Proyecto Final - Coderhouse - SQL

Estudiante: Ma. Florencia Torres Nieva

Comisión: 43420

Profesor: Gabriel Almiñana

Tutor: José Ignacio López Saez

Índice

Introducción

Este informe representa la conclusión de los conocimientos adquiridos durante el curso de SQL de la escuela de Coderhouse. En él, se explora el proceso de codificación e implementación de una base de datos relacional.

Se registra la información con la creación de una base completa y funcional, y se van utilizando distintos métodos que contribuyen a la implementación de un negocio.

Objetivo

El objetivo principal de este proyecto es crear una base de datos utilizando MySQL Workbench. Esta base de datos estará diseñada para satisfacer las necesidades que una tienda virtual de venta de productos veganos y ecológicos pueda necesitar, corroborando cuales son los productos más vendidos, zonas de entrega, entre otras cosas.

Para lograr este objetivo, se llevarán a cabo las siguientes tareas:

Creación de Tablas: para la gestión de clientes, pedidos, productos, repartidores y otros aspectos adicionales como un blog con el que los clientes pueden tener acompañamiento nutricional.
Stored Procedures y Funciones: para automatizar tareas y mejorar la eficiencia de la base de datos.
Vistas: presentarán datos y métricas clave para el negocio.
Triggers: para rastrear eventos (incidentes) y aplicar acciones específicas.

Situación problemática

Brindar a *Bionest* una plataforma sólida para gestionar su inventario de productos, atender las necesidades de sus clientes y optimizar sus procesos comerciales. Además, hacer el seguimiento detallado de su catálogo de productos, la gestión de pedidos, la interacción con los clientes y la recopilación de datos relevantes para tomar decisiones informadas.

Modelo de negocio

Bionest es una plataforma en línea dedicada a cerrar la brecha entre los consumidores de productos veganos y ecológicos y los productores y proveedores de estos productos. La misión es ofrecer una amplia variedad de productos de alta calidad, promoviendo un estilo de vida saludable y sostenible.

En *Bionest*, se ofrece una extensa selección de productos que abarcan desde vegetales y frutas frescas, hasta quesos de origen vegetal envasados al vacío. Nuestro catálogo se adapta a las necesidades de nuestros clientes, proporcionando opciones frescas y deliciosas para quienes siguen una dieta basada en plantas y buscan alimentos orgánicos y veganos.

Por otra parte, también se busca empoderar a los clientes con conocimientos valiosos sobre la alimentación basada en plantas, incluyendo un blog que ofrece información interesante y recursos sobre nutrición, recetas saludables y consejos para llevar un estilo de vida sostenible.

