Capstone Project: Real-Time Payments Data Platform

Objective

In 2 weeks, you will design and implement a **production-like financial data pipeline**. It must handle **real-time payments**, detect **fraud/anomalies**, store data in a **Bronze–Silver–Gold architecture**, and produce **daily settlement reports**.

You must use **all tools learned**:
Python · SQL · Data Modeling · dbt · Kafka · Airflow · Spark · Docker · Linux shell scripting

End-to-End Flow

1. **Simulated Payments Producer (Python + Kafka)**
    - Write a Python script to generate **JSON payment events**.
    - Publish them into Kafka topic payments.raw.
    - Example fields:

```
{
  "transaction_id": "TXN123456789",
  "ts_event": "2025-09-24T09:45:31Z",
  "card_hash": "a8c9f09d11ffcc45",
  "merchant_id": "M98765",
  "amount": 249.75,
  "currency": "USD",
  "mcc": "5411",
  "channel": "POS",
  "auth_result": "APPROVED",
  "location": "New York, USA"
}
```

2. **Kafka Setup**
    - Run Kafka with Docker Compose.
    - Create topics:

        - payments.raw → for valid incoming transactions.
        - payments.deadletter → for invalid or rejected transactions.

3. **Spark Streaming Job (Python + Spark)**
    - Read from payments.raw.
    - Validate schema and apply **fraud rules** (e.g., high amount, velocity, blacklisted merchant).
    - Write valid events to **Bronze layer** (Parquet, partitioned by date).
    - Route flagged/invalid events to payments.deadletter.
    - Use checkpoints for recovery.
4. **Bronze → Silver Job (Spark batch)**
    - Run hourly via Airflow.
    - Deduplicate on transaction_id.
    - Standardize data types (cast amounts, enforce ISO currency).
    - Clean invalid fields, add processing timestamp.
    - Write out **Silver Parquet tables**.
5. **Silver → Gold Load (Spark + Postgres)**
    - Run daily via Airflow.
    - Load clean Silver data into **Postgres warehouse**.
    - Build a **star schema**:
        - Dimensions: dim_date, dim_card, dim_merchant
        - Facts: fact_transactions, fact_settlement_daily, fact_fraud_signals
6. **dbt Models and Tests**
    - Create dbt project on top of Postgres.
    - Write models for facts and dimensions.
    - Add dbt tests:
        - not_null on IDs
        - unique on primary keys
        - relationships between facts and dims
        - Custom: amounts ≥ 0, valid currency codes

    - Generate dbt docs.
7. **Airflow Orchestration**
    - DAG 1 (hourly): Bronze → Silver
    - DAG 2 (daily): Silver → Gold → dbt → Export daily CSV
    - Configure retries, SLAs, failure alerts.
8. **Daily Outputs**
    - Settlement report CSV: sums per merchant, currency, and fees.
    - Fraud alerts report: flagged transactions.
9. **Documentation**
    - **Architecture diagram** showing Kafka → Spark → Bronze → Silver → Postgres → dbt → Reports.
    - **Runbook**: how to start services, backfill data, and recover from failure.
    - Explain fraud rules clearly. Sample Rules
      Sample Rules:
        - **High Amount Rule**
            - Flag if amount > 10,000 for any merchant.
        - **Velocity Rule**
            - Flag if the same card_hash has more than 5 transactions in 1 minute.
        - **Cross-border Rule**
            - If the same card_hash is used in two different countries within 10 minutes.
        - **Blacklisted Merchant Rule**
            - Flag if merchant_id is in a blacklist table.
        - **Decline Rate Rule**
            - If a card has > 50% declined transactions in the last 10 attempts.