CS 513: Theory & Practice of Data Cleaning

Final Project – Phase 2 Report

University of Illinois at Urbana-Champaign, Summer 2025

Team Information

• Team ID: 112
• Members:
• Ajithlal Parackel (ap91@illinois.edu)
• Austin Offenberger (austino3@illinois.edu)

Description of Data Cleaning Performed

Menu:

Description of Data Cleaning Performed

The data cleaning workflow consists of a series of transformations applied to multiple columns. These can be grouped into four main high-level steps:

1. **Trimming Whitespace:** The `value.trim()` operation was applied to 19 different columns to remove any leading or trailing whitespace.
2. **Standardizing Text:** The workflow standardized text in several columns.
   - **Replacing Whitespace:** The `grel:value.replace(/\s+/, '_')` operation was used to replace all whitespace with underscores.
   - **Removing Brackets and Punctuation:** The `grel:value.replace(/[\[\]]/, '')` and `grel:value.replace(/[\[\]";?()*!]/, '_')` operations were used to remove specific characters.
3. **Standardizing Case:** The `value.toTitlecase()` operation was applied to ensure consistent capitalization.
4. **Type and Format Conversion:**
   - **Converting to Number:** The `value.toNumber()` operation was applied to convert their data type to a number.

    - **Converting and Formatting Date:** The `value.toDate()` and `grel:toString(toDate(value), "yyyy-MM-dd")` operations were used to convert text to a standard date format.

5. **Removed rows** with place as blank or null.

6. **Cluster event and place column:** Cluster and merge similar column values for place and events(eg NY, New York, [New York] to New York) to improve the quality of search.

Rationale for each high-level data cleaning step

1. **Trimming Whitespace:** The presence of leading or trailing spaces can cause different entries to be treated as unique, even if the core content is identical. This could lead to inaccurate counts, failed joins, and inconsistent filtering. This step is a prerequisite for any use case that relies on consistent data values for accurate analysis or lookups.
2. **Standardizing Text:** These transformations were performed to make text data more uniform and easier to work with. This is particularly useful for use cases that involve searching, tagging, or grouping records based on these text fields.
3. **Standardizing Case:** Using `toTitlecase()` ensures that the values in columns like `name`, `sponsor`, and `location` are presented consistently. For example, "new york" and "New York" would be treated as separate entities without this step.
4. **Type and Format Conversion:**
    - **Number Conversion:** The conversion of `page_count`, `dish_count`, and `id` to a numeric type is essential. Without it, these values would be treated as text, making it impossible to perform mathematical operations, such as summing counts or calculating averages.
    - **Date Formatting:** Converting the `date` column to a standardized `yyyy-MM-dd` format ensures that all date entries are interpreted correctly. Inconsistent date formats can lead to significant errors in chronological sorting, time-series analysis, and filtering by date ranges.

5. Remove menus without place. Since our U1 is for finding the price change per location for a dish, it is necessary to have the place information.

6. Clustering and merging locations is critical for U1, as we want to avoid treating differently spelled versions of the same place as separate entities in our queries.

MenuItem:

Description of Data Cleaning Performed

1. **Whitespace Trimming:** The `value.trim()` operation was applied to a wide range of columns.

2. **Type Conversion:**
   - The `value.toNumber()` operation was used to convert the data in the `id`, `menu_page_id`, and `dish_id` columns from text to a numerical data type.
   - The `value.toDate()` operation was applied to the `created_at` and `updated_at` columns to convert them from text strings to date objects.
3. **Row Removal:** Rows with a blank `price` column were removed

Rationale for each high-level data cleaning step

1. **Trimming Whitespace:** The presence of leading or trailing spaces can cause different entries to be treated as unique, even if the core content is identical. This could lead to inaccurate counts, failed joins, and inconsistent filtering. This step is a prerequisite for any use case that relies on consistent data values for accurate analysis or lookups.
2. **Type Conversion:** This is a crucial step for data integrity and functionality.
   - **Number Conversion:** The conversion to a numeric type is essential. Without it, these values would be treated as text, making it impossible to perform mathematical operations, such as summing counts or calculating averages.
   - **Date Formatting:** Converting the `date` column to a standardized `yyyy-MM-dd` format ensures that all date entries are interpreted correctly. Inconsistent date formats can lead to significant errors in chronological sorting, time-series analysis, and filtering by date ranges.
3. **Row Removal:** The removal of rows with a blank `price` is a key step for ensuring data completeness and quality for a specific analytical purpose. Missing prices are unusable and can skew results/cause errors in calculations. Therefore, removing these incomplete records is a necessary step to produce a clean and reliable dataset for any price-related analysis.

