DATABASE MANAGEMENT SYSTEM

PROJECT REPORT

SUBMITTED BY: Barsha Gyawali

ROLL : ACE080BCT019

SECTION: A

Train Ticket Management System

1. Introduction

The Train Ticket Management System is a database application developed to manage train travel information efficiently. It stores details about passengers, trains, stations, bookings, and payments. The system allows passengers to reserve train tickets, trains to run between different stations, and payments for each booking. This project demonstrates database design concepts such as Entity-Relationship modeling, primary and foreign keys, SQL queries, joins, aggregate functions, views, and transactions.

2. Objectives

- To design a relational database with proper relationships between tables.
- To create an ER diagram.
- To perform SQL operations such as JOIN and GROUP BY.
- To implement database view and transaction control.

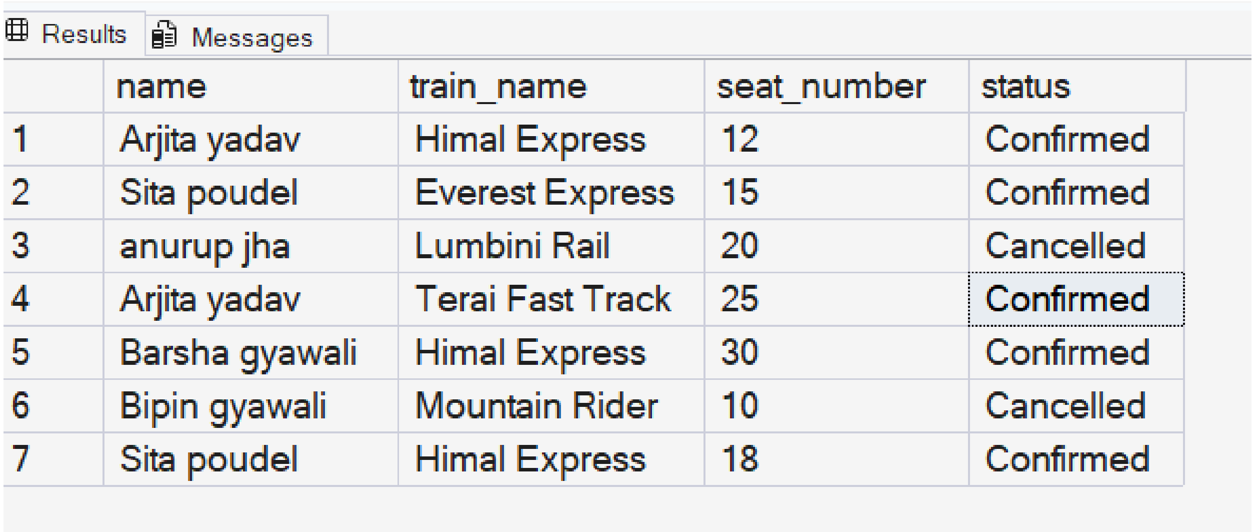
3. Domain Description

The system manages multiple passengers who book train tickets for different trains running between stations. Each booking is connected to a passenger and a train, and each payment is related to a booking. The domain contains several entities and relationships, making it suitable for implementing train ticket management system.

QUERIES:

Inner join:

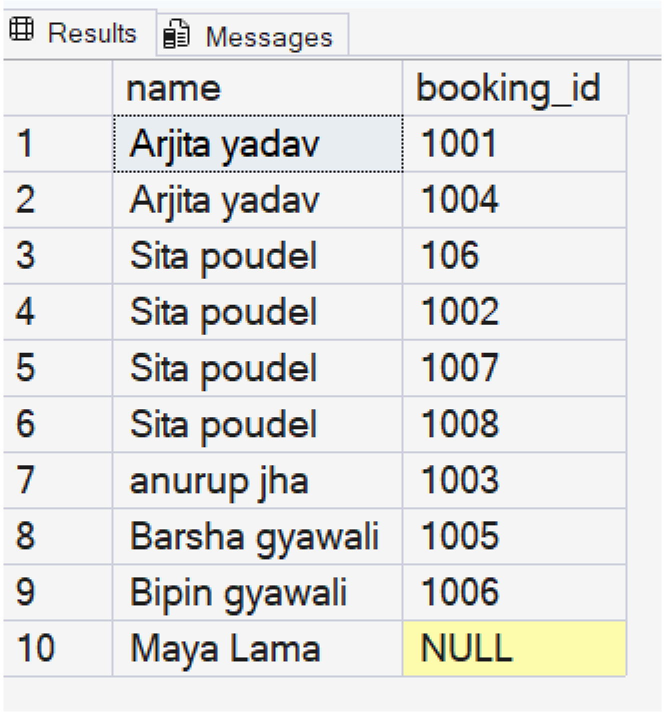
show booking detail with passengers and train:



	name	train_name	seat_number	status
1	Arjita yadav	Himal Express	12	Confirmed
2	Sita poudel	Everest Express	15	Confirmed
3	anurup jha	Lumbini Rail	20	Cancelled
4	Arjita yadav	Terai Fast Track	25	Confirmed
5	Barsha gyawali	Himal Express	30	Confirmed
6	Bipin gyawali	Mountain Rider	10	Cancelled
7	Sita poudel	Himal Express	18	Confirmed

