**Railway System Database**

**Mentored By**

|  |  |
| --- | --- |
| **Vinayak M** | **Archana H / Dr. Natesh** |
| **Assistant Professor-Senior** | **Assistant Professor-Senior** |
| **DSCA, MIT** | **DSCA, MIT** |

**ABSTRACT**

Railway systems are an important mode of transportation, used by millions of people around the world every day. The efficiency and safety of railway systems depend on various factors, such as train schedules, routes, and passenger bookings. In this context, the objective of this project is to develop a database system for a railway system that models different train stations, railway tracks between connecting stations, the train details, rail routes, and the schedule of the trains, and passenger booking information.

To achieve this objective, a database schema is designed to store information about the different train stations, train details, routes, schedules, and passenger bookings. The schema was implemented using the Structured Query Language (SQL) to perform operations such as inserting, updating, and retrieving data from the database.

The results of this project include a fully functional railway system database that can store and retrieve information about the different train stations, train details, routes, schedules, and passenger bookings. The significance of this project lies in its ability to improve the efficiency and safety of railway systems by providing a centralized database system to manage various operations.

In conclusion, this project provides a useful contribution to the field of transportation by providing a comprehensive database system for a railway system. The project demonstrates the importance of database systems in managing large-scale transportation systems and highlights the need for efficient and safe railway systems.

Contents

**1. Introduction 1**

**2. Synopsis 2**

**2.1 Proposed System**

**2.2 Objectives**

**3. Data Requirements 3**

**4. Functional Requirements 7**

**5. Detailed Design 9**

**5.1 ER Diagram**

**5.2 Schema Diagram 10**

**5.3 Data Dictionary 10**

**5.4 Relational Model Implementation 12**

**5.5 Queries**

**5.7 Triggers**

**5.8 Stored Procedures**

**5.9 Stored Functions**

**6. Conclusion and Future Work 26**

Chapter 1

Introduction

Railway systems are a crucial mode of transportation, used by millions of people around the world every day. The efficient functioning of railway systems depends on various factors, such as train schedules, routes, and passenger bookings. However, managing a large-scale railway system can be a challenging task, as it requires the coordination of various departments and operations. This is where database systems can play a critical role in managing railway systems efficiently and safely.

Currently, many railway systems use manual or semi-automated systems to manage their operations, which can be time-consuming and error-prone. For example, managing train schedules and routes manually can lead to delays and errors, resulting in unhappy passengers and financial losses for the railway company. Similarly, managing passenger bookings manually can lead to overbooking, under booking, and incorrect bookings, which can again result in delays and financial losses.

The objective of this project is to develop a centralized database system for a railway system that can store and manage various aspects of railway stations such as train stations, railway tracks, train details, rail routes, schedules, and passenger bookings. The database system will be implemented using a Relational Database Management System (RDBMS) and the Structured Query Language (SQL) will be used to perform operations such as inserting, updating, and retrieving data from the database. By developing a database system for a railway system, this project aims to provide a comprehensive solution for managing railway systems efficiently and safely.

Chapter 2

Synopsis

**2.1 Proposed System**

The proposed system for the railway system database aims to address the problem of the lack of an efficient and centralized database system for managing railway systems. The current manual or semi-automated systems used by many railway systems can be time-consuming and error-prone, leading to delays and financial losses for the railway company. The proposed system will provide real-time information about train schedules and routes, making it easier for the railway company to manage their operations efficiently. Similarly, the system will store and retrieve passenger booking information, making it easier to manage passenger bookings and avoid errors.

**Tables:**

* Stations
* Trains
* Routes
* Schedule
* Passengers

**2.2 Objectives**

**The main objectives of the work are:**

* To create a database schema for managing train stations, railway tracks, train details, rail routes, schedules, and passenger bookings.
* To implement the database using MySQL.
* To provide accurate information about train schedules and routes to improve the efficiency of railway operations.
* To store and retrieve passenger booking information to avoid errors and efficiently manage passenger bookings.
* To provide a user-friendly interface for querying and updating the database.
* To test the database system for its functionality and performance and refine it as necessary.