Diagrama Entidad - Relación

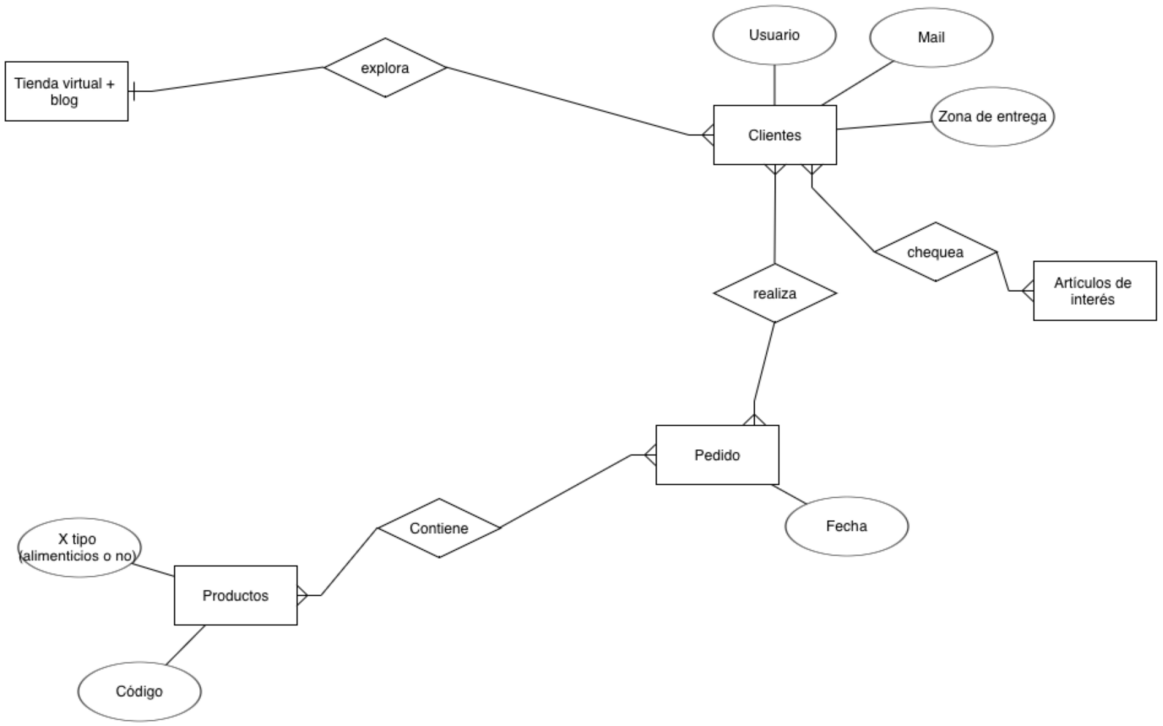
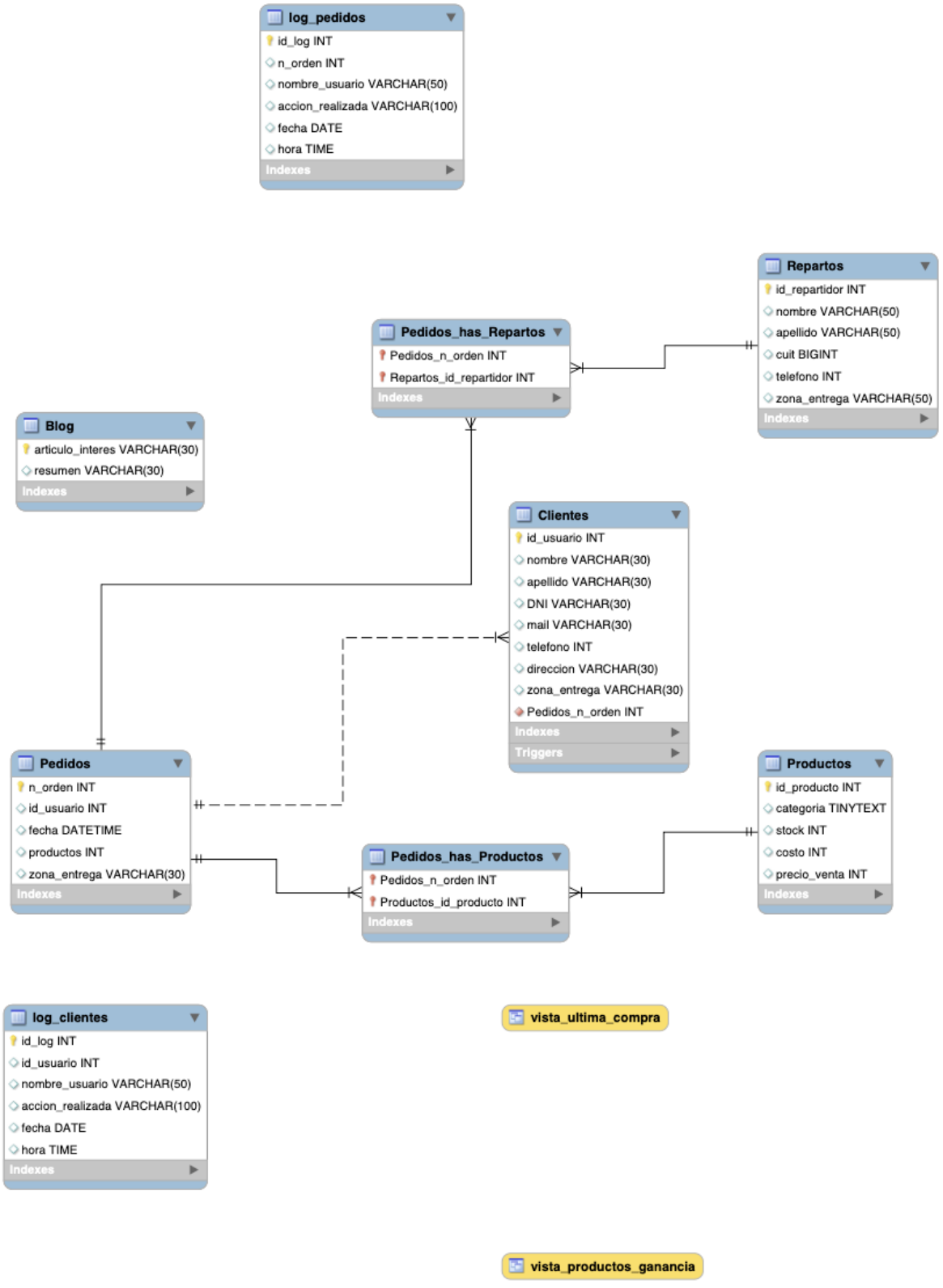


