

Railway Ticket Management System

Srinivasa Sai Giddaluri (23EEB0A22)

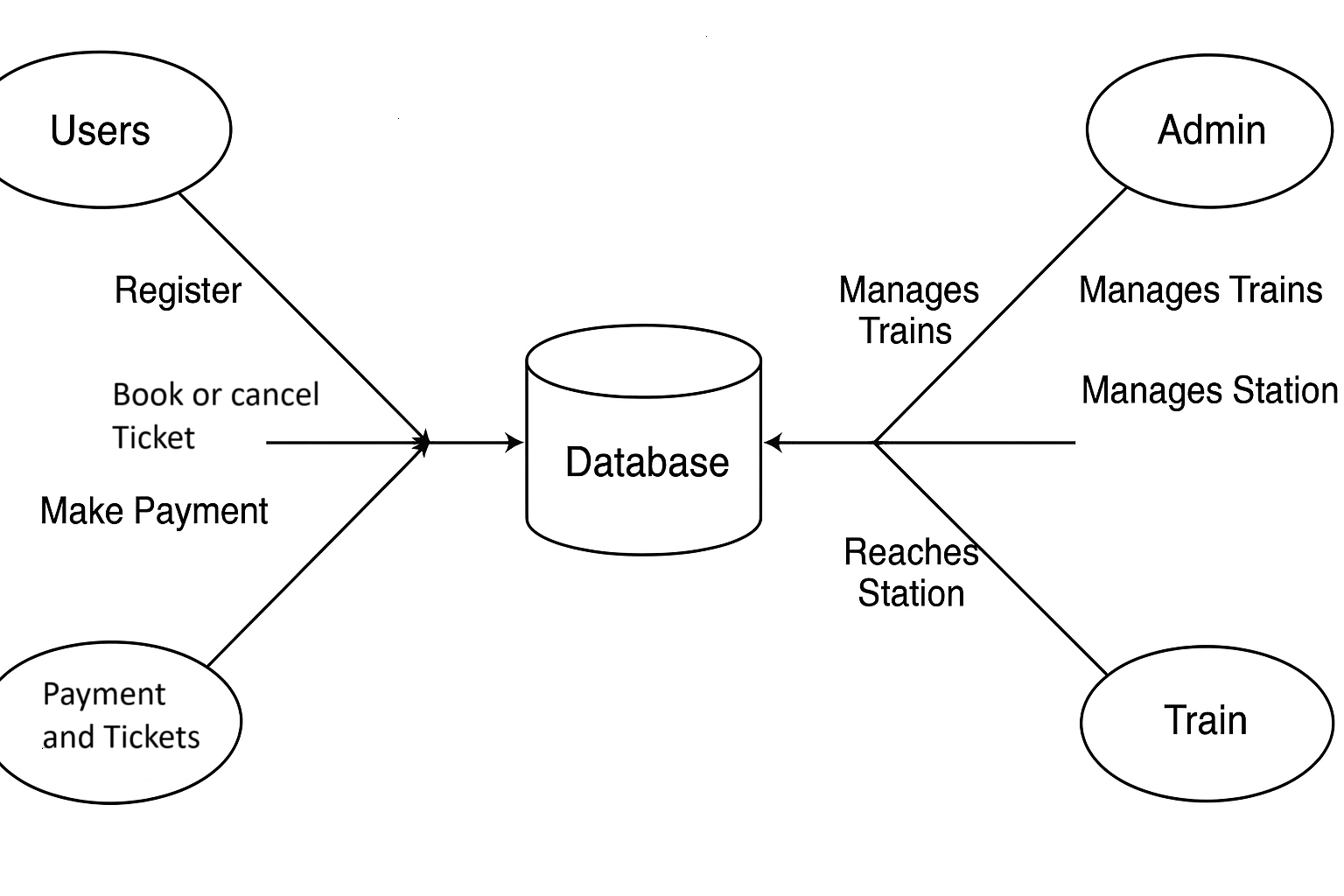
April 26th 2025

Introduction

A **Railway Ticket Management System**, is a software application which helps efficiently organize and manage all the data involved in booking train tickets. From storing passenger details to tracking train schedules and payments. It serves both passengers and railway authorities by digitizing ticket-related operations, reducing manual errors, and improving overall efficiency.

