Junior Data Engineer Assessment

Context:

You've joined a mid-sized e-commerce platform that focuses on selling used smartphones. Your team is responsible for building data pipelines and ensuring that data from various sources (websites, internal systems, 3rd-party APIs) is ingested, transformed, and stored efficiently.

This case study consists of **two main tasks** designed to evaluate your practical and conceptual understanding of core data engineering topics, the **time estimate is 3 hours** so you can plan ahead for it:

- Part 1: **Web Scraping Task** (Coding)
- Part 2: **Data Pipeline Architecture** (Design)

Part 1: Web Scraping Task (Coding)

Objective:

Scrape pricing data for **used smartphones** (e.g., iPhone, Samsung Galaxy) from at least **two e-commerce websites** (e.g., mobile misr, dubizzle, Facebook Marketplace).

Requirements:

1. Write a Python script using requests, BeautifulSoup, agentic AI library, or Scrapy (or any library you choose) to:
    - Search for used phones (e.g., "iPhone 13", "Samsung S22").
    - Collect at least the following data for each listing:
        - Product name, Price, Seller or listing title, Listing URL, Location (if available)
    - Store the output in **CSV or JSON format**.
2. Ensure your script handles:
    - Pagination (if applicable)
    - Basic error handling and retries
    - Logging (simple print statements or logging module)
3. Document how to run the script and what sites were scraped.

Part 2: Data Streaming Pipeline + Warehouse Architecture (Design)

Objective:

Design a **data pipeline** that collects and processes data **in near-real-time** from multiple sources including:

- Web-scraped data (like the one above)
- Internal sales transactions
- User activity logs from the mobile app
- An AI-powered trade-in pricing engine
- Manual data collected from the market

All data must be ingested and stored in a **data warehouse** for analytics.

Requirements:

1. Streaming Architecture Diagram

Design a **data flow diagram** that includes:

- Sources
- Stream ingestion layer (e.g., Kafka, Kinesis)
- Transformation layer (e.g., Spark Streaming, Flink)
- Storage layer (Data Warehouse: e.g., Snowflake, Redshift, BigQuery)
- Orchestration tools (e.g., Airflow, DBT for transformation)

Use any diagram tool (draw.io, Lucidchart, Miro) or markdown if submitting as text.

2. Explain the Flow

Explain an overview of your architecture:

- Why each component was chosen
- How real-time and batch data are handled
- How data quality and schema evolution are managed
- Partitioning and performance considerations in the warehouse

3. Schema Design (Optional but Preferred)

Propose a simple star/snowflake schema for the warehouse including:

- Fact table (e.g., used_phone_sales)
- Dimension tables (product, location, source, etc.)

Evaluation Criteria

Area	Criteria
Web Scraping	Code readability, error handling, pagination handling, data completeness
Data Pipeline Design	Clarity of architecture, technology choices, scalability
Data Warehousing	Schema design, warehouse suitability, partitioning strategy
Communication	Clarity in the written explanation and diagram presentation

---

Submission Instructions

Please submit the following:

1. Python script + output file (CSV or JSON) in a GitHub repo or zip file.
2. A PDF or Markdown document with:
    - Your architecture diagram (image or link)
    - Your design explanation
    - Optional: schema design

Notes:

- If you're unsure which e-commerce sites to scrape, feel free to choose two publicly accessible ones.
- Don’t worry about legal limitations; you are not actually deploying this scraper in production.
- Keep cloud tools realistic but don't feel limited to using one cloud provider.
- This is an opportunity to demonstrate your skills and style of thinking you are not asked to do a production-ready nor a very detailed output.