Diagrama Entidad - Relación de MySQL Workbench



Listado de tablas

Tabla CLIENTES	Campo abreviado	Nombre de campo completo	Clave Primaria	Clave Foránea	Tipo de dato
	id_usuario	Identificador único de cliente	✓		INT
	nombre				VARCHAR(30)
	apellido				VARCHAR(30)
	DNI				VARCHAR(30)
	mail (unique index)				VARCHAR(50)
	telefono				INT
	direccion				VARCHAR(50)
	zona_entrega			✓	VARCHAR(50)
Descripción	tabla que abarca datos personales de los clientes, la zona de entrega permite lograr entregas más eficientes y ahorro de viajes				

Tabla PEDIDOS	Campo abreviado	Nombre de campo completo	Clave Primaria	Clave Foránea	Tipo de dato
	n_orden	Identificador único de orden	✓		INT
	fecha				DATETIME
	id_producto			✓	INT
	id_repartidor			✓	VARCHAR(50)
	zona_entrega			✓	
Descripción	tabla de pedidos identificados por el número de orden				

Tabla PRODUCTOS	Campo abreviado	Nombre de campo completo	Clave Primaria	Clave Foránea	Tipo de dato
	id_producto	Identificador único de producto	✓		INT
	categoria				VARCHAR(30)
	stock				INT
	costo				INT
	precio_venta				INT
Descripción	productos divididos por categorías que tienen su propio código de identificación				

Tabla REPARTOS	Campo abreviado	Nombre de campo completo	Clave Primaria	Clave Foránea	Tipo de dato
	id_repartidor	Identificador único de repartidor	✓		INT
	nombre				VARCHAR(30)
	apellido				VARCHAR(30)
	cuit				INT
	telefono				INT
	zona_entrega			✓	VARCHAR(50)
Descripción	tabla que abarca datos personales de los repartidores, la zona de entrega permite lograr entregas más eficientes y ahorro de viajes				

Tabla BLOG	Campo abreviado	Nombre de campo completo	Clave Primaria	Clave Foránea	Tipo de dato
	articulo_interes	Título del artículo	✓		VARCHAR(30)
	resumen	Breve resumen del mismo			TEXT
Descripción	tener una tabla con los artículos del blog y un resumen de ellos permite no repetir temas e incluso facilita la búsqueda de artículos que necesitan actualizaciones.				

MySQL script: Creación de tablas + inserción de datos

Vistas

1ra vista: *vista_ultima_compra*

Creación de vista para chequear la información de los clientes y la fecha de última compra.

```sql
CREATE VIEW vista_ultima_compra AS
SELECT
    c.id_usuario,
    c.nombre,
    c.apellido,
    c.DNI,
    c.mail,
    c.telefono,
    c.direccion,
    c.zona_entrega,
    MAX(p.fecha) AS ultima_compra -- así obtengo la fecha de la última compra x cliente
FROM
    Clientes c
LEFT JOIN -- con esto uno las tablas Clientes y Pedidos x la columna zona_entrega
    Pedidos p ON c.zona_entrega = p.zona_entrega
GROUP BY
    c.id_usuario;
SELECT * FROM vista_ultima_compra;
```

2da vista: *vista_productos_ganancia*

Con esta vista, chequeo la información de los productos y calculo la ganancia por unidad y total para cada producto.

```sql
CREATE VIEW vista_productos_ganancia AS
SELECT
    id_producto,
    categoria,
    stock,
    costo,
    precio_venta,
    (precio_venta - costo) AS ganancia_por_unidad, -- uso operaciones simples para determinarlo
    (precio_venta - costo) * stock AS ganancia_total
FROM
    Productos;

SELECT * FROM vista_productos_ganancia;
```