This system allows passengers to easily **search for trains**, **check seat availability**, **book tickets**, **make payments**, and even **cancel bookings** if plans change — all from the comfort of their home. It also keeps users updated with their **ticket status** (like Confirmed, RAC, or Waiting), making travel planning much more predictable and stress-free.

From the railway staff’s side, it helps **manage train schedules**, **track passenger information**, **handle payments**, and **monitor station data** efficiently. This reduces paperwork, prevents overbooking, and improves overall service quality.



# Problem Statement

We aim to develop a Database System for Online Railway Ticket Registration and Management Platform. The primary responsibilities of this system include:

1. **Allowing users to register and maintain personal profiles** for ticket booking and travel history tracking.
2. **Enabling passengers to book, cancel, and view train tickets** for different trains and travel dates.
3. **Maintaining payment records** of the passengers who have booked the tickets according to the class (2AC/Sleeper etc) details.
4. **Providing admins the ability to manage trains and stations**, update capacities, add or remove routes
5. **Supporting train and route scheduling**, including train numbers, arrival/departure times, and station stops.
6. **Tracking ticket status** (Confirmed, RAC, or Waiting) based on real-time availability and cancellations.

Key Features of the Railway Ticket Management System

**Passenger Information Management**

The system maintains complete passenger records, including full name, gender, age, contact number, date of birth, and address. Each passenger is uniquely identified and their information can be securely accessed and updated by authorized personnel.

**Ticket Booking and Reservation**

Passengers can search for trains, check availability, and book tickets. Once booked, each ticket is assigned a unique ID and PNR number, and linked to both the user and the selected train. The system prevents duplicate bookings and ensures seat allocation is managed in real-time.

**Train and Schedule Management**

Administrators can create and manage train profiles, including train number, name, capacity, and destination. The system also handles route details such as intermediate stations, arrival and departure times, and stop durations, ensuring accurate and conflict-free scheduling.

**Payment and Billing System**

When a ticket is booked, the system generates a bill and records the amount paid along with the payment method and time. It supports multiple payments per ticket and ensures all payment records are stored securely and can be reviewed when needed.

**Ticket Status Tracking**

Each ticket is assigned a current status such as Confirmed, RAC, or Waiting, depending on seat availability. The system updates the status automatically based on booking activity and cancellations, allowing passengers to track their position in the queue easily.

**Station and Route Mapping**

The system maintains a list of all train stations, including station codes, names, and timing details. Each train is linked to one or more stations, and the route is tracked to provide passengers with accurate travel information, including halt times and route coverage.

**Admin Control and Monitoring**

Administrators have access to manage train data, user records, station schedules, and ticket information. The system offers real-time insights into booking activity, available seats, and overall system usage, helping in efficient operation and decision-making.

Entity-Relationship Model

A diagram of a company

AI-generated content may be incorrect.

**ENTITIES**

1. USER
   * Represents: a passenger account in the system.
   * Attributes: USER\_ID, USERNAME, GENDER, MOBILE\_NO, AGE, DOB
   * Primary Key: USER\_ID
2. STATION
   * Represents: a railway station where trains start, stop, or end.
   * Attributes: STATION\_ID, STATION\_NAME, PLACE, STATE, ZONE, NO\_OF\_PLATFORMS
   * Primary Key: STATION\_ID
3. TRAIN
   * Represents: a train service traveling between stations.
   * Attributes: TRAIN\_ID, TRAIN\_NAME, START\_STATION\_ID, END\_STATION\_ID, CAPACITY, SCHEDULED\_TIME
   * Primary Key: TRAIN\_ID
4. TICKET\_STATUS
   * Represents: the booking status of a ticket.
   * Attributes: STATUS\_ID, STATUS\_NAME
   * Primary Key: STATUS\_ID
5. TICKET
   * Represents: a booked journey on a train.
   * Attributes: TICKET\_ID, PNR, USER\_ID, TRAIN\_ID, STATUS\_ID, TRAVEL\_DATE, CLASS, FARE
   * Primary Key: TICKET\_ID
   * Foreign Keys:
     + USER\_ID → USER
     + TRAIN\_ID → TRAIN
     + STATUS\_ID → TICKET\_STATUS
     + CLASS → BASE\_FARE, EXTRA\_FEE
6. PAYMENT
   * Represents: a payment transaction made by a user for a ticket.
   * Attributes: PAYMENT\_ID, TICKET\_ID, USER\_ID, AMOUNT, METHOD, PAYMENT\_DATE, PAYMENT\_STATUS
   * Primary Key: PAYMENT\_ID
   * Foreign Keys:
     + TICKET\_ID → TICKET
     + USER\_ID → USER
7. PRICING
   * Represents: fare rules per travel CLASS.
   * Attributes: CLASS, BASE\_FARE, EXTRA\_FARE
   * Primary Key: CLASS

