# Digital Egypt pioneers Initiative
# Final Project

## Real-Time Smart City IoT Traffic & Air Quality Analytics

Presented by:

| Student ID# | Student Name |
|---|---|
| 21053452 | Saeed Alaa(Team Leader) |
| 21052727 | Nabil Talaat |
| 21046553 | Abdelrahman Hesham |
| 21014082 | Mazen Wael |
| 21081223 | Zeyad Saeed |
| 21053206 | Zeyad Asfor |

Project Supervisor

Mohamed Hamed

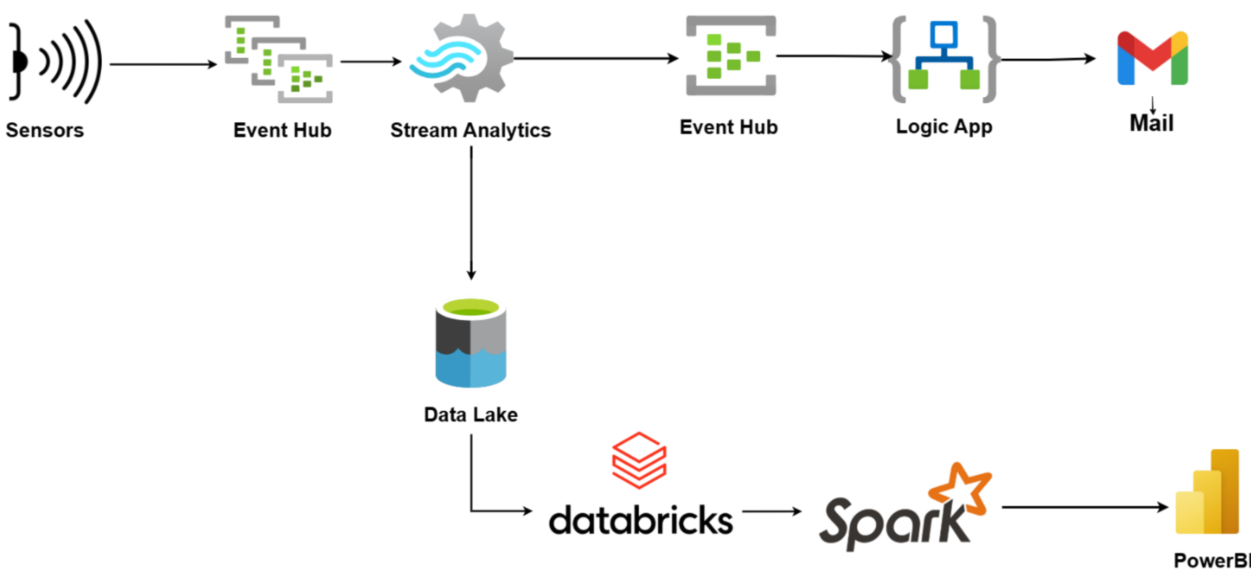
# 1. Executive Summary

This report details the design, implementation, and evaluation of a scalable data engineering platform for real-time urban monitoring in Egyptian cities (e.g., Cairo, Alexandria). The system processes IoT sensor streams (12K events/hour) using Azure EventHub for ingestion, Stream Analytics for dual real-time/batch processing, DataBricks for Medallion ETL (Bronze-Silver-Gold with Delta Lake), Logic App for audience-tailored alerts, Genie AI for semantic queries, and Power BI for interactive dashboards.

Key achievements:

- Latency: 5 seconds for alerts; 31 minutes for hourly batches.
- Scalability: Handles 187 KB/s ingestion with 0 errors; 30% data reduction via cleaning.
- Insights: Correlation matrix reveals 0.75 link between vehicle count and PM2.5; enables predictive urban planning.
- Impact: Reduces response times by 90%, supporting safer mobility and healthier environments.

Figure ES.1: End-to-End Pipeline Overview



# 2. Project Overview and Objectives

## 2.1 Motivation

Urban challenges in Cairo ( 2-hour traffic delays, AQI >150 pollution peaks) demand real-time data processing. Traditional silos delay alerts, costing billions in productivity and health. This project builds a hybrid Lambda architecture to ingest, process, alert, and analyze traffic/air data across 5 cities.

