Historic Hospital Data Collection Pipeline

Overview

This pipeline processes raw hospital data to produce a consolidated and cleaned dataset of historic hospitals. The final output is stored in `/output/processed_data/processed_hospitals_combined.csv`. The pipeline is designed to standardize, enrich, and validate hospital data for historical research purposes.

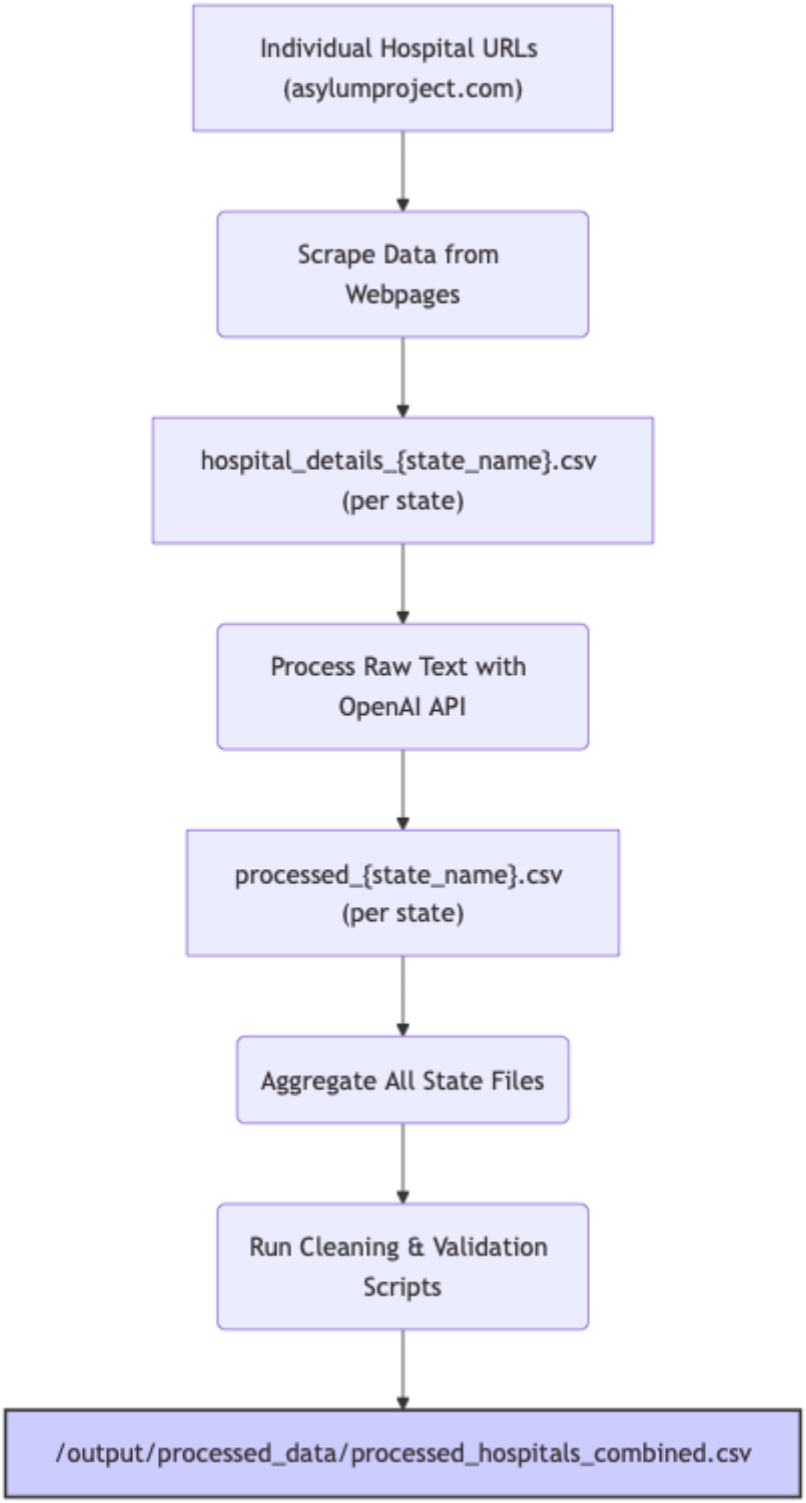
Final Output Data Dictionary

Below is the data dictionary for the final output file, `/output/processed_data/processed_hospitals_combined.csv`:

Column Name	Description
`state`	The U.S. state where the hospital is located.
`city`	The city where the hospital is located.
`hospital_name`	The primary name of the hospital.
`alt_name1` to `alt_name5`	Alternative names or aliases for the hospital.
`hospital_type`	The type of hospital (e.g., State Hospitals, Reform Schools).
`url`	URL to the hospital's historical page or reference.
`current_status`	The current operational status of the hospital (e.g., Active, Closed).
`building_style`	The architectural style of the hospital's building(s).
`architecture_style`	Specific architectural details, if available.
`final_year_opened`	The year the hospital first opened.
`final_year_closed`	The year the hospital closed, if applicable.
`final_hospital_age`	The total operational age of the hospital in years.
`final_number_of_beds`	The maximum number of beds recorded for the hospital.
`final_number_of_patients`	The maximum number of patients recorded for the hospital.
`incomplete_page_flag`	Flag indicating incomplete or ambiguous data (1 = incomplete, 0 = complete).
`kirkbride_flag`	Flag indicating if the hospital follows the Kirkbride architectural plan.
`hand_check_flag`	Flag indicating if the record requires manual verification.
---	---

