Anjali Jain
anjali9@illinois.edu
Team - Team232
Team Name - Mocking Bird
Team Members - Single member Project

Final Project Phase-II(Summer 2024)

CS513: Theory & Practice of Data Cleaning

Overview
For my Summer Final Project, the NYPL-menus dataset was assigned to me. This dataset consists of four tables: Dish, Menu, MenuItem, and MenuPages. It provides a detailed historical overview of dining, featuring 17,545 menus from 233 venues across 3,714 locations, with 423,397 unique dishes. Notable trends include frequent appearances of coffee, tea, and celery. Predominantly featuring daily menus and special occasions like anniversaries, the dataset offers insights into culinary trends, pricing patterns, and dining habits over time. Analysis revealed the top occasions, status distribution, popular dishes, and the impressive longevity of celery, which appeared from year 1 to 2928. This comprehensive collection is a valuable resource for food historians and researchers. For more details refer Phase I document.
About Dataset
NYPL-menus a. Content: "What's on the menu?": A mix of simple bibliographic description of the menus (created by The New York Public Library) and the culinary and economic content of the menus themselves (transcribed by you). b. Source: http://menus.nypl.org/data
Target (Main) use case U1: Data cleaning is necessary and sufficient
Use Case U1 - Analyze historical pricing trends of dishes across various menus over time

In order to perform analysis on historical pricing trends of dishes across various menus over time, a detailed and cleaned dataset is must which includes accurate dates, consistent currency formats, and complete price information.
Project Phase II begins here
Description of Data Cleaning Performed

To achieve the outcome for Use Case 1, I performed data cleaning on the NYPL dataset consisting of historical pricing trends of various dishes across multiple menus over time.

The primary objective was to ensure data accuracy, consistency, and completeness to facilitate meaningful analysis of the historical pricing trends. I used OpenRefine and Python (Jupyter Notebook) for this task.

Data Cleaning Steps :

1. Load CSV Files into OpenRefine :All 4 dataset DISH, Menu, MenuItem and MenuPage were imported into separate projects within OpenRefine a powerful tool for data cleaning and transformation.

2. Data Cleaning with OpenRefine .

   A. DISH.csv

- Remove Leading and Trailing Spaces and White spaces collapsed :Removed leading and trailing spaces, as well as collapsed white spaces, for all fields.
- Transform ID :Transformed the id column to a numeric format.
- Identify and Remove Duplicates : Went to the Facet menu and chose Text facet for the id column to see if there were any duplicate IDs. No duplicates were found.
- Transform name column : Transformed the name column to title case and removed special characters using GREL(/[()\[\]{}*?'"-]/, “").
- Standardize Date Formats : Standardized the date formats in the first_appeared and last_appeared columns by common transform to date.
- Ensure Numeric Consistency : Ensured lowest_price and highest_price, also menus_appeared and times_appeared are in a consistent numeric format by common transform to number.

   B. MenuItem.csv

- Remove Leading and Trailing Spaces and White spaces collapsed :Removed leading and trailing spaces, as well as collapsed white spaces, for all fields.
- On analysis I observed dataset is almost clean, therefore I performed generic operation to ensure consistency.
- Ensure Numeric Consistency : Ensured id,menu_page_id,price,high_price,dish_id,xpos and ypos are in a consistent numeric format by common transform to number.

C. MenuPage.csv

- Remove Leading and Trailing Spaces and White spaces collapsed :Removed leading and trailing spaces, as well as collapsed white spaces, for all fields.
- On analysis I observed dataset is almost clean, therefore I performed generic operation to ensure consistency.
- Ensure Numeric Consistency : Ensured id,menu_id,page_number,image_id,full_height & full_width are in a consistent numeric format by common transform to number.
- To maintains consistency in string data, aiding in data management uuid common transform to lowercase.

