Entrega Final del Proyecto
Base de datos - Tienda de Libros

Profesor: Cesar Aracena

Tutor: Nicolas E. Costante

Estudiante: Carolina Ponce

Índice

Introducción

El presente documento es el proyecto final que se basa en lo aprendido en el curso de SQL de Coderhouse.

Situación Problemática

Se propone crear una tienda de libros, la cual inicialmente no cuenta con un registro de las ventas realizadas, de información de cada vendedor de la tienda, de los clientes y de los productos que posee la tienda. A futuro le costará crear estrategias de negocio.

Objetivo

El objetivo que se propone es, crear una base de datos la cual permita recolectar los distintos datos que puedan aportar cada venta, vendedores o clientes, con ellos analizar las estadísticas de las ventas, la eficiencia de los vendedores, para así poder implementar una mejor estrategia de negocios.

Modelo de negocio

Para poder lograr el objetivo se propone crear una base de datos para la tienda de libros, donde se pueda facilitar la búsqueda de los productos, las ventas realizadas, quién realizó cada venta, las formas de pago, vendedores y clientes.

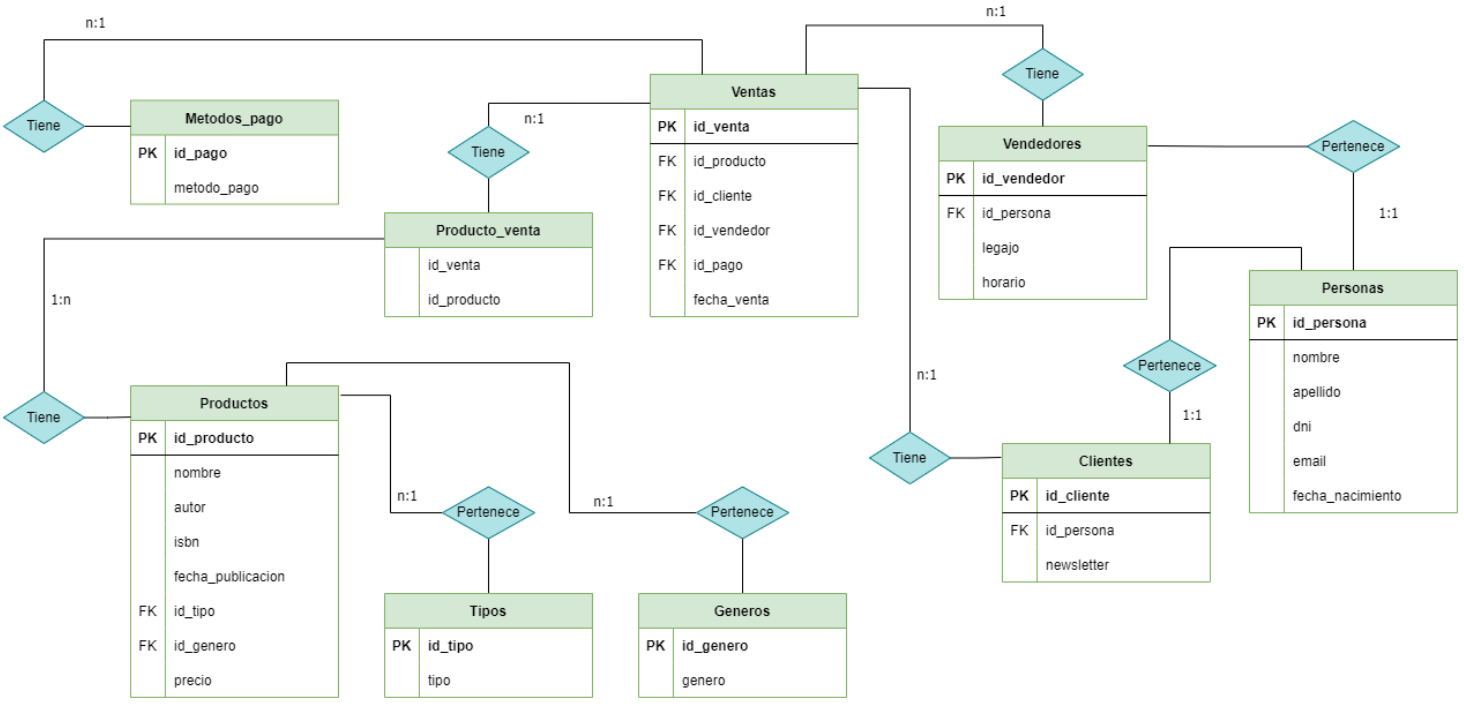
Las tablas que se pueden implementar son:

