# MySQL Train Booking Project

**1.Project Objective:**

The objective of this project is to design and implement a Train Ticket Booking Database System using MySQL. The system allows passengers to book tickets, view train schedules, manage reservations, and generate reports such as seat availability and revenue summaries.

This project provides hands-on experience in:

* Database design (ER model)
* Writing SQL queries, views, stored procedures
* Data analysis using Excel

**2. Modules Covered**

| **Module** | **Description** |
| --- | --- |
| **Passenger Module** | Stores personal info of passengers |
| **Train Module** | Contains train numbers, names, routes, and timings |
| **Booking Module** | Manages ticket booking details including date, train, and seat info |
| **Seat Availability** | Tracks seat count for different classes |
| **Admin Reporting** | Generates reports for booked tickets, route-wise demand. |

**3. Key Tables**

* passengers(passenger\_id, name, age, gender, contact)
* trains(train\_id, train\_name, source, destination, departure\_time, arrival\_time)

**4. Sample Queries**

* Book a ticket and update seat availability
* Show available trains from A to B on a specific date
* Find the number of passengers on a specific train

**5. Tools & Technologies**

* **Database:** MySQL
* **Reporting:** Excel Dashboard (for analysis)
* **SQL Concepts:** Joins, Views, CTEs, Stored Procedures, etc

**6. Expected Outcomes**

* A well-structured relational database system
* Efficient SQL queries for data management and analysis
  + Total bookings by route/class
  + Revenue per month
  + Train utilization rates

**7. Future Enhancements**

* Add login system for admin and users
* Implement waitlisting and cancellation logic
* Integrate live train running status (via API)

SQL QUERIES

-- Create Database

CREATE DATABASE IF NOT EXISTS TrainBooking;

USE TrainBooking;

-- Create Tables

CREATE TABLE IF NOT EXISTS Stations (

station\_id INT AUTO\_INCREMENT PRIMARY KEY,

station\_name VARCHAR(100),

city VARCHAR(100)

);

CREATE TABLE IF NOT EXISTS Trains (

train\_id INT AUTO\_INCREMENT PRIMARY KEY,

train\_name VARCHAR(100),

source\_station\_id INT,

destination\_station\_id INT,

departure\_time TIME,

arrival\_time TIME,

FOREIGN KEY (source\_station\_id) REFERENCES Stations(station\_id),

FOREIGN KEY (destination\_station\_id) REFERENCES Stations(station\_id)

);

CREATE TABLE IF NOT EXISTS Passengers (

passenger\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

age INT,

gender ENUM('Male', 'Female', 'Other')

);

CREATE TABLE IF NOT EXISTS Tickets (

ticket\_id INT AUTO\_INCREMENT PRIMARY KEY,

passenger\_id INT,

train\_id INT,

travel\_date DATE,

seat\_class ENUM('Sleeper', 'AC', 'First Class'),

booking\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (passenger\_id) REFERENCES Passengers(passenger\_id),

FOREIGN KEY (train\_id) REFERENCES Trains(train\_id)

);

-- Insert Sample Data

INSERT INTO Stations (station\_id, station\_name, city) VALUES

(1, 'Chennai Egmore', 'Mumbai'),

(2, 'Coimbatore Junction', 'Bangalore'),

(3, 'Madurai Junction', 'Delhi'),

(4, 'Tiruchirappalli Junction', 'Chennai'),

(5, 'Salem Junction', 'Chennai'),

(6, 'Erode Junction', 'Chennai'),

(7, 'Tambaram', 'Trivandrum'),

(8, 'Villupuram Junction', 'Trivandrum'),

(9, 'Katpadi Junction', 'Bangalore'),

(10, 'Dindigul Junction', 'Delhi'),

(11, 'Karur Junction', 'Mumbai'),

(12, 'Nagercoil Junction', 'Delhi'),

(13, 'Thanjavur Junction', 'Trivandrum'),

(14, 'Chengalpattu Junction', 'Bangalore'),

(15, 'Arakkonam Junction', 'Chennai'),

(16, 'Kanyakumari', 'Bangalore'),

(17, 'Rameswaram', 'Chennai'),

(18, 'Royapuram Chennai', 'Bangalore'),

(19, 'Central Railway Station', 'Mumbai'),

(20, 'Karaikudi Junction', 'Mumbai');

-- Create Stored Procedures

DELIMITER $$

CREATE PROCEDURE GetBookingsByDate(IN book\_date DATE)