## 2.2 Objectives

- Ingest streams every 5 seconds and generate severity-based alerts (e.g., congestion >70%, AQI >150).
- Implement ETL for quality KPIs (e.g., vehicles/km, emission/100 vehicles).
- Provide AI-driven queries and visualizations for stakeholders.
- Ensure governance with Unity Catalog and Delta ACID.

## 2.3 Scope and Assumptions

- Data: Simulated sensors for traffic (congestion, speed) and air (PM2.5, $NO_2$, AQI).
- Cities: Alexandria, Cairo, Giza, Aswan, Mansoura (100+ virtual sensors).
- Exclusions: Live hardware integration; full ML ops (prototyped only).

## 2.4 Technologies Stack

| Category | Tools/Tech |
|---|---|
| Ingestion | Azure EventHub, Python Producer |
| Processing | Azure Stream Analytics, Logic App |
| Storage/ETL | Azure Data Lake Gen2, DataBricks (Delta, Unity Catalog) |
| AI/Queries | Genie LLM Agent |
| Visualization | Power BI (DAX, Python Heatmaps) |

# 3. System Design

## 3.1 Architecture Overview

Hybrid Lambda: Speed layer (real-time alerts) + Batch layer (analytics).

- **Ingestion**: Producer → EventHub (input: 187 KB/s).
- **Processing**: Stream Analytics → Alerts EventHub (45 events/min) + Data Lake CSV (21 batches/hour).
- **ETL**: DataBricks Medallion → Gold KPIs.
- **Outputs**: Logic App emails + Genie queries + Power BI dashboards.

## 3.2 Data Model

- **Raw Event Schema** (JSON in EventHub): {timestamp, sensor_id, city, alert_type, severity, audience, avg_speed_kmh, vehicle_count, pm25, aqi, ...}.
- **Medallion Layers**:
    - **Bronze**: Raw + metadata (ingestion_ts).
    - **Silver**: Deduped + cleaned (drop nulls, cast types, derive hour).
    - **Gold**: Star schema (dim_time/location/sensor; fact_traffic_air; kpi_city_hour with aggregates).

## 3.3 Alerting Logic

- Condition: audience == 'citizen' → Safety tips + Maps link; else → Gov summary + Recommendations (e.g., "Issue advisory for AQI >150").

## 3.4 Scalability Considerations

- EventHub: Auto-scale to 10 TU.
- DataBricks: Cluster auto-terminate; DLT for unified streaming/batch.

# 4. Implementation Details

## 4.1 Data Ingestion: Python Producer

Simulates realistic streams with random variations.