Process Flow Diagrams

Diagram 1: Overview of the Process Flow



Core Business Logic

The pipeline processes hospital data using the following core logic:

1. **Data Cleaning**:

- Remove duplicate entries based on key identifiers (e.g., `hospital_name`, `state`, `city`).
- Standardize missing values and ensure consistent formatting across columns.

2. **Data Enrichment**:
   - Derive additional attributes such as `final_hospital_age`, `final_number_of_beds`, and `final_number_of_patients` based on available data.
   - Identify and flag incomplete or ambiguous records using the `incomplete_page_flag`.
3. **Data Validation**:
   - Validate URLs to ensure they are accessible and correctly formatted.
   - Cross-check `current_status` and `final_year_closed` for logical consistency (e.g., a hospital marked as "Active" should not have a `final_year_closed`).
4. **Final Variable Calculation**:

   The following functions are used to calculate the final variables. Each function processes specific columns in the dataset to derive the most accurate and reliable values for the final output.

   - **`final_year_opened`**:
     - **Logic**: The function `get_year_opened(row)` iterates through a prioritized list of columns (`['Opened', 'year_opened_LLM', 'Established', 'Construction Began']`) to find the earliest known year the hospital was operational. It ensures the year is within a valid range (1700–2025) to filter out erroneous data.
     - **Impact**: This ensures that the `final_year_opened` reflects the most reliable and earliest available data about when the hospital began operations.

```
def get_year_opened(row):
    for col in ['Opened', 'year_opened_LLM', 'Established',
'Construction Began']:
        if pd.notnull(row[col]) and 1700 <= row[col] <=
2025:
            return row[col]
    return np.nan
```

   - **`final_year_closed`**:
     - **Logic**: The function `get_year_closed(row)` checks columns like `['Closed', 'year_closed_LLM']` to determine the latest known year the hospital ceased operations. It validates the year to ensure it falls within the range of 1700–2025.
     - **Impact**: This ensures that the `final_year_closed` accurately represents the year the hospital stopped functioning, or remains blank if the hospital is still active.

```
def get_year_closed(row):
    for col in ['Closed', 'year_closed_LLM']:
```

```
            if pd.notnull(row[col]) and 1700 <= row[col] <=
2025:
                return row[col]
        return np.nan
```

- `final_hospital_age`:
  - **Logic**: This is calculated as:
    - `final_year_closed - final_year_opened` if the hospital is closed.
    - `current_year - final_year_opened` if the hospital is still active.
  - **Impact**: This provides a clear measure of the hospital's operational lifespan, whether it is still active or has been closed.
- `final_number_of_beds`:
  - **Logic**: The function `get_number_of_beds(row)` uses the column `number_of_beds_LLM` to determine the maximum number of beds recorded for the hospital. It ensures the value is valid (greater than 0) before returning it.
  - **Impact**: This ensures that `final_number_of_beds` reflects the most reliable data about the hospital's capacity.

```
def get_number_of_beds(row):
    return row['number_of_beds_LLM'] if
pd.notnull(row['number_of_beds_LLM']) and
row['number_of_beds_LLM'] > 0 else np.nan
```

- `final_number_of_patients`:
  - **Logic**: The function `get_number_of_patients(row)` iterates through a prioritized list of columns (`['number_of_patients_LLM', 'peak_patient_population_LLM', 'Peak Patient Population']`) to find the maximum number of patients recorded for the hospital. It ensures the value is valid (greater than 0) before returning it.
  - **Impact**: This ensures that `final_number_of_patients` captures the peak patient population, providing insight into the hospital's historical usage.

```
def get_number_of_patients(row):
    for col in ['number_of_patients_LLM',
'peak_patient_population_LLM', 'Peak Patient Population']:
        if pd.notnull(row[col]) and row[col] > 0:
            return row[col]
    return np.nan
```

5. **Flagging**:

- `incomplete_page_flag`: Set to `1` if the hospital's data is incomplete or requires manual review.
- `kirkbride_flag`: Set to `1` if the hospital follows the Kirkbride architectural plan.
- `hand_check_flag`: Set to `1` if the record requires manual verification.

How to Use

1. Place raw hospital data files in the appropriate input directory.
2. Run the pipeline script to process the data.
3. Review the output file located at `/output/processed_data/processed_hospitals_combined.csv`.
4. Use the flags (`incomplete_page_flag`, `hand_check_flag`) to identify records requiring manual review.

Notes

- The pipeline assumes that all input data is in CSV format and adheres to a predefined schema.
- Any discrepancies or missing data should be flagged for manual review.
- For further customization or debugging, refer to the pipeline scripts in the `/scripts` directory.