- ➔ Ventas
- ➔ Personas
- ➔ Vendedores
- ➔ Clientes
- ➔ Productos

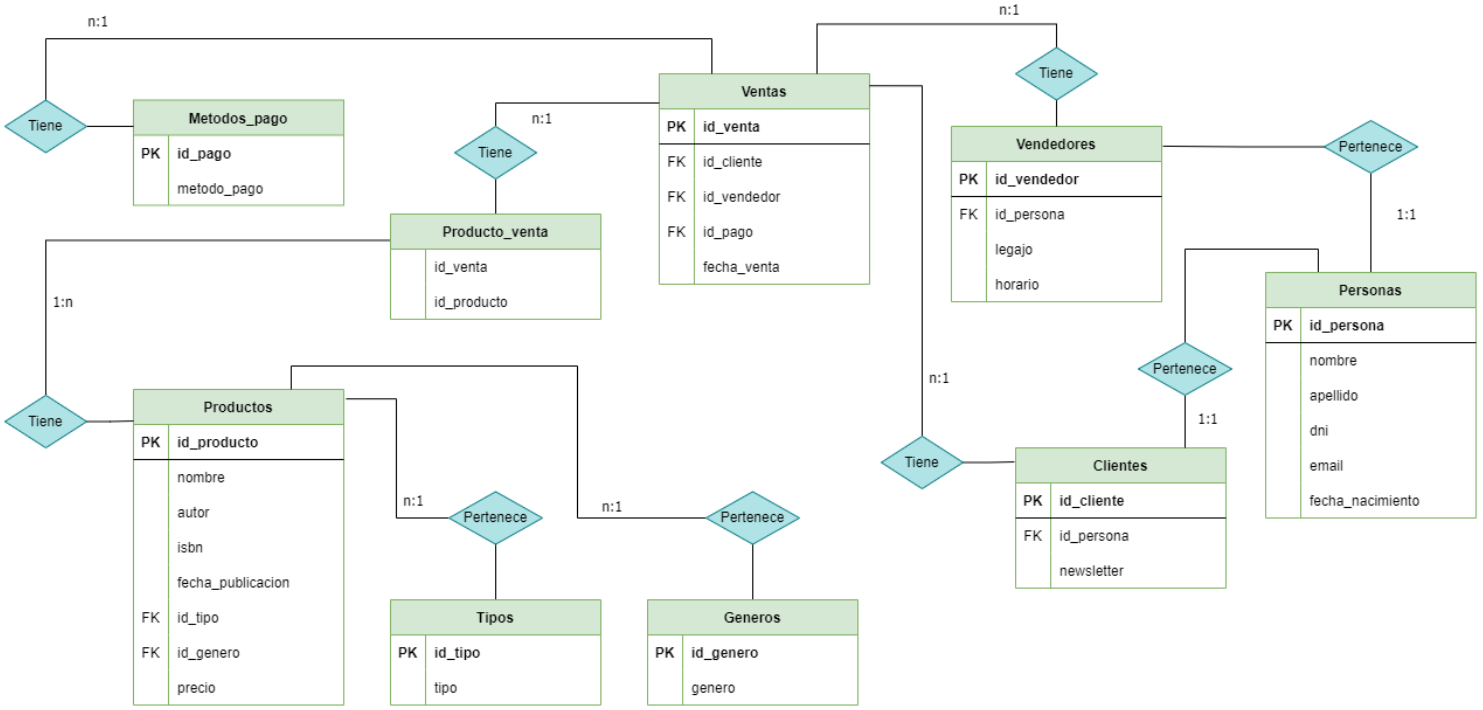
- ➔ Tipos
- ➔ Géneros
- ➔ Productos por ventas
- ➔ Formas de pago