LEFT JOIN:

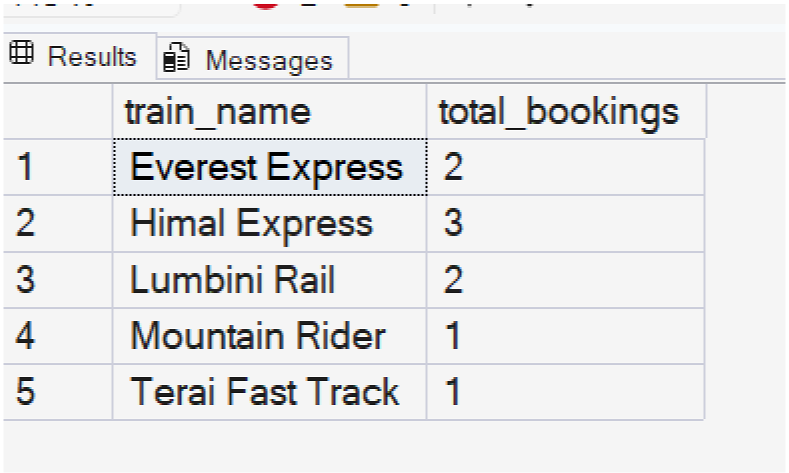
--Show all passengers and their bookings:



	name	booking_id
1	Arjita yadav	1001
2	Arjita yadav	1004
3	Sita poudel	106
4	Sita poudel	1002
5	Sita poudel	1007
6	Sita poudel	1008
7	anurup jha	1003
8	Barsha gyawali	1005
9	Bipin gyawali	1006
10	Maya Lama	NULL

AGGREGATE COUNT + GROUP BY:

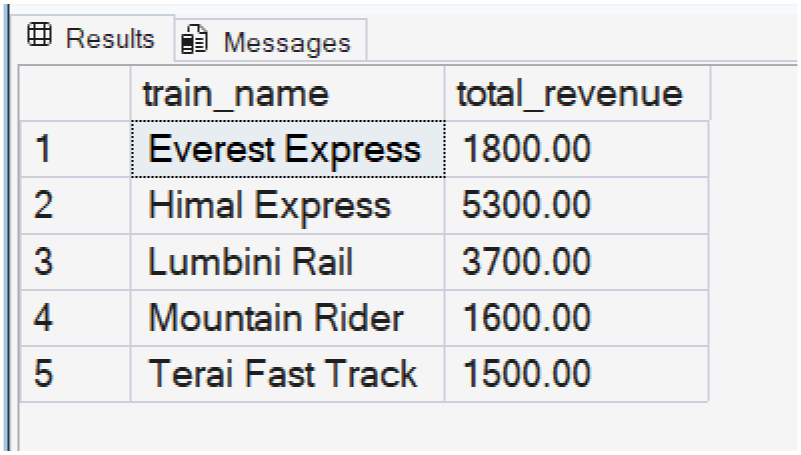
--Total bookings per train:



	train_name	total_bookings
1	Everest Express	2
2	Himal Express	3
3	Lumbini Rail	2
4	Mountain Rider	1
5	Terai Fast Track	1

SUM

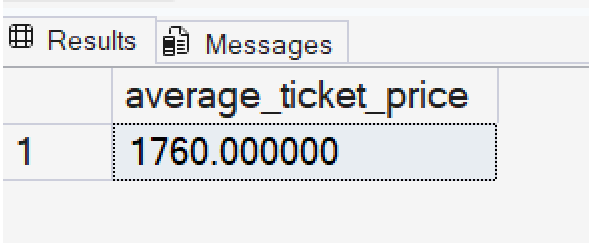
-- Total AMOUNT Per Train



	train_name	total_revenue
1	Everest Express	1800.00
2	Himal Express	5300.00
3	Lumbini Rail	3700.00
4	Mountain Rider	1600.00
5	Terai Fast Track	1500.00

