PharmaRoute: Online Medicine Delivery Database System

This project focuses on building a comprehensive, real-time online medicine delivery system using MySQL. The system will include robust database design, and advanced SQL features such as triggers, stored procedures, views, and complex data retrieval tasks. The objective is to provide hands-on experience with database management and optimization in a real-world e-commerce scenario.

Tables Design and Details

1. **Users**
    - **Columns:**
        - user_id (Primary Key, AUTO_INCREMENT)
        - name (VARCHAR)
        - email (VARCHAR, UNIQUE, INDEXED)
        - phone_number (VARCHAR, UNIQUE)
        - user_type (ENUM: 'Customer', 'Admin', 'Pharmacy', 'DeliveryPerson')
        - address (TEXT)
        - registered_on (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
    - **Associations:**
        - Foreign Key in Orders, Payments, Feedback, Deliveries.
2. **Medications**
    - **Columns:**
        - medication_id (Primary Key, AUTO_INCREMENT)
        - name (VARCHAR)
        - description (TEXT)
        - price (DECIMAL)
        - stock_quantity (INT)
        - category (VARCHAR)
        - prescription_required (BOOLEAN)
    - **Associations:**
        - Linked to Orders and Feedback.
3. **Orders**
    - **Columns:**
        - order_id (Primary Key, AUTO_INCREMENT)
        - customer_id (Foreign Key → Users)
        - pharmacy_id (Foreign Key → Users)

        - order_status (ENUM: 'Pending', 'Processed', 'Shipped', 'Delivered', 'Cancelled')
        - order_date (DATETIME, INDEXED)
        - delivery_address (TEXT)
        - payment_status (ENUM: 'Pending', 'Completed', 'Refunded')
    - **Associations:**
        - Linked to Users (Customer, Pharmacy) and Payments.
4. **OrderDetails**
    - **Columns:**
        - order_detail_id (Primary Key, AUTO_INCREMENT)
        - order_id (Foreign Key → Orders)
        - medication_id (Foreign Key → Medications)
        - quantity (INT)
        - unit_price (DECIMAL)
        - total_price (DECIMAL)
    - **Associations:**
        - Linked to Orders and Medications.
5. **Payments**
    - **Columns:**
        - payment_id (Primary Key, AUTO_INCREMENT)
        - order_id (Foreign Key → Orders)
        - amount (DECIMAL)
        - payment_date (DATETIME, INDEXED)
        - payment_method (ENUM: 'Credit Card', 'Debit Card', 'Wallet', 'Cash')
        - payment_status (ENUM: 'Pending', 'Completed', 'Refunded')
    - **Associations:**
        - Linked to Orders.
6. **Feedback**
    - **Columns:**
        - feedback_id (Primary Key, AUTO_INCREMENT)
        - order_id (Foreign Key → Orders)
        - user_id (Foreign Key → Users)
        - rating (INT)
        - comments (TEXT)
    - **Associations:**
        - Linked to Orders and Users.
7. **Deliveries**
    - **Columns:**
        - delivery_id (Primary Key, AUTO_INCREMENT)
        - order_id (Foreign Key → Orders)

        - delivery_person_id (Foreign Key → Users)
        - delivery_status (ENUM: 'Pending', 'Out for Delivery', 'Delivered')
        - delivery_date (DATETIME)
    - **Associations:**
        - Linked to Orders and Users (DeliveryPerson).
8. **OperationLogs**
    - **Columns:**
        - log_id (Primary Key, AUTO_INCREMENT)
        - operation_type (VARCHAR)
        - table_name (VARCHAR)
        - operation_time (TIMESTAMP)
        - changed_data (JSON)
    - **Associations:**
        - Standalone logging table.

Tasks

Phase 1: Database and Table Design

- Create the **MedDeliverDB** database.
- Design all 10 tables with optimized data types and constraints.
- Establish foreign key relationships between tables.
- Add ON DELETE CASCADE and ON UPDATE CASCADE constraints.
- Normalize the database to eliminate redundancy.
- Partition **Orders** table by **order_date** for large datasets.
- Create indexes on foreign key columns in all tables.
- Insert sample data for testing (100 rows per table).

Phase 2: Index Optimization

- Index **email** and **phone_number** columns in **Users**.
- Index **medication_id** and **name** in **Medications**.
- Create a composite index for **order_status** and **order_date** in **Orders**.
- Index **payment_status** and **payment_date** in **Payments**.
- Index **rating** in **Feedback** for frequent rating queries.
- Add a composite index for **delivery_person_id** and **delivery_status** in **Deliveries**.

Phase 3: Trigger Implementation

- Create a trigger to log all **INSERT** operations on the **Orders** table.
- Create a trigger to update the pharmacy's stock quantity after a new order.
- Create a trigger to log changes to **Payments**.
- Create a trigger to mark coupons as **Used** after application.
- Create a trigger to insert **delivery updates** into **Deliveries**.
- Test triggers with sample operations.
- Set up error handling within triggers.

Phase 4: Stored Procedure Development

- Write a procedure to calculate monthly earnings for a pharmacy.
- Write a procedure to fetch customers with the most orders.
- Create a procedure to generate a monthly performance report for delivery persons.
- Test procedures with different input parameters.

Phase 5: View Creation

- Create a view for completed orders with feedback and payment details.
- Create a view of the pharmacy's performance metrics.
- Create a view showing customers and their order statistics.

Phase 6: Develop code for retrieving the below information:

- Retrieve pharmacies with the highest average ratings.
- Fetch orders with payments above the average.
- Identify the most active customers.
- Rank orders by payment amount for each pharmacy.
- Calculate the percentage of orders completed vs. canceled per pharmacy.
- Use CTEs to fetch pharmacies with more than 50 orders.
- Find top-rated pharmacies with most earnings.
- Use window functions to rank customers based on order frequency.
- Calculate the total earnings per month for each pharmacy, using **RANK()**.
- Retrieve the latest 5 orders for each customer using a subquery.
- Combine multiple subqueries to get a list of orders, feedback, and payments, excluding canceled orders.

Dashboarding and Reporting

To create a comprehensive and visually appealing dashboard for the MedDeliverDB project, each visual can be supported with meaningful visualizations.