Diagrama Entidad - Relación

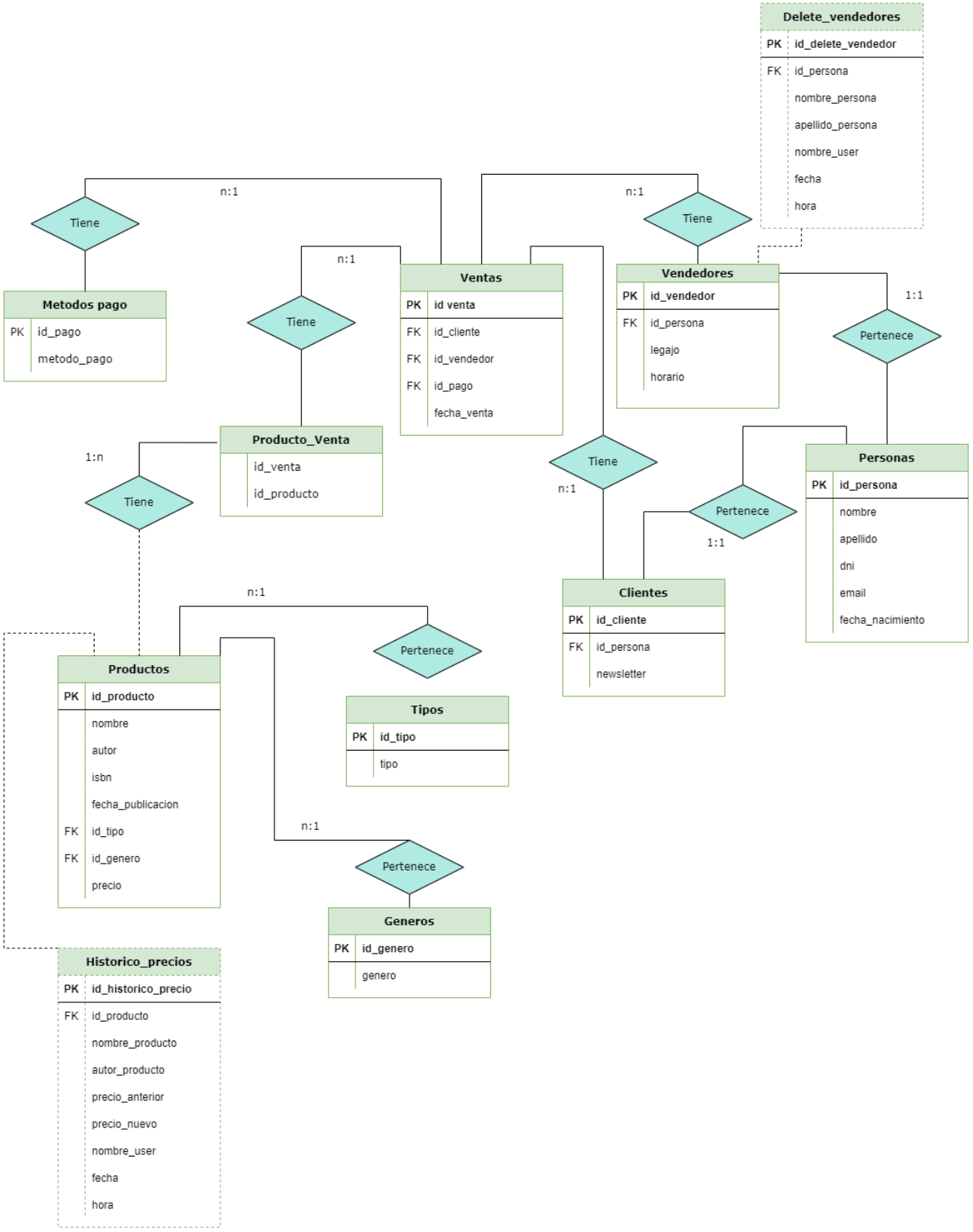
Primera versión:



Segunda versión: se eliminó la columna "id_producto" de la tabla "Ventas".



Última versión:



Listado de tablas

Tabla	VENTAS						
Descripción	TABLA DE HECHO DONDE SE REGISTRAN CADA VENTA						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_venta	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primario de cada venta
FK	id_cliente	INT		NOT NULL			id de cada cliente
FK	id_vendedor	INT		NOT NULL			id de cada vendedor
FK	id_pago	INT		NOT NULL			id de la forma de pago
	fecha_de_venta	DATETIME		NOT NULL			fecha de cada venta realizada