AVERAGE

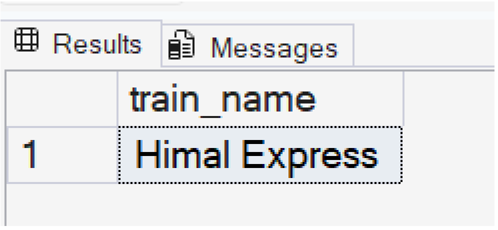
-- Average Ticket Price



	average_ticket_price
1	1760.000000

SUBQUERY

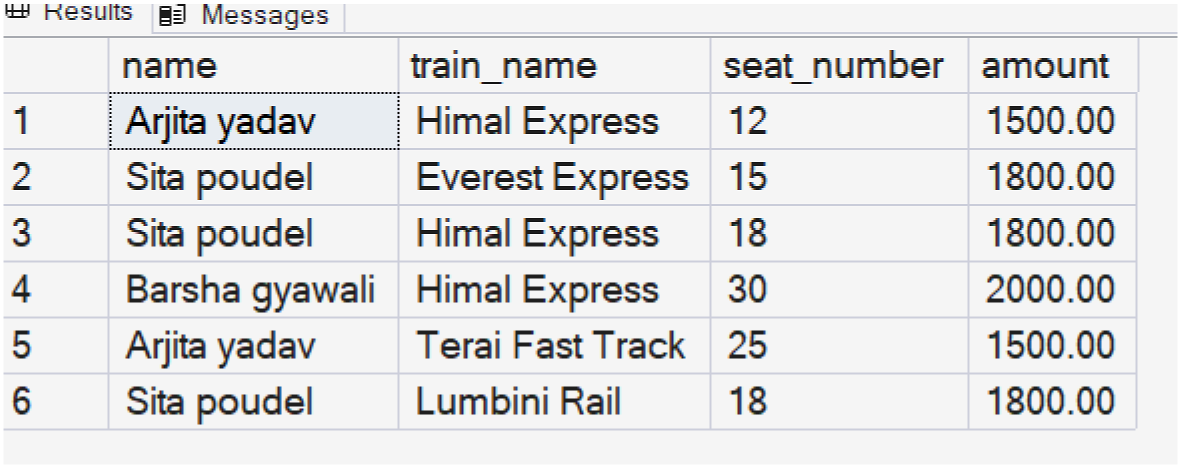
--Select the train that has the highest count.



	train_name
1	Himal Express

VIEW

--Show Confirmed booking detail



	name	train_name	seat_number	amount
1	Arjita yadav	Himal Express	12	1500.00
2	Sita poudel	Everest Express	15	1800.00
3	Sita poudel	Himal Express	18	1800.00
4	Barsha gyawali	Himal Express	30	2000.00
5	Arjita yadav	Terai Fast Track	25	1500.00
6	Sita poudel	Lumbini Rail	18	1800.00

4.Entity List

- Passengers
- Trains
- Stations

- Bookings
- Payments

5.ER Diagram Explanation

Passenger — Books — Booking (1:M relationship)

Train — Has — Booking (1:M relationship)

Station — Assigned to — Train (1:M relationship)

Booking — Has — Payment (1:1 relationship)

6. Relational Schema (With Data Types)

• **PASSENGERS**: passenger _id (PK), name VARCHAR(100), email VARCHAR(100), phone VARCHAR(15)

• **STATIONS**: station _id (PK), station _name VARCHAR(100), city VARCHAR(100)

• **TRAINS**: train _id (PK), train _name VARCHAR(100), total _seats INT, ticket _price DECIMAL(10,2), source _station _id (FK), destination_ station _id (FK)

• **BOOKINGS**: booking _id (PK), passenger _id (FK), train _id (FK), booking _date DATE , seat _number INT, status VARCHAR(20)

• **PAYMENTS**: payment _id (PK), booking _id (FK), amount DECIMAL(10,2), payment _date DATE, payment _method VARCHAR(50)

