Objetivo: Siguiendo con el aprendizaje sobre sentencias SQL, los alumnos cargarán una base de datos proporcionada y realizarán una serie de consultas y manipulaciones de datos, aplicando diferentes tipos de sentencias SQL como `SELECT`, `INSERT`, `UPDATE`, `DELETE`, y la gestión de transacciones con `COMMIT` y `ROLLBACK`.

Instrucciones

1. **Cargar la Base de Datos**
   - **Descarga e importa** la base de datos proporcionada en tu entorno de trabajo (MySQL o H2 Database).
   - **Revisa** la estructura de las tablas y familiarízate con los campos y relaciones para comprender el contexto.

     database_script.sql

---

2. **Ejercicio de Consultas y Manipulación de Datos**

   Realiza los siguientes ejercicios en el mismo orden. Cada ejercicio debe ser **documentado** con la consulta ejecutada y los resultados obtenidos. Explica brevemente cada operación y el motivo de los resultados.

   - **Consulta SELECT básica**: Muestra todos los registros de una tabla específica (por ejemplo, `CUSTOMER`).
   - **SELECT con filtro**:
     - Realiza una consulta que obtenga solo los clientes cuya edad sea mayor de 30 años.
     - Ordena los resultados alfabéticamente por nombre.
   - **Uso de operadores en WHERE**:
     - Selecciona los clientes cuya ciudad sea “Málaga” o “Granada”.
     - Selecciona los productos cuyo precio esté entre 20 y 50 unidades monetarias.
   - **Uso de operadores avanzados**:
     - Usa el operador `LIKE` para seleccionar todos los registros de `CUSTOMER` cuyo apellido comience por "L".
     - Usa el operador `IN` para mostrar los clientes que vivan en "Sevilla", "Córdoba" o "Jaén".
   - **Consultas con Ordenación**:
     - Realiza una consulta que muestre todos los empleados ordenados por salario en orden descendente.
     - Realiza una consulta que muestre los productos en orden ascendente por nombre.
   - **Funciones de agregado**:
     - Calcula el número total de clientes en la tabla.
     - Calcula la media de edad de los clientes.
     - Obtén el precio máximo y mínimo de los productos.
   - **Cláusula GROUP BY**:
     - Muestra el número de clientes por cada ciudad.
     - Agrupa los productos por categoría y muestra la cantidad de productos en cada categoría.

- **JOIN**:
  - Realiza una consulta que muestre el nombre del cliente y su dirección, combinando las tablas `CUSTOMER` y `ADDRESS`.
  - Usa un `LEFT JOIN` para listar todos los pedidos junto con el nombre del cliente, aunque algunos pedidos no tengan cliente asignado.
- **Transacciones**:
  - Realiza una transacción que incluya la inserción de un nuevo cliente y un nuevo pedido asociado a este cliente.
  - Realiza un `ROLLBACK` en caso de error durante la transacción.
  - Usa `COMMIT` para guardar los cambios si la transacción es exitosa.

---

3. **Desarrollo de la Aplicación**
   - **Inserción de Nuevos Clientes desde la Consola**: Permitir la inserción de clientes en la base de datos (Nombre, Apellido, Email, Dirección y Teléfono), ingresados por medio de la consola.
   - **Visualización de Clientes**: Permitir la visualización de todos los clientes almacenados en la base de datos, mostrando su información en una lista en la consola.
   - **Desarrollo de Consultas Adicionales**:
     - Crear consultas para buscar clientes por nombre o email.
     - Consultar detalles de pedidos para un cliente específico y calcular el precio total de cada pedido.
   - **Gestión de Transacciones**:
     - Deshabilitar el modo autocommit y utilizar confirmaciones (`COMMIT`) y deshacer (`ROLLBACK`) para mantener la integridad de los datos durante la inserción y eliminación de registros.
   - **Creación/Eliminación de Objetos para Almacenar Resultados**:
     - Usa `executeUpdate()`, que devuelve un número entero con el conteo de filas afectadas,
     - Usar `cerrarConexion(dbConnection)` para liberar los recursos al finalizar.
4. **Entrega de Resultados**
   - **Documenta** cada consulta y operación en un archivo de texto o en un documento PDF, incluyendo explicaciones sobre cada sentencia SQL y capturas de pantalla de los resultados obtenidos.
   - **Código Fuente**: Entrega el código fuente de la aplicación desarrollada.
   - **Documentación**: Incluye una descripción de cómo funciona la aplicación, cómo se gestionan las transacciones y las pruebas realizadas que demuestren el funcionamiento del sistema.
   - **Pruebas**: Asegúrate de incluir pruebas que demuestren el correcto funcionamiento de la aplicación y la gestión de la base de datos.

