base de datos SQL. Se utilizó Microsoft Excel para visualizar la información mediante tablas dinámicas y gráficos, lo que permitió obtener insights clave sobre el comportamiento de la tienda online, incluyendo tendencias de ventas, identificación de productos populares y perfil de los clientes.

5.1 Top Clientes por Gasto Total



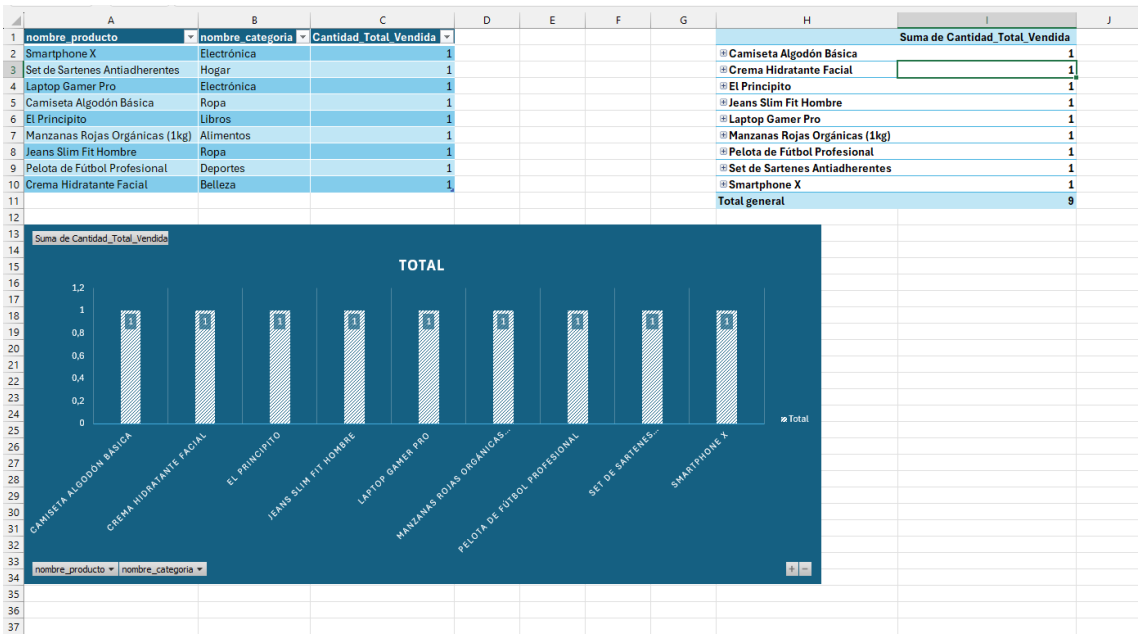
id_cliente	nombre	apellido	email	Gasto_Total
1	Juan	Pérez	juan.perez@example.com	$ 79.999,00
2	Ana	García	ana.garcia@example.com	$ 6.000,00
5	Pedro	Sánchez	pedro.sanchez@example.com	$ 3.000,00
4	María	López	maria.lopez@example.com	$ 350,00

	Suma de Gasto_Total	Suma de id_cliente
ana.garcia@example.com	$ 6.000,00	2
juan.perez@example.com	$ 79.999,00	1
maria.lopez@example.com	$ 350,00	4
pedro.sanchez@example.com	$ 3.000,00	5
Total general	$ 89.349,00	12

- Esta tabla dinámica y el análisis asociado (top_clientes) muestran el gasto total acumulado por cada cliente en la tienda online.
- Se observa claramente que Juan Pérez es, por un margen significativo, el cliente con mayor gasto, habiendo contribuido con aproximadamente $80.000, lo que representa más del 89% del total general de gastos. Esta información es crucial para identificar a los clientes de alto valor (VIP) y diseñar estrategias de fidelización personalizadas, como