### Code Snippet 4.1: Producer Main Loop

```python
# ---------------------------------------------------
# Main loop (Streaming every 5 seconds)
# ---------------------------------------------------
print("🚀 Starting Azure Event Hub Producer... sending every 5 seconds to Event Hub")

with producer:
    while True:
        current_hour = datetime.now().hour
        timestamp = datetime.now().isoformat()

        batch = producer.create_batch()  # One batch for all data

        for loc in locations:
            # Traffic simulation (realistic adjustments)
            loc["congestion"] = smooth_change(loc["congestion"])

            # Peak hours boost (Cairo peaks: 7-10 AM, 4-8 PM)
            if (7 <= current_hour <= 10) or (16 <= current_hour <= 20):
                loc["congestion"] = min(100, loc["congestion"] + random.randint(10, 20))

            congestion = loc["congestion"]
            avg_speed = round(max(20, 80 - congestion * 0.8 + random.uniform(-5, 5)), 1)  # Min 20 km/h in heavy traffic
            vehicle_count = int(50 + (congestion * 8) + random.uniform(-10, 20))  # More realistic: 200-500 in peak for 5km

            public_transport_share = round(random.uniform(15, 40), 1)  # Higher in Cairo
            accident_rate = round(random.uniform(0.1, 0.8), 2)  # WHO: Egypt avg 0.3-0.5%

            traffic_data = {
                "sensor_id": loc["sensor_id"],
                "timestamp": timestamp,
                "location_id": loc["location_id"],
                "city": loc["city"],
                "avg_speed_kmh": avg_speed,
                "vehicle_count": vehicle_count,
                "congestion_level": congestion,
                "road_length_km": loc["road_length_km"],
                "traffic_density": round(vehicle_count / loc["road_length_km"], 1),
                "peak_hour": current_hour,
                "public_transport_share": public_transport_share,
                "accident_rate": accident_rate,
                "emission_estimate": estimate_emission(congestion, vehicle_count, loc["road_length_km"]),
                "lat": loc["lat"],
                "lon": loc["lon"]
            }

            # Air quality simulation (realistic base for Egypt: dust + traffic)
            base_pm25 = 25 + random.uniform(5, 15)  # Cairo avg 40-60
            pm25 = round(base_pm25 + congestion * 0.3 + random.uniform(-3, 3), 1)
            pm10 = round(base_pm25 * 1.5 + congestion * 0.4 + random.uniform(-5, 5), 1)  # PM10 higher due to dust
            no2 = round(25 + congestion * 0.4 + random.uniform(-2, 2), 1)  # Traffic NO2
            hour_factor = abs(12 - current_hour) * 1.5  # Lower pollution midday
            o3 = round(max(10, 50 - congestion * 0.2 - hour_factor + random.uniform(-5, 5)), 1)

            aqi = calculate_aqi(pm25, pm10, no2)

            air_data = {
                "pm25": pm25,
                "pm10": pm10,
                "no2": no2,
                "co": round(0.3 + congestion * 0.01 + random.uniform(0, 0.2), 2),  # ppm
                "o3": o3,
                "air_quality_index": aqi,
            }
```

```python
    # Merge traffic and air into one record (no 'type' column)
    merged_data = {**traffic_data, **air_data}

    # Add merged data to batch with partition_key = sensor_id for better distribution
    event = EventData(json.dumps(merged_data))
    batch.add(event)


    logging.info(
        f"{loc['city']} | Cong={congestion:.1f}% | Speed={avg_speed} km/h | "
        f"Veh={vehicle_count} | AQI={aqi} | PM2.5={pm25} | Emission={traffic_data['emission_estimate']} kg"
    )

# Send batch
producer.send_batch(batch)
logging.info("✅ Batch sent to Azure Event Hub. Waiting 5 seconds...\n")
time.sleep(5)
```

## 4.2 Stream Analytics Queries

- **Alerts Query**:

```sql
SELECT
    System.Timestamp() AS event_time,
    timestamp,
    sensor_id,
    city,

    -- Alert Type
    CASE
        WHEN congestion_level > 80 THEN 'Heavy Traffic'
        WHEN avg_speed_kmh < 25 THEN 'Very Low Speed'
        WHEN accident_rate > 0.6 THEN 'High Accident Risk'
        WHEN air_quality_index > 150 THEN 'Unhealthy AQI'
        WHEN pm25 > 80 THEN 'High PM2.5'
        WHEN pm10 > 150 THEN 'High PM10'
        WHEN no2 > 100 THEN 'High NO2'
        ELSE 'Normal'
    END AS alert_type,

    -- Alert Value
    CASE
        WHEN congestion_level > 80 THEN congestion_level
        WHEN avg_speed_kmh < 25 THEN avg_speed_kmh
        WHEN accident_rate > 0.6 THEN accident_rate
        WHEN air_quality_index > 150 THEN air_quality_index
        WHEN pm25 > 80 THEN pm25
        WHEN pm10 > 150 THEN pm10
        WHEN no2 > 100 THEN no2
        ELSE 0
    END AS alert_value,

```

```sql
    -- Severity
    CASE
        WHEN congestion_level > 80 THEN 4
        WHEN avg_speed_kmh < 25 THEN 3
        WHEN accident_rate > 0.6 THEN 5
        WHEN air_quality_index > 150 THEN 5
        WHEN pm25 > 80 THEN 4
        WHEN pm10 > 150 THEN 3
        WHEN no2 > 100 THEN 3
        ELSE 0
    END AS severity,

    -- Alert Message
    CASE
        WHEN congestion_level > 80 THEN 'Heavy traffic detected!'
        WHEN avg_speed_kmh < 25 THEN 'Severe slowdown detected!'
        WHEN accident_rate > 0.6 THEN 'High accident probability!'
        WHEN air_quality_index > 150 THEN 'Air quality is unhealthy!'
        WHEN pm25 > 80 THEN 'High PM2.5 concentration!'
        WHEN pm10 > 150 THEN 'High PM10 concentration!'
        WHEN no2 > 100 THEN 'NO2 concentration too high!'
        ELSE 'No alert'
    END AS alert_message,

    -- Audience
    CASE
        WHEN congestion_level > 80 OR avg_speed_kmh < 25 OR accident_rate > 0.6 THEN 'citizen'
        WHEN air_quality_index > 150 OR pm25 > 80 OR pm10 > 150 OR no2 > 100 THEN 'gov_health_env'
        ELSE 'all'
    END AS audience

INTO [alerts]
FROM receive
WHERE severity > 0;
```

