CASE STUDY 6

Title: Smart Hospital Patient Flow & Resource Utilization Analytics using PySpark

A multi-specialty hospital chain wants to optimize patient waiting time, doctor utilization, and bed occupancy.
They export their operational data daily as:

```
hospital_visits.csv
```

Columns:

```
visit_id
patient_id
doctor_id
department
hospital_city
checkin_time
consult_start_time
consult_end_time
discharge_time
visit_status
payment_amount
payment_mode
insurance_provider
rating
```

Where:

- visit_status = REGISTERED, IN_CONSULTATION, DISCHARGED, CANCELLED
- payment_amount is string (may contain commas, empty, invalid)
- Times are in mixed formats
- Some visits are DISCHARGED without consult_end_time
- Some visits have consult_start_time earlier than checkin_time
- Some visits have discharge_time earlier than consult_end_time
- rating exists only for DISCHARGED visits

This is real hospital operational data chaos.

Your job is to build a **Patient Flow Optimization & Revenue Integrity Pipeline** using PySpark.

PHASE 1 – Ingestion

1. Read hospital_visits.csv as all StringType.
2. Print schema and total row count.
3. Display sample rows.
4. Identify visible data quality issues.

PHASE 2 – Cleaning

1. Trim all string columns.

2. Clean payment_amount:

    - Remove commas
    - Convert to IntegerType
    - Replace invalid and empty values with null

3. Parse time columns into:

```
checkin_time_clean
consult_start_time_clean
consult_end_time_clean
discharge_time_clean
```

Support formats:

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

Keep original time columns for audit.

4. Create validity flags:

```
checkin_valid
consult_start_valid
consult_end_valid
discharge_valid
```

PHASE 3 – Derived Time Metrics

Compute:

```
waiting_time = consult_start_time - checkin_time
consultation_time = consult_end_time - consult_start_time
```

```
total_visit_time = discharge_time - checkin_time
```

In minutes.

Negative or null values must be flagged.

PHASE 4 – Data Quality Audit

1. Count visits where:
    - consult_start_time < checkin_time
    - consult_end_time < consult_start_time
    - discharge_time < consult_end_time
2. Count DISCHARGED visits with missing consult_end_time.
3. Count invalid payment_amount.
4. Count CANCELLED visits.

Deliverable: Patient Flow Data Quality Report.

PHASE 5 – Operational Analytics

1. Average waiting time per department.
2. Average consultation time per doctor.
3. Departments with longest patient queues.
4. Cities with longest discharge delays.
5. Cancellation rate per department.

PHASE 6 – Window Functions

1. Rank doctors by:
    - Shortest consultation time
    - Highest patient ratings
2. Rank departments by:
    - Average waiting time
    - Cancellation rate
3. Identify:
    - Top 3 efficient doctors per department

- Bottom 3 overloaded departments

PHASE 7 – Revenue Analytics

1. Total revenue per department.
2. Revenue per insurance provider.
3. Average bill amount per city.
4. Compare insured vs non-insured revenue.

PHASE 8 – Performance Engineering

1. Cache cleaned dataset.
2. Use explain(True) on:
    - Doctor ranking query
    - Department waiting-time query
3. Identify shuffle stages.
4. Repartition by department.
5. Compare execution plans before and after.

PHASE 9 – RDD

1. Convert DISCHARGED visits to RDD.
2. Compute:
    - Total hospital revenue using reduce.
    - Visit count per city using map-reduce.
3. Explain why DataFrames are superior for healthcare analytics.