MenuPage:

Description of Data Cleaning Performed:

1. **Remove rows with empty page number**.
2. **Text Transformation (Type Conversion):** The `value.toNumber()` operation was applied. This step converted the data in these columns from text strings to numerical data types.

Rationale for each high-level data cleaning step:

1. **Remove rows with empty page number**- Page rows with page number is confusing and make it hard to locate a dish in a menu.
2. **Type Conversion:** The conversion to a numeric type is essential. Without it, these values would be treated as text, making it impossible to perform mathematical operations, such as summing counts or calculating averages.

Dish:

Due to the size of the Dish Table, it was difficult to use OpenRefine; therefore, we opted to use Python for data cleaning.

Description of Data Cleaning Performed

1. **Normalization of Dish Names:** This process involves a series of string transformations on the `name` column, including converting to lowercase, removing specific punctuation and non-alphanumeric characters, and standardizing whitespace.
2. **Filtering of Empty/Null Values:** Rows with empty, null, or 'nan' values in the `name` column are removed from the dataset.
3. **Clustering**: The `name` column had K-Means clustering applied, clustering like named dishes.
4. **Removal of a Column:** The `description` column is explicitly dropped from the final DataFrame.

Rationale for each high-level data cleaning step

1. **Normalization of Dish Names:**
    a. **Trimming Whitespace:** The presence of leading or trailing spaces can cause different entries to be treated as unique, even if the core content is identical. This could lead to inaccurate counts, failed joins, and inconsistent filtering. This step is a prerequisite for any use case that relies on consistent data values for accurate analysis or lookups.
    b. **Standardizing Text:** These transformations were performed to make text data more uniform and easier to work with. This is particularly useful for use cases that involve searching, tagging, or grouping records based on these text fields.
    c. **Standardizing Case:** Converting to lowercase ensures that the values are presented consistently. For example, "new york" and "New York" would be treated as separate entities without this step.
2. **Filtering of Empty/Null Values:** This step is a fundamental aspect of data integrity. Rows with empty or null values in the `name` are unusable for the analysis of dish data. Retaining these rows could lead to errors in calculations, inaccurate cluster assignments, or failed processing in subsequent steps. Therefore, this step is required to ensure that the dataset is of high quality and that downstream operations can execute without errors.
3. **Clustering:** This step is done to group like-named dishes under a single name. This is done so that analysis on a specific dish can be done, regardless of how it was represented in the original dataset.
4. **Removal of a Column:** The `description` column is removed because all rows are empty, and to simplify the dataset

Document Data Quality Changes

Menu:

Column	Description	Number of Cells Changed
`id`, `page_count`, `dish_count`	**Type Conversion:** The number of cells that were originally stored as text but could be successfully converted to a number.	Id = ~17545 page_count =~17545 dish_count = ~17545
`date`	**Type & Format Conversion:** The number of cells that were not in the `yyyy-MM-dd` format.	date = ~586
All Columns	**Whitespace Trimming:** The number of cells that had leading or trailing whitespace	name = ~9 sponsor= ~14 event = ~3 place=~8 physical_description=~384 currency_symbol ~4 call_number=~9 notes=~125
`name`, `sponsor`, `event`, `venue`, `place`, `occasion`, `location`	**Text Standardization & Case:** The number of cells with multiple internal spaces, special characters, or	name = ~629 sponsor= ~8109 event=~7827 venue=~8109 place=~7337 occasion=~3752 location =~1270

	inconsistent casing.	
`event,place`	**Cluster event and place:** Clustering event and place together.	Event= ~5266 Place=~3808

Demonstrating Improved Data Quality

Integrity Constraints (ICs):

- **IC-M1 (Numeric Type):** The `id`, `page_count`, and `dish_count` columns must contain only numerical data.
- **IC-M2 (Date Format):** The `date` column must be a valid date formatted as `yyyy-MM-dd`.
- **IC-M3 (Text Consistency):** Columns such as `name`, `event`, and `place` must be consistently formatted (e.g., title-cased, no extra whitespace, no problematic special characters).
- **IC-M4 (Format):** All columns must be free of leading or trailing whitespace.

