Event Ticketing System

🎟 Event Ticketing System Project Requirements:

In an event ticketing system, customers can purchase tickets for various events hosted at different venues. Each customer has a unique identifier, name, email address, phone number, date of birth, and registration date.

Events are identified by unique identifiers, event names, descriptions, categories (e.g., concert, theater, sports, conference), start date/time, end date/time, and status (upcoming, ongoing, completed, cancelled). Each event is hosted at a specific venue and organized by event organizers.

Venues have unique identifiers, names, addresses, cities, capacities, and facility types (indoor, outdoor, hybrid). Each venue can host multiple events but has a maximum seating capacity that limits ticket sales.

Event organizers are entities that create and manage events. They have unique identifiers, organization names, contact information, and specialization areas (music, sports, corporate events, etc.).

Tickets represent individual bookings and contain details such as the purchasing customer, the event, seat information (section, row, seat number), ticket type (VIP, regular, student, senior), price, purchase date/time, and ticket status (active, used, refunded, cancelled). The system must track available seats for each event and prevent overselling.

The ticketing system supports searching for events by name, category, date range, venue, and organizer. Customers should be able to view available events in specific categories, search by partial event names, and filter by price ranges.

The system tracks sales analytics, such as total revenue per event, average ticket prices by category, most popular events, customer purchasing patterns, and venue utilization rates. It also monitors ticket sales trends over time and identifies peak booking periods.

The system should provide views that simplify common operations, such as displaying upcoming events with available tickets, showing sold-out events, customer purchase history, and revenue summaries by venue or organizer.

Strategic indexes should be implemented to optimize frequent queries such as searching events by date, filtering by venue, checking ticket availability, and generating sales reports.

To support business intelligence, the system must handle complex queries using subqueries, such as finding customers who have purchased tickets to more events than average, identifying events with below-average sales, or discovering venues that have never hosted sold-out events.

The event ticketing system incorporates advanced database features including stored procedures for ticket purchasing and refund processing, triggers for automatically updating seat availability when tickets are sold or cancelled, and robust transaction management to ensure data consistency during high-volume ticket sales periods.

🎯 Project Overview

Design and implement a comprehensive event ticketing database system that demonstrates mastery of all SQL concepts from basic database design to advanced optimization techniques in a unified, real-world application.

✓ Core Requirements

☞ Core Queries

Basic Retrieval:

1. Find all upcoming events in a specific category
2. List all sold-out events with venue details
3. Search for events by partial name match
4. Find customers who registered in the current year

Advanced Operations:

5. Calculate average ticket price per event category

6. Identify the highest-grossing event

7. Find organizers whose events have never sold out

8. Generate customer purchasing statistics and spending patterns

---

Required Components

- Create **3 useful views** that simplify complex business queries
- Implement **4 strategic indexes** with detailed performance justifications
- Write **2 subqueries** addressing real business intelligence needs

✓ Technical Requirements

Your project must demonstrate mastery of these SQL concepts:

1 DDL (Data Definition Language)

- Write **CREATE TABLE** statements for all system entities: customers, events, venues, organizers, tickets
- Include comprehensive constraints: **PRIMARY KEY**, **FOREIGN KEY**, **UNIQUE**, **CHECK**, **NOT NULL**
- Implement proper data types and relationships

2 DML (Data Manipulation Language)

- Write **INSERT** statements to populate all tables with realistic sample data
- Write an **UPDATE** statement to modify event details or customer information
- Write a **DELETE** statement to remove cancelled events or refunded tickets

3 Basic SELECT with WHERE

- Write queries retrieving specific data using various WHERE conditions
- Filter events by date, venue, price range, or availability status

4 Advanced Filtering

- Write queries using **BETWEEN** to filter date ranges or price ranges
- Write queries using **IN** to match multiple categories or venues
- Write queries using **LIKE** to perform partial text searches on event names

5 Sorting and Ordering

- Write **SELECT** queries with **ORDER BY** to sort events by date, price, or popularity
- Implement multi-column sorting for complex result sets

6 Aggregation Functions

- Write queries using **COUNT**, **AVG**, **SUM**, **MIN**, **MAX** for sales analytics
- Calculate total revenue, average prices, and ticket sales statistics

7 GROUP BY with HAVING

- Write queries grouping data by event category, venue, or time period
- Use **HAVING** to filter groups based on aggregate conditions (e.g., events with sales > $10,000)

8 Joins

- **INNER JOIN**: Combine event and venue data, customer and ticket information
- **LEFT JOIN**: Show all events including those with no ticket sales
- **RIGHT JOIN**: Display all customers including those who haven't purchased tickets

9 Set Operations

- **UNION**: Combine current and archived event data
- **INTERSECT**: Find customers who attended both concerts and sports events
- **EXCEPT/MINUS**: Identify events with available tickets vs. sold-out events

10 Subqueries

- **WHERE subquery**: Find events with above-average ticket sales
- **FROM subquery**: Create derived tables for complex analytics
- **SELECT subquery**: Calculate running totals and comparative metrics

11 Views

Create **3 strategic views** such as:

- **UpcomingEventsView**: Available events with venue and organizer details
- **CustomerPurchaseHistoryView**: Complete customer transaction history
- **VenueRevenueAnalyticsView**: Revenue and utilization statistics by venue

12 Indexes

Create **4 performance-optimized indexes** with justifications:

- **Event search optimization**: Index on event_name, category, event_date
- **Ticket lookup optimization**: Index on customer_id, event_id
- **Revenue reporting optimization**: Index on purchase_date, ticket_price
- **Venue utilization optimization**: Index on venue_id, event_date

Bonus Features (Optional)

- **Stored Procedures**: Implement procedures for ticket purchasing, refund processing, and event creation
- **Triggers**: Automatic seat availability updates, revenue calculations, and audit trail maintenance
- **Transaction Management**: Handle concurrent ticket purchases and prevent overselling
- **Advanced Analytics**: Customer segmentation, seasonal trends analysis, and predictive modeling queries
- **Security Features**: User roles, access controls, and data encryption considerations

---

Instructions for Students:

Work through each requirement carefully. Your final submission should include:

• **Database schema (DDL)** with all constraints

• **Example DML statements** with sample data

• **All core queries** as SQL scripts

• **ERD Entity Relational Diagram**

• **3 created views** with practical applications

• **4 strategic indexes** with detailed explanations

• **2 meaningful subqueries** solving business problems

• **Optional bonus features** if you choose