Stored Procedures

1er SP: *ordenar_tabla*

Procedure para ordenar una tabla según un campo (columna) y en una dirección específica. En el ejemplo, se ordena la tabla Clientes, por nombre en orden ascendente.

```
DELIMITER //

CREATE PROCEDURE ordenar_tabla(
    IN tabla_nombre VARCHAR(50), -- Nombre de la tabla que se desea ordenar
    IN campo_orden VARCHAR(50), -- Campo por el cual se desea ordenar la tabla
    IN direccion VARCHAR(50) -- Dirección de ordenamiento (ASC para ascendente, DESC para descendente)
)
BEGIN
    SET @query = CONCAT('SELECT * FROM ', tabla_nombre, ' ORDER BY ', campo_orden, ' ', direccion, ';'); -- Se concatenan los valores de
    PREPARE stmt FROM @query; -- asocio query con stmt
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt; -- dsps del execute con esto aseguro que los recursos se liberen correctamente
END
//

CALL ordenar_tabla('Clientes', 'nombre', 'ASC');
```

2do SP: *modificar_tabla*

Procedure para modificar una tabla. En el ejemplo, modifico la tabla Clientes, agregando la información de una nueva clienta.

```
DELIMITER //

CREATE PROCEDURE modificar_tabla(
    IN accion INT, -- accion a realizar: 1 para inserción, 2 para eliminación
    IN tabla_nombre VARCHAR(50), -- nombre de la tabla en la cual quiero realizar la modificación
    IN valores_csv VARCHAR(1000) -- registros en formato CSV
)
BEGIN
    IF accion = 1 THEN
        SET @query = CONCAT('INSERT INTO ', tabla_nombre, ' (id_usuario, nombre, apellido, DNI, mail, telefono, direccion, zona_entrega) VALUES (', valores_csv, ');');
    ELSEIF accion = 2 THEN
        SET @query = CONCAT('DELETE FROM ', tabla_nombre, ' WHERE id_usuario IN (', valores_csv, ');');
    END IF;

    PREPARE stmt FROM @query;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END
//

CALL modificar_tabla(1, 'Clientes', '5,"Karen","Filippino","33435612","karen@hotmail.com.ar","11236789","Rodrigo Pertegas 34","Villa Ballester"'); -- inserto un nuevo cliente en la tabla 'Clientes'
```

Triggers

1er trigger: *tr_before_insert_log_clientes*

Este trigger registrará la adición de nuevos registros en Clientes ANTES de que se realice la operación.

```
DELIMITER //
CREATE TRIGGER tr_before_insert_log_clientes
BEFORE INSERT ON Clientes -- se va a activar antes de insertar datos en la tabla
FOR EACH ROW
BEGIN
    INSERT INTO log_clientes (id_usuario, nombre_usuario, accion_realizada, fecha, hora)
    VALUES (NEW.id_usuario, USER(), 'INSERCIÓN', CURDATE(), CURTIME());
    -- va a registrar la acción como INSERCIÓN y x USER obtendrá el usuario, tmb almacena la fecha y hora x CURDATE y CURTIME
END
//
```

2do trigger: *tr_after_delete_log_pedidos*

Este trigger registrará la eliminación de datos en la tabla Pedidos después de que se realice esa operación.

```
DELIMITER //
CREATE TRIGGER tr_after_delete_log_pedidos
AFTER DELETE ON Pedidos -- se va a activar dsps de eliminar datos en la tabla
FOR EACH ROW
BEGIN
    INSERT INTO log_pedidos (n_orden, nombre_usuario, accion_realizada, fecha, hora)
    VALUES (OLD.n_orden, USER(), 'ELIMINACIÓN', CURDATE(), CURTIME());
        -- va a registrar la acción como ELIMINACIÓN y x USER obtendrá el usuario, tmb almacena la fecha y hora x CURDATE y CURTIME
END
//
```

Funciones

1ra función: *getpedidos*

Función para obtener el número de pedidos por cliente.

```
DELIMITER //
CREATE FUNCTION getpedidos (cliente_id INT) /* establezco parámetro de entrada cliente_id) */
RETURNS INT /* quiero que la función me devuelva un número */
DETERMINISTIC
BEGIN
        DECLARE num_pedidos int; /* el resultado se almacena en esta variable num_pedidos */
    SET num_pedidos = (
                SELECT COUNT(*) /* COUNT para contar el N de filas en la tabal Pedidos */
        FROM Pedidos
        WHERE id_usuario = cliente_id /* con id_usuario filtro por el cliente seleccionado */
        );
    RETURN num_pedidos;
END
//
select getpedidos(3);
```