**Chapter 3**

**Data Requirements**

The Data Requirement Document (DRD) is a central document of the project, in which all information relating to data is gathered for agreement by the key stakeholders and then for guidance and information for those involved in the project. The data requirements are listed in the following sections and describe the essential data requirements for the Railway System Database.

**3.1** **Purpose, Scope, and Overview**

**Purpose:**

• The purpose of the railway system database is to provide an efficient and centralized system for managing railway systems.

• The database will provide real-time information about train schedules and routes, making it easier for the railway company to manage their operations efficiently.

• The system will store and retrieve passenger booking information, making it easier to manage passenger bookings and avoid errors.

**Scope:**

• The scope of the project is to design and develop a database system that will be used by the railway company to manage their operations.

• The database will include tables for Stations, Trains, Routes, Schedule, and Passengers.

• The system will provide real-time information about train schedules and routes, and will store and retrieve passenger booking information.

**Overview:**

The abstract describes the need for a centralized database system for managing railway systems efficiently and safely. The proposed system aims to address the problem of the lack of an efficient and centralized database system currently used by many railway systems. The system will store and manage various aspects of railway stations such as train stations, railway tracks, train details, rail routes, schedules, and passenger bookings. By developing a database system for a railway system, this project aims to provide a comprehensive solution for managing railway systems efficiently and safely. The proposed system will provide real-time information about train schedules and routes, making it easier for the railway company to manage their operations efficiently. Similarly, the system will store and retrieve passenger booking information, making it easier to manage passenger bookings and avoid errors. The database will consist of tables for Stations, Trains, Routes, Schedule, and Passengers

**3.2 Requirements**

**3.2.1 Route**

|  |  |  |
| --- | --- | --- |
| **Data Name** | **Description** | **Example** |
| route\_id | An unique identifier for each route. | 10001 |
| origin\_station\_id | The unique identifier of the origin station of the route. References the station\_id field in the Station table. | 10010 |
| destination\_station\_id | The unique identifier of the destination station of the route. References the station\_id field in the Station table. | 10020 |
| Distance | The distance in kilometers between the origin and destination stations. Must be a non-negative real number. | 120.5 |

**3.2.2 Passenger**

|  |  |  |
| --- | --- | --- |
| **Data Name** | **Description** | **Example** |
| passenger\_id | An unique identifier for each passenger. | 10104 |
| name | The name of the passenger. | Abhinav Kumar |
| contact\_info | The contact information of the passenger, including the country code, area code, and phone number. Must be in the format **+CC-NNNNNNNNNN**, where **CC** is the country code and **NNNNNNNNNN** is the phone number. | +91-9471191100 |

**3.2.3 Trip**

|  |  |  |
| --- | --- | --- |
| **Data Name** | **Description** | **Example** |
| trip\_id | An unique number to identify each trip | 10001 |
| passenger\_id | An unique number to identify each passenger | 20001 |
| schedule\_id | An unique number to identify each schedule | 30001 |

**3.2.4 Schedule**

|  |  |  |
| --- | --- | --- |
| **Data Name** | **Description** | **Example** |
| *schedule\_id* | An unique numeric value to identify each schedule. | 24674 |
| *train\_id* | Numeric identifier of the train for the schedule. | 10214 |
| *route\_id* | Numeric identifier of the route for the schedule. | 10002 |
| *departure\_time* | The time when the train departs from the origin station. | 2023-06-01 10:00:00 |
| *arrival\_time* | The time when the train arrives at the destination station. | 2023-06-01 11:30:00 |

**3.2.5 Train**

|  |  |  |
| --- | --- | --- |
| **Data Name** | **Description** | **Example** |
| train\_id | A unique numeric identifier for each train. | 10103 |
| name | The name of the train. | Shatabdi Express |
| type | The type of train, which can be Local, Express, or Freight. | Express |
| capacity | The maximum number of passengers that the train can carry. | 200 |

**3.2.6 Station**

|  |  |  |
| --- | --- | --- |
| **Data Name** | **Description** | **Example** |
| station\_id | Numeric identifier of the station. | 10010 |
| name | Name of the station. | New Delhi Railway |
| location | Location of the station: North, South, East or West. | North |