Tabla	PERSONAS						
Descripción	TABLA DIMENSIONAL DONDE SE GUARDAN LOS DATOS DE CADA PERSONA						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_persona	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria de cada persona
	nombre	VARCHAR	32	NOT NULL			nombre de pila
	apellido	VARCHAR	32	NOT NULL			apellido
	dni	INT		NOT NULL			dni de cada persona
	email	VARCHAR	32				email
	fecha_de_nacimiento	DATE					fecha de nacimiento

Tabla	VENDEDORES						
Descripción	TABLA DIMENSIONAL DONDE SE GUARDA LOS DATOS DEL VENDEDOR						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_vendedor	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primario de cada vendedor
FK	id_persona	INT		NOT NULL			id de la persona vendedora
	legajo	INT		NOT NULL	UNIQUE		legajo
	horario	VARCHAR	16			TARDE	horario de trabajo de cada vendedor

Tabla	CLIENTES						
Descripción	TABLA DIMENSIONAL DONDE SE REGISTRAN LOS DATOS DEL CLIENTE						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_cliente	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria de cada cliente
FK	id_persona	INT		NOT NULL			id de la persona cliente
	newsletter	BOOLEAN		NOT NULL		0	TINYINT(1), un valor cero se considera falso y los valores distintos de cero, verdadero. Indicador para saber si el cliente desea recibir un newsletter.

Tabla	PRODUCTOS						
Descripción	TABLA DIMENSIONAL DONDE SE GUARDAN LAS ESPECIFICACIONES DE CADA PRODUCTO						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_producto	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria de cada producto
	nombre	VARCHAR	32	NOT NULL			nombre del libro
	autor	VARCHAR	32	NOT NULL			autor
	isbn	VARCHAR	32	NOT NULL	UNIQUE		número Internacional normalizado para libros
	fecha_de_publicacion	DATE		NOT NULL			fecha de publicación del libro
FK	id_tipo	INT		NOT NULL			id del tipo de producto
FK	id_genero	INT		NOT NULL			id del género del producto
	precio	DECIMAL	7, 2	NOT NULL			precio

Tabla	TIPOS						
Descripción	TABLA DIMENSIONAL DONDE SE GUARDAN LOS TIPOS DE PRODUCTOS						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_tipo	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria del tipo de producto
	tipo	VARCHAR	32	NOT NULL	UNIQUE		tipo de producto (libro, revista, comic)

Tabla	GENEROS						
Descripción	TABLA DIMENSIONAL DONDE SE GUARDAN LOS GENEROS DE LOS PRODUCTOS						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_genero	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria del género del producto
	genero	VARCHAR	16	NOT NULL	UNIQUE		tipo de género

Tabla	PRODUCTO POR VENTA						
Descripción	TABLA INTERMEDIA DONDE SE PUEDE REGISTRAR LOS PRODUCTOS QUE PERTENECEN A CADA VENTA REALIZADA						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
FK	id_venta	INT		NOT NULL			id de la venta realizada
FK	id_producto	INT		NOT NULL			id del producto para la venta realizada

Tabla	MÉTODOS DE PAGO						
Descripción	TABLA DIMENSIONAL DE LAS FORMAS DE PAGO						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_pago	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria de los tipos de pagos
	metodo_de_pago	VARCHAR	32	NOT NULL	UNIQUE		formas de pago

Tabla	DELETE VENDEDORES						
Descripción	TABLA DIMENSIONAL DONDE SE REGISTRAN LOS VENDEDORES ELIMINADOS (TRIGGER)						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_delete_vendedor	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria de los vendedores eliminados
FK	id_persona	INT		NOT NULL			id foránea de cada persona
	nombre_persona	VARCHAR	32	NOT NULL			nombre de cada vendedor eliminado
	apellido_persona	VARCHAR	32	NOT NULL			apellido de cada vendedor eliminado
	nombre_user	VARCHAR	32	NOT NULL			nombre del usuario que realizó la acción
	fecha	DATE					fecha en que se realizó la acción
	hora	TIME					hora en que se realizó la acción

