Interview Assignment: Backend Developer (Python FastAPI)

Objective Evaluate the candidate's ability to validate data using FastAPI and implement cron jobs.

Database Schema Overview The database consists of a **fact table** storing advertising performance metrics and multiple **dimension tables** for metadata.

Fact Table: `fact_ad_metrics_daily`

This table contains daily aggregated advertising metrics.

- `date_id` (FK) - Links to `dim_date`
- `region_id` (FK) - Links to `dim_region`
- `age_id` (FK) - Links to `dim_age_group`
- `gender_id` (FK) - Links to `dim_gender`
- `platform_id` (FK) - Links to `dim_platform`
- `placement_id` (FK) - Links to `dim_placement`
- `device_type_id` (FK) - Links to `dim_device_type`
- `impressions` - Total ad impressions
- `clicks` - Number of clicks received
- `cost` - Total spend for the day
- `conversions` - Number of conversions
- `likes` - Total number of likes

Dimension Tables:

- `dim_date (date_id, date_value)` - Stores dates
- `dim_region (region_id, region_name)` - Stores region names
- `dim_age_group (age_id, age_range)` - Stores age brackets
- `dim_gender (gender_id, gender_name)` - Stores gender values
- `dim_platform (platform_id, platform_name)` - Stores ad platforms
- `dim_placement (placement_id, placement_name)` - Stores ad placements
- `dim_device_type (device_type_id, device_type_name)` - Stores device types

Assignment Tasks

Task 1: Data Validation using FastAPI

1. Implement a simple FastAPI service that includes:
    - **A GET endpoint** to retrieve ad metrics based on filters (date range, region, platform, etc.).
2. Use **Pydantic models** to validate incoming API requests.
3. Implement meaningful error handling to reject invalid data.

Task 2: Cron Job for Logging

1. Implement a cron job using `APScheduler` or `Celery` that runs every **6 hours** to:
    - Log a timestamp indicating the job ran successfully.
2. Ensure the cron job is **asynchronous and non-blocking** to prevent performance bottlenecks.

Expected Deliverables

1. A FastAPI project with:
    - Pydantic models for request validation.
    - A GET endpoint for data retrieval.
    - A cron job setup for periodic logging.
2. Code hosted in a public repository (GitHub/GitLab).

Evaluation Criteria

- Proper use of FastAPI and data validation.
- Implementation of an async cron job.
- Code structure and documentation.