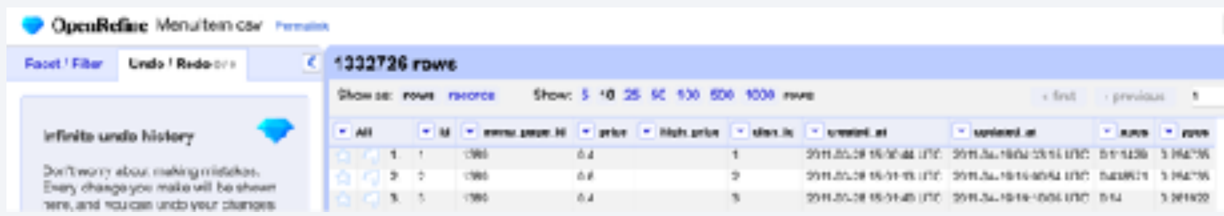
D. Menu.csv

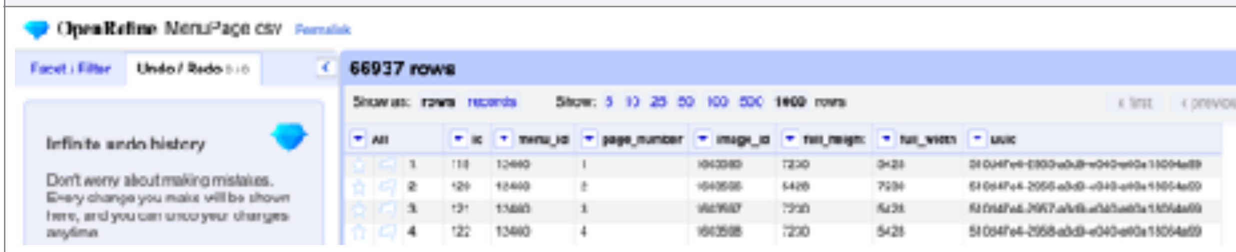
- Remove Leading and Trailing Spaces and White spaces collapsed :Removed leading and trailing spaces, as well as collapsed white spaces, for all fields.
- This dataset consumed my big part of data cleaning effort considering it was most dirty data set where I have to perform 115 steps to clean as per my requirement.
- I ensured field id ,page_count and dish_count are in a consistent numeric format by common transform to number.
- Date column is transformed to date in order to maintain standardize date format.
- Columns like name, sponsor, event, venue, place, occasion and notes are common transform to titlecase.
- Performed clustering multiple times on column name, sponsor, event, venue and place using method Key Collision :fingerprint, ngram-fingerprint, metaphone 3, colgne-phonetic, Daiktch-Mokotoff and Beider-Morse & Nearest Neighbor: levenshtein and ppm.
- For above columns also removed special characters using GREL(/[()\[\]{}*?'"-]/, “”:;).
- Due to lack of availability of data and relevance to support U1 use case removed keywords, language, location and location_type columns.

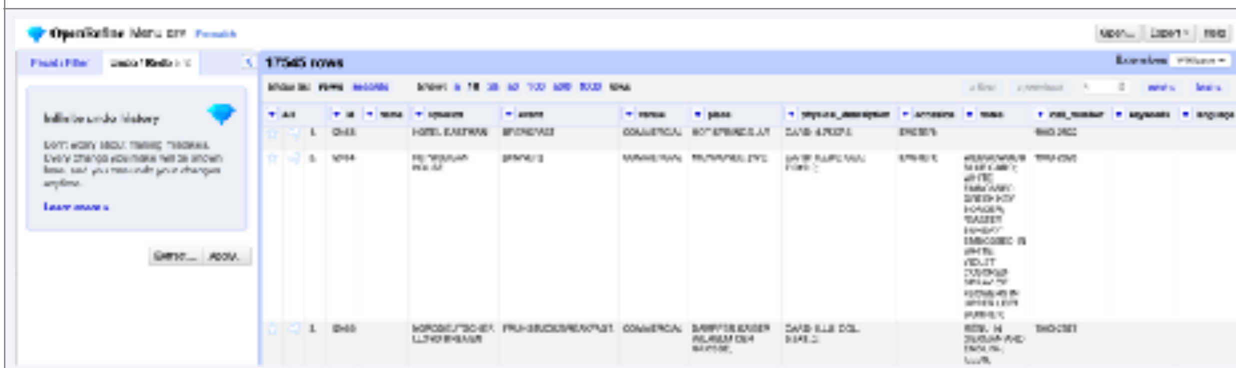
In summary, while not all steps were directly essential for Use Case 1, they significantly improved overall data quality. All transformations mentioned above ensured the dataset is clean, consistent, and reliable, supporting accurate analysis for trend analysis.