Tabla	HISTÓRICO PRECIO						
Descripción	TABLA DIMENSIONAL DONDE SE REGISTRAN LAS MODIFICACIONES DE PRECIO DE PRODUCTOS (TRIGGER)						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	id_historico_precio	INT		NOT NULL	UNIQUE	AUTO_INCREMENT	id primaria de los cambios de precios
FK	id_producto	INT		NOT NULL			id foránea de cada producto modificado
	nombre_producto	VARCHAR	64	NOT NULL			nombre de cada producto modificado
	autor_producto	VARCHAR	32	NOT NULL			autor de cada producto modificado
	precio_anterior	DECIMAL	7, 2	NOT NULL			precio anterior del producto
	precio_nuevo		7, 2	NOT NULL			precio nuevo modificado del producto
	nombre_user	VARCHAR	32	NOT NULL			nombre del usuario que realizó la acción
	fecha	DATE					fecha en que se realizó la acción
	hora	TIME					hora en que se realizó la acción

Listado de Vistas

Vista: Productos más vendidos

Objetivo: muestra los nombres de los 5 productos más vendidos.
Tablas que la componen: productos y producto_venta.

```
CREATE OR REPLACE VIEW productos_mas_vendidos AS
    (SELECT nombre, COUNT(nombre)
    FROM productos p JOIN producto_venta pv ON (p.id_producto = pv.id_producto)
    GROUP BY nombre ORDER BY COUNT(nombre) DESC
    LIMIT 5);
```

Vista: Vendedores con más ventas

Objetivo: muestra de forma descendente los nombres y apellidos de los vendedores que vendieron algún producto y la cantidad de ventas que tuvo cada uno. Permite a posterior poder brindarles algún beneficio.
Tablas que la componen: ventas, vendedores y personas.

```
CREATE OR REPLACE VIEW vendedores_mas_ventas AS
    (SELECT CONCAT(nombre," ", apellido) AS NombreYApellido, COUNT(v.id_vendedor) AS CantidadVendida
    FROM ventas v JOIN vendedores vd ON (v.id_vendedor = vd.id_vendedor) JOIN personas p ON (p.id_persona = vd.id_persona)
    GROUP BY v.id_vendedor ORDER BY COUNT(v.id_vendedor) DESC);
```

Vista: Precios con IVA

Objetivo: muestra el nombre, precio sin iva y precio con iva de cada producto.
Tablas que la componen: productos.

```
CREATE OR REPLACE VIEW precios_con_iva AS
    (SELECT nombre, precio AS PrecioSinIVA, ((precio*0.21)+precio) AS PrecioConIVA
    FROM productos);
```

Vista: Clientes con más compras

Objetivo: muestra nombre y apellido, y el email de aquellos 5 clientes que compraron más productos. Posteriormente se le puede ofrecer al cliente algún beneficio.

Tablas que la componen: ventas, clientes y personas.

```
CREATE OR REPLACE VIEW clientes_mas_compras AS
    (SELECT CONCAT(nombre," ", apellido) AS NombreYApellido, email, COUNT(c.id_cliente) AS ComprasRealizadas
    FROM ventas v JOIN clientes c ON (v.id_cliente = c.id_cliente) JOIN personas p ON (p.id_persona = c.id_cliente)
    GROUP BY c.id_cliente ORDER BY COUNT(c.id_cliente) DESC
    LIMIT 5);
```

Vista: Cantidad de vendedores por turno

Objetivo: muestra la cantidad que hay de vendedores en los distintos turnos. Permite analizar una reestructuración o adquisición de recurso humano.

Tablas que la componen: vendedores.

```
CREATE OR REPLACE VIEW cantidad_por_turno AS
    (SELECT horario, COUNT(horario)
    FROM vendedores
    GROUP BY horario  ORDER BY COUNT(horario));
```

Vista: Productos vendidos por mes

Objetivo: muestra la cantidad de productos vendidos por mes, en un año específico, en este caso el año 2021.

Tablas que la componen: producto_venta y ventas.

```
CREATE OR REPLACE VIEW productos_vendidos_mes AS
    (SELECT COUNT(pv.id_venta) AS ProductosVendidos, MONTH(v.fecha_venta) AS Mes, YEAR(v.fecha_venta) AS Anio
    FROM producto_venta pv JOIN ventas v ON (pv.id_venta = v.id_ventas)
    WHERE YEAR(v.fecha_venta) = 2021
    GROUP BY MONTH(v.fecha_venta) ORDER BY MONTH(v.fecha_venta));
```

Vista: Productos menos vendidos

Objetivo: muestra los productos menos vendidos. Con la consiguiente información se pueden establecer estrategias para la venta de los mismos. Tablas que la componen: productos y producto_venta.

```
CREATE OR REPLACE VIEW productos_menos_vendidos AS
    (SELECT nombre, autor, COUNT(pv.id_producto) AS CantidadVendidos
    FROM productos pr JOIN producto_venta pv ON (pr.id_producto = pv.id_producto)
    GROUP BY pv.id_producto ORDER BY COUNT(pv.id_producto) ASC LIMIT 10
    );
```

Listado de Funciones

Función: Bonificación por compra

Función que permite recibir por parámetro el id de una venta, suma el precio de cada producto que tiene esa venta y le proporciona un descuento si supera los $20000, los $10000 o el precio sigue siendo el mismo.

