DB Project

By

Samra Kalwar

Database Syatem

Mam Warda Aslam

# Project Introduction: FrostByte Resort Management System

The project titled **"FrostByte – Ice Skating Resort & Competition Management System"** is a real-world inspired database solution designed for managing the operations of a winter sports resort located in **Skardu, Pakistan**.

This system aims to provide a centralized way to handle multiple resort activities including **tourist bookings**, **skating sessions**, **equipment rentals**, **training lessons with coaches**, and the **organization of skating competitions**.

FrostByte also includes detailed tracking of **participants**, **teams**, **judges**, and **scores**, making it ideal for storing and analyzing performance data from both **past and upcoming competitions**. In addition to public-facing operations, the system also supports internal resort management through modules for **staff scheduling**, **maintenance tracking**, and **inventory control**.

The goal of this project is to build a normalized, relational database in 3NF that reflects a functional, scalable, and efficient real-life winter resort management system — one that could potentially be applied in any tourism-based sports facility.

The reason Skardu was chosen as the location is because of its growing popularity in hosting national ice sports events and its natural environment that supports seasonal tourism and ice-skating competitions.

This project demonstrates not just database theory but also real-world modeling of a system that handles **users, events, logistics, and performance tracking** — all while maintaining data integrity and efficiency.

# **Objectives:**

The primary objective of this project is to design a relational database system that effectively manages the core functions of an ice skating resort. The system is centered around **FrostByte**, a winter resort located in **Skardu, Pakistan**, and is designed to streamline operations such as:

* Managing **tourist bookings** for rooms and skating sessions
* Scheduling and tracking **training sessions** with professional coaches
* Organizing and recording **skating competitions**, including participants, teams, judges, and scores
* Handling **equipment rentals**, inventory status, and return logs
* Logging **maintenance tasks** and staff responsibilities
* Storing **historical data** for past competitions, scores, and participant performances

The system is built with **data normalization (up to 3NF)**, ensuring minimum redundancy, consistency, and efficient data retrieval for queries and reports. It also enables scalability for future modules such as online bookings, event promotion, and seasonal packages.

# Conclusion:

The **FrostByte – Ice Skating Resort & Competition Management System** demonstrates how database technology can be used to solve real-world problems within the tourism and sports event sector. From booking and training to competition scoring and internal operations, this system offers a complete digital solution for managing a modern, winter sports facility.

The ERD and database design ensure that all components of the resort — tourists, events, equipment, staff, and facilities — are logically linked and efficiently managed. With data integrity, relational accuracy, and real-world relevance, the system reflects the full scope of a working resort's digital backbone.

Overall, this project not only satisfies academic database requirements, but also proposes a practical model for future implementations in adventure tourism, seasonal sports, or professional athletic management systems.

# Tools & Features Used:

The following tools, technologies, and SQL features were used to implement and test the FrostByte database system:

## **1. SQL Server Management Studio (SSMS)**

* The entire project was built and tested using **Microsoft SQL Server Management Studio (SSMS)**.
* All tables, keys, and relationships were implemented and verified using **SQL scripts**.

## **2. Table Creation:**

Tables were created for all 15+ entities including:

Tourist, Booking, SkatingSession, Competition, Participant, Team, Coach, Lesson, Equipment, Rental, Judge, Score, Staff, Maintenance, and FrostByteResort.

## 3. Basic SQL Queries Applied

 **INSERT** queries were used to populate the tables with sample data.

 **SELECT** queries were used to view data, apply filters, and test relationships.

 **DELETE** queries were used to test the safe removal of records while preserving referential integrity.

## 4. Stored Procedures:

1. **AssignLesson**

**Where Used:**

* Called when a tourist books a new lesson with a coach via the “Book Lesson” PHP form.
* Automatically inserts a new record into the Lesson table.

**Why:**

* Encapsulates logic to consistently assign lessons without writing repetitive SQL.

1. **GetLessonsByCoach**

**Where Used:**

* Executed on the “Coach Dashboard” to display all past lessons conducted by a specific coach.

**Why:**

* Allows quick and filtered retrieval of lesson history for administrative or reporting purposes.

1. **GetAvailableEquipment**

**Where Used:**

* Triggered when a tourist opens the rental form and selects equipment type.
* Lists all currently available equipment filtered by type.

**Why:**

* Prevents manual filtering and enhances user experience by showing only usable equipment

1. **RegisterTourist**

**Where Used:**

* Called during tourist registration via the HTML form.

**Why:**

* Simplifies the registration logic and ensures the RegistrationDate is always set to the current date.

1. **ViewCompetitionParticipants**

 **Where Used:**

* Invoked in the competition management panel to list all participants in a given competition.

**Why:**

* Allows the competition manager to monitor participation, team assignments, and rankings quickly.

## Triggers:

1. **trg\_UpdateAvailabilityOnRental**

**Where Used:**

* Fires **after a new record is inserted** into the Rental table (i.e., when equipment is rented).

**What It Does:**

* Automatically updates the corresponding Equipment.Availability to 'N' (Not Available).

**Why:**

Ensures that once an equipment item is rented, it's no longer available for others until returned.

1. **trg\_UpdateAvailabilityOnReturn**

**Where Used:**

* Executes **after a record in the Rental table is updated**, specifically when the Status is changed to 'Returned'.

**What It Does:**

* Sets the corresponding Equipment.Availability to 'Y' (Available) again.

**Why:**

* Helps automate the process of releasing equipment back into the pool of available inventory without manual intervention.

# **Normalization & Constraints**

 All tables were **normalized to 3NF** to avoid redundancy and dependency issues.

 **Primary keys**, **foreign keys**, and **constraints** were applied to ensure referential integrity.

# ERD





**Table: FrostByteResort**

* **1NF**: Each attribute contains atomic values (e.g., Name, Location, etc.)
* **2NF**: All fields depend solely on ResortID (primary key)
* **3NF**: No transitive dependencies exist

**Table: Tourist**

* **1NF**: No multivalued fields; all values are atomic
* **2NF**: All attributes are fully dependent on TouristID
* **3NF**: No non-key attribute depends on another non-key attribute

**Table: Rental**

* **1NF**: All values are atomic (e.g., RentalDate, Status)
* **2NF**: Composite dependencies are eliminated — everything hinges on RentalID
* **3NF**: All dependencies lead directly to RentalID

**Table: Coach**

* **1NF**: Atomic fields like Name, Rating
* **2NF**: All values depend on CoachID
* **3NF**: No transitive dependencies

**Table: Competition**

* **1NF**: No repeating groups; all attributes atomic
* **2NF**: Fully dependent on CompetitionID
* **3NF**: No indirect dependencies (e.g., Status doesn't depend on Category)

**Table:Booking**

* **1NF**: Single values in each field (Type, Status, etc.)
* **2NF**: All fields depend on BookingID
* **3NF**: No transitive dependencies





