3. Perform IC check on cleaned dataset.

Snapshot of Dirty Data set in OpenRefine







Data Cleaning Steps Rationale & Relevance :

Dataset	Cleaning Steps	Rationale	Relevance to use case U1
DISH.csv	Remove Leading and Trailing Spaces:	Cleaning up extra spaces prevents issues in data merging and comparisons	While this step may not be directly related to pricing trends, it ensures overall data integrity, made analysis more reliable.Therefore its useful but not required

Dataset	Cleaning Steps	Rationale	Relevance to use case U1
	Transform ID to number	Transforming id to numeric format ensures it can be correctly used in calculations and merges.	While the transformation of `id` is crucial for data integrity, the name transformation primarily enhances data presentation and usability, which indirectly supports the analysis process.
	Identify and Remove Duplicates	Ensuring each dish has a unique id is crucial for accurate data merging and analysis. Duplicate IDs can lead to incorrect aggregations and misleading results.	This step is essential to avoid duplication errors when analyzing historical pricing trends across various menus over time.
	Standardize Date Formats	Consistent date formats in first_appeared and last_appeared columns are necessary for accurate time-based analysis.	Standardizing dates allows for precise calculations of when dishes appeared and their pricing trends over specific periods, which is directly relevant to analyzing historical pricing trends.
	Ensure Numeric Consistency	Having lowest_price, highest_price, menus_appeared, and times_appeared in a consistent numeric format ensures accurate calculations and comparisons.	Accurate numeric data is crucial for analyzing trends and patterns in dish pricing and appearances across menus, directly supporting the use case.

Dataset	Cleaning Steps	Rationale	Relevance to use case U1
	Transform Name Columns	Standardizing name to title case and removing special characters ensures consistency and readability.	Useful but not required Steps as it enhances data integrity and readability, which is beneficial for a clean dataset but not directly tied to the specific use case of analyzing pricing trends.
MenuItem.csv	Remove Leading and Trailing Spaces:	Cleaning up extra spaces prevents issues in data merging and comparisons	While this step may not be directly related to pricing trends, it ensures overall data integrity, made analysis more reliable.Therefore its useful but not required although there was no change captured .
	Ensure Numeric Consistency	Ensured menu_page_id is in numeric format for proper identification and relational operations.	Important for linking and aggregating data accurately across different menus and pages.
	Ensure Numeric Consistency	Converts price values to numeric format to ensure they can be used in mathematical operations and analyses.	Essential for analyzing historical pricing trends since the prices need to be accurate numeric values.
	Ensure Numeric Consistency	Ensures high_price values are numeric, enabling precise comparisons and calculations.	Important for comparing price ranges and understanding pricing trends over time.
	Ensure Numeric Consistency	Converts dish_id values to numeric format for consistent identification and relation to DISH.csv and other datasets.	Crucial for accurately linking dishes to their appearances and prices across menus.

Dataset	Cleaning Steps	Rationale	Relevance to use case U1
	Ensure Numeric Consistency	Ensures xpos values are numeric for proper spatial analysis if needed.	Not directly relevant to pricing trends but ensures overall data integrity.
	Ensure Numeric Consistency	Converts ypos values to numeric format, ensuring consistency in spatial data.	Similar to xpos, it is not directly related to pricing trends but contributes to data quality.It is useful but not required however, these transformations ensure overall data quality and integrity, which supports reliable analysis.
	Ensure Numeric Consistency	Ensures id values are numeric for consistent and accurate identification.	Critical for uniquely identifying each record and preventing duplicates, which supports accurate trend analysis.
MenuPage.csv	Remove Leading and Trailing Spaces	Cleaning up extra spaces prevents issues in data merging and comparisons	While this step may not be directly related to pricing trends, it ensures overall data integrity, made analysis more reliable.Therefore its useful but not required,Also MenuPage.csv was already clean therefore no change was captured.
	ID Transformation	Ensures id values are in numeric format for consistent identification.	Important for uniquely identifying each record and preventing duplicates, which supports accurate trend analysis.It is essential for unique identification and preventing duplicates.