RELATIONSHIPS

1. BOOKED
   * Between: USER (partially participates) and TICKET (totally participates)
   * Cardinality: 1 USER → M TICKET
   * Assumption: A user may book multiple tickets; every ticket must be booked by exactly one user.
   * Implementation: USER\_ID (PK of USER) is added as FK in TICKET.
2. STARTS\_FROM
   * Between: TRAIN (totally participates) and STATION (partially participates)
   * Cardinality: 1 TRAIN → 1 STATION
   * Assumption: Every train has exactly one start station; a station may be the start for many trains.
   * Implementation: START\_STATION\_ID (PK of STATION) is added as FK in TRAIN.
3. ENDS\_AT
   * Between: TRAIN (totally participates) and STATION (partially participates)
   * Cardinality: 1 TRAIN → 1 STATION
   * Assumption: Every train has exactly one end station; a station may be the end for many trains.
   * Implementation: END\_STATION\_ID (PK of STATION) is added as FK in TRAIN.
4. STOPS\_AT
   * Between: TRAIN (partially participates) and STATION (partially participates)
   * Cardinality: M TRAIN ↔ M STATION
   * Assumption: A train may stop at multiple stations; a station may serve many trains.
   * Implementation: Associative entity TRAIN\_STOP with composite PK (TRAIN\_ID, STATION\_ID) and attributes ARR\_TIME, DEP\_TIME, PLATFORM.
5. REQUIRES
   * Between: TICKET (partially participates) and PAYMENT (totally participates)
   * Cardinality: 1 TICKET → M PAYMENT
   * Assumption: A ticket may involve one or more payment transactions; every payment must be linked to one ticket.
   * Implementation: TICKET\_ID (PK of TICKET) is added as FK in PAYMENT.
6. PAYS
   * Between: USER (partially participates) and PAYMENT (totally participates)
   * Cardinality: 1 USER → M PAYMENT
   * Assumption: A user may make multiple payments; every payment must be made by one user.
   * Implementation: USER\_ID (PK of USER) is added as FK in PAYMENT.
7. HAS\_STATUS
   * Between: TICKET (partially participates) and TICKET\_STATUS (partially participates)
   * Cardinality: M TICKET → 1 TICKET\_STATUS
   * Assumption: Many tickets share the same status; each ticket has exactly one status.
   * Implementation: STATUS\_ID (PK of TICKET\_STATUS) is added as FK in TICKET.

8. ISSUED\_FOR

* Between: TICKET (totally participates) and TRAIN (partially participates)
* Cardinality: M TICKET → 1 TRAIN
* Assumption: Every ticket must be booked for exactly one train. A train can have many tickets issued over time.
* Implementation: TRAIN\_ID (PK of TRAIN) is added as a foreign key in TICKET.

9. PRICED\_AS

* Between: TICKET (partially participates) and PRICING (partially participates)
* Cardinality: M TICKET → 1 PRICING
* Assumption: Many tickets may use the same class (Sleeper, AC, etc.), and pricing is defined per class.
* A ticket refers to a class, and price can be calculated using base + extra fare for that class.
* Implementation: CLASS (PK of PRICING) is added as a foreign key in TICKET.

**Relational schema and Normalization**

A computer screen shot of a computer

AI-generated content may be incorrect.