Formato de entrega: Entrega un archivo comprimido en formato `.zip` o `.rar` con el nombre **ApellidosNombre_TareaSQL**.

Aquí tienes un ejemplo adaptado para la tabla `PRODUCT` de la base de datos que te proporcioné. Este código en Java permite **insertar nuevos productos** desde la consola, **mostrar los productos** almacenados en la base de datos, y cierra la conexión una vez completada cada operación. Asume que los datos de conexión (`DRIVER`, `URL_CONEXION`, `usuario`, y `password`) están definidos.

```
import java.sql.*;
import java.util.Scanner;

public class ProductoManager {

    public static void insertarProductoDesdeConsola() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Ingresar información del producto:");
        System.out.print("Nombre del producto: ");
        String nombre = scanner.nextLine();
        System.out.print("Precio: ");
        double precio = scanner.nextDouble();
        scanner.nextLine(); // Limpiar el buffer
        System.out.print("Categoría: ");
        String categoria = scanner.nextLine();

        try {
            insertarProducto(nombre, precio, categoria);
        } catch (SQLException e) {
            System.out.println("Error al insertar el producto: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Error al cargar el controlador de la base de datos: " + e.getMessage());
        }
    }

    public static void insertarProducto(String nombre, double precio, String categoria) throws
SQLException, ClassNotFoundException {
        Connection dbConnection = null;
        try {
            dbConnection = obtenerConexion();
            dbConnection.setAutoCommit(false); // Deshabilitar el modo autocommit

         // Insertar el producto
        String insertProductoSQL = "INSERT INTO PRODUCT (product_name, price, category)
VALUES (?, ?, ?)";
        PreparedStatement productoStatement =
dbConnection.prepareStatement(insertProductoSQL);
        productoStatement.setString(1, nombre);
        productoStatement.setDouble(2, precio);
        productoStatement.setString(3, categoria);

        int filasAfectadasProducto = productoStatement.executeUpdate();

        // Insertar la orden de pedido
        String insertOrdenSQL = "INSERT INTO ORDER (order_number, customer_name) VALUES
(?, ?)";
        PreparedStatement ordenStatement = dbConnection.prepareStatement(insertOrdenSQL);
        ordenStatement.setString(1, numeroOrden);
        ordenStatement.setString(2, cliente);

        int filasAfectadasOrden = ordenStatement.executeUpdate();

        // Confirmar la transacción si ambas inserciones son exitosas
        if (filasAfectadasProducto > 0 && filasAfectadasOrden > 0) {
            dbConnection.commit(); // Confirma la transacción si ambas inserciones se realizaron con
éxito
            System.out.println("Producto y orden insertados con éxito.");
        } else {
            System.out.println("No se pudieron insertar el producto y/o la orden.");
            dbConnection.rollback(); // Deshace la transacción en caso de fallo en alguna de las
```

```
inserciones
        }

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
            if (dbConnection != null) {
                dbConnection.rollback(); // Deshacer la transacción en caso de error
            }
        } finally {
            // Cerrar la conexión al finalizar
            cerrarConexion(dbConnection);
        }
    }

    public static void mostrarProductos() {
        Connection dbConnection = null;

        try {
            dbConnection = obtenerConexion();
            Statement statement = dbConnection.createStatement();
            String selectTableSQL = "SELECT * FROM PRODUCT";
            ResultSet rs = statement.executeQuery(selectTableSQL);

            while (rs.next()) {
                int id = rs.getInt("product_id");
                String nombre = rs.getString("product_name");
                double precio = rs.getDouble("price");
                String categoria = rs.getString("category");
                System.out.println("ID Producto: " + id);
                System.out.println("Nombre: " + nombre);
                System.out.println("Precio: " + precio);
                System.out.println("Categoría: " + categoria);
                System.out.println("-----------------------");
            }

        } catch (SQLException e) {
            System.out.println("Error al mostrar los datos de los productos: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Error al cargar el controlador de la base de datos: " + e.getMessage());
        } finally {
            cerrarConexion(dbConnection); // Cerrar la conexión al finalizar
        }
    }

    private static Connection obtenerConexion() throws SQLException, ClassNotFoundException {
        Class.forName("com.mysql.cj.jdbc.Driver"); // Cambiar DRIVER según tu configuración
        return DriverManager.getConnection("jdbc:mysql://localhost:3306/tu_base_datos", "usuario",
"password");
    }

    private static void cerrarConexion(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                System.out.println("Error al cerrar la conexión: " + e.getMessage());
            }
        }
    }
}
```