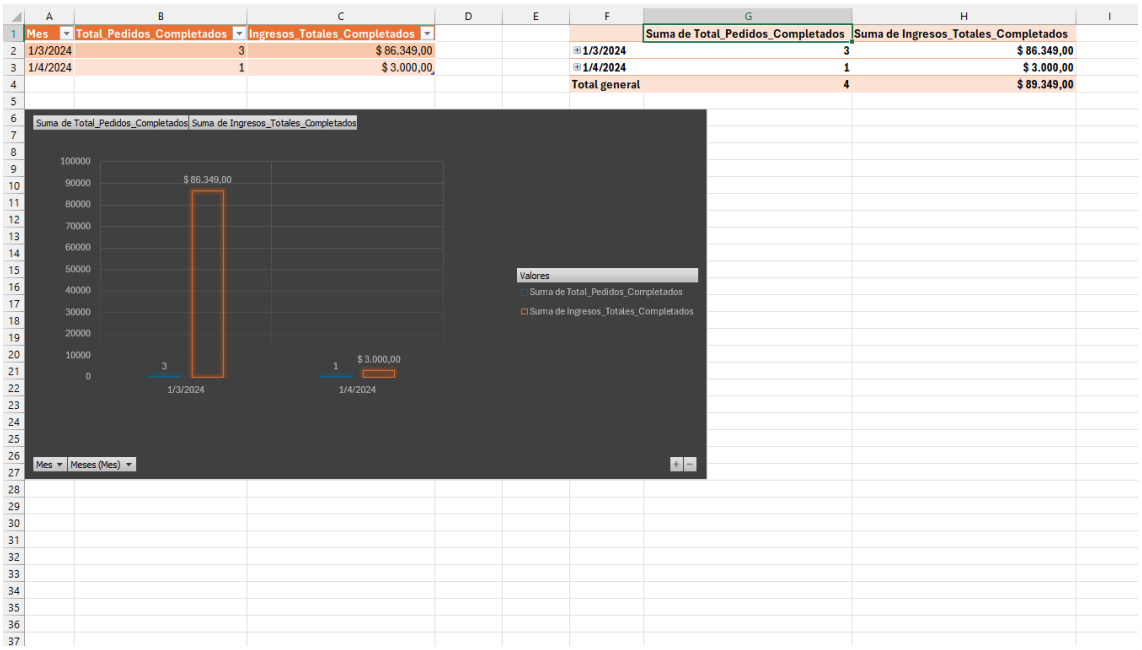
programas de recompensas, descuentos exclusivos o promociones dirigidas a este segmento clave, maximizando el valor de vida del cliente.

5.2 Productos Más Vendidos



- Este gráfico de columnas y tabla dinámica (top_productos) visualizan la cantidad total de unidades vendidas por cada producto.
- Los datos muestran que todos los productos registrados se han vendido en una cantidad total de una unidad cada uno. Esto indica que, en esta etapa inicial del negocio, aún no se ha consolidado un patrón claro de preferencia o un producto "estrella" que domine el volumen de ventas. Esta observación es valiosa para comprender la etapa temprana de la tienda y sugiere que se podrían implementar campañas promocionales segmentadas para productos con mayor margen de ganancia, o explorar estrategias de venta cruzada (cross-selling) y agrupamiento de artículos similares para incentivar la compra de múltiples productos.

5.3 Ventas Totales por Mes



- Este gráfico de columnas y la tabla dinámica asociada (ventas_mes) comparan la cantidad de pedidos completados y los ingresos totales generados mes a mes.

- La visualización de las ventas mensuales es fundamental para identificar tendencias y estacionalidades. Esta información es clave para comprender qué períodos o campañas generan mayores ingresos y cuáles muestran caídas de rendimiento. Permite reforzar estrategias exitosas, planificar el inventario y los recursos humanos en función de la demanda esperada, y detectar la necesidad de acciones correctivas en meses de bajo rendimiento.

6. Conclusiones y Reflexiones

El desarrollo de este proyecto final ha sido una experiencia integral y enriquecedora, permitiendo aplicar y consolidar los conocimientos adquiridos en el diseño e implementación de bases de datos relacionales. Desde la concepción del modelo E-R hasta la generación de informes analíticos, cada etapa ha representado un desafío y una oportunidad de aprendizaje. Hemos logrado construir una base de datos robusta y escalable para una tienda online, capaz de gestionar de manera eficiente información vital sobre clientes, productos, pedidos, proveedores y otros elementos clave. La inclusión de objetos avanzados como vistas, procedimientos almacenados, triggers y funciones no solo optimiza la operatividad del sistema, sino que también demuestra la capacidad de implementar lógica de negocio compleja directamente en la base de datos, asegurando la integridad y automatización de procesos críticos.

Particularmente, la integración de un esquema dimensional (Fact_Ventas y Dim_ tablas) ha sido un punto destacable, abriendo la puerta a un análisis exploratorio más profundo y a la extracción de *insights* valiosos sobre el comportamiento de compra y las tendencias de ventas. Aunque los datos iniciales puedan mostrar patrones incipientes, la estructura ya está preparada para un crecimiento futuro y para soportar decisiones estratégicas basadas en información consolidada.

Este proyecto me ha permitido comprender la importancia de una buena planificación del esquema, la depuración meticulosa de scripts SQL y la presentación clara de los resultados. Aunque siempre hay espacio para la mejora, considero que esta base de datos es un cimiento sólido que podría expandirse con nuevas funcionalidades, como la integración con sistemas de marketing, pasarelas de pago más complejas, o dashboards de visualización en tiempo real para una gestión aún más proactiva.

En definitiva, este trabajo refleja no solo la adquisición de habilidades técnicas en SQL y gestión de bases de datos, sino también la capacidad de abordar un problema de negocio real con una solución estructurada y adaptable.