**Chapter 4**

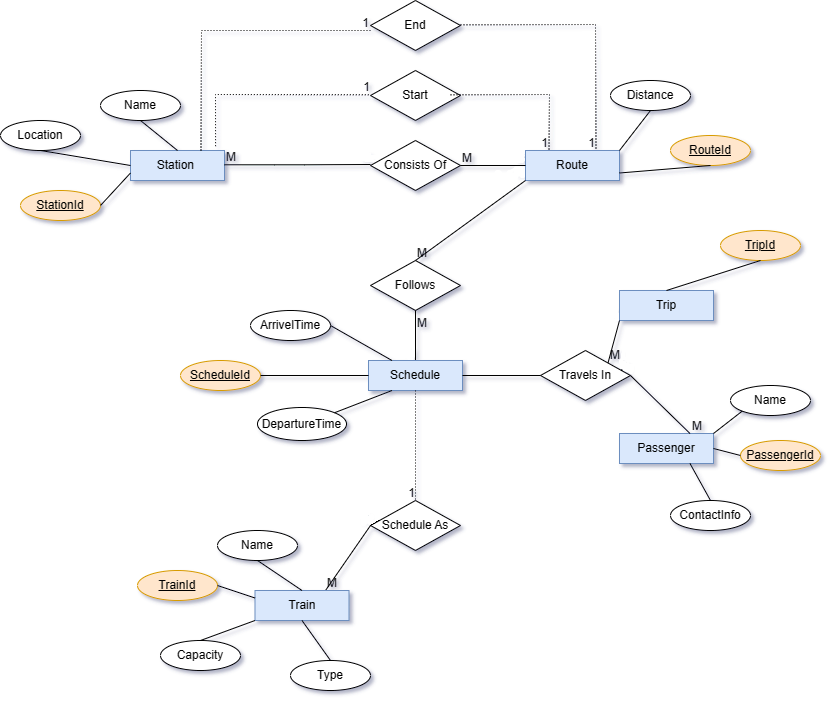
**Functional Requirements**

| **Requirement** | **Input** | **Processing** | **Output** |
| --- | --- | --- | --- |
| **Add a new station** | Name, Location | Check if the station with the same name and location doesn't already exist, if not, add a new station to the Station table with the given name and location | If station is successfully added, return success message, else return error message |
| **Update station information** | Station ID, Name, Location | Check if the station with the given ID exists, if yes, update its name and location with the given values | If station information is successfully updated, return success message, else return error message |
| **Delete a station** | Station ID | Check if the station with the given ID exists, if yes, delete the station from the Station table | If station is successfully deleted, return success message, else return error message |
| **Add a new train** | Name, Type, Capacity | Check if a train with the same name doesn't already exist, if not, add a new train to the Train table with the given name, type, and capacity | If train is successfully added, return success message, else return error message |
| **Update train information** | Train ID, Name, Type, Capacity | Check if the train with the given ID exists, if yes, update its name, type, and capacity with the given values | If train information is successfully updated, return success message,else return error message |
| **Delete a train** | Train ID | Check if the train with the given ID exists, if yes, delete the train from the Train table | If train is successfully deleted, return success message, else return error message |
| **Add a new route** | Origin station ID, Destination station ID, Distance | Check if the origin and destination stations exist, if yes, add a new route to the Route table with the given origin station ID, destination station ID, and distance | If route is successfully added, return success message, else return error message |
| **Update route information** | Route ID, Origin station ID, Destination station ID, Distance | Check if the route with the given ID exists, if yes, update its origin station ID, destination station ID, and distance with the given values | If route information is successfully updated, return success message, else return error message |
| **Delete a route** | Route ID | Check if the route with the given ID exists, if yes, delete the route from the Route table | If route is successfully deleted, return success message, else return error message |
| **Book a new trip** | Passenger name, Contact info, Train ID, Route ID, Departure time, Arrival time | Check if the passenger with the given name and contact info exists, if not, add a new passenger to the Passenger table. Check if the train and route with the given IDs exist, and if the departure time is after the current time. If all checks pass, add a new trip to the Trip table with the given passenger ID, train ID, route ID, departure time, and arrival time | If trip is successfully booked, return success message with trip details, else return error message |

