CASE STUDY 7

Title: Telecom Network Quality and Customer Experience Analytics using PySpark

A telecom operator collects daily network performance logs from mobile towers.
They want to reduce call drops, identify weak coverage areas, and improve customer experience.

They export one CSV file per day:

```
network_logs.csv
```

Columns:

```
event_id
subscriber_id
tower_id
city
network_type
signal_strength
download_speed_mbps
upload_speed_mbps
latency_ms
call_drop
event_time
device_type
```

Where:

- network_type = 4G, 5G, 3G
- signal_strength is in dBm and arrives as string
- speeds and latency arrive as strings and contain invalid values
- call_drop can be YES or NO
- event_time has mixed formats
- Some devices produce duplicate logs
- Some towers are consistently underperforming

Your job is to build a **Network Health & Customer Experience Pipeline** using PySpark.

Only one input is assumed:

```
network_logs.csv
```

PHASE 1 – Ingestion

1. Read network_logs.csv as all StringType.
2. Print schema and row count.
3. Show sample rows.

PHASE 2 – Cleaning

1. Trim string columns.
2. Normalize:
    - city
    - network_type
    - device_type
    - call_drop
3. Clean numeric fields safely:
    - signal_strength → IntegerType
    - download_speed_mbps → DoubleType
    - upload_speed_mbps → DoubleType
    - latency_ms → IntegerType

Invalid values must become null, not crash the job.

4. Parse event_time into:

```
event_time_clean (TimestampType)
```

Support multiple formats:

- yyyy-MM-dd HH:mm:ss
- dd/MM/yyyy HH:mm:ss
- yyyy/MM/dd HH:mm:ss

PHASE 3 – Validation

1. Count invalid values for each numeric field.
2. Count invalid timestamps.
3. Remove duplicate logs based on:

```
event_id
```

PHASE 4 – Network Performance KPIs

Compute:

1. Average download speed per city.
2. Average latency per city.
3. Call drop rate per city.
4. Call drop rate per tower.
5. Identify top 10 worst towers by:
    - highest call drop rate
    - highest latency
    - lowest download speed

PHASE 5 – Customer Experience Analysis

Compute for each subscriber_id:

1. Number of events recorded.
2. Average download speed.
3. Average latency.
4. Call drop count.
5. Identify subscribers with poor experience.

Define “poor experience” logic such as:

- call_drop_count high
- avg_download_speed low
- avg_latency high

PHASE 6 – Window Functions

1. Rank towers within each city by call drop rate.
2. Rank subscribers within each city by worst experience.
3. Use lag() to detect sudden deterioration in signal_strength for a tower.

PHASE 7 – Anomaly Detection

Detect towers where:

- latency suddenly spikes
- download speed suddenly drops

- call drops spike

Use window functions with:

- lag
- rolling averages
