Real-Time Traffic Analytics & Smart Lighting Control System**

1. Project Overview

The *Real-Time Traffic Analytics & Smart Lighting Control System* is an end-to-end data engineering and IoT analytics pipeline designed to monitor city traffic conditions, detect congestion, optimize lighting consumption, and generate automated alerts.
The project integrates **IoT devices, MQTT, Kafka, Spark Structured Streaming, SQL Server, Parquet data lakes, machine learning (optional), and a lighting actuator** to create a complete real-time smart-city platform.

This system collects live data from simulated IoT sensors deployed across several streets (vehicle count, speed, weather, light level, energy demand, solar power), processes the data in real-time using Spark Streaming, stores it in a scalable data lake and relational database, and triggers automated alerts and lighting adjustments.

2. Project Objectives

1. Build a reliable real-time ingestion pipeline using MQTT + Kafka.
2. Perform continuous data transformation, feature engineering, and alert detection using Spark Streaming.
3. Store raw and processed data in SQL Server and Parquet-based data lakes.
4. Automate database loading using SSIS (SQL Server Integration Services).
5. Implement an IoT lighting actuator that responds to congestion alerts.
6. Enable proactive monitoring via Telegram notifications.
7. Provide a scalable architecture suitable for smart-city analytics.

3. System Architecture

The project follows a multi-layer architecture:

Data Ingestion Layer

- IoT sensors publish JSON messages via **MQTT** to `traffic/data/menofia_national_university`.
- An **MQTT → Kafka Bridge** forwards these messages to a Kafka topic (`traffic_topic`).

Streaming Processing Layer

- Apache Spark Structured Streaming reads the Kafka topic in micro-batches.
- Data is parsed, cleaned, and enriched (hour, day, peak hours, congestion logic, energy calculations).
- Real-time windowed aggregations compute:
    - total vehicles per window
    - average speed
    - congestion level
    - lighting energy consumption

Alerting Layer

Spark generates alerts based on congestion severity:

- `High` or `Medium` alerts are pushed to the `traffic-alerts` Kafka topic.
- A **Telegram bot** sends real-time notifications.
- A **lighting controller actuator** adjusts lighting levels via MQTT.

Storage Layer

Two main storage sinks:

1. **Parquet Data Lake**
    a. Stores all transformed data
    b. Used for long-term analytics + later ML work
2. **SQL Server (Normalized Tables)**
    Spark divides each batch into three relational tables using `foreachBatch`:
    a. traffic_sensors_data
    b. traffic_weather_conditions
    c. traffic_energy_analysis

ETL Layer – SSIS

SQL Server Integration Services automates:

- Restoring database from `.bak` (Final_Project_des.bak)

- Creating missing tables
- Loading CSV files exported from Spark or control pipeline
- Running Bulk Insert operations
- Archiving processed files
- Monitoring failures

This ensures strong enterprise-grade data integration.

4. Key Components & Technologies

Component	Purpose
MQTT	IoT message transport for simulated traffic sensors
Kafka	Distributed message broker for high-throughput ingestion
Spark Structured Streaming	Real-time processing & windowed analytics
SQL Server	Relational storage for reporting & dashboards
Parquet (Data Lake)	Scalable storage for analytical workloads
SSIS	Automated ETL pipeline for SQL Server loading
Docker Compose	Containerized environment for Kafka, Zookeeper, Jupyter
Telegram Bot	Real-time alerts to the user
Lighting Actuator (MQTT)	IoT device simulation that receives alert decisions

5. Data Processing Logic

Each streaming record contains:

- timestamp
- street_name
- vehicle_count
- vehicle_speed
- light_level
- weather
- solar_energy_level
- lighting_demand

Computed Features

- hour_of_day
- day_of_week
- Is_peak_hour (rush hours: 7–9 AM, 4–6 PM)
- congestion_level (High, Medium, Low)
- Is_congested (binary)
- lighting_demand_kW (converted from text)
- lighting_consumption_kWh (demand − solar)
- energy_alert_flag

Alert Generation

Alerts occur when:

- average speed drops below 25
- or vehicle_count exceeds threshold
- or lighting energy consumption is high

Alerts are structured JSON messages published to Kafka and MQTT.

6. Output Storage

A) Parquet Data Lake

- Stores clean, processed, schematized data
- Supports ML, BI, dashboarding, and analytics
- Automatic checkpointing for fault tolerance

B) SQL Server (3-Table Normalized Schema)

traffic_sensors_data: vehicle-level metrics
traffic_weather_conditions: environmental data
traffic_energy_analysis: energy usage & solar compensation

Used for reporting, dashboards, and relational queries.

7. SSIS ETL Pipeline

The SSIS package automates the backend database processes:

- Restores the `.bak` file (Final_Project_des.bak)
- Validates tables and re-creates them if needed
- Loads CSV/Parquet-derived datasets into SQL tables
- Performs bulk insert operations for large throughput
- Moves processed files into archive folders
- Logs errors, supports retry logic

This ensures the system is "production-grade" and ready for enterprise deployment.

8. IoT Actuator Integration

The lighting actuator subscribes to:

```
lighting/<street_name>/set
```

When Spark detects congestion:

- Lighting level increases (MEDIUM or HIGH)
- State is saved into JSON state file
- MQTT messages simulate smart-city lighting control

9. Alerts & Notification System

Alerts from Spark are consumed by:

- Kafka-based alert notifier
- Telegram bot
- Lighting actuator

Alert message example:

```
{
  "street_name": "El-Tahrir",
  "alert_level": "HIGH",
  "vehicles_per_window": 92,
  "avg_speed": 14.8
}
```

10. Final Deliverables

- Complete Kafka–Spark–SQL pipeline
- Parquet data lake
- SQL Server normalized database
- SSIS ETL package
- Real-time alerting system
- Lighting control actuator
- Telegram monitoring bot
- Fully documented source code
- Presentation (Milestone 1, 2, 3)
- Live running system

11. Conclusion

This project successfully delivers a scalable, real-time smart-city traffic monitoring and analytics ecosystem.
It integrates streaming data engineering, IoT communication, automated alerting, optimized lighting control, and enterprise-grade ETL using SSIS.
The architecture is scalable, fault-tolerant, and ready for further expansion into machine learning and predictive analytics (Milestone 4).