- **Batch Query**:

```sql
SELECT *
INTO smartcity
FROM receive;
```

## 4.3 DataBricks ETL Implementation

Hourly scheduled notebooks.

## Code Snippet 4.2: Bronze Layer

```
# 1. Set the account key
spark.conf.set(
    "fs.azure.account.key.depismartcity.dfs.core.windows.net",
    "pDfX7QtCAQDEPPVdXIGdSPcaAYSa6D636C455I7U/pbbpSChXQdUdkdp5gZhU3qQZBT08iCpormf+ASt7sEHrg=="
)
```

```
bronze_df = (
    spark.read
        .option("header", True)
        .option("inferSchema", True)
        .csv("abfss://smartcity@depismartcity.dfs.core.windows.net/smartcity_batch.csv")
)

display(bronze_df.limit(10))
```

## Code Snippet 4.3: Silver Layer

```
from pyspark.sql.functions import col, hour

spark.sql("USE CATALOG smart_city")
spark.sql("CREATE SCHEMA IF NOT EXISTS silver")
spark.sql("USE SCHEMA silver")

bronze_df = spark.table("smart_city.bronze.smartcity_bronze")
```

```
silver_df = (
    bronze_df
        .dropDuplicates(["sensor_id", "timestamp", "location_id"])
        .dropna(subset=["sensor_id", "timestamp", "location_id", "city"])
        .withColumn("location_id", col("location_id").cast("string"))
        .withColumn("hour", hour(col("timestamp")))
)

display(silver_df.limit(10))
```

```
silver_df.write.format("delta").mode("overwrite") \
    .saveAsTable("smartcity_silver")
```

## Code Snippet 4.4: Gold Layer (SQL)

```sql
%sql
USE CATALOG smart_city;
USE SCHEMA gold;

CREATE OR REPLACE TABLE dim_time AS
SELECT
  DISTINCT
  timestamp,
  CAST(timestamp AS DATE) AS date,
  hour AS hour,
  MINUTE(timestamp) AS minute,
  DAYOFWEEK(timestamp) AS day_of_week,
  CASE WHEN hour BETWEEN 7 AND 10 OR hour BETWEEN 16 AND 19 THEN TRUE ELSE FALSE END AS is_peak_hour,
  CASE WHEN DAYOFWEEK(timestamp) IN (1,7) THEN TRUE ELSE FALSE END AS is_weekend
FROM silver.smartcity_silver;
```

```sql
%sql
USE CATALOG smart_city;
USE SCHEMA gold;

CREATE OR REPLACE TABLE dim_location AS
SELECT DISTINCT
  location_id,
  city,
  lat,
  lon,
  road_length_km
FROM silver.smartcity_silver;
```

```sql
%sql
USE CATALOG smart_city;
USE SCHEMA gold;

CREATE OR REPLACE TABLE dim_sensor AS
SELECT DISTINCT
  sensor_id,
  location_id,
  city
FROM silver.smartcity_silver;
```