### 1. USER

**FDs:**  
USER\_ID → USERNAME, GENDER, MOBILE\_NO, AGE, DOB  
**Primary Key:** USER\_ID  
**Normal Form:** Relation is in **BCNF**

### 2. STATION

**FDs:**  
STATION\_ID → STATION\_NAME, PLACE, STATE, ZONE, NO\_OF\_PLATFORMS  
**Primary Key:** STATION\_ID  
**Normal Form:** Relation is in **BCNF**

### 3. TRAIN

**FDs:**  
TRAIN\_ID → TRAIN\_NAME, START\_STATION\_ID, END\_STATION\_ID, CAPACITY, SCHEDULED\_TIME  
**Primary Key:** TRAIN\_ID  
**Normal Form:** Relation is in **BCNF**

### 4. TRAIN\_STOP

**FDs:**  
(TRAIN\_ID, STATION\_ID) → ARR\_TIME, DEP\_TIME, PLATFORM  
**Primary Key:** (TRAIN\_ID, STATION\_ID)  
**Normal Form:** Relation is in **BCNF**

### 5. TICKET\_STATUS

**FDs:**  
STATUS\_ID → STATUS\_NAME  
**Primary Key:** STATUS\_ID  
**Normal Form:** Relation is in **BCNF**

### 6. TICKET

**FDs:**  
TICKET\_ID → PNR, USER\_ID, TRAIN\_ID, STATUS\_ID, TRAVEL\_DATE, CLASS, FARE  
**Primary Key:** TICKET\_ID  
**Normal Form:** Relation is in **BCNF**

### 7. PAYMENT

**FDs:**  
PAYMENT\_ID → TICKET\_ID, USER\_ID, AMOUNT, METHOD, PAYMENT\_DATE, PAYMENT\_STATUS  
**Primary Key:** PAYMENT\_ID  
**Normal Form:** Relation is in **BCNF**

### 8. PRICING

**FDs:**  
CLASS → BASE\_FARE, EXTRA\_FARE  
**Primary Key:** CLASS  
**Normal Form:** Relation is in **BCNF**

### 9. TRAIN\_CLASS\_FARE

(If implemented separately for train-specific pricing)  
**FDs:**  
(TRAIN\_ID, CLASS) → FARE  
**Primary Key:** (TRAIN\_ID, CLASS)  
**Normal Form:** Relation is in **BCNF**

### 10. BOOKED

**FDs:**  
(USER\_ID, TRAIN\_ID) → (no other attributes)  
**Primary Key:** (USER\_ID, TRAIN\_ID)  
**Normal Form:** Relation is in **BCNF**

## SQL Queries to Create Tables

CREATE TABLE USER (

USER\_ID INT PRIMARY KEY,

USERNAME VARCHAR(50) NOT NULL,

GENDER CHAR(1),

MOBILE\_NO VARCHAR(15),

AGE INT,

DOB DATE

);

CREATE TABLE STATION (

STATION\_ID INT PRIMARY KEY,

STATION\_NAME VARCHAR(50) NOT NULL,

PLACE VARCHAR(50),

STATE VARCHAR(50),

ZONE VARCHAR(50),

NO\_OF\_PLATFORMS INT

);

CREATE TABLE TRAIN (

TRAIN\_ID INT PRIMARY KEY,

TRAIN\_NAME VARCHAR(100) NOT NULL,

START\_STATION\_ID INT NOT NULL,

END\_STATION\_ID INT NOT NULL,

CAPACITY INT,

SCHEDULED\_TIME TIMESTAMP,

FOREIGN KEY (START\_STATION\_ID) REFERENCES STATION(STATION\_ID),

FOREIGN KEY (END\_STATION\_ID) REFERENCES STATION(STATION\_ID)

);

CREATE TABLE TRAIN\_STOP (

TRAIN\_ID INT NOT NULL,

STATION\_ID INT NOT NULL,

ARR\_TIME TIME NOT NULL,

DEP\_TIME TIME NOT NULL,

PLATFORM INT,

PRIMARY KEY (TRAIN\_ID, STATION\_ID),

FOREIGN KEY (TRAIN\_ID) REFERENCES TRAIN(TRAIN\_ID),

FOREIGN KEY (STATION\_ID) REFERENCES STATION(STATION\_ID)