2da función: *getproductos_x_categoria*

Función que concatenará el id_producto con la categoría, de acuerdo a la consulta de categoría.

```
DELIMITER //
CREATE FUNCTION getproductos_x_categoria(categoria VARCHAR(50)) /* establezco parámetro de entrada categoria, que es e
RETURNS VARCHAR(100) /* quiero que la función me devuelva una rta alfanumérica */
DETERMINISTIC
BEGIN
    DECLARE productos_lista VARCHAR(100); /* el resultado se almacena en esta variable productos_lista */

    SELECT GROUP_CONCAT(p.id_producto SEPARATOR ', ') /* Subconsulta: para concatenar los nombres de los productos */
    INTO productos_lista /* almaceno el resultado de la concatenación en la variable productos_lista */
    FROM productos p
    WHERE p.categoria = categoria; /* para filtrar los registros de la tabla Productos según categoria. Solo se van a

    RETURN productos_lista;
END
//
select getproductos_x_categoria('Limpieza');
```

Backup script: click aquí

Informes: análisis de datos

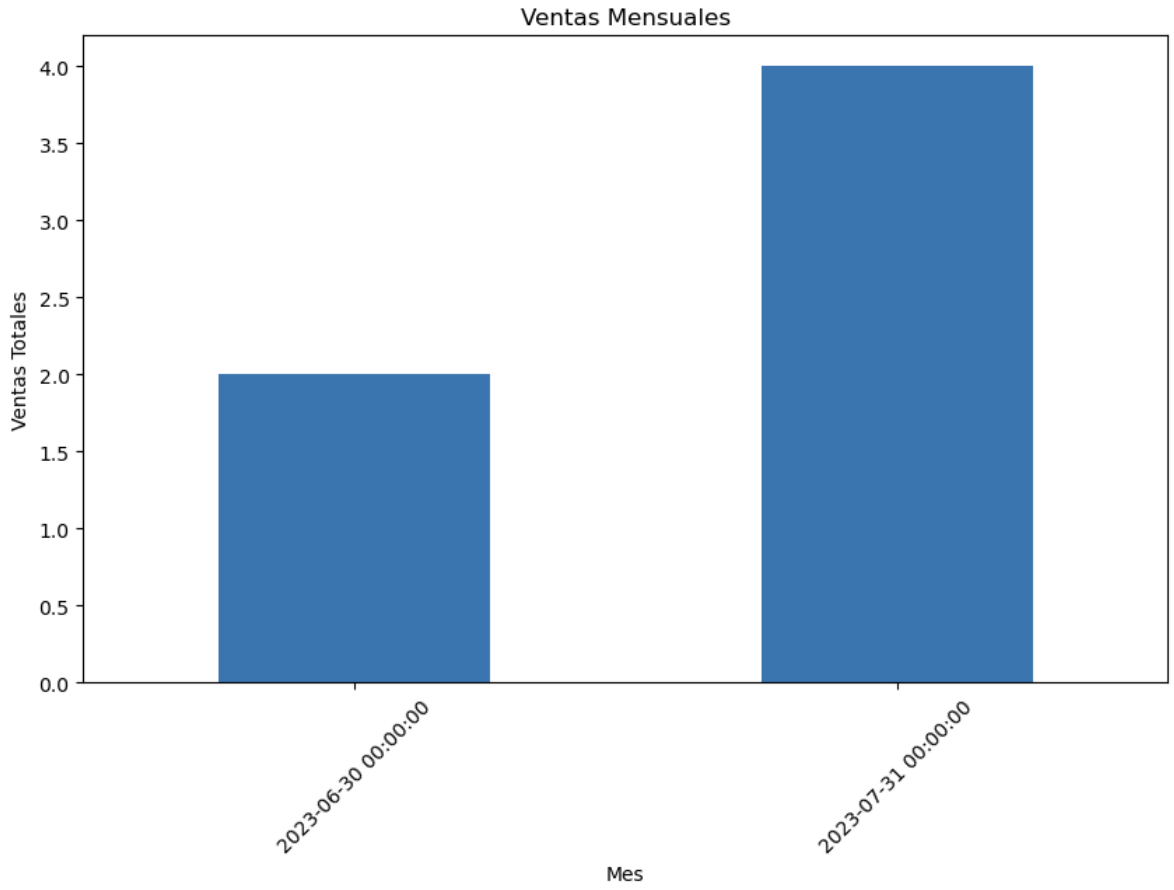
A continuación, se presentarán una serie de gráficos generados mediante el procesamiento y análisis de la información obtenida de las vistas creadas anteriormente. El lenguaje utilizado en este caso fue Python, en la aplicación Jupyter Notebook.

