Part 3 – Database Design

Project Title: Smart Radar Traffic Monitoring System

Overview

The database design for the *Smart Radar Traffic Monitoring System* is built to efficiently handle large-scale traffic data generated by radar sensors.
It supports **real-time streaming** and **batch processing** workflows, enabling quick analysis, alerting, and visualization.

The system uses a **two-layer storage architecture**:

1. **Azure Data Lake** — for storing raw and processed Parquet files.
2. **PostgreSQL Data Warehouse** — for structured queries and Power BI dashboards.

Database Objectives

- Store, clean, and organize incoming radar data.
- Log all types of traffic violations and real-time alerts.
- Provide fast access for analytics and dashboard queries.
- Ensure data integrity and scalability for large volumes.

Main Tables

Table Name	Description	Key Fields
vehicles	Stores all raw vehicle detection data collected by simulated radars.	plate_number, color, speed, location, timestamp, seat_belt, phone_usage
violations	Records detected violations, such as speeding or phone usage.	violation_id, plate_number, violation_type, timestamp, location

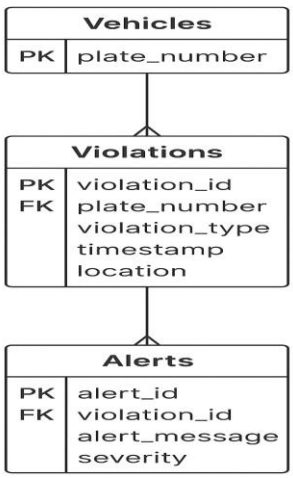
Table Name	Description	Key Fields
alerts	Stores alerts triggered in real-time by the streaming system (e.g., wanted vehicles).	alert_id, plate_number, alert_message, severity, alert_time
aggregated_reports	Stores daily or weekly summarized data for dashboard use.	report_date, total_vehicles, total_violations, avg_speed, top_location

---

Relationships (ERD Summary)

- **One Vehicle → Many Violations**
  Each vehicle may appear multiple times in the system if it commits more than one violation.
- **One Violation → May Trigger One Alert**
  A severe violation (e.g., a wanted vehicle) can generate an alert notification.
- **Aggregated Reports** summarize historical data from **vehicles** and **violations** tables.

Example ERD Structure



Data Flow Integration

1. **Data Simulation & Ingestion:**
   Python scripts generate synthetic radar data → sent to Kafka topics.
2. **Batch Processing:**
   PySpark reads Kafka data → cleans & writes to Azure Data Lake (Parquet).
3. **Streaming Processing:**
   Real-time Spark job consumes Kafka stream → detects violations → writes alerts to PostgreSQL.
4. **Visualization Layer:**
   Power BI connects to PostgreSQL → generates visual reports and insights.

Data Integrity & Quality Rules

- Records with missing or invalid plate_number or negative speed are removed.
- Duplicated events are filtered using unique event IDs.
- All timestamps are standardized to UTC format.
- Referential integrity between **violations** and **vehicles** is enforced.