**Chapter 5**

**Detailed Design**

**5.1 ER Diagram -**

****

**5.2 Schema Diagram -**

**Station (StationId**, Name, Location)

**Train (TrainId**, Name, Type, Capacity)

**Route (RouteId**, OriginStationId, DestinationStationId, Distance)

*OriginStationId, DestinationStationId References Station*

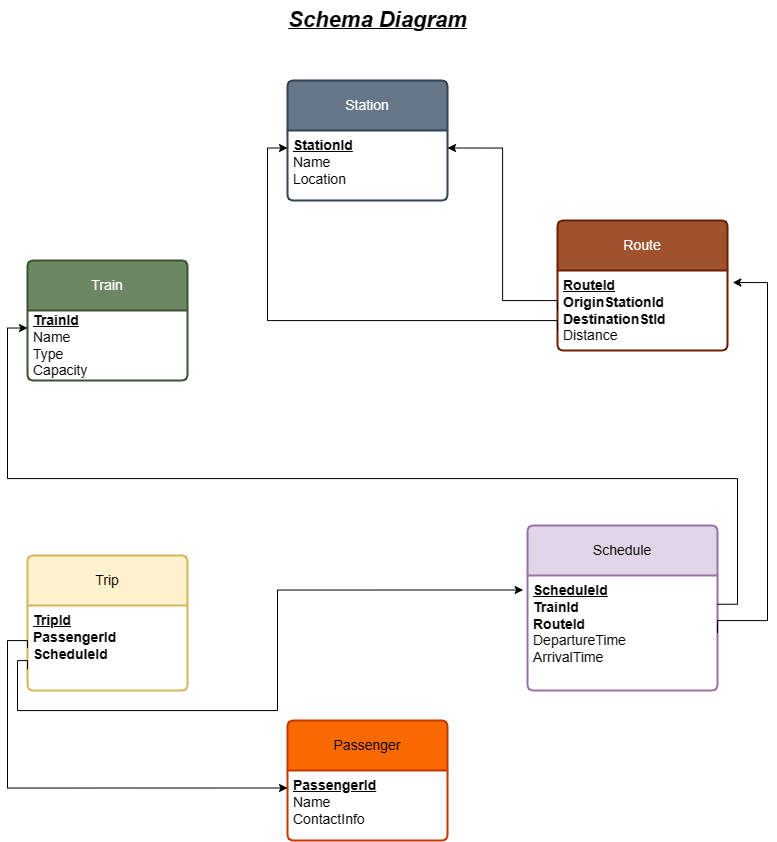
**Trip (TripId*,*** PassengerId, ScheduleId)

*PassengerId References Passenger and ScheduleId References Schedule*

**Schedule (ScheduleId**, TrainId, RouteId , DepartureTime , ArrivalTime )

*TrainId References Train and RouteId References Route*

**Passenger (**PassengerId, Name, ContactInfo)

****

**5.3 Data Dictionary**

**STATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraint Type** | **Constraint Name** |
| station\_id | NUMBER(10) | PRIMARY KEY | pk\_station |
| name | VARCHAR2(255) | NOT NULL |  |
| location | VARCHAR2(255) | NOT NULL |  |
|  |  | UNIQUE | uq\_station\_name\_location |
|  |  | CHECK | ck\_station\_location |

**TRAIN**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraint Type** | **Constraint Name** |
| train\_id | NUMBER(10) | PRIMARY KEY | pk\_train |
| name | VARCHAR2(255) | NOT NULL |  |
| type | VARCHAR2(255) | NOT NULL |  |
| capacity | NUMBER(10) | NOT NULL |  |
|  |  | CHECK | ck\_train\_type |