- **IC-M5** Place value should not be blank or empty.

IC Violation Report:

Integrity Constraint	Violations Before Cleaning	Violations After Cleaning	Difference
IC-M1: Numeric data	0	0	*[Equal to the "Before" count]*
IC-M2: Date Formatting	586	477	111
IC-M3: Text Consistency	1175	921	254
IC-M4: Whitespace Issues	25	0	25

IC-M5: Place is null	9422	0	9422

MenuItem:

Quantifying the results of your efforts

Column(s)	Type of Change	Number of Cells Changed (Conceptual)	Number of cells changed(count)
`id`, `menu_page_id`, `price`, `high_price`, `dish_id`, `created_at`, `updated_at`, `xpos`, `ypos`	**Whitespace Trimming**	The number of cells that had leading or trailing whitespace.	0
`id`, `menu_page_id`, `dish_id`, `price`	**Type Conversion to Number**	The number of cells that were originally stored as text but were successfully converted to a numerical type.	Id= ~1332726 menu_page_id= ~1332726 dish_id =~1332726 price= ~1332726
`created_at`, `updated_at`	**Type Conversion to Date**	The number of cells that were not in a valid date format and needed conversion.	0
Rows with `price` column	**Row Removal**	The number of rows where the `price` column was blank.	price=445916

Demonstrating that data quality has been improved

Integrity Constraints (ICs):

- **IC-MI1 (Completeness):** The `price` column must not contain any blank values.
- **IC-MI2 (Type Consistency):** The `id`, `menu_page_id`, and `dish_id` columns must contain only numerical data.

- **IC-MI3 (Date Format):** The `created_at` and `updated_at` columns must be valid date objects.
- **IC-MI4 (Format):** All columns must be free of leading or trailing whitespace.

Foreign Key Checks:

- **FK-MI1:** All MenuItem entries are linked to a valid MenuPage
- **FK-MI2:** All MenuItem entries are linked to a valid Dish

IC Violation Report

Integrity Constraint	Violations Before Cleaning	Violations After Cleaning	Difference
IC-MI1: Blank `price` values	445916	0	445916
IC-MI2: Non-numeric IDs	0	0	*[Equal to the "Before" count]*
IC-MI3: Invalid Dates	0	0	*[Equal to the "Before" count]*
IC-MI4: Whitespace Issues	0	0	*[Equal to the "Before" count]*
FK-MI1	0	38729	38729
FK-MI2	244	84	160

MenuPage:

Quantifying the results of your efforts:

Column(s)	Type of Change	Number of Cells Changed (Conceptual)	Cells Changed

`id`, `menu_id`, `page_number`	**Type Conversion to Number**	The number of cells that were originally stored as text but were successfully converted to a number.	id: 66937 menu_id: 66937 page_number: 65735
`full_height`, `full_width`	**Type Conversion to Number**	The number of cells that were originally stored as text but were successfully converted to a number.	full_height: 66608 full_width: 66608
`page_number`	**Remove rows with blank page number**	Manu page without page number is not valid.	1202

Demonstrating that data quality has been improved:

Integrity Constraints (ICs):

- **IC-MP1 (Type Consistency):** The `id`, `menu_id`, `page_number`, `full_height`, and `full_width` columns must contain only numerical data.
- **IC-MP2 (Format):** menu pages should have page number.

Foreign Key Checks:

- **FK-MP1:** All MenuPage entries are linked to a valid Menu

IC Violation Report

Integrity Constraint	Violations Before Cleaning	Violations After Cleaning	Difference
IC-MP1: Non-numeric Values	0	0	*[Equal to the "Before" count]*
IC-MP2: menu pages with empty page_number	1202	0	1202
FK-MP1	5803	36965	31162

Dish:

Quantifying Data Quality Changes

Column	Type of Change	Number of Cells Changed (Conceptual)	Rationale
`name`	Normalization & Filtering	The number of rows dropped due to empty values, plus the number of cells where the value was modified for case, spacing, or punctuation.	Ensures data consistency and removes unusable records.
`name_cluster`	Addition of a new column	All cells in the cleaned dataset's new column contain a value.	Adds a new feature to the dataset for grouping similar items based on clustering.
`description`	Column Removal	All cells in this column were removed from the dataset.	Simplifies the dataset by removing a column that may not be relevant to the use case.