Dataset	Cleaning Steps	Rationale	Relevance to use case U1
	menu_id Transformation	Converts menu_id values to numeric format for proper identification and relational operations.	Crucial for linking and aggregating data accurately across different menus and pages.Important for linking and aggregating data across different menus.
	page_number Transformation	Ensures page_number values are numeric, which is necessary for ordering and analyzing pages correctly.	Important for organizing and linking data to specific pages in the menus.
	image_id Transformation	Converts image_id values to numeric format, ensuring consistency for image identification and linking.	Useful but not required although it helps in associating images with corresponding menu pages, though not directly related to pricing trends.
	full_height Transformation	Ensures full_height values are numeric for proper spatial analysis if needed.	Not directly relevant to pricing trends but ensures overall data integrity.Although this step is useful but not required ensure overall data quality and consistency, supporting reliable analysis.
	full_width Transformation	Converts full_width values to numeric format, ensuring consistency in spatial data.	Similar to full_height, it is not directly related to pricing trends but contributes to data quality.useful but not required.
	uuid	Converts uuid values to lowercase for consistency in string data.	Ensures uniformity in UUID representation, aiding in data consistency and integrity.Useful but not required steps.

Dataset	Cleaning Steps	Rationale	Relevance to use case U1
Menu.csv	All	Removes leading and trailing spaces which can cause inconsistencies during analysis.	Trimming whitespace helps in accurate data analysis, particularly when filtering or grouping data by these columns this is required step for this dataset as this the dirties dataset.
	id	Converts textual representations of numbers into actual numeric values for better data handling.	Ensures the id column is correctly formatted for any numeric operations.
	columns: name, sponsor, event, venue, place, occasion, currency, status, page_count, dish_count	Converted text to title case for uniformity.	Moderately necessary. While not directly impacting pricing trends, it ensures data is consistently formatted for easier reading and interpretation.
	Columns: name, sponsor, event, venue, place, physical_description, occasion	Cleans text by removing unwanted special characters like [],/\{}()”’;:*-	Useful. Helps prevent issues in data analysis caused by special characters.
	date	Converts text to date format for accurate date-based analysis.	Crucial step as date conversion is essential for analyzing trends over time.
	Columns: name, event, venue, place, occasion	Standardizes specific values for consistency by performing mass edits.	Important step to ensures consistency in data, which is vital for accurate grouping and analysis.
	Columns: keyword, language, location, location_type	I removed unnecessary columns that do not contribute to the analysis.	Necessary step to simplifies the dataset by removing irrelevant information.

Data Quality Changes

Dataset	Field Name	Quality Changes
DISH.csv	All	Text transform on 9,045 cells in column name :value.trim()
	first_appeared	Text transform on 367,905 cells in column first_appeared :value.toDate()
	last_appeared	Text transform on 368,076 cells in column last_appeared :value.toDate()
	lowest_price	Text transform on 394,297 cells in column lowest_price :value.toNumber()
	highest_price	Text transform on 394,297 cells in column highest_price :value.toNumber( )
	menus_appeared	Text transform on 423,397 cells in column menu_appeared :value.toNumb er()
	times_appeared	Text transform on 423,397 cells in column times_appeared :value.toNumb er()
	id	Text transform on 423,397 cells in column id :value.toNumber()
	name	Text transform on 6,415 cells in column name :value.replace(/ [\p{Zs}\s]+/,’’)
	name	Text transform on 281,551 cells in column name :value.toTitlecase()
	name	Text transform on 86,969 cells in column name :grel:value.replace(/[()\ [\]{}*?'"-]/, “")
MenuItem.csv	menu_page_id	Text transform on 1,332,726 cells in column menu_page_id: value.toNumber()

	price	Text transform on 886,810 cells in column price: value.toNumber()
	high_price	Text transform on 91,905 cells in column high_price: value.toNumber()
	dish_id	Text transform on 1,332,485 cells in column dish_id: value.toNumber()
	xpos	Text transform on 1,332,726 cells in column xpos: value.toNumber()
	ypos	Text transform on 1,332,726 cells in column ypos: value.toNumber()
	id	Text transform on 1,332,726 cells in column id: value.toNumber()
MenuPage.csv	id	Text transform on 66,937 cells in column id: value.toNumber()
	menu_id	Text transform on 66,937 cells in column menu_id: value.toNumber()
	page_number	Text transform on 65,735 cells in column page_number: value.toNumber()
	image_id	Text transform on 66,914 cells in column image_id: value.toNumber()
	full_height	Text transform on 66,608 cells in column full_height: value.toNumber()
	full_width	Text transform on 66,608 cells in column full_width: value.toNumber()
	uuid	Text transform on 1 cells in column uuid: value.toLowercase()