**ROUTE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraint Type** | **Constraint Name** |
| route\_id | NUMBER(10) | PRIMARY KEY | pk\_route |
| origin\_station\_id | NUMBER(10) | NOT NULL |  |
| destination\_station\_id | NUMBER(10) | NOT NULL |  |
| distance | REAL | NOT NULL |  |
|  |  | CHECK | ck\_route\_distance |
|  |  | FOREIGN KEY | origin\_station\_id |
|  |  |  | destination\_station\_id |

**SCHEDULE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraint Type** | **Constraint Name** |
| schedule\_id | NUMBER(10) | PRIMARY KEY | pk\_schedule |
| train\_id | NUMBER(10) | NOT NULL |  |
| route\_id | NUMBER(10) | NOT NULL |  |
| departure\_time | TIMESTAMP | NOT NULL |  |
| arrival\_time | TIMESTAMP | NOT NULL |  |
|  |  | CHECK | ck\_schedule\_arrival |
|  |  | FOREIGN KEY | train\_id |
|  |  |  | route\_id |

**PASSENGER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraint Type** | **Constraint Name** |
| passenger\_id | NUMBER(10) | PRIMARY KEY | pk\_passenger |
| name | VARCHAR2(255) | NOT NULL |  |
| contact\_info | VARCHAR2(255) | NOT NULL |  |
|  |  | CHECK | ck\_passenger\_contact |

**TRIP**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraint Type** | **Constraint Name** |
| trip\_id | NUMBER(10) | PRIMARY KEY | pk\_trip |
| passenger\_id | NUMBER(10) | NOT NULL |  |
| schedule\_id | NUMBER(10) | NOT NULL |  |
|  |  | FOREIGN KEY | passenger\_id |
|  |  |  | schedule\_id |

**5.4 Relational Model Implementation**

* **Create Table Commands -**

**CREATE TABLE Station** (

station\_id NUMBER(10) CONSTRAINT pk\_station PRIMARY KEY,

name VARCHAR2(255) NOT NULL,

location VARCHAR2(255) NOT NULL,

CONSTRAINT uq\_station\_name\_location UNIQUE (name, location),

CONSTRAINT ck\_station\_location CHECK (location IN ('North', 'South', 'East', 'West'))

);

**CREATE TABLE Train** (

train\_id NUMBER(10) CONSTRAINT pk\_train PRIMARY KEY,

name VARCHAR2(255) NOT NULL,

type VARCHAR2(255) NOT NULL,

capacity NUMBER(10) NOT NULL,

CONSTRAINT ck\_train\_type CHECK (type IN ('Local', 'Express', 'Freight'))

);

**CREATE TABLE Route** (

route\_id NUMBER(10) CONSTRAINT pk\_route PRIMARY KEY,

origin\_station\_id NUMBER(10) NOT NULL,

destination\_station\_id NUMBER(10) NOT NULL,

distance REAL NOT NULL,

CONSTRAINT ck\_route\_distance CHECK (distance >= 0),

FOREIGN KEY (origin\_station\_id) REFERENCES Station(station\_id),

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

);

**CREATE TABLE Schedule** (

schedule\_id NUMBER(10) CONSTRAINT pk\_schedule PRIMARY KEY,

train\_id NUMBER(10) NOT NULL,

route\_id NUMBER(10) NOT NULL,

departure\_time TIMESTAMP NOT NULL,

arrival\_time TIMESTAMP NOT NULL,

CONSTRAINT ck\_schedule\_arrival CHECK (arrival\_time > departure\_time),

FOREIGN KEY (train\_id) REFERENCES Train(train\_id),

FOREIGN KEY (route\_id) REFERENCES Route(route\_id)

);

**CREATE TABLE Passenger** (

passenger\_id NUMBER(10) CONSTRAINT pk\_passenger PRIMARY KEY,

name VARCHAR2(255) NOT NULL,

contact\_info VARCHAR2(255) NOT NULL,CONSTRAINT ck\_passenger\_contact CHECK (contact\_info LIKE '+%-%-%' AND LENGTH(contact\_info) = 12));

**CREATE TABLE Trip** (

trip\_id NUMBER(10) CONSTRAINT pk\_trip PRIMARY KEY,

passenger\_id NUMBER(10) NOT NULL,

schedule\_id NUMBER(10) NOT NULL,

CONSTRAINT fk\_trip\_passenger FOREIGN KEY (passenger\_id) REFERENCES Passenger(passenger\_id),