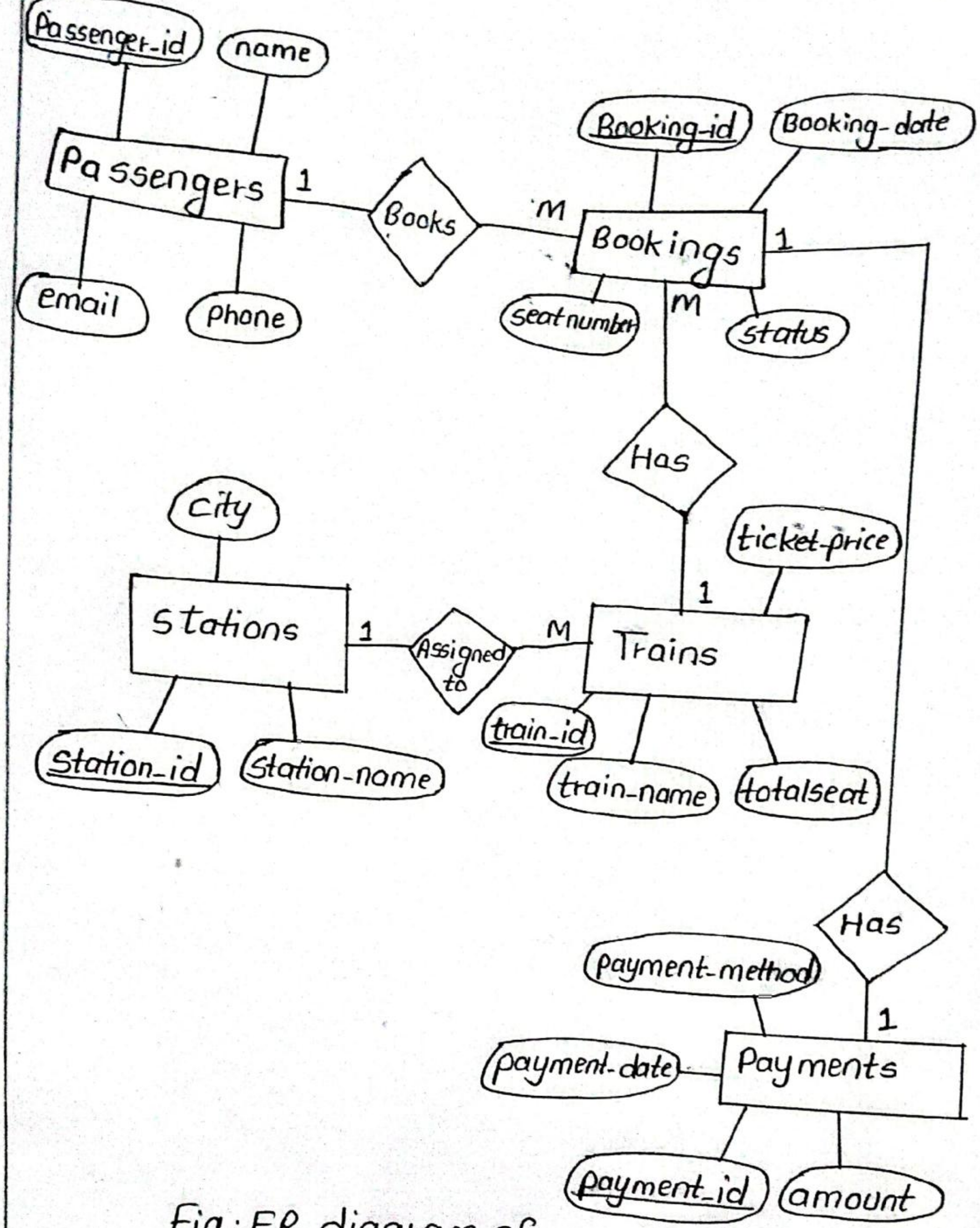


Fig: ER diagram of Train Ticket Management system.

7. SQL Implementation

- Tables created using PRIMARY KEY and FOREIGN KEY constraints (Passengers, Stations, Trains, Bookings, Payments).
- Sample data inserted successfully into all tables with proper foreign key relationships.

8. Required SQL Queries

- **INNER JOIN** – Displays passenger name, train name, and seat number booked and status.
- **LEFT JOIN** – Displays all trains including those without any bookings.
- **COUNT with GROUP BY** – Counts total bookings per train.
- **SUM with GROUP BY** – Calculates total amount generated per train.
- **AVG Function** – Calculates average ticket price of trains.
- **Subquery** – Displays the train that has the highest count.
- **VIEW** – Created a view to display confirmed bookings with passenger and train details.
- **Transaction (COMMIT / ROLLBACK)** – Show booking and payment process using transaction control.

9. Normalization

The database for the Train Ticket Management System is designed up to Third Normal Form (3NF). All tables contain atomic attributes with no repeating groups. Each non-key attribute fully depends on the primary key. This ensures the database structure is efficient, consistent, and free from redundancy**.**

10. Final Tables

Entity	**Primary Key**	**Foreign Key**	**Relationship**
Passengers	Passenger _id	–	Makes Booking (1:N)
Trains	Train _id	–	Has Booking (1:N)
Stations	Station _id	–	Source/Destination in Booking (1:N)
Bookings	Booking _id	Passenger _id, train _id, source _station _id, destination _station _id	Has Payment (1:1)
Payments	Payment _id	booking_ id	Linked to Booking

11. Conclusion

The Train Ticket Management System successfully demonstrates the design and implementation of a relational database for managing train reservations. It shows the use of Entity-Relationship modeling, primary and foreign keys, SQL queries, joins, aggregate functions, view creation, and transaction control while maintaining data consistency and integrity. The system efficiently manages passengers, trains, stations, bookings, and payments, making the database structured and easy to use.

Github link: https://github.com/barsha19-er/DBMS-PROJECT-