Objetivo: proporcionar un descuento del 10% y del 20% para aquellas ventas que superen los $10000 y los $20000 respectivamente.

Tablas que la componen: producto_venta y productos.

```
CREATE  FUNCTION `bonificacion_por_compra`(venta INT) RETURNS decimal(7,2)
    READS SQL DATA
BEGIN
    DECLARE precio DECIMAL(7,2);
    DECLARE resultado  DECIMAL(7,2);
    SET precio = (SELECT SUM(ps.precio)
                FROM producto_venta pv JOIN productos ps ON (pv.id_producto = ps.id_producto)
                WHERE pv.id_venta = venta);
    IF precio >= 20000 THEN
        SET resultado = (precio -(precio * 0.2));
    ELSEIF precio >= 10000 THEN
        SET resultado = (precio -(precio * 0.1));
    ELSE
        SET resultado = precio;
    END IF;
RETURN resultado;
END
```

Función: Ventas por año

Función que permite conocer cuántas ventas realizó por año cada vendedor.
Objetivo: permite analizar si es necesario incentivar a los vendedores para que realicen más ventas.
Tablas que la componen: ventas.

```
CREATE FUNCTION `ventas_por_anio`(vendedor INT, anio INT) RETURNS varchar(128)
BEGIN
    DECLARE cant_ventas INT;
    DECLARE resultado VARCHAR (128);
    SET cant_ventas = (SELECT COUNT(id_vendedor) FROM libreria.ventas WHERE id_vendedor = vendedor AND YEAR(fecha_venta) = anio);
    IF cant_ventas >= 2 THEN
        SET resultado = "El vendedor recibirá una bonificación por tener mas de 2 ventas";
    ELSE
        SET resultado = "LLamado de atención al vendedor";
    END IF;
RETURN resultado;
END
```

Listado de Store Procedures

Store Procedure: Ingresar nuevo vendedor

Objetivo: permite ingresar un vendedor a la tabla vendedores, a la misma vez que se inserta la persona asociada a la tabla personas.
Tablas que la componen: personas y vendedores.

```
CREATE PROCEDURE `ingresar_nuevo_vendedor`(
        IN nombre_param VARCHAR(32),
        IN apellido_param VARCHAR(32),
        IN dni_param INT,
        IN email_param VARCHAR(32),
        IN fecha_nacimiento_param DATE,
        IN legajo_param INT,
        IN horario_param VARCHAR(16)
)
```

```
BEGIN
    DECLARE id_ultima_persona INT;
    DECLARE email_new VARCHAR(32);
    DECLARE fecha_nacimiento_new DATE;
    DECLARE horario_new VARCHAR(16);

    IF(nombre_param <> '') AND (apellido_param <> '') THEN
        IF email_param = '' THEN
            SET email_new = NULL;
        ELSE
            SET email_new = email_param;
        END IF;
        IF fecha_nacimiento_param = '' THEN
            SET fecha_nacimiento_new = NULL;
        ELSE
            SET fecha_nacimiento_new = fecha_nacimiento_param;
        END IF;
        IF horario_param = '' OR horario_param <> 'mañana' THEN
            SET horario_new = 'tarde';
        ELSE
            SET horario_new = horario_param;
        END IF;

        INSERT INTO personas (id_persona, nombre, apellido, dni, email, fecha_nacimiento) VALUE
            (NULL, nombre_param, apellido_param, dni_param, email_new, fecha_nacimiento_new);
        SET id_ultima_persona = LAST_INSERT_ID();

        INSERT INTO vendedores (id_vendedor, id_persona, legajo, horario) VALUE
            (NULL, id_ultima_persona, legajo_param, horario_new);
    END IF;
END
```