Para ello se realizó, en primera instancia, el *Export Results* de las queries consultadas para luego llevar esos archivos .csv a Jupyter.

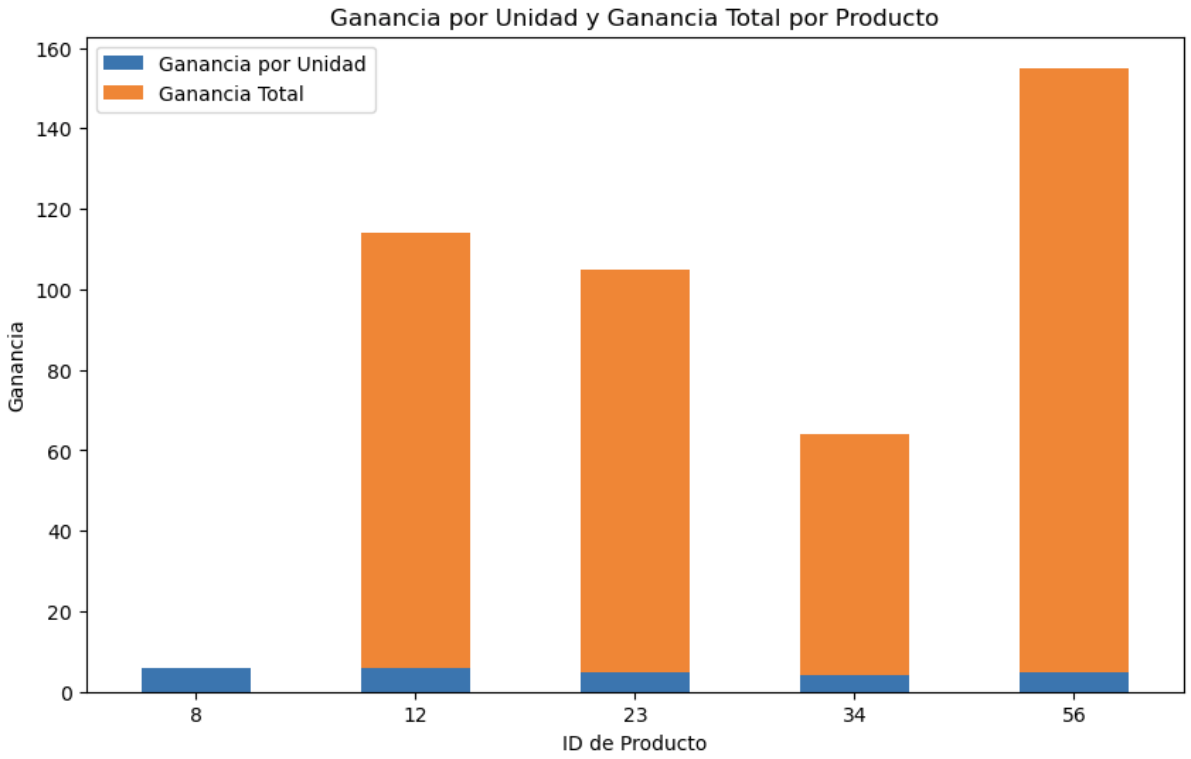
Inicialmente, se analiza la evolución de las compras a lo largo del tiempo, utilizando los resultados de *vista_ultima_compra*:

Al tener una cantidad de datos muy reducida se ve solo una función lineal ascendente (ya que las compras aumentaron en julio con respecto a junio). Pero este gráfico sirve para hacer un análisis de los meses más fuertes, los más flojos y determinar, por ejemplo, cuando es el mejor momento para tomarse vacaciones.

