Transport Bookings Database Project Report

Student ID: 23073211
Student Name: Prabhu Sekhar Reddy Settipalli
Github Link: https://github.com/prabhu124-sep/Transport-Bookings-Database-Project

---

1. Project Justification

This project models a realistic multi-modal transport booking system using a normalized relational database with three linked tables:

- **Passengers:** Demographics and registration details.
- **Vehicles:** Fleet composition and operational attributes.
- **Bookings:** All individual journey records, connecting passengers and vehicles on booked trips.

The schema supports analysis of passenger behavior, vehicle utilization, route popularity, and booking status, suitable for operational analytics in a public transport or commercial fleet context.

---

2. ER Diagram Overview

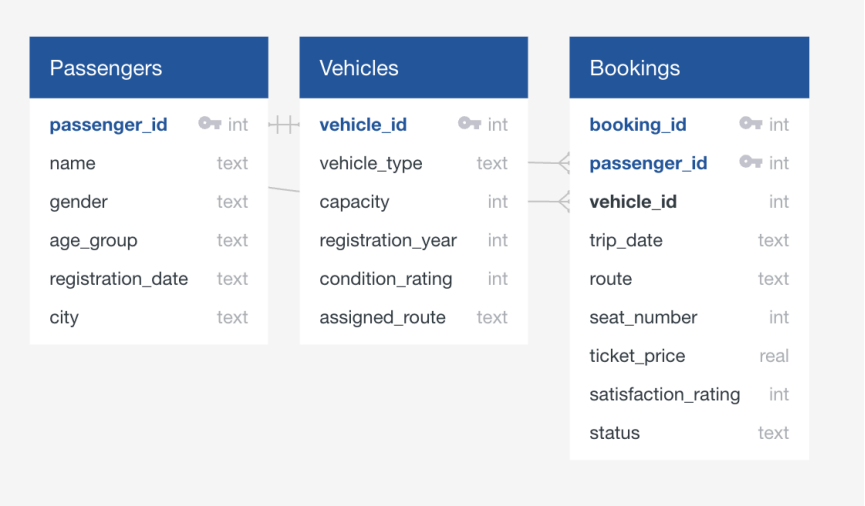

Figure 01

Relationships and Structure:

- **One-to-Many:** Passengers → Bookings; Vehicles → Bookings.
- The Bookings table uses a **composite primary key (booking_id, passenger_id)** for uniqueness and is joined to both Passengers (by passenger_id) and Vehicles (by vehicle_id), ensuring referential integrity.
- Key foreign keys and composite keys are clearly depicted.

---

3. Database Realism and Data Preparation

Synthetic Data Generation:

- Data generated using Python (Faker/random/pandas) to simulate 600 passengers, 40 vehicles, and 1200 bookings.
- *Missing Values:* Introduced in gender and city (Passengers), condition_rating (Vehicles), and status & satisfaction_rating (Bookings)—mimicking realistic operational data gaps.
- *Handling:* Text NULLs replaced with 'unknown', numeric NULLs with 0 before querying and analysis.

---

4. Ethics and Data Privacy

- All data is **synthetic**; no real individuals or travel data are used.
- Personally identifiable information is avoided; only generic labels, attributes, or randomly generated values populate fields.
- *Transparency:* All data cleaning, imputation, and randomization is documented in the project appendix and code.
- *Fairness:* Category assignments (e.g., age group, booking status, satisfaction) are random, minimizing bias.

---

5. Database Schema and Keys

- **Passengers:** Indexed by unique passenger_id.
- **Vehicles:** Indexed by unique vehicle_id.
- **Bookings:** Composite key (booking_id, passenger_id); foreign keys enforce links to Passengers and Vehicles.
- Data types include integer, real, and text, with date fields as ISO-formatted strings for SQLite compatibility.

---

6. Data Types

- **Integers:** IDs, seat numbers, vehicle capacities, and condition/satisfaction ratings.
- **Text:** Names, genders, cities, types, routes, booking statuses, dates (as strings).
- **Real:** Ticket price (Bookings).

---

7. SQL Analysis and Results Overview

7.1 Average Ticket Price Per Vehicle

```
SELECT Vehicles.vehicle_type, AVG(Bookings.ticket_price) AS avg_price,
COUNT(Bookings.booking_id) AS total_bookings
FROM Bookings
JOIN Vehicles ON Bookings.vehicle_id = Vehicles.vehicle_id
GROUP BY Vehicles.vehicle_id
ORDER BY avg_price DESC
LIMIT 5;
```

Findings: Provided mean prices by vehicle and usage, example:

vehicle_type	avg_price	total_bookings
Bus	81.48	24
Van	78.15	24

7.2 Seat Number Distribution

```
SELECT seat_number, COUNT(*) AS frequency
FROM Bookings
GROUP BY seat_number
ORDER BY seat_number;
```

Shows which seats are most/least commonly booked over the sample period.

7.3 Most Used Routes

```
SELECT route, COUNT(*) AS num_bookings
FROM Bookings
GROUP BY route
ORDER BY num_bookings DESC;
```

Identifies most popular transport routes (e.g., Route D had 144 bookings, Route J 136, …).

7.4 Booking Status Counts

```
SELECT status, COUNT(*) AS count
```

```
FROM Bookings
GROUP BY status
ORDER BY count DESC;
```

Distribution of confirmed, pending, cancelled, and missing/unknown booking statuses.

7.5 Passengers With Most Bookings

```
SELECT Passengers.name, COUNT(Bookings.booking_id) AS num_bookings
FROM Bookings
JOIN Passengers ON Bookings.passenger_id = Passengers.passenger_id
GROUP BY Passengers.passenger_id
ORDER BY num_bookings DESC
LIMIT 10;
```

Lists high-frequency travelers and their booking counts.

---

8. Insights and Observations

- **Vehicle and route utilization** patterns are clearly visible, supporting operational planning.
- **Data cleaning** ensures NULL/missing values do not bias summary statistics.
- The use of **composite keys and foreign keys** enables robust multi-table analysis and ensures referential integrity.
- Complete transparency in how synthetic data, missing values, and potential bias were handled.

---

9. Conclusion

This transport booking project demonstrates:

- Best practices in SQL schema design (composite/foreign keys, normalized structure)
- Realistic synthetic data creation and cleaning for analytics
- Effective exploratory SQL queries and results interpretation
- Ethical use of artificial/generated data sources

Appendix

A. Table Creation Queries

```sql
-- Passengers
CREATE TABLE Passengers (
    passenger_id INTEGER PRIMARY KEY,
    name TEXT,
    gender TEXT,
    age_group TEXT,
    registration_date TEXT,
    city TEXT
);

-- Vehicles
CREATE TABLE Vehicles (
    vehicle_id INTEGER PRIMARY KEY,
    vehicle_type TEXT,
    capacity INTEGER,
    registration_year INTEGER,
    condition_rating INTEGER,
    assigned_route TEXT
);

-- Bookings
CREATE TABLE Bookings (
    booking_id INTEGER,
    passenger_id INTEGER,
    vehicle_id INTEGER,
    trip_date TEXT,
    route TEXT,
    seat_number INTEGER,
    ticket_price REAL,
    satisfaction_rating INTEGER,
    status TEXT,
    PRIMARY KEY (booking_id, passenger_id),
    FOREIGN KEY (passenger_id) REFERENCES Passengers(passenger_id),
    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id)
);
```