CONSTRAINT fk\_trip\_schedule FOREIGN KEY (schedule\_id) REFERENCES Schedule(schedule\_id),

);

* **Insert Table Commands –**

**Station -**

INSERT INTO Station (station\_id, name, location) VALUES (10010, 'New Delhi Railway', 'North');

INSERT INTO Station (station\_id, name, location) VALUES (10020, 'Mumbai Central', 'West');

INSERT INTO Station (station\_id, name, location) VALUES (10030, 'Howrah Junction', 'East');

INSERT INTO Station (station\_id, name, location) VALUES (10040, 'Chennai Central', 'South');

INSERT INTO Station (station\_id, name, location) VALUES (10050, 'Bangalore City', 'South');

INSERT INTO Station (station\_id, name, location) VALUES (10060, 'Ahmedabad Junction', 'West');

INSERT INTO Station (station\_id, name, location) VALUES (10070, 'Guwahati', 'East');

INSERT INTO Station (station\_id, name, location) VALUES (10080, 'Jaipur Junction', 'North');

INSERT INTO Station (station\_id, name, location) VALUES (10090, 'Hyderabad Deccan', 'South');

INSERT INTO Station (station\_id, name, location) VALUES (10100, 'Patna Junction', 'East');

**Train -**

INSERT INTO Train (train\_id, name, type, capacity) VALUES (10103, 'Rajdhani Express', 'Express', 300);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (10214, 'Konkan Kanya Express', 'Express', 400);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (10322, 'Mumbai Local', 'Local', 200);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (10545, 'Garib Rath', 'Express', 500);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (10984, 'Shatabdi Express', 'Express', 350);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (11099, 'Deccan Odyssey', 'Express', 450);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (11670, 'Kanyakumari Express', 'Express', 400);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (11903, 'Himalayan Queen', 'Express', 300);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (12067, 'Jan Shatabdi Express', 'Express', 350);

INSERT INTO Train (train\_id, name, type, capacity) VALUES (12879, 'Mumbai Mail', 'Express', 500);

**Route -**

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10001, 10010, 10020, 120.5);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10002, 10020, 10030, 250.0);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10003, 10030, 10040, 350.8);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10004, 10040, 10050, 200.3);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10005, 10050, 10060, 150.0);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10006, 10060, 10070, 280.4);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10007, 10070, 10080, 420.0);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10008, 10080, 10090, 310.2);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10009, 10090, 10100, 180.0);

INSERT INTO Route (route\_id, origin\_station\_id, destination\_station\_id, distance)

VALUES (10010, 10100, 10010, 400.1);

**Schedule -**

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (24674, 10214, 10002, '2023-05-10 09:00:00', '2023-05-10 11:30:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (48235, 11903, 10010, '2023-05-11 13:00:00', '2023-05-12 03:45:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (88253, 10545, 10008, '2023-05-09 17:30:00', '2023-05-10 06:15:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (12809, 10103, 10001, '2023-05-12 06:45:00', '2023-05-12 11:00:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (73526, 12067, 10009, '2023-05-08 22:00:00', '2023-05-09 14:45:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (98150, 10322, 10003, '2023-05-11 18:30:00', '2023-05-12 04:15:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (39215, 11099, 10006, '2023-05-10 21:15:00', '2023-05-11 12:00:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (68123, 10984, 10005, '2023-05-08 19:00:00', '2023-05-09 06:45:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (57319, 10545, 10002, '2023-05-09 07:30:00', '2023-05-09 10:00:00');

INSERT INTO Schedule (schedule\_id, train\_id, route\_id, departure\_time, arrival\_time)

VALUES (87690, 12879, 10010, '2023-05-11 12:30:00', '2023-05-11 21:15:00');