Store Procedure: Ordenar campo de la tabla personas

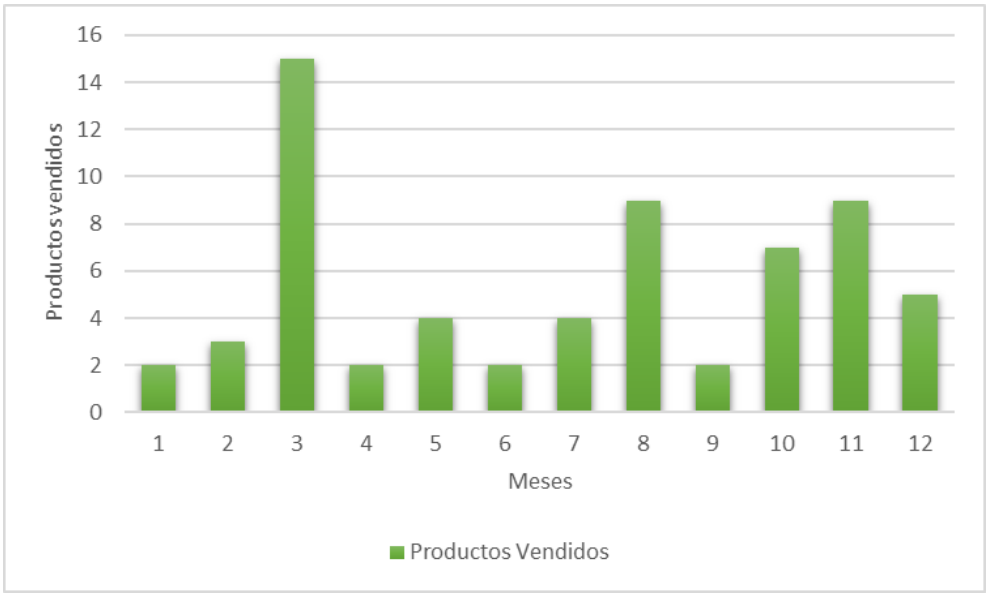
Objetivo: permite ordenar la tabla personas, según un determinado campo y determinado orden (ascendente o descendente).

Tablas que la componen: personas.

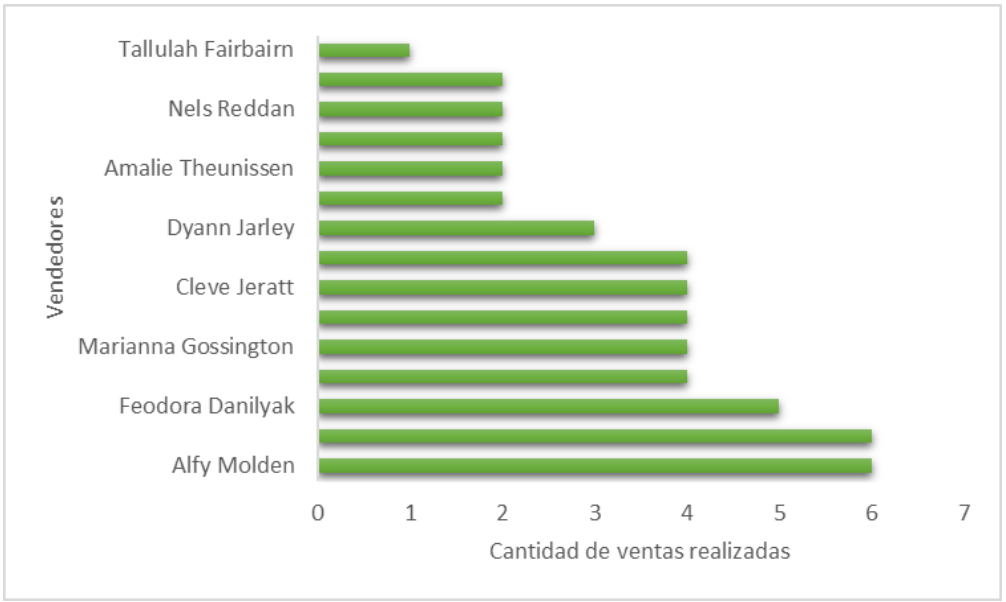
```
CREATE PROCEDURE `ordenar_campo`(IN campo varchar(32), IN orden VARCHAR(16))
BEGIN
    IF campo <> '' THEN
        SET @orden_campo = CONCAT('ORDER BY ', campo, ' ');
        IF orden = 'ASC' OR orden = 'DESC'  THEN
            SET @orden_campo =  CONCAT(@orden_campo, orden);
        END IF;
    ELSE
        SET @orden_campo = '';
    END IF;
    SET @clausula = concat('SELECT * FROM personas ', @orden_campo);
    PREPARE runSQL FROM @clausula;
    EXECUTE runSQL;
    DEALLOCATE PREPARE runSQL;
END
```