);

CREATE TABLE TICKET\_STATUS (

STATUS\_ID INT PRIMARY KEY,

STATUS\_NAME VARCHAR(50) NOT NULL UNIQUE

);

CREATE TABLE TICKET (

TICKET\_ID INT PRIMARY KEY,

PNR VARCHAR(20) NOT NULL UNIQUE,

USER\_ID INT NOT NULL,

TRAIN\_ID INT NOT NULL,

STATUS\_ID INT NOT NULL,

TRAVEL\_DATE DATE NOT NULL,

CLASS VARCHAR(10) NOT NULL,

FARE DECIMAL(10,2) NOT NULL,

FOREIGN KEY (USER\_ID) REFERENCES USER(USER\_ID),

FOREIGN KEY (TRAIN\_ID) REFERENCES TRAIN(TRAIN\_ID),

FOREIGN KEY (STATUS\_ID) REFERENCES TICKET\_STATUS(STATUS\_ID),

FOREIGN KEY (CLASS) REFERENCES PRICING(CLASS)

);

CREATE TABLE PAYMENT (

PAYMENT\_ID INT PRIMARY KEY,

TICKET\_ID INT NOT NULL,

USER\_ID INT NOT NULL,

AMOUNT DECIMAL(10,2) NOT NULL,

METHOD VARCHAR(50) NOT NULL,

PAYMENT\_DATE DATETIME NOT NULL,

PAYMENT\_STATUS VARCHAR(20) NOT NULL CHECK (PAYMENT\_STATUS IN ('PAID','PENDING','REFUNDED')),

FOREIGN KEY (TICKET\_ID) REFERENCES TICKET(TICKET\_ID),

FOREIGN KEY (USER\_ID) REFERENCES USER(USER\_ID)

);

CREATE TABLE BOOKED (

USER\_ID INT NOT NULL,

TRAIN\_ID INT NOT NULL,

PRIMARY KEY (USER\_ID, TRAIN\_ID),

FOREIGN KEY (USER\_ID) REFERENCES USER(USER\_ID),

FOREIGN KEY (TRAIN\_ID) REFERENCES TRAIN(TRAIN\_ID)

);

CREATE TABLE PRICING (

CLASS VARCHAR(10) PRIMARY KEY,

BASE\_FARE DECIMAL(10,2) NOT NULL,

EXTRA\_FEE DECIMAL(10,2) default 0.0

);

SQL queries to insert data

-**-User**

INSERT INTO USER VALUES

(101,'RAMESH','M','99121836899',35,'1990-08-10'),

(118,'PARDHIV','M','7356282857',20,'2005-06-21'),

(301, 'VENKATESH', 'M', '9845123456', 65, '1960-07-04'),

(211,'GEETHA','F','9898346582',25,'2000-11-08');

--Station

INSERT INTO STATION VALUES (1,'SECUNDERABAD','SECUNDERABAD','TELANGANA','SOUTH',10),

(7,'MUMBAI ','MUMBAI','MAHARASTRA','WEST',8),

(17,'HAZUR SAHIB NANDED','NANDED','MAHARASTRA','WEST',6);

--Train

INSERT INTO TRAIN VALUES

(12733, 'GODAVARI EXPRESS', 1, 7, 1200, '2025-07-05 05:00:00'),

(17299, 'NANDED EXPRESS', 17, 1, 1000, '2025-07-05 09:00:00'),

(14388, 'TIRUPATHI INTERCITY', 21, 8, 800, '2025-07-06 06:30:00');

--Ticket\_Status

INSERT INTO TICKET\_STATUS VALUES

(1, 'CONFIRMED'),

(2, 'WAITING'),

(3, 'CANCELLED');

--Booked

INSERT INTO BOOKED VALUES

(101, 12733),

(118, 14388),

--Train\_Stops

INSERT INTO TRAIN\_STOP VALUES

(12733, 1, '05:00:00', '05:10:00', 1),

(12733, 17, '10:00:00', '10:05:00', 2),

(12733, 7, '20:00:00', '20:10:00', 4);