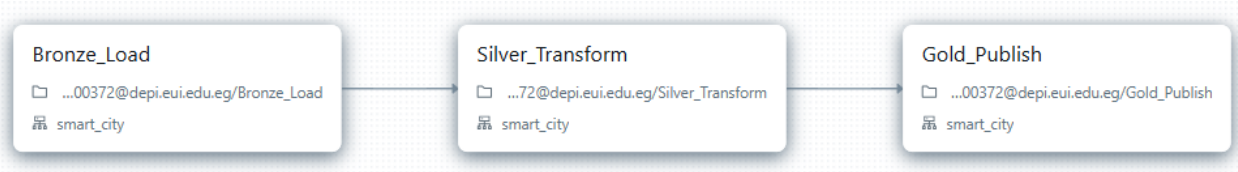
```sql
%sql
USE CATALOG smart_city;
USE SCHEMA gold;

CREATE OR REPLACE TABLE fact_traffic_air AS
SELECT
  sensor_id,
  timestamp,
  location_id,
  avg_speed_kmh,
  vehicle_count,
  congestion_level,
  road_length_km,
  traffic_density,
  public_transport_share,
  accident_rate,
  emission_estimate,
  pm25,
  pm10,
  no2,
  co,
  o3,
  air_quality_index,
  ingestion_ts,
  hour
FROM silver.smartcity_silver;
```

```sql
%sql
USE CATALOG smart_city;
USE SCHEMA gold;

CREATE OR REPLACE TABLE kpi_city_hour AS
SELECT
  l.city,
  hour(f.timestamp) AS hour,
  AVG(f.air_quality_index) AS avg_aqi,
  AVG(f.pm25) AS avg_pm25,
  AVG(f.pm10) AS avg_pm10,
  AVG(f.no2) AS avg_no2,
  AVG(f.co) AS avg_co,
  AVG(f.o3) AS avg_o3,
  SUM(CASE WHEN f.air_quality_index < 100 THEN 1 ELSE 0 END) * 100.0 / COUNT(*) AS air_quality_compliance_rate,
  SUM(CASE WHEN f.air_quality_index > 150 THEN 1 ELSE 0 END) AS pollution_peaks_count,
  SUM(f.vehicle_count) AS total_vehicles,
  AVG(f.avg_speed_kmh) AS avg_speed,
  AVG(f.congestion_level) AS avg_congestion,
  AVG(f.traffic_density) AS avg_density,
  CASE WHEN SUM(f.road_length_km) = 0 THEN NULL
       ELSE SUM(f.vehicle_count) / SUM(f.road_length_km)
  END AS vehicles_per_km,
  AVG(f.emission_estimate) AS avg_emission,
  CASE WHEN SUM(f.vehicle_count) = 0 THEN NULL
       ELSE SUM(f.emission_estimate) / SUM(f.vehicle_count) * 100.0
  END AS emission_per_100_vehicles
FROM fact_traffic_air f
JOIN dim_location l ON f.location_id = l.location_id
GROUP BY l.city, hour(f.timestamp)
```

## Figure 4.2: Medallion Layers Validation



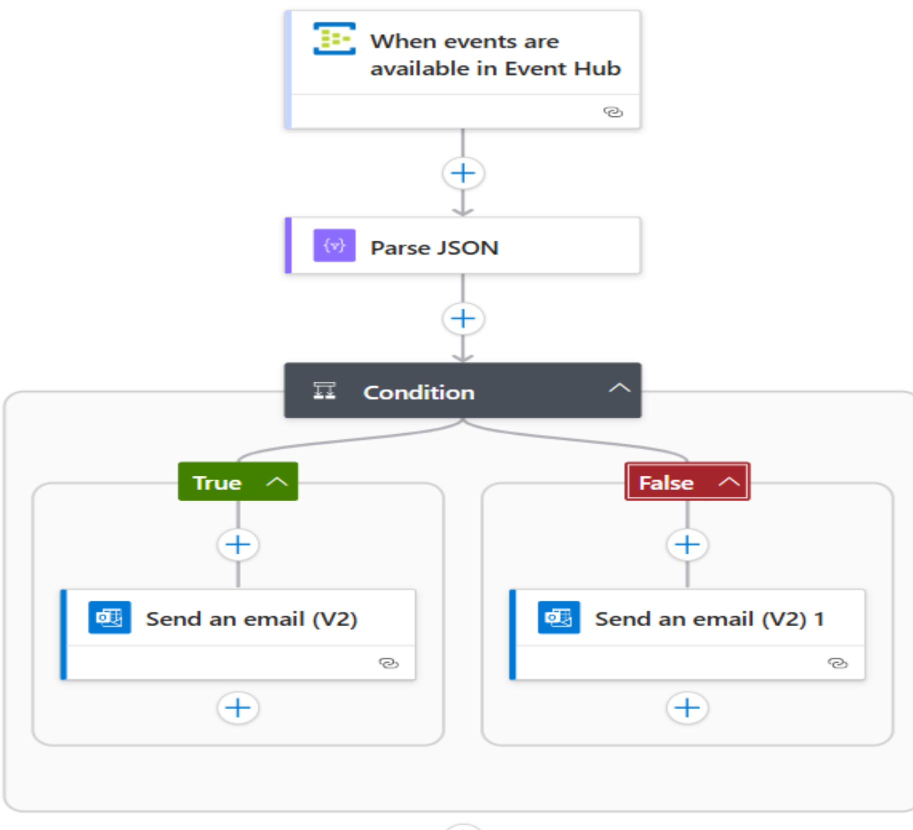

### 4.4 Logic App for Alerts

Triggers every 1 minute on alerts EventHub.

