SQL & PL/SQL Project: Noida Metro Ticketing System

# Introduction

This project simulates a Metro Ticketing & Passenger Management System for the Noida Metro (Aqua Line and extensions). It includes SQL schema design, PL/SQL procedures, triggers, queries, and sample data inserts to demonstrate real-world database applications in Oracle SQL Developer.

# Step 1: Project Scope

The project models:  
- Metro Lines  
- Stations  
- Passengers  
- Tickets  
- Journeys  
- Train Schedules

# Step 2: SQL Schema

-- Table for Metro Lines  
CREATE TABLE metro\_line (  
 line\_id INT PRIMARY KEY,  
 line\_name VARCHAR2(50),  
 color VARCHAR2(20)  
);  
  
-- Table for Stations  
CREATE TABLE station (  
 station\_id INT PRIMARY KEY,  
 station\_name VARCHAR2(100),  
 line\_id INT REFERENCES metro\_line(line\_id),  
 opening\_date DATE  
);  
  
-- Table for Passengers  
CREATE TABLE passenger (  
 passenger\_id INT PRIMARY KEY,  
 name VARCHAR2(100),  
 card\_type VARCHAR2(20), -- SmartCard / QR Ticket  
 balance NUMBER(10,2)  
);  
  
-- Table for Tickets  
CREATE TABLE ticket (  
 ticket\_id INT PRIMARY KEY,  
 passenger\_id INT REFERENCES passenger(passenger\_id),  
 start\_station INT REFERENCES station(station\_id),  
 end\_station INT REFERENCES station(station\_id),  
 fare NUMBER(10,2),  
 journey\_date DATE  
);  
  
-- Table for Train Schedules  
CREATE TABLE train\_schedule (  
 train\_id INT PRIMARY KEY,  
 line\_id INT REFERENCES metro\_line(line\_id),  
 departure\_time TIMESTAMP,  
 arrival\_time TIMESTAMP  
);

## Step 2.1: Sequences (Oracle Auto-Increment)

CREATE SEQUENCE ticket\_seq START WITH 1 INCREMENT BY 1;  
CREATE SEQUENCE audit\_seq START WITH 1 INCREMENT BY 1;  
CREATE SEQUENCE passenger\_seq START WITH 1 INCREMENT BY 1;  
CREATE SEQUENCE station\_seq START WITH 1 INCREMENT BY 1;  
CREATE SEQUENCE line\_seq START WITH 1 INCREMENT BY 1;

## Step 2.2: Sample Data Inserts

-- Insert Metro Lines  
INSERT INTO metro\_line VALUES (line\_seq.NEXTVAL, 'Aqua Line', 'Blue-Green');  
INSERT INTO metro\_line VALUES (line\_seq.NEXTVAL, 'Blue Line', 'Blue');  
  
-- Insert Stations  
INSERT INTO station VALUES (station\_seq.NEXTVAL, 'Sector-51', 1, TO\_DATE('2019-01-25','YYYY-MM-DD'));  
INSERT INTO station VALUES (station\_seq.NEXTVAL, 'Sector-52', 2, TO\_DATE('2011-01-12','YYYY-MM-DD'));  
INSERT INTO station VALUES (station\_seq.NEXTVAL, 'Pari Chowk', 1, TO\_DATE('2019-01-25','YYYY-MM-DD'));  
INSERT INTO station VALUES (station\_seq.NEXTVAL, 'Depot Station', 1, TO\_DATE('2019-01-25','YYYY-MM-DD'));  
  
-- Insert Passengers  
INSERT INTO passenger VALUES (passenger\_seq.NEXTVAL, 'Rahul Sharma', 'SmartCard', 500);  
INSERT INTO passenger VALUES (passenger\_seq.NEXTVAL, 'Priya Singh', 'SmartCard', 200);  
INSERT INTO passenger VALUES (passenger\_seq.NEXTVAL, 'Amit Kumar', 'QR Ticket', 0);

# Step 3: PL/SQL Logic

## 1. Procedure to Book a Ticket

CREATE OR REPLACE PROCEDURE book\_ticket (  
 p\_passenger\_id IN INT,  
 p\_start\_station IN INT,  
 p\_end\_station IN INT  
) AS  
 v\_fare NUMBER(10,2);  
 v\_balance NUMBER(10,2);  
BEGIN  
 -- Calculate fare (simple fixed rate for demo)  
 v\_fare := ABS(p\_end\_station - p\_start\_station) \* 10;  
   
 SELECT balance INTO v\_balance  
 FROM passenger  
 WHERE passenger\_id = p\_passenger\_id;  
  
 IF v\_balance >= v\_fare THEN  
 -- Deduct balance  
 UPDATE passenger  
 SET balance = balance - v\_fare  
 WHERE passenger\_id = p\_passenger\_id;  
  
 -- Insert ticket record  
 INSERT INTO ticket(ticket\_id, passenger\_id, start\_station, end\_station, fare, journey\_date)  
 VALUES (ticket\_seq.NEXTVAL, p\_passenger\_id, p\_start\_station, p\_end\_station, v\_fare, SYSDATE);  
   
 DBMS\_OUTPUT.PUT\_LINE('Ticket booked successfully! Fare: ' || v\_fare);  
 ELSE  
 DBMS\_OUTPUT.PUT\_LINE('Insufficient balance.');  
 END IF;  
END;  
/

## 2. Trigger for Ticket Audit Log

CREATE TABLE ticket\_audit (  
 audit\_id INT PRIMARY KEY,  
 ticket\_id INT,  
 log\_time TIMESTAMP,  
 action VARCHAR2(50)  
);  
  
CREATE OR REPLACE TRIGGER trg\_ticket\_insert  
AFTER INSERT ON ticket  
FOR EACH ROW  
BEGIN  
 INSERT INTO ticket\_audit(audit\_id, ticket\_id, log\_time, action)  
 VALUES (audit\_seq.NEXTVAL, :NEW.ticket\_id, SYSTIMESTAMP, 'Ticket Booked');  
END;  
/

# Step 4: Sample Queries

-- Total passengers per station  
SELECT s.station\_name, COUNT(t.ticket\_id) AS total\_trips  
FROM ticket t  
JOIN station s ON t.start\_station = s.station\_id  
GROUP BY s.station\_name;  
  
-- Daily revenue report  
SELECT journey\_date, SUM(fare) AS total\_revenue  
FROM ticket  
GROUP BY journey\_date;

# Step 5: Testing Procedure

BEGIN  
 book\_ticket(1, 1, 3); -- Rahul books from Sector-51 to Pari Chowk  
 book\_ticket(2, 2, 4); -- Priya books from Sector-52 to Depot  
END;  
/