--Pricing

INSERT INTO PRICING (CLASS, BASE\_FARE, EXTRA\_FARE) VALUES

('1AC', 2000.00, 300.00),

('2AC', 1500.00, 200.00),

('3AC', 1000.00, 100.00),

('SLEEPER', 500.00, 50.00),

('GENERAL', 200.00, 0.00);

--Ticket  
INSERT INTO TICKET VALUES

(1001, 'PNR1001', 101, 12733, 1, '2025-07-06', '1AC', 2300.00),

(1002, 'PNR1002', 118, 14388, 2, '2025-07-07', 'GENERAL', 200.00),

(1003, 'PNR1003', 211, 17299, 1, '2025-07-07', '2AC', 1700.00),

--Payment

INSERT INTO PAYMENT VALUES

(501, 1001, 101, 2300.00, 'CREDIT\_CARD', '2025-07-01 14:20:00', 'PAID'),

(502, 1002, 118, 200.00, 'UPI', '2025-07-02 09:00:00', 'PENDING'),

(503, 1003, 211, 1700.00, 'DEBIT\_CARD', '2025-07-02 10:30:00', 'PAID'),

Database In Action

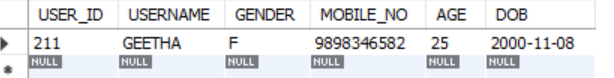
1. **Passenger Information Management**

i)All Female passengers

SELECT \*

FROM USER

WHERE GENDER='F';

ii)All senior citizens travelling

SELECT \*

FROM USER

WHERE AGE>60;



2. **Ticket Booking and Reservation**

SELECT

U.USERNAME,

TR.TRAIN\_NAME,

T.CLASS,

T.TRAVEL\_DATE,

TS.STATUS\_NAME

FROM TICKET T

JOIN USER U ON T.USER\_ID = U.USER\_ID

JOIN TRAIN TR ON T.TRAIN\_ID = TR.TRAIN\_ID

JOIN TICKET\_STATUS TS ON T.STATUS\_ID = TS.STATUS\_ID

ORDER BY T.TRAVEL\_DATE, U.USERNAME;

A table with text on it

AI-generated content may be incorrect.

**3.Train and Schedule Management**

**--Insert new stop**

INSERT INTO TRAIN\_STOP

(TRAIN\_ID, STATION\_ID, ARR\_TIME, DEP\_TIME, PLATFORM)

VALUES

(17299, 21, '15:00:00', '15:10:00', 2);

-- Verify full route for 17299

SELECT

S.STATION\_NAME,

TS.ARR\_TIME,

TS.DEP\_TIME,

TS.PLATFORM

FROM TRAIN\_STOP TS

JOIN STATION S ON TS.STATION\_ID = S.STATION\_ID

WHERE TS.TRAIN\_ID = 17299

ORDER BY TS.ARR\_TIME;

**A screenshot of a computer

AI-generated content may be incorrect.**

4.**Payment and Billing System**

--list of all tickets booked in July,2025

SELECT

T.TICKET\_ID,

T.USER\_ID,

T.FARE,

COALESCE(SUM(P.AMOUNT), 0) AS PAID FROM TICKET T

LEFT JOIN PAYMENT P ON T.TICKET\_ID = P.TICKET\_ID

WHERE T.TRAVEL\_DATE BETWEEN '2025-07-01'

AND '2025-07-31'

GROUP BY T.TICKET\_ID, T.USER\_ID, T.FARE;

A screenshot of a computer

AI-generated content may be incorrect.

**5.Ticket Status Tracking**

SELECT

TR.TRAIN\_NAME,

TS.STATUS\_NAME,

COUNT(\*) AS NUM\_TICKETS

FROM TICKET T

JOIN TRAIN TR ON T.TRAIN\_ID = TR.TRAIN\_ID

JOIN TICKET\_STATUS TS ON T.STATUS\_ID = TS.STATUS\_ID

WHERE T.TRAVEL\_DATE = '2025-07-07'

GROUP BY TR.TRAIN\_NAME, TS.STATUS\_NAME

ORDER BY TR.TRAIN\_NAME, TS.STATUS\_NAME;

A close up of a sign

AI-generated content may be incorrect.