Demonstrating Improved Data Quality

To demonstrate that data quality has been improved, we can define and check for violations of several integrity constraints (ICs) that are directly addressed by the cleaning steps.

Integrity Constraints (ICs):

- **IC-D1 (Completeness):** The `name` column must not contain any null, empty, or 'nan' values.
- **IC-D2 (Format):** All values in the `name` column must be lowercase, have no leading/trailing whitespace, and have a single space between words.

IC Violation Report

Integrity Constraint	Violations Before Cleaning	Violations After Cleaning	Difference
IC-D1: Completeness (`name` column)	*[Number of rows with empty or 'nan' names]*	0	*[Equal to the "Before" count]*
IC-D2: Format (`name` column)	411963	0	411963

Workflow Models

1.Outer(Overall workflow)

1. Load Data into SQLite

- **Input:** Initial CSV files
- **Process:** Import raw data into a SQLite database.
- **Output:** SQLite database with raw data loaded.
- **Purpose:** Create a structured, queryable data store as a foundation for integrity checks and cleaning.

2. Find Integrity Constraint (IC) Violations

- **Input:** SQLite database with raw data.
- **Process:** Detect data quality issues such as missing prices, invalid dates, or duplicate records.
- **Output:** Report listing integrity constraint violations.
- **Purpose:** Identify data quality problems that could affect accurate price change analysis.

3. Load Data into OpenRefine

- **Input:** Initial CSV files
- **Process:** Import data into OpenRefine for visual inspection and manual cleaning.
- **Output:** OpenRefine project with raw sales data loaded.
- **Purpose:** Facilitate interactive correction of inconsistent data.

4. Manual Cleanup in OpenRefine

- **Input:** OpenRefine project with raw data.
- **Process:** User performs manual fixes like correcting price typos, standardizing date formats, and merging duplicate entries.
- **Output:** Cleaned dataset within OpenRefine.
- **Purpose:** Improve data quality to ensure reliable price trend computations.

5. Clean Data Using Python Scripts

- **Input:** Dish.csv(too large for open refine clustering).
- **Process:** Run Python scripts to normalize and cluster duplicate dish names.
- **Output:** Further cleaned and processed dataset ready for analysis.
- **Purpose:** Clean data which can’t be done by open refine.