Gráficos

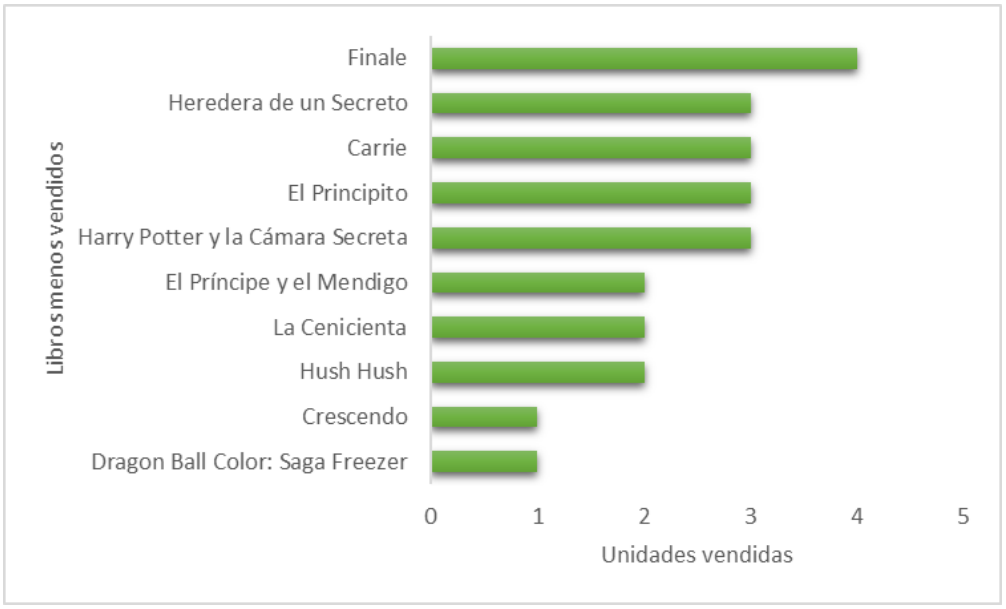
A través de la vista “productos_vendidos_mes” se puede observar la cantidad de productos vendidos por mes, en el año 2021. La gran cantidad de productos vendidos en el mes de abril puede coincidir con el evento “Feria del libro”. Este análisis puede ayudar a crear estrategias para los consiguientes al mes de abril, donde se pueden aplicar descuentos para no perder ventas.



A través del siguiente gráfico podemos observar aquellos vendedores que obtuvieron como mínimo una venta y quienes obtuvieron mayor cantidad de ventas. Permite a posterior la oportunidad de brindarles algún beneficio por el buen trabajo realizado. Dicho gráfico fue realizado gracias a los resultados de la vista “vendedores con más ventas”.

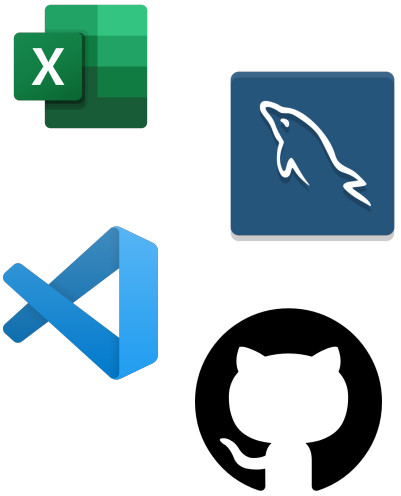


Por medio de la vista "Productos menos vendidos", se pueden observar aquellos ejemplares que no tuvieron tantas ventas. Con la consiguiente información se pueden establecer estrategias para la venta de los mismos.



Herramientas y tecnologías usadas

Microsoft Excel 365
MySQL Workbench 8.0 Versión 8.0.30
Visual Studio Code 1.74.2
Github



Anexo

Link a Github:
https://github.com/Caromponce/Curso-SQL