**Passenger -**

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (10105, 'Abhinav Kumar', '+91-9834567823');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (10225, 'Nitya Singh', '+91-9652345901');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (10456, 'Mohit Verma', '+91-9810789546');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (10892, 'Priya Sharma', '+91-9543210789');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (11003, 'Varun Gupta', '+91-9876543201');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (11427, 'Aryan Singh', '+91-9432106789');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (11863, 'Priyanshi Gupta', '+91-9923456781');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (12005, 'Karan Chaudhary', '+91-9111111111');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (12506, 'Alisha Singh', '+91-9012345678');

INSERT INTO Passenger (passenger\_id, name, contact\_info) VALUES (12930, 'Rishabh Kumar', '+91-9765432109');

**Trip -**

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (84627, 10892, 48235);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (73491, 11863, 73526);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (91248, 10225, 12809);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (58379, 12005, 68123);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (24768, 10105, 24674);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (49025, 10456, 88253);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (62381, 12506, 57319);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (92106, 11003, 39215);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (38954, 12930, 87690);

INSERT INTO Trip (trip\_id, passenger\_id, schedule\_id) VALUES (85436, 11427, 98150);

**5.5 Queries**

* **Display the name and capacity of all Express trains:**

SELECT name, capacity FROM Train WHERE type = 'Express';

* **Display the name and location of all stations in the North region:**

SELECT name, location FROM Station WHERE location = 'North';

* **Display the name, distance, and origin station of all routes with a distance greater than 500 km:**

SELECT Route.name, distance, Station.name AS origin\_station

FROM Route

JOIN Station ON Station.station\_id = Route.origin\_station\_id

WHERE distance > 500;

* **Display the name and type of all trains that can carry more than 500 passengers:**

SELECT name, type FROM Train WHERE capacity > 500;

* **Display the name and location of all stations that have a unique name-location combination:**

SELECT name, location FROM Station WHERE (name, location) IN

(SELECT name, location FROM Station GROUP BY name, location HAVING COUNT(\*) = 1);

* **Display the name and contact information of all passengers who have booked a trip:**

SELECT name, contact\_info FROM Passenger WHERE passenger\_id IN (SELECT passenger\_id FROM Trip);

* **Display the total distance of all routes that originate from each station, ordered by station name:**

SELECT Station.name, SUM(Route.distance) AS total\_distance

FROM Station

JOIN Route ON Route.origin\_station\_id = Station.station\_id

GROUP BY Station.name

ORDER BY Station.name;

* **Display the name, departure time, and arrival time of all schedules for a given train, ordered by departure time:**

SELECT Schedule.schedule\_id, Train.name, Schedule.departure\_time, Schedule.arrival\_time

FROM Schedule

JOIN Train ON Train.train\_id = Schedule.train\_id

WHERE Train.name = 'My Train'

ORDER BY Schedule.departure\_time;

* **Display the name and total distance of all routes that originate from each station in the North region, ordered by total distance:**

SELECT Station.name, SUM(Route.distance) AS total\_distance

FROM Station

JOIN Route ON Route.origin\_station\_id = Station.station\_id

WHERE Station.location = 'North'

GROUP BY Station.name

ORDER BY total\_distance DESC;

* **Display the name and number of trips booked by each passenger, ordered by number of trips:**

SELECT Passenger.name, COUNT(\*) AS num\_trips

FROM Passenger

JOIN Trip ON Trip.passenger\_id = Passenger.passenger\_id

GROUP BY Passenger.name

ORDER BY num\_trips DESC;

**5.6 Triggers**

* **Trigger to enforce capacity constraint for Train table:**

CREATE OR REPLACE TRIGGER train\_capacity\_trigger

BEFORE INSERT OR UPDATE ON Train

FOR EACH ROW

BEGIN

IF :new.capacity < 1 THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Train capacity must be greater than or equal to 1');

END IF;

END;

* **Trigger to enforce contact\_info format for Passenger table:**

CREATE OR REPLACE TRIGGER passenger\_contact\_trigger

BEFORE INSERT OR UPDATE ON Passenger

FOR EACH ROW

BEGIN

IF NOT REGEXP\_LIKE(:new.contact\_info, '^\+\d{1,3}\-\d{3}\-\d{4}$') THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Contact info format must be "+XXX-XXX-XXXX"');

