DPDzero Data Ops Assignment

Objective:

Build a simplified end-to-end data pipeline that mimics a real-world operational use case. Your goal is to fetch, clean, transform, and report daily collection call data with basic validations and metrics.

Scenario:

Your organization runs a daily call campaign for loan collections. You receive daily CSV dumps from multiple sources:

- Call Logs
- Agent Roster
- Disposition Summary

Mock Data Files:

1. call_logs.csv
   - Columns: call_id, agent_id, org_id, installment_id, status, duration, created_ts, call_date
   - Each row represents a single call attempt with status and duration logged.
   - Includes randomized timestamps and call statuses (completed, connected, failed, etc.).

2. agent_roster.csv
   - Columns: agent_id, users_first_name, users_last_name, users_office_location, org_id
   - Contains static agent-level metadata like names, office location, and org ID.

3. disposition_summary.csv
   - Columns: agent_id, org_id, call_date, login_time
   - Represents agent logins (presence) on a given date with optional login time.

Tasks:

1. Data Ingestion and Validation:

- Read all 3 files into pandas.
- Ensure call_date, agent_id, and org_id are present and correctly formatted.
- Flag missing or duplicate entries.

2. Join Logic:

- Merge the datasets using agent_id, org_id, and call_date.
- Ensure no data loss in joins; explain how you handled mismatches.

3. Feature Engineering:

- For each agent on each date, compute:
  * Total Calls Made
  * Unique Loans Contacted
  * Connect Rate = Completed Calls / Total Calls
  * Avg Call Duration (in minutes)
  * Presence (1 if login_time exists, else 0)

4. Output:

- Save the report as agent_performance_summary.csv
- Format a Slack-style summary message like:

  Agent Summary for 2025-04-28

  Top Performer: Ravi Sharma (98% connect rate)

  Total Active Agents: 45

  Average Duration: 6.5 min

Bonus (Optional):

- Add CLI args to accept file paths.

- Add logging (info, error).

- Use modular functions/classes.

Evaluation Criteria:

- Code cleanliness and modularity

- Correctness of joins and logic

- Efficiency and readability

- Data validation handling

- Reporting format clarity