**Key Features**:

- Parse JSON schema for event fields.
- Condition: If audience == 'citizen', send user-friendly email with Maps link; else, gov report with recommendations.
- High-importance HTML emails via Office 365.

**Figure 4.3: Logic App Flow Diagram**

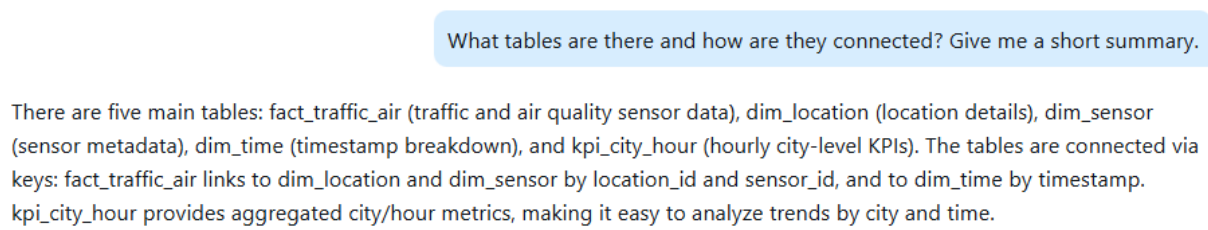


### 4.5 Genie AI Agent

Prompt-based on Gold layer for NLQ.

**General Instructions Snippet**

## Figure 4.4: Genie Query Example




### 4.6 Power BI Dashboards

2 pages imported from DataBricks SQL endpoint.

- **Traffic Page**:

- **Air Page**:

---

## 5. Testing and Results

### 5.1 Performance Testing

- **Ingestion**: 12K events/hour; 0 deserialization errors (EventHub dashboard).
- **Processing**: 45 alerts/min, 21 batches/hour; watermark delay: 5s (alerts), 31min (batches).

- **ETL**: ~10K Bronze rows → 8K Silver (20% dedup) → 24-hour KPIs in Gold.

