Data Engineering Assignment

Question 1: coding challenge

Write some ETL able to ingest events into a data lake in a format that is efficient for querying them by type of event and time. Consider that there are a few hundred types of events and that hour resolution is enough.

Incoming events are represented as JSON and different events follow different schemas with some commonalities. All events have a `data` and `meta` keys with at least:

- `data.id`: it's a unique event id
- `data.type`: type of the event. You will get all the `foo` events in `raw-event-foo`
- `data.attributes`: contains all the particular event fields
- `meta.created_at`: event time in epoch millis

Apart from these bare-minimum attributes, `data.attributes` can have more event-specific fields and `meta` can have more control attributes. For instance:

```
{
  "data": {
    "id": "0aecd029-1eae-42b1-9ef0-97173247c0e4",
    "type": "user_created",
    "attributes": {
      "user_id": "5db24a33-4e2f-4469-b627-7478497fc7af",
      "name": "John",
      "favourite_ice_cream": "vanilla"
    }
  },
  "meta": {
    "created_at": 1549650668000,
    "service": "service_a"
  }
}
```

If any of the required attributes is missing or malformed, the event should be discarded or stored separately as such.

The events can be read from a streaming source or, for simplicity, from files but we are interested in the considerations related to scale it (billions of daily events).

Question 2: case studies

Context: a business that has about 10M unique daily users whose activity produces 1 billion domain events per day and every event is 10KB of data. This business is cloud native and very much invested into the AWS platform, so all AWS products are also available to be used alongside open source technologies.

We want to build a system that allows us to **track** and **analyze** all the domain events associated with a user.

In the following scenarios you must design the architecture and describe the tools needed and make some assumptions to complement the requirements. You **must** consider cost-effectiveness of the proposed solution. Remember that all choices of tooling and technology and components should be properly reasoned.

scenario 1: design a system to ingest the domain events produced by the app and allow to perform analytical queries (OLAP load) on them.

scenario 2: scenario 1 is extended with the requirement of being able to expose all historical events for a given user through an API. This API will be used by internal teams and it has to support very few requests.

scenario 3: product guys like our new API and now they ask us if they can use it to render some statistics to every user in their mobile devices (remember, 10M daily users). Those statistics are still daily aggregates, for example, the number of products being contacted in the last N days, the number of times that user's products got favorited per day, number of products sold, etc.

scenario 4: we have a re-ranking ML model that has been trained to increase the relevance of products being searched. Apart from a set of static features like text in the description, the model requires some real-time statistics (impressions, likes, conversations in the last hour, day and week). How do you offer these features in real time to feed the model in production? How could you do it by doing the minimum set of changes on the design of previous scenarios?

Question 3: “a day of very low probability”

All the events of our data lake are stored in S3 partitioned by event type and time. Subsets of these data flow to different systems to perform different business functions.

However, one day, something very unlikely happens: most of AWS is down! Right now, the only service up & running is S3. No EC2, no EMR, no nothing!

Unfortunately, your CEO desperately needs some daily KPIs computed for a meeting of utmost importance.

You need to calculate those KPIs (remember that we have 1000M events per day) and the only local computing power you have is a couple laptops.

What would you do? **Be creative and show off!**