ASSIGNMENT - 3

DATABASE MANAGEMENT SYSTEM

SIDDARTH.P

192324264

Public Transportation Management System

Description

The **Public Transportation Management System (PTMS)** database is designed to manage operations related to bus or train schedules, ticket bookings, passenger data, routes, vehicles, and payments. It ensures efficient management of schedules, seat availability, and ticketing, while enabling accurate reporting and analysis. Automation and optimization are incorporated through stored procedures and triggers.

Database Tables Design

1. Passengers

Stores passenger information for identification and contact purposes.

Column	Data Type	Description
passenger_id	INT (PK)	Unique identifier for each passenger
first_name	VARCHAR(50)	Passenger's first name
last_name	VARCHAR(50)	Passenger's last name
phone_number	VARCHAR(15)	Contact number
email	VARCHAR(100)	Email address

2. Routes

Tracks route details for vehicles.

Column	Data Type	Description
route_id	INT (PK)	Unique identifier for each route
departure_station	VARCHAR(50)	Station of departure
arrival_station	VARCHAR(50)	Station of arrival
distance	DECIMAL(5,2)	Distance of the route (in km)

3. Schedules

Stores details about schedules for routes and vehicles.

Column	Data Type	Description
schedule_id	INT (PK)	Unique identifier for each schedule
route_id	INT (FK)	References Routes(route_id)
vehicle_id	INT (FK)	References Vehicles(vehicle_id)
departure_time	DATETIME	Scheduled departure time
arrival_time	DATETIME	Scheduled arrival time

4. Bookings

Tracks ticket bookings, linking passengers to schedules.

Column	Data Type	Description
booking_id	INT (PK)	Unique identifier for each booking
passenger_id	INT (FK)	References Passengers(passenger_id)
schedule_id	INT (FK)	References Schedules(schedule_id)
seat_number	VARCHAR(10)	Assigned seat number
booking_date	DATE	Date of booking
class	VARCHAR(20)	Travel class (e.g., Economy)

5. Vehicles

Stores details of the vehicles assigned to routes.

Column	Data Type	Description
vehicle_id	INT (PK)	Unique identifier for each vehicle
vehicle_type	VARCHAR(20)	Type of vehicle (e.g., Bus, Train)
total_seats	INT	Total number of seats available

6. Payments

Manages payment records for bookings.

Column	Data Type	Description
payment_id	INT (PK)	Unique identifier for each payment
booking_id	INT (FK)	References Bookings(booking_id)
payment_date	DATE	Date of payment
amount	DECIMAL(10,2)	Total amount paid
payment_status	VARCHAR(20)	Payment status (e.g., Paid)

Constraints for Referential Integrity

- **Foreign Keys:**
  - passenger_id in **Bookings** references **Passengers(passenger_id)**.
  - route_id in **Schedules** references **Routes(route_id)**.
  - schedule_id in **Bookings** references **Schedules(schedule_id)**.
  - booking_id in **Payments** references **Bookings(booking_id)**.
- **Primary Keys:**
  - Each table includes a primary key for unique record identification.
- **Check Constraints:**
  - Ensure non-negative values for distance, total_seats, and amount.

Stored Procedures

a. Check Seat Availability

Confirms if seats are available for a specific schedule.

```
CREATE PROCEDURE CheckSeatAvailability(IN scheduleId INT)
BEGIN
  SELECT total_seats - COUNT(booking_id) AS available_seats
  FROM Bookings
  WHERE schedule_id = scheduleId;
END;
```

b. Book Ticket

Registers a booking, assigns a seat, and updates seat availability.

```
CREATE PROCEDURE BookTicket(
  IN passengerId INT,
  IN scheduleId INT,
  IN seatNumber VARCHAR(10),
  IN travelClass VARCHAR(20)
)
BEGIN
  INSERT INTO Bookings (passenger_id, schedule_id, seat_number, booking_date, class)
  VALUES (passengerId, scheduleId, seatNumber, CURDATE(), travelClass);
END;
```

c. Cancel Booking

Cancels a booking and restores seat availability.

```
CREATE PROCEDURE CancelBooking(IN bookingId INT)
BEGIN
  DELETE FROM Bookings WHERE booking_id = bookingId;
END;
```

Triggers

a. Update Seat Availability on Booking

Automatically updates seat availability after a new booking.

```
CREATE TRIGGER AfterBookingInsert
AFTER INSERT ON Bookings
FOR EACH ROW
BEGIN
  UPDATE Vehicles
  SET total_seats = total_seats - 1
  WHERE vehicle_id = (SELECT vehicle_id FROM Schedules WHERE schedule_id =
NEW.schedule_id);
END;
```

b. Update Seat Availability on Cancellation

Automatically restores seat availability upon booking cancellation.

```
CREATE TRIGGER AfterBookingDelete
AFTER DELETE ON Bookings
FOR EACH ROW
BEGIN
  UPDATE Vehicles
```

```
    SET total_seats = total_seats + 1
    WHERE vehicle_id = (SELECT vehicle_id FROM Schedules WHERE schedule_id =
OLD.schedule_id);
END;
```

SQL Queries for Reports

a. Route Popularity Report

Identifies the most booked routes.

```
SELECT r.departure_station, r.arrival_station, COUNT(b.booking_id) AS bookings
FROM Routes r
JOIN Schedules s ON r.route_id = s.route_id
JOIN Bookings b ON s.schedule_id = b.schedule_id
GROUP BY r.departure_station, r.arrival_station
ORDER BY bookings DESC;
```

b. Occupancy Rates

Calculates occupancy rates for schedules.

```
SELECT schedule_id,
    (COUNT(booking_id) / v.total_seats) * 100 AS occupancy_rate
FROM Bookings b
JOIN Schedules s ON b.schedule_id = s.schedule_id
JOIN Vehicles v ON s.vehicle_id = v.vehicle_id
GROUP BY schedule_id;
```

c. Revenue by Route

Summarizes revenue by route and time period.

```
SELECT r.departure_station, r.arrival_station, SUM(p.amount) AS total_revenue
FROM Routes r
JOIN Schedules s ON r.route_id = s.route_id
JOIN Bookings b ON s.schedule_id = b.schedule_id
JOIN Payments p ON b.booking_id = p.booking_id
WHERE p.payment_date BETWEEN '2024-01-01' AND '2024-12-31'
GROUP BY r.departure_station, r.arrival_station
ORDER BY total_revenue DESC;
```

Conclusion

This **Public Transportation Management System** database ensures efficient handling of schedules, routes, bookings, and payments, supporting both operational needs and analytical reporting. The use of triggers and stored procedures automates critical processes like seat availability updates and ticket bookings, ensuring data integrity. Furthermore, robust reporting capabilities allow for insights into route popularity, occupancy, and revenue, aiding in better decision-making and resource allocation.