The Order Management System (OMS)

OMS is designed to manage and display orders in a user-friendly interface, allowing users to view, add, edit, and expire orders. It features a grid to display all orders with the ability to perform batch or individual operations on orders. This system also maintains a history of all orders by marking orders as expired rather than deleting them permanently, supporting audit and historical tracking.

Key Features

- **Order Grid Display**: A comprehensive grid to view all orders with columns for Order ID, Order Number, Order Date, Amount, Vendor, and Shop.
- **Add Order:** Allows users to add new orders with specific details.
- **Edit Order**: Enables editing of selected order fields, with Order Number displayed as a read-only field.
- **Delete Order (Expire)**: Provides options to delete individual order, marking them as expired to retain order history.
- **Batch Delete (Expire) Orders**: Users can select multiple orders and expire them in a single operation.

OrderManagementDB

This database manages order information and associated vendor details for an order management system. It includes two tables, OrderDetails and Vendors, along with various stored procedures to handle data operations like inserting, retrieving, updating, and deleting orders and vendors.

Tables Overview

1. OrderDetails Table

This table stores detailed information about each order, including order identification, amount, vendor reference, shop information, and audit timestamps.

Columns:

Column	Data Type	Constraints	Description
OrderId	INT	Primary Key, Not Null, Auto Increment	Unique identifier for each order. Automatically increments with each insert.

OrderDate	DATETIME		Date when the order was placed.
OrderNumber	INT	Unique Key, Auto-generated	Unique order number, incremented by 100s. Set via trigger trg_set_order_number.
Amount	DECIMAL(10,2)		Total monetary value of the order.
VendorId	INT	Foreign Key to Vendors.VendorId	References the vendor responsible for the order.
ShopId	INT		Identifier for the shop where the order was placed.
CreatedDateTime	DATETIME		Timestamp of when the record was created.
UpdatedDateTime	DATETIME		Timestamp of the last update to the record.
ExpiredDateTime	DATETIME		Timestamp indicating when the record was marked as expired.

Column Explanations:

- **OrderId**: Primary key, unique for each order, auto-increments with each new entry.
- **OrderNumber**: Unique key generated by the trg_set_order_number trigger, which automatically sets and increments the value in steps of 100 on each insert.
- **Amount**: Stored as a decimal with a precision of 10 and scale of 2.
- **Audit Columns**:
    - **CreatedDateTime** records when the order was first inserted.
    - **UpdatedDateTime** stores the timestamp of the last modification.
    - **ExpiredDateTime** indicates when the record is marked as expired, useful for soft deletion or data lifecycle tracking.

2. Vendors Table

This table holds information about vendors associated with orders in the system.

Columns:

Column	Data Type	Constraints	Description

VendorId	INT	Primary Key, Not Null	Unique identifier for each vendor.
VendorName	VARCHAR	Not Null, Variable Length	Name of the vendor

Column Explanations:

- **VendorId**: Primary key for each vendor. This ID is referenced by the VendorId column in the OrderDetails table to link each order to a specific vendor.
- **VendorName**: Stores the name of the vendor.

Stored Procedures Overview

The OrderManagementDB database includes six stored procedures designed to manage order data and vendor information. Each procedure is optimised to handle specific data operations within the database.

1. usp_insert_order_details.sql

- **Description**: Inserts a new order record into the OrderDetails table.
- **Parameters**: Accepts all required fields for an order entry.
- **Behaviour**: Inserts data into OrderDetails table. Order Number is generated automatically based on trg_set_order_number trigger.

2. usp_get_order_details

- **Description**: Retrieves a list of all orders stored in the OrderDetails table.
- **Returns**: A result set containing details of each order, including OrderId, OrderDate, OrderNumber, Amount, VendorId, and ShopId.

3. usp_get_order_summary

- **Description**: Retrieves a summary of a single order based on the OrderId.
- **Parameters**: OrderId (int) - ID of the order to retrieve.
- **Returns**: A single row with a summary of the specified order.

4. usp_get_vendors

- **Description**: Retrieves a list of all vendors stored in the Vendors table.
- **Returns**: A result set containing VendorId and VendorName for each vendor.

5. usp_update_order_details

- **Description**: Updates details of an existing order in the OrderDetails table.

- **Parameters**: Accepts all fields that can be updated, including OrderDate, Amount, VendorId, ShopId, and audit fields.
- **Behaviour**: Updates the specified order and modifies the UpdatedDateTime column to the current timestamp.

6. usp_delete_order_details

- **Description**: Deletes one or more orders based on the specified OrderId values.
- **Parameters**: OrderId (single or list of integers) - ID(s) of the orders to delete.
- **Behaviour**: Deletes the specified orders. Also handles bulk deletion if provided with a list of OrderId values.

Trigger: trg_set_order_number

This trigger is executed on each insert to the OrderDetails table, automatically assigning a unique OrderNumber that increments in steps of 100 for every new entry.

- **Trigger Name**: trg_set_order_number
- **Purpose**: Ensures each order has a unique OrderNumber, auto-incremented by 100.
- **Execution**: Fires on each insert operation to OrderDetails.