END IF;

END;

* **Trigger to update distance in Route table based on origin and destination station location:**

CREATE OR REPLACE TRIGGER route\_distance\_trigger

AFTER INSERT OR UPDATE ON Route

FOR EACH ROW

DECLARE

origin\_loc VARCHAR2(255);

dest\_loc VARCHAR2(255);

BEGIN

SELECT location INTO origin\_loc FROM Station WHERE station\_id = :new.origin\_station\_id;

SELECT location INTO dest\_loc FROM Station WHERE station\_id = :new.destination\_station\_id;

IF origin\_loc = 'North' AND dest\_loc = 'South' THEN

:new.distance := 1000;

ELSIF origin\_loc = 'South' AND dest\_loc = 'North' THEN

:new.distance := 1000;

ELSIF origin\_loc = 'East' AND dest\_loc = 'West' THEN

:new.distance := 1500;

ELSIF origin\_loc = 'West' AND dest\_loc = 'East' THEN

:new.distance := 1500;

ELSE

RAISE\_APPLICATION\_ERROR(-20000, 'Invalid origin and destination station locations');

END IF;

END;

**5.7 Stored Procedures**

* **Procedure to add new station -**

CREATE OR REPLACE PROCEDURE add\_station(

p\_station\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_location IN VARCHAR2

)

AS

BEGIN

INSERT INTO Station(station\_id, name, location)

VALUES(p\_station\_id, p\_name, p\_location);

COMMIT;END;

* **Procedure to update the origin station of a route:**

CREATE OR REPLACE PROCEDURE update\_route\_origin(

p\_route\_id IN NUMBER,

p\_new\_origin\_id IN NUMBER

)

AS

BEGIN

UPDATE Route

SET origin\_station\_id = p\_new\_origin\_id

WHERE route\_id = p\_route\_id;

COMMIT;

END;

* **Procedure to add a new trip for a passenger:**

CREATE OR REPLACE PROCEDURE add\_trip(

p\_passenger\_id IN NUMBER,

p\_schedule\_id IN NUMBER

)

AS

BEGIN

INSERT INTO Trip(trip\_id, passenger\_id, schedule\_id)

VALUES(TRIP\_SEQ.NEXTVAL, p\_passenger\_id, p\_schedule\_id);

COMMIT;

END;

**5.8 Stored Functions**

* **Function to retrieve the total capacity of a train:**

CREATE OR REPLACE FUNCTION get\_train\_capacity(p\_train\_id IN NUMBER)

RETURN NUMBER

AS

v\_capacity NUMBER;

BEGIN

SELECT capacity INTO v\_capacity

FROM Train

WHERE train\_id = p\_train\_id;

RETURN v\_capacity;

END;

* **Function to get the distance between two stations in a route:**

CREATE OR REPLACE FUNCTION get\_station\_distance(

p\_route\_id IN NUMBER,

p\_origin\_id IN NUMBER,

p\_destination\_id IN NUMBER

)

RETURN REAL

AS

v\_distance REAL;

BEGIN

SELECT distance INTO v\_distance

FROM Route

WHERE route\_id = p\_route\_id

AND origin\_station\_id = p\_origin\_id

AND destination\_station\_id = p\_destination\_id;

RETURN v\_distance;

END;

**Chapter 6**

**6.1 Conclusion**

In summary, the proposed railway system database project includes the creation of several tables that will store information about stations, trains, routes, schedules, passengers, and trips. These tables will enable the railway company to manage their operations more efficiently by providing real-time information on train schedules and passenger bookings. The database will help minimize errors and avoid financial losses that may arise from manual or semi-automated systems.

**6.2 Scope for future work**

In the future, the railway company could consider adding additional features to the database to enhance its functionality. For example, they could integrate a payment system for passengers to book and pay for their trips online. The company could also consider adding a feature that allows customers to track their luggage during the journey. Additionally, the company could use data analytics tools to analyze passenger data to identify patterns and trends and make data-driven decisions. Overall, the proposed railway system database project lays the foundation for a more efficient and modernized railway system that benefits both the company and its customers.

**Thank You**