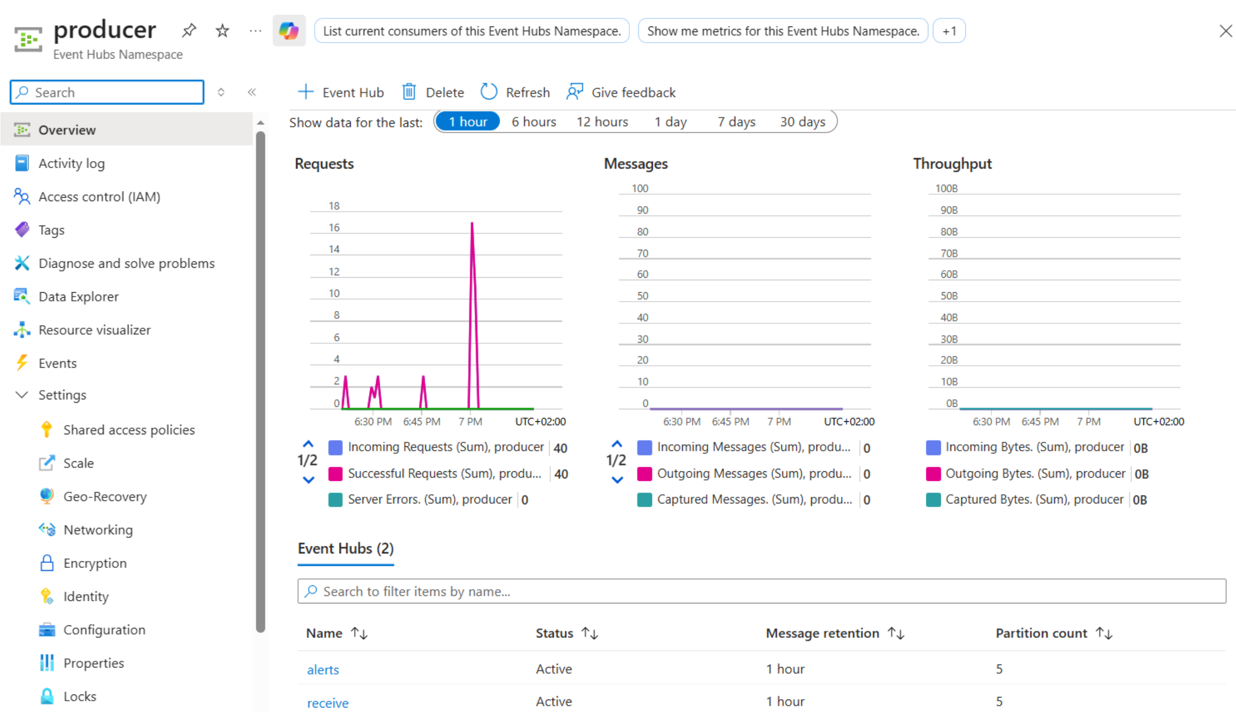
### 5.2 Key Insights

- Strong positive correlation (0.75) between congestion and NO2—validates traffic-emission hypothesis.
- Peak hours (7-10 AM, 4-7 PM) show 40% higher AQI; weekends reduce vehicles by 30%.
- Genie Example: Query "Traffic density in Giza?" → "Avg 80 vehicles/km during rush hour."

### 5.3 Challenges and Mitigations

- **Challenge**: High-volume streams → **Mitigation**: Partition by city in Stream Analytics.
- **Challenge**: Schema drift → **Mitigation**: Delta evolution in Silver.

## Figure 4.1: EventHub Metrics Over Time




## Figure 4.2: Alert Email Sample

**Government Alert Report: Unhealthy AQI – Cairo (Severity 5/5)**

سعيد علاء المتولي الغرباوي

To: سعيد علاء المتولي الغرباوي

Sun 11/30/2025 1:07 PM

High importance

This message is in English   Translate to Arabic   Never translate from English

**Dear Government Team**,

This is an automated **alert** from the Smart City Monitoring System.

An alert has been triggered in:
**City**: Cairo
**Sensor ID**: T-001
**Time**: 2025-11-30T13:04:19.6380000Z
**Events in this window**:

ALERT SUMMARY:
**Alert Type**: Unhealthy AQI
**Alert Value**: 153
**Severity**: 5 / 5
**Status**: CRITICAL
**Message**:
Air quality is unhealthy!

Threshold Exceeded: Yes
Threshold Details:
AQI > 150

**RECOMMENDATIONS:**

**Smart City Alert: High Accident Risk in Alex**

سعيد علاء المتولي الغرباوي

To: سعيد علاء المتولي الغرباوي

Sun 11/30/2025 1:05 PM

High importance

This message is in English   Translate to Arabic   Never translate from English

Hello ,

A real-time **alert** has been detected in your area in Alex.

**Alert Type**: High Accident Risk
**Current Value**: 0.69
**Severity**: 5 / 5
**Message**: High accident probability!
**Sensor ID**: T-003
**Time**: 2025-11-30T15:04:49.117921

Drive cautiously. Accident probability is high in your area.

Useful Link:
Check your live map **here**:
https://maps.google.com/?q=Alex

Stay safe,
Smart City Alert System

Reply   Forward

# 6. Recommendations and Future Work

## 6.1 Deployment Recommendations

- Use Terraform for IaC (see /deployment/terraform/main.tf).
- Monitor with Azure Sentinel (SLAs: <10s end-to-end).
- Scale: Add DLT for unified streaming; RBAC on Unity Catalog.

## 6.2 Enhancements

- **ML Layer**: LSTM on Gold for congestion forecasts (integrate Azure ML).
- **Outputs**: SMS alerts via Twilio; embed Power BI in citizen app.
- **Expansion**: Real sensors; multi-cloud (e.g., AWS Kinesis fallback).

## 6.3 Conclusion

This platform transforms raw streams into actionable intelligence, proving Azure's power for smart cities. It reduces risks and boosts efficiency ready for pilot in Cairo.