PROYECTO FINAL SQL

Sistema de gestión para manejo de ventas online.

El sistema deberá tener en cuenta los siguientes puntos:

- Ingreso de clientes.
- Ingresos de productos.
- Gestión de stock.
- Precio o bonificación de envió dependiendo zona y monto.
- Cancelación de pedido / compra.
- Usado por vendedor para gestionar, no por cliente.
- Sin manejo de imágenes, solo datos.
- Facturación.

LINK A REPOSITORIO:

https://github.com/rodriguezjulian/SQL-FINAL.git

Para la población de tablas se utilizo https://www.mockaroo.com/ y chat GPT a fin de obtener datos rápidamente.

Vistas y sus campos a mostrar

vw_ClientesV3;						
ID del cliente (C)	Nombre	Cuit_Cuil	Descripcion_Zona_Geografica	Descripcion_Condicion_Fiscal	Metodo de pago	Medio de pago
vw_VendedoresV1						
ID del vendedor	Razon_Social	CUIT				
vw_ArticulosV1						
ID del articulo	Descripcion	Color	Stock	Precio	ID del vendedor	Razon_Social
vw_ordenes;						
ID de orden	Nombre del C	ID del vendedor	Razon_Social	Descripcion del articulo	Unidades	
vw_metodosPago;						
ID metodo de pago	Metodo de P	Medio de P				
vw_zonaGeografica						
ID zona geografica	Descripcion de Z					
vw_condificionFiscal;						
ID condicion fiscal	Descripcion CF					

Rodriguez Julian Manuel

Explicación SP

```
DELIMITER $$
CREATE PROCEDURE ordenar_clientes(IN campo VARCHAR(50), IN orden VARCHAR(4))
BEGIN
 SET @sql = CONCAT('SELECT * FROM cliente ORDER BY ', campo, ' ', orden);
 PREPARE stmt FROM @sql;
 EXECUTE stmt;
 DEALLOCATE PREPARE stmt;
END$$
```

1. **`DELIMITER $$`**: Esta línea establece el delimitador a "$$" en lugar del delimitador predeterminado ";" para que el stored procedure pueda contener instrucciones SQL múltiples.
2. **`CREATE PROCEDURE ordenar_clientes(IN campo VARCHAR(50), IN orden VARCHAR(4))`**: Esta línea crea un nuevo stored procedure llamado "ordenar_clientes" con dos parámetros de entrada: "campo" y "orden". "campo" es una cadena que indica el nombre del campo por el que se ordenará la tabla, mientras que "orden" es una cadena que indica el orden de clasificación (ASC o DESC).
3. **`BEGIN`**: Esta línea indica el inicio del cuerpo del stored procedure.
4. **`SET @sql = CONCAT('SELECT * FROM cliente ORDER BY ', campo, ' ', orden);`**: Esta línea crea una cadena SQL dinámica que ordenará la tabla "cliente" según el campo y el orden especificados en los parámetros de entrada. La función CONCAT se utiliza para concatenar las cadenas.
5. **`PREPARE stmt FROM @sql;`**: Esta línea prepara la consulta SQL dinámica almacenada en la variable "@sql" para su posterior ejecución. La consulta preparada se almacena en la variable "stmt".
6. **`EXECUTE stmt;`**: Esta línea ejecuta la consulta SQL preparada almacenada en la variable "stmt".
7. **`DEALLOCATE PREPARE stmt;`**: Esta línea libera la memoria utilizada por la consulta preparada almacenada en la variable "stmt".
8. **`END$$`**: Esta línea indica el final del cuerpo del stored procedure.
9. **`DELIMITER;`**: Esta línea restaura el delimitador predeterminado ";".

TRIGGER tg_articulo_backup

El código crea un trigger llamado **`eliminar_factura_trigger`** que se activará automáticamente después de que se elimine un registro en la tabla **`factura`**. Cuando se dispare el trigger, se insertará una copia de los valores de la fila eliminada en la tabla **`factura_backUp`**, junto con la fecha y hora actual de la

operación y el usuario que la ejecutó. Este proceso se repetirá para cada fila eliminada de la tabla **`factura`**.

El código crea un trigger llamado **`tg_articulo_backup`** que se activará automáticamente después de que se inserte un registro en la tabla **`articulo`**. Cuando se dispare el trigger, se insertará una copia de los valores de la fila insertada en la tabla **`articulo_backUp`**, junto con la fecha y hora actuales de la operación y el usuario que la ejecutó. Este proceso se repetirá para cada fila insertada en la tabla **`articulo`**.