Data Engineering Take-Home Exercise

PriceHubble helps B2B clients make smart real estate decisions, such as evaluating property values, tracking market trends, spotting investment opportunities, and improving property management.

As a Senior Data Engineer in the Property Intelligence Tribe, you'll have a big impact by gathering and combining property listing and transaction data from different sources. You'll ensure data quality and enhance it with details like location and market trends to give a complete view of the residential property market. This data helps our clients with property valuation decisions and building trusted Automated Valuation Models (AVMs) for assessing property values.

This take-home project is designed to give you a feel for the kind of work you'd be doing at PriceHubble. We hope you enjoy it!

Objective:
Your task is to build a data pipeline using the provided JSONL file. You'll process the data and insert it into a DuckDB table. You can choose any tech stack, as long as the code is reproducible and can be run by the reviewers.

Requirements:

Programming Language:

- The code should be written in Python.

Input:

- You will receive a JSONL file with raw data (***scraping_data.jsonl***).
- The input file contains one property offer per row, in JSON format. Each row has the following columns:

Column	Type	Description
id	string	Unique ID of the property
raw_price	string	Price info (e.g., “530 000€/mo.”)
living_area	float	The area of the property in square meters

property_type	string	Type of property (e.g., house, studio)
municipality	string	City or town where the property is located
scraping_date	string	Date the data was scraped (YYYY-MM-DD)

Example input:

```Python
{
   "id": "0000a4fb",
   "raw_price": "530 000€/mo.",
   "living_area": 84.0,
   "property_type": "apartment",
   "municipality": "Solothurn",
   "scraping_date": "2021-02-17"
}
```

DuckDB Table:

- Create a DuckDB table matching the output data structure.
- Ensure the table is populated correctly after running the pipeline.

The output should match the following structure:

Column	Type	Nullable	Description
id	string	No	Property ID
scraping_date	string	No	Date when the data was scraped
property_type	string	No	Type of property
municipality	string	No	Municipality of the property
price	float	No	Converted price in numeric format

living_area	float	No	Area of the property
price_per_square_meter	float	No	Price per square meter

Example output:

```Python
{
    "id": "0000a4fb",
    "scraping_date": "2021-02-17",
    "property_type": "apartment",
    "municipality": "Solothurn",
    "price": 530000.0,
    "living_area": 84.0,
    "price_per_square_meter": 6309.52
}
```

Filtering Criteria:

- Only include rows where:
  - price_per_square_meter is between 500 and 15,000.
  - property_type is either “apartment” or “house”.
  - scraping_date is after March 5, 2020.

Orchestration:

- Use any orchestration tool of your choice (e.g., Airflow, Prefect, Dagster, Meltano).

Reproducibility:

- Ensure the pipeline can be easily run by the reviewers.
- Use Docker (or an alternative solution) to set up a reproducible environment.
- Provide clear instructions on setting up and running the pipeline, including dependencies.

Deliverables:

Code:

- Complete Python code for the data pipeline, including extraction, loading, and transformation steps.
- Any necessary setup scripts (e.g., Dockerfiles, configuration files).
- Orchestration files (e.g., DAGs or Prefect flows).

Documentation:

- A README with clear instructions on setting up and running the pipeline:
    - Prerequisites (e.g., Docker, libraries, tools).
    - How to run the pipeline manually or through an orchestrator.
    - Example outputs or logs from the pipeline.

Evaluation Criteria:

1. **Code Quality:**
    - Clear, well-structured Python code following best practices.
2. **Reproducibility:**
    - The pipeline should be easy to set up and run without complex configurations.
3. **Scalability:**
    - The solution should handle large datasets effectively.
4. **Documentation:**
    - Instructions should be clear and easy to follow.

Additional Info:

- You’re free to use any libraries or tools you're comfortable with.
- Docker is recommended to ensure the environment is reproducible.

Time Limit:

- Expect to spend 4 to 6 hours. Focus on simplicity and clarity.