Arlo Data Engineering Challenge

Overview

Welcome to the Arlo Data Engineering Challenge!

Arlo ingests high-volume medical and prescription claims every day, enriches them with clinical and network context, and then runs real-time models so our underwriting engine can price new groups within minutes. For this challenge you'll simulate a slice of that workflow using the publicly available **CMS DE-SynPUF** dataset (synthetic Medicare claims). See the Appendix below for instructions on how to download the dataset you will be using. Additionally, you can find the documentation for this dataset here.

Please timebox this challenge to 4-6 hours of focused work. You will continue working on this code in a subsequent, synchronous interview! We recognize the challenge is long and don't expect everyone to complete it.

The Challenge

Utilizing the above dataset, you must develop a solution that can answer the following questions at the **member/beneficiary and year** level:

- **Annual spend**

  Total allowed, total paid, inpatient stays, outpatient visits, RX fills for a given Beneficiary ID (desynpuf_id) & calendar year.

- **Top diagnoses**

  The 5 diagnoses that generated the highest paid dollars for that member-year.

- **Provider breadth**

  Count of distinct billing providers the member saw that year.

(Hints: Inpatient stays ≈ distinct `clm_id` rows in the IP file; outpatient visits ≈ distinct `clm_id` rows in the OP file; RX fills ≈ rows in the PDE file.)

(Note: These files use ICD9 codes instead of the modern ICD10 codes, make sure to take this into account when using online help.)

Roughly speaking, you will need to:

1. **Ingest**
    a. A job that converts the raw CSV to Delta/Parquet (Spark/Polars/dbt—any scalable tool is fine)
    b. Normalize column names and types
    c. Pick a partition strategy. Tell us why you chose those partition keys (e.g., write-conflict, query-skipping).
2. **Model**
    a. Design a schema that can answer the aforementioned questions without rescanning the raw files.
    b. How you model is up to you—dim/fact star schema, Data Vault, something else. Explain your reasoning.
4. **Incremental refresh**
    a. Describe how you would handle data updates. In other words, assume new claims keep rolling in for beneficiaries.
5. **(Optional) Serve**
    a. Expose a minimal FastAPI endpoint `GET /patient/{bene_id}?year=YYYY`
    b. The endpoint can return something as follows

```
{
  "bene_id": "200284",
  "year": 2009,
  "total_allowed": 13250.43,
  "total_paid": 11206.07,
  "inpatient_stays": 1,
```

```
  "outpatient_visits": 3,
  "rx_fills": 12,
  "unique_providers": 4,
  "top_diagnoses": [
    {"code": "250.00", "description": "Diabetes mellitus", "spend": 2423.50},
    ...
  ]
}
```

Deliverables

1. Git repo with runnable code (notebooks or `.py`)
2. **Short design brief (or Loom video, if easier)** covering:

- Storage & partition rationale (call out at least one alternative you considered)
- Incremental-load and schema-evolution strategy.
- One optimization you'd ship to manage 10× scale (cost or latency).

Appendix: Download Dataset

We'll work with **CMS DE-SynPUF** Samples 1 & 2 for the years 2008-2010 (≈1.2 GB zipped). Here is a GitHub Gist with a simple shell script that can simplify this. Alternatively, you may download them from here.