# 🚍 Ahmedabad BRTS Transport Analyst Project

# **Case Study**

## 1. Problem Statement

Ahmedabad’s BRTS (Bus Rapid Transit System) generates a large amount of operational data (routes, trips, stops, vehicles, delays, and passenger feedback). However, raw data alone does not provide actionable insights for decision-making. The goal of this project is to

* Analyze bus delays and punctuality.
* Measure passenger satisfaction through ratings.
* Identify top/bottom performing routes and drivers.
* Build an interactive **Power BI dashboard** for transport analysts and city planners.

## 2. Data Collection & Preparation

* **Scraping:** Bus timetables scraped from official **Ahmedabad BRTS website** using **Python (Selenium)**.
* **Enrichment:** Added **latitude/longitude** for stops via **Geopy**.
* **Simulation:** Generated random **delays (1–10 min)** and **weather conditions** for trips.
* **Passenger Feedback:** 450+ random names with ratings (1–5).
* **Storage:** Data stored in structured CSV files.

Key Data Sources:

* brts\_routes.csv → Route details.
* brts\_stops.csv → Stops + stop sequence.
* brts\_route\_stops.csv → Route–stop mapping.
* brts\_trip\_data.csv → Trip timings + delays + weather.
* brts\_passenger\_feedback.csv → Ratings & feedback.
* brts\_vehicles.csv → Vehicle & driver details

## 3. Database Design (PostgreSQL)

A **normalized relational schema** was implemented:

### Tables

1. **route** → (route\_id, route\_name, start\_stop, end\_stop)
2. **stops** → (stop\_id, stop\_name, latitude, longitude)
3. **route\_stops** → (route\_id, stop\_id, stop\_sequence)
4. **vehicles** → (vehicle\_id, route\_id, vehicle\_type, driver\_name, driver\_license)
5. **trip\_data** → (trip\_id, route\_id, scheduled\_departure, scheduled\_arrival, actual\_departure, actual\_arrival, delay\_sec, weather\_condition, time\_slot)
6. **passenger\_feedback** → (feedback\_id, trip\_id, passenger\_name, rating)

### Design Highlights

* **route\_stops →** bridge table for many-to-many relationships.
* **trip\_data →** stored TIME only (simplified).
* Data cleaning for missing lat/long using **SQL Window Functions (LAG/LEAD)**

## 4. SQL Analysis

Queries designed for analysis from beginner → advanced:

**Beginner (CRUD, Filters, Aggregates)**

* Insert a new passenger feedback for trip\_id = 25 with name "Jay Patel" and rating 4.
* Update the driver\_name for vehicle\_id = 5 to "Rishi Ladani".
* Show all trips delayed more than 5 minutes (delay\_sec > 300), ordered by delay descending.
* Count the total number of unique stops across all routes.
* Find the average passenger rating per route, but show only those routes where avg\_rating < 3.5.

**Intermediate (Joins, Subqueries, Basic CTEs, CASE)**

* List all trips with their route\_name, vehicle\_type, and driver\_name (JOIN route + vehicles + trip\_data).
* Find trips where delay\_sec is greater than the average delay of all trips (subquery).
* Rank all vehicles by their average delay using a Window Function (RANK).
* Categorize each trip into Morning, Afternoon, Evening, or Night based on scheduled\_departure using a CASE expression.
* Create a CTE that calculates the average delay per route, and then select only the top 3 routes with the highest delay.
* Find trips that are faster than scheduled (actual\_arrival < scheduled\_arrival).
* Use a CASE expression to classify delay severity (0–200 = On Time, 201–500 = Slight Delay, 501+ = Heavy Delay).
* For each route, calculate the average trip delay and classify it as Low Delay (<300), Medium Delay (300–600), or High Delay (>600).
* Identify the route with the maximum number of unique stops using route\_stops and stops tables.
* Identify the routes where more than 40% of trips are delayed (delay\_sec > 0).
* For each driver, calculate the percentage of trips where the delay exceeded 10 minutes (600 sec).

**Advanced (Window Functions, Recursive CTEs, Ranking, Analytical Queries)**

* Using a Window Function, calculate the running total of trips for each route, ordered by scheduled\_departure.
* Find the total number of trips for each route using a Window Function (without using GROUP BY).
* Find the top 3 vehicles with the highest total delay time across all trips.
* Find the driver(s) whose vehicles have the least average delay across trips.
* For each time slot (Morning, Afternoon, Evening, Night), find the most delayed route.
* List the first and last stop of every route (start → end) using window functions.
* Full stop sequence for route\_id = 5 → table: “Q8\_Full\_Path\_for\_route\_5” (recursive CTE).
* Top 5 passengers with lowest avg ratings → table: “Q9\_Top5\_Passengers\_Lowest\_Avg\_Ratings”.
* Find the top 3 routes with the highest average delay per trip.
* Identify the stop that is part of the maximum number of different routes.
* Find the passengers who always gave the same rating (never changed their rating).
* Find the top 5 trips with the worst delay-to-rating ratio (delay\_sec ÷ passenger rating).

## 5. Power BI Dashboard

Designed a **multi-page interactive dashboard** with modern theme (light background, **blue** for reliability, **orange** for delays)

### Pages & Visuals

### **Page 1: Overview**

* + KPIs: Total Trips | Avg Delay | % Delayed Trips | Avg Passenger Rating
  + City Map: Routes & stops with delay overlay
  + Donut: On-Time vs Delayed vs Early Trips
  + Bar: Top 3 Delayed Routes
  + Line: Delay Trend by Time of Day
* **Page 2: Route Performance**
* Route table: Trips, Avg Delay, Avg Rating, Stops
* Routes ranked by Avg Delay (bar)
* Route with Maximum Stops (highlight card)
* Delay by Time Slot (heatmap/line chart)
* **Page 3: Trip Analytics**
* Delay Severity Distribution (donut/stacked bar)
* Running Total Trips per Route (line chart)
* Top 5 Worst Trips by Delay-to-Rating Ratio (table)
* KPI: % Faster-than-schedule trips (dataset shows none)
* **Page 4: Driver & Vehicle**
* Driver table: Avg Delay, Trips >10 min delay
* Vehicle Delay Ranking (bar)
* Delays by Vehicle Type (pie → only BRTS bus type)
* Best Performing Driver (highlight card)
* **Page 5: Passenger Feedback**
* Passenger table: Avg Rating, Feedback Count
* Top 5 Passengers with Lowest Ratings (bar)
* Delay vs Rating Correlation (scatter)
* Consistent Passengers (KPI → fixed rating behavior)

6. Tools & Skills

## **Python:** Selenium, Pandas, Geopy (scraping + preprocessing).

## **PostgreSQL:** SQL, CTEs, Window Functions, CASE, Subqueries.

## **Power BI:** DAX measures, KPI cards, advanced visuals.

## **Data Analytics Skills:** Cleaning, Joins, Aggregates, Visualization, Dashboard design.

## 7. Final Deliverables

* PostgreSQL Database with normalized schema.
* 28 SQL Queries (CRUD, aggregates, windows, recursive CTEs).
* Final CSV datasets.
* Power BI Dashboard (5-page interactive report).

**Result: A complete end-to-end transport analytics solution — from web scraping to PostgreSQL to Power BI**