Menu.csv	All	Text transform on 14 cells in column location: value.trim() Text transform on 14 cells in column sponsor: value.trim() Text transform on 125 cells in column notes: value.trim() Text transform on 9 cells in column call_number: value.trim() Text transform on 384 cells in column physical_description: value.trim() Text transform on 4 cells in column currency_symbol: value.trim() Text transform on 3 cells in column event: value.trim() Text transform on 9 cells in column name: value.trim() Text transform on 8 cells in column place: value.trim() Text transform on 127 cells in column sponsor: value.replace(/\s+/,' ‘)
	Id	Text transform on 17,545 cells in column id: value.toNumber()
	name	Text transform on 629 cells in column name: value.toTitlecase()
	name	Text transform on 798 cells in column name: grel:value.replace(/[()\[\]{} *?'"-]/, “")
	sponsor	Text transform on 8,683 cells in column sponsor: value.toTitlecase()
	sponsor	Text transform on 3,113 cells in column sponsor: grel:value.replace(/[()\[\]{} *?'"-]/, "")
	event	Text transform on 7,829 cells in column event: value.toTitlecase()

	event	Text transform on 649 cells in column event: grel:value.replace(/[()\[\]{} *?'"-]/, “")
	event	Text transform on 247 cells in column event: value.toTitlecase()
	event	Text transform on 79 cells in column event: grel:value.replace(/[\[\]\{\}\(\) *\?\-\′;\:"]/,"")
	sponsor	Text transform on 658 cells in column sponsor: value.toTitlecase()
	sponsor	Text transform on 57 cells in column sponsor: grel:value.replace(/[\[\]\{\}\(\) *\?\-\';\:"]/,"")
	name	Text transform on 144 cells in column name: value.toTitlecase()
	name	Text transform on 12 cells in column name: grel:value.replace(/[\[\]\{\}\(\) *\?\-\';\:"]/,"")
	venue	Text transform on 8,109 cells in column venue: value.toTitlecase()
	venue	Text transform on 1,974 cells in column venue: grel:value.replace(/[\[\]\{\}\(\) *\?\-\′;\:"]/,"")
	venue	Text transform on 148 cells in column venue: value.toTitlecase()
	place	Text transform on 7,337 cells in column place: value.toTitlecase()
	place	Text transform on 2,497 cells in column place: grel:value.replace(/[\[\]\{\}\(\) *\?\-\′;\:"]/,"")

	place	Text transform on 1,379 cells in column place: value.toTitlecase()
	physical_description	Text transform on 151 cells in column physical_description: grel:value.replace(/[()\[\]{} *?'"-]/, "")
	occasion	Text transform on 3,752 cells in column occasion: value.toTitlecase()
	occasion	Text transform on 2,524 cells in column occasion: grel:value.replace(/[\[\]\{\}\(\)*\?\-\';\:"]/,"")
	occasion	Text transform on 467 cells in column occasion: value.toTitlecase()
	notes	Text transform on 9,458 cells in column notes: value.toTitlecase()
	keyword	Remove column keyword
	language	Remove column language
	Date	Text transform on 16,959 cells in column date: value.toDate()
	location	Remove column location
	location_type	Remove column location_type
	currency	Text transform on 118 cells in column currency: value.toTitlecase()
	status	Text transform on 17,545 cells in column status: value.toTitlecase()
	page_count	Text transform on 17,545 cells in column page_count: value.toTitlecase()
	dish_count	Text transform on 17,545 cells in column dish_count: value.toTitlecase()

	name	Mass edit 286 cells in column name Mass edit 621 cells in column name Mass edit 790 cells in column name Mass edit 4 cells in column name Mass edit 43 cells in column name Mass edit 7 cells in column name Mass edit 64 cells in column name
	name	Text transform on 6 cells in column name: grel:value.replace("United States Ship