Otra opción de gráfico para observar lo mismo podría ser con barras:

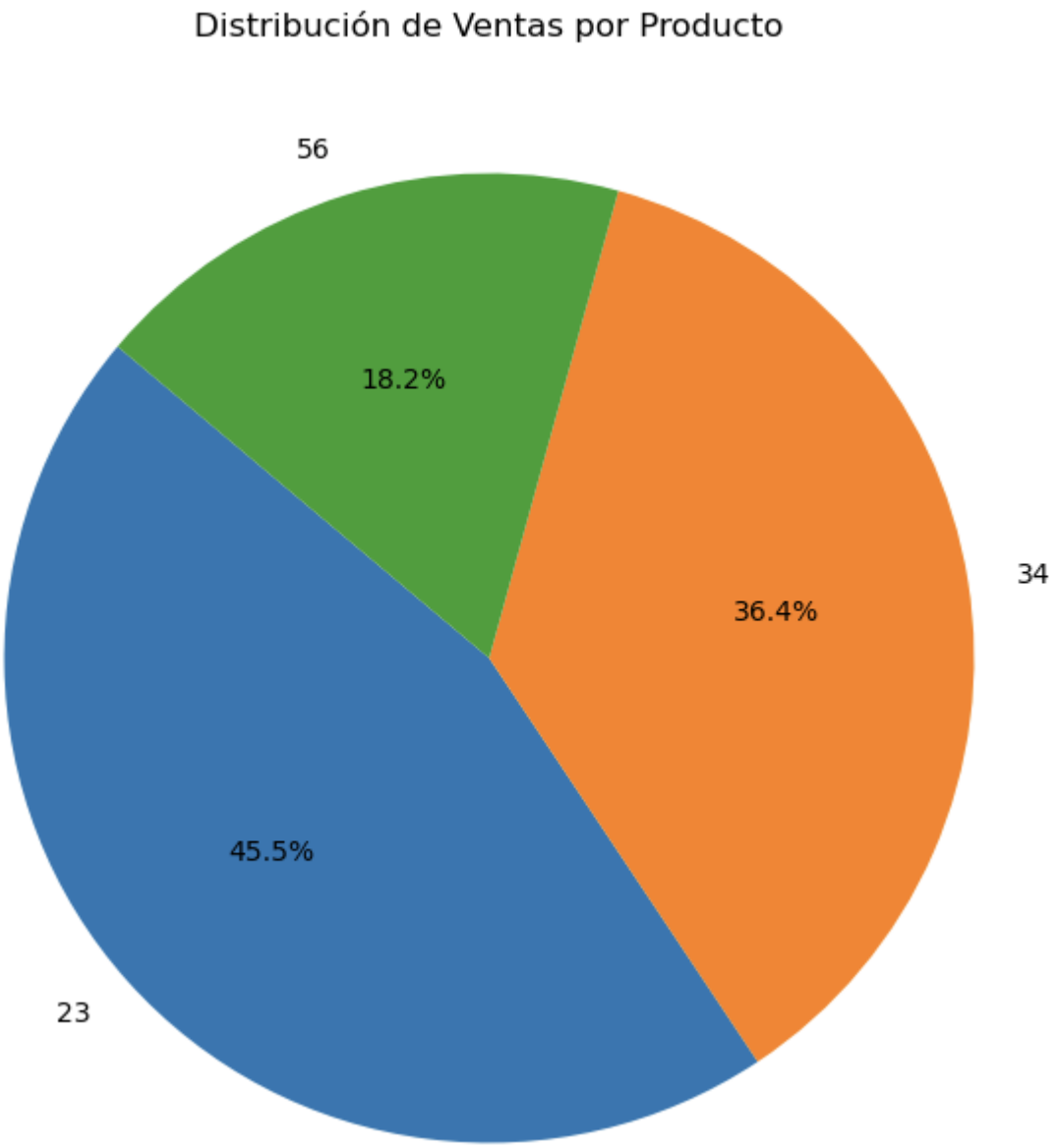


Luego, se analizan los resultados obtenidos de *vista_productos_ganancia* y se analiza la ganancia por unidad y la total:



Se puede observar que del producto con ID 8 no hay stock y, de esta manera, se puede hacer algo al respecto: comunicarse con los proveedores del mismo.

Adicionalmente, examinando los datos de la tabla Pedidos, puedo advertir lo siguiente:



El producto con ID 23 cuenta con el mayor volumen de ventas. De nuevo, se tiene una cantidad reducida de datos y el nulo número de ventas de algunos productos puede estar relacionado a que no hay stock de los mismos. Sin embargo, en condiciones normales, puede ser un buen indicador de los productos que más se venden.

Para más detalles, consultar aquí.

Herramientas utilizadas

- **MySQL Workbench**
- **Google Docs**
- **Anaconda (plataforma que brinda un entorno de desarrollo)**
- **Jupyter Notebook (aplicacion web para programacion en Python)**
- **Creación del logo**