BEGIN

SELECT T.ticket\_id, P.name, TR.train\_name, T.seat\_class

FROM Tickets T

JOIN Passengers P ON T.passenger\_id = P.passenger\_id

JOIN Trains TR ON T.train\_id = TR.train\_id

WHERE T.travel\_date = book\_date;

END $$

CREATE PROCEDURE GetTrainSummary()

BEGIN

SELECT TR.train\_name,

COUNT(T.ticket\_id) AS total\_tickets,

AVG(P.age) AS avg\_passenger\_age

FROM Tickets T

JOIN Passengers P ON T.passenger\_id = P.passenger\_id

JOIN Trains TR ON T.train\_id = TR.train\_id

GROUP BY TR.train\_name;

END $$

DELIMITER ;

-- Total Bookings

SELECT COUNT(\*) AS total\_bookings FROM Tickets;

-- Unique Passengers

SELECT COUNT(DISTINCT passenger\_id) AS unique\_passengers FROM Tickets;

-- Bookings by Train

SELECT TR.train\_name, COUNT(\*) AS total\_bookings

FROM Tickets T

JOIN Trains TR ON T.train\_id = TR.train\_id

GROUP BY TR.train\_name;

-- Bookings by Seat Class

SELECT seat\_class, COUNT(\*) AS total FROM Tickets GROUP BY seat\_class;

-- Daily Bookings Trend

SELECT travel\_date, COUNT(\*) AS daily\_bookings

FROM Tickets

GROUP BY travel\_date

ORDER BY travel\_date;

-- Average Passenger Age by Train

SELECT TR.train\_name, AVG(P.age) AS avg\_age

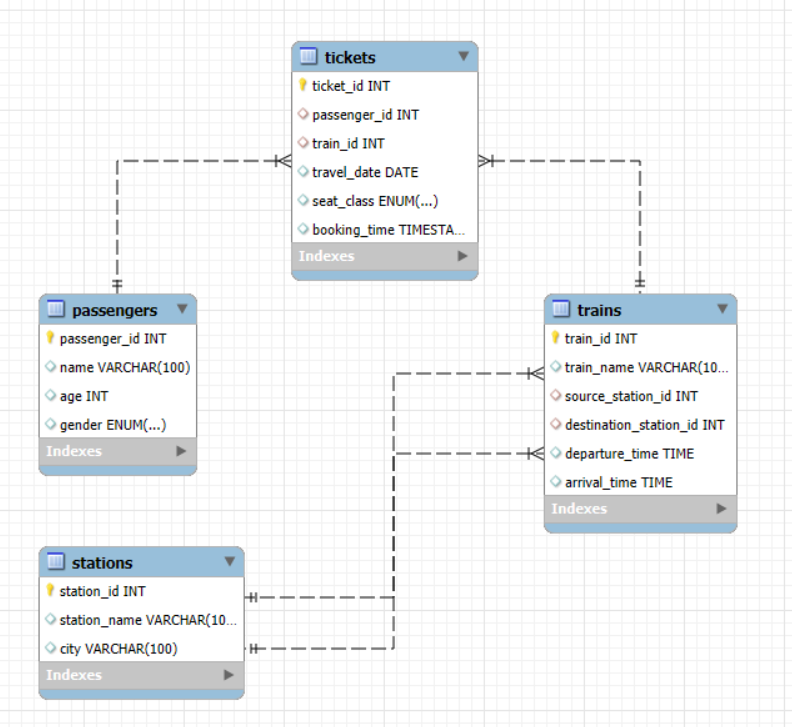
FROM Tickets T

JOIN Passengers P ON T.passenger\_id = P.passenger\_id

JOIN Trains TR ON T.train\_id = TR.train\_id

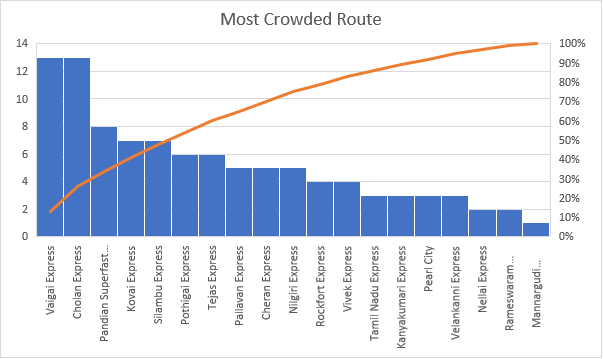
GROUP BY TR.train\_name;

**ER Diagram:**



**Analysis**

* To find the most crowded route.
* To find day Bookings Trend
* To Find passenger average age wise travel through different trains.
* To find Trains with average passenger age > 30
* Passengers with more than 1 booking.



# ----End of Project----