6. Load Cleaned Data Back into SQLite Without Foreign Keys

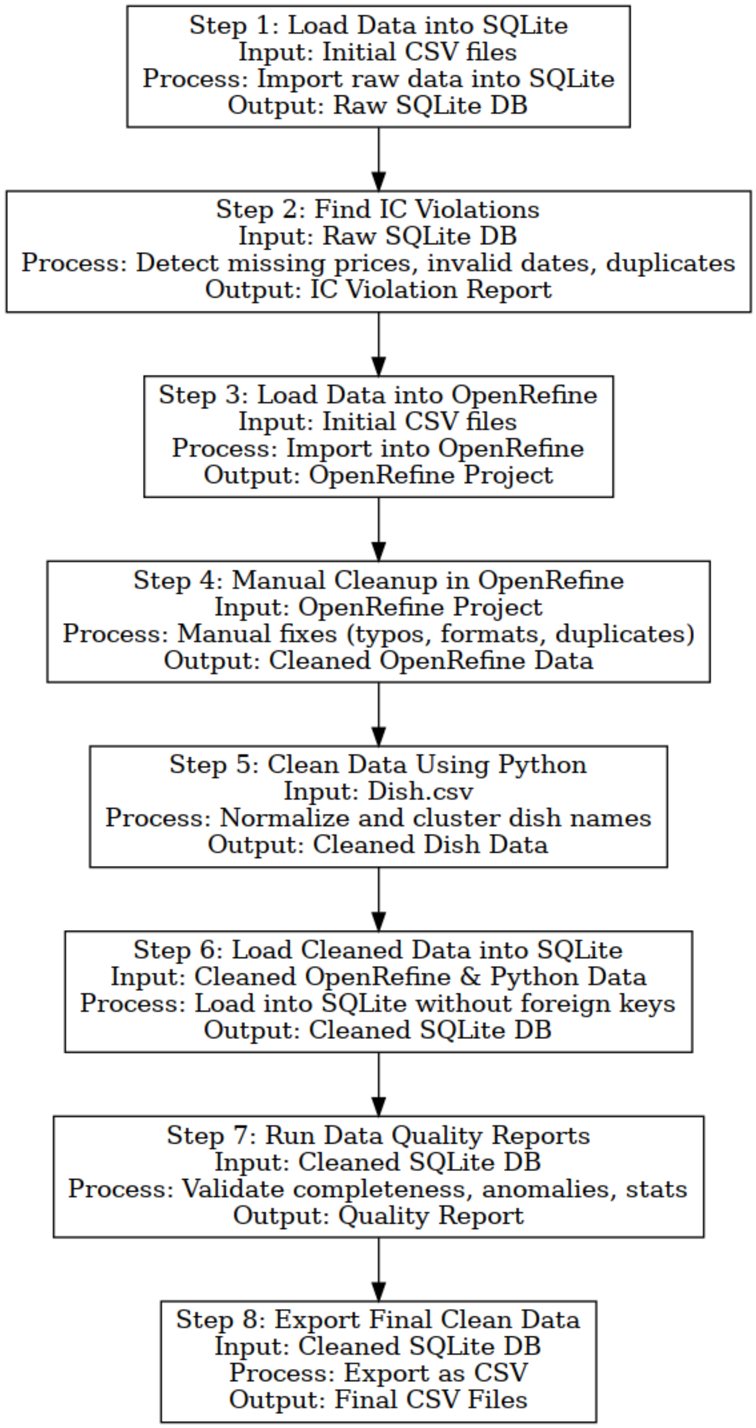
- **Input:** Cleaned dataset from Python and Openrefine.
- **Process:** Load cleaned data into SQLite database with foreign key constraints disabled to simplify updates and reprocessing.
- **Output:** SQLite database with cleaned data.
- **Purpose:** Prepare the dataset for final quality checks and querying.

7. Run Data Quality Reports

- **Input:** Cleaned SQLite database without foreign key constraints.
- **Process:** Execute SQL queries to validate data completeness, check for remaining anomalies, and compute summary statistics.
- **Output:** Quality report summarizing data readiness.
- **Purpose:** Ensure dataset integrity before price change analysis.

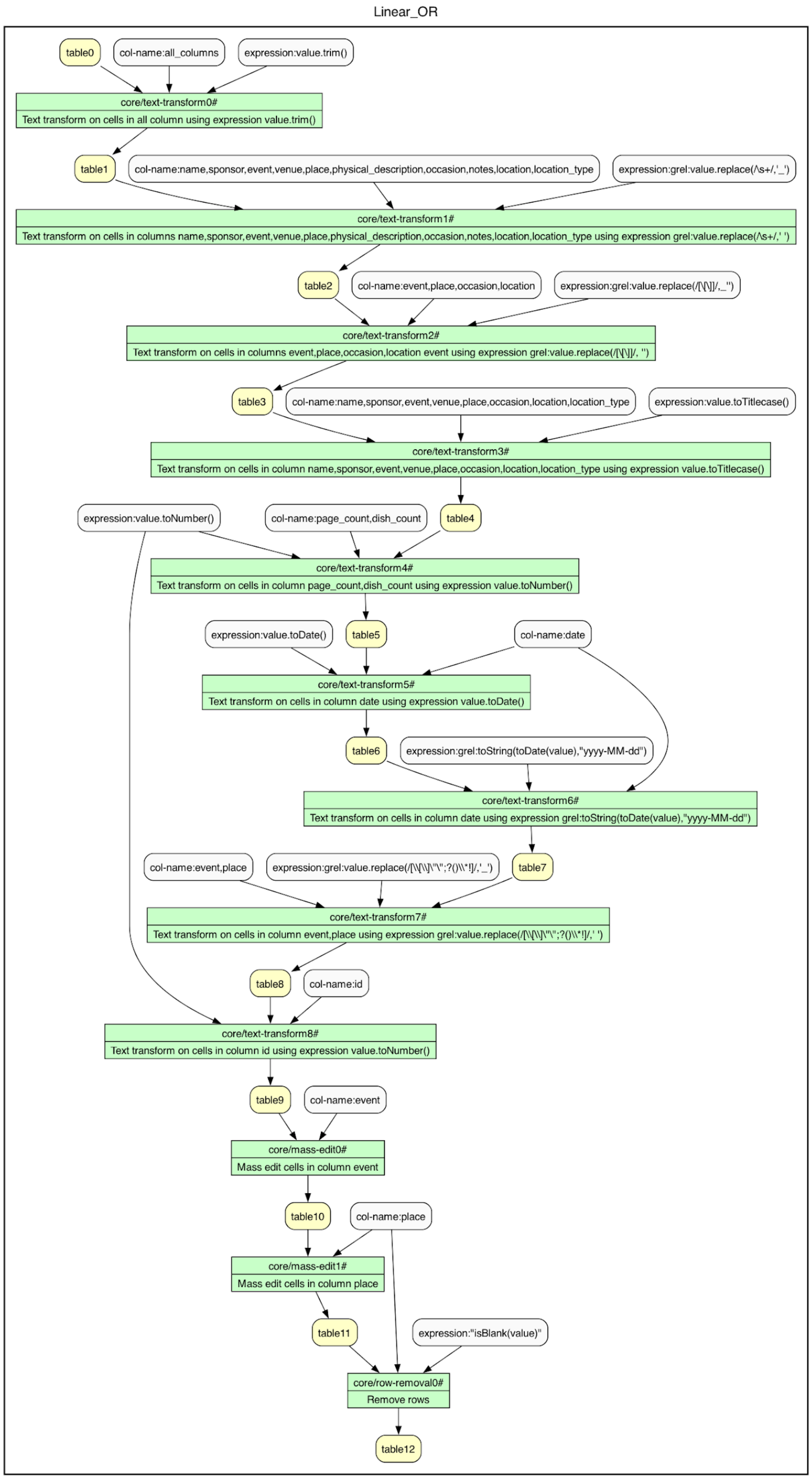
8. Export Final Clean Data as CSV

- **Input:** Cleaned SQLite database.
- **Process:** Export the final dataset as CSV files.
- **Output:** CSV files containing clean, validated sales data.
- **Purpose:** Provide reliable input data for downstream analysis of average price changes over time.

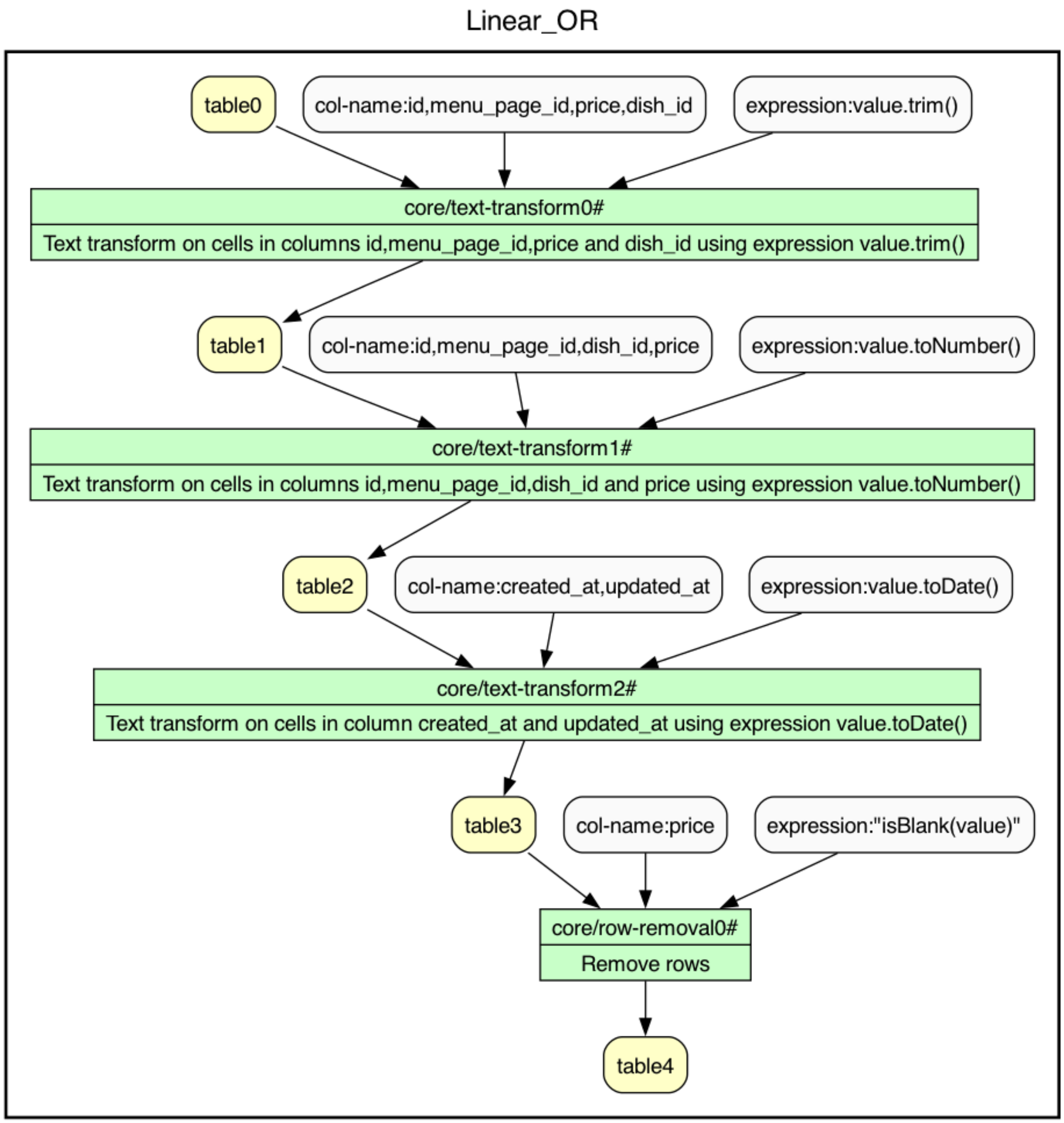


1.Inner Workflows

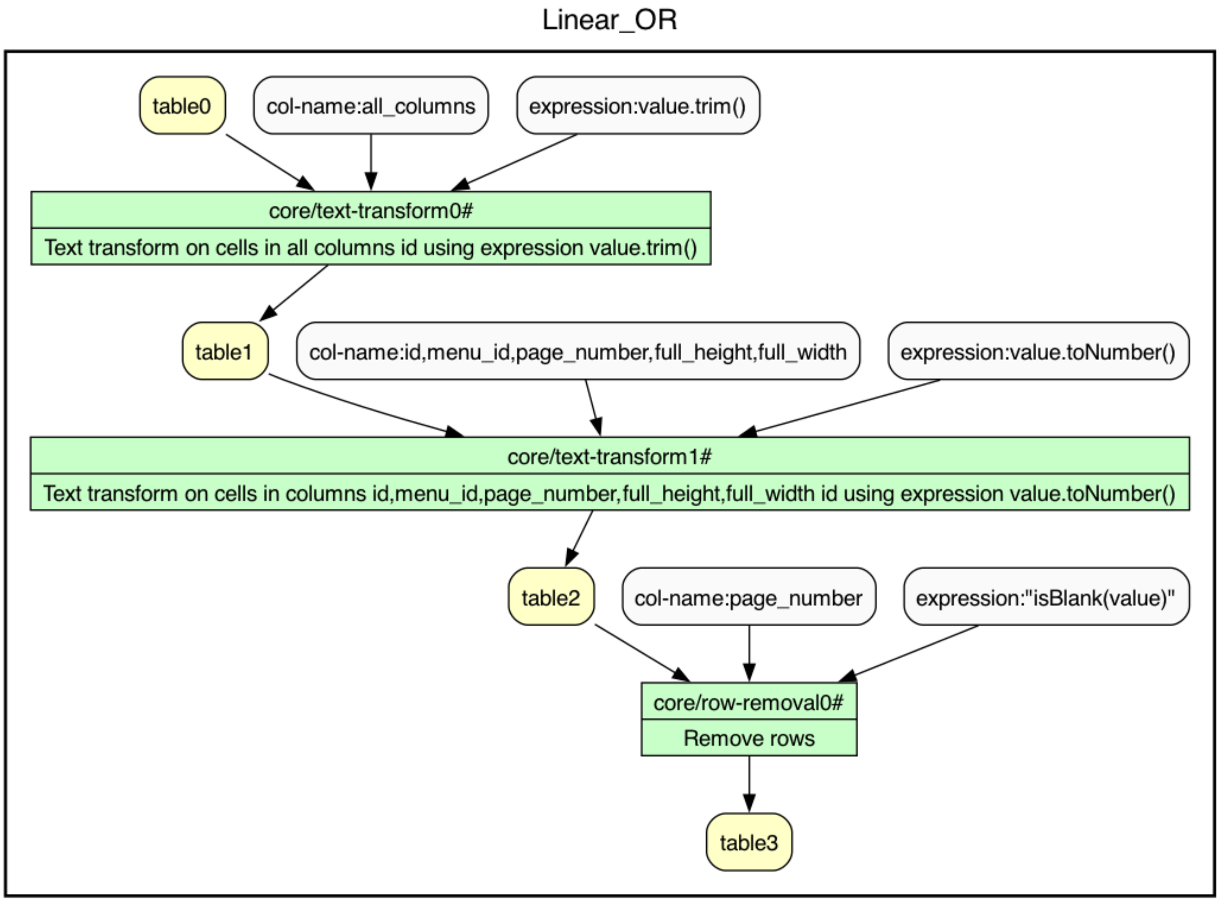
Menu(using or2yw)



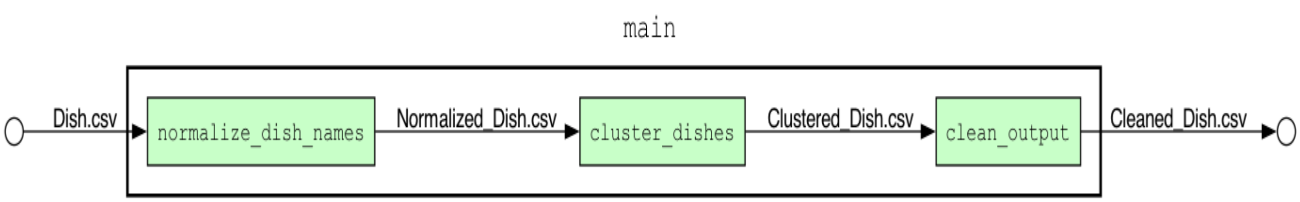
MenuItem(using or2yw)



MenuPage(using or2yw)



Dish(Using graphviz and yw jar file)



Conclusions & Summary

In this project, we explored the strengths and limitations of various data cleaning tools introduced in CS 513. One of the key lessons learned is that no single tool provides a comprehensive solution for all data cleaning needs.

OpenRefine proved highly effective for identifying and resolving single-field integrity constraint (IC) violations, but it lacks support for relational ICs. SQL, on the other hand, excels at enforcing relational constraints but is not inherently designed for flexible data cleaning workflows. To bridge these gaps, we developed a hybrid approach using Python to integrate OpenRefine and SQL. This allowed us to process the entire NYPL restaurants dataset, though the system was not optimized for speed or efficiency.

We recognize that for larger datasets, distributed data processing platforms like Apache Spark or Google BigQuery would be essential. While these were beyond the scope of our project, they represent promising directions for scaling our workflow.

Despite these limitations, we successfully cleaned the NYPL restaurants dataset in alignment with use case U1 from our Phase I submission. The cleaned data is suitable for exploratory data mining and unsupervised learning applications. With further refinement, subsets of this data could support more critical analytical tasks.

Additionally, we built a Python application capable of cleaning similar datasets with a single command. This tool ensures reproducibility and efficiency, offering a more robust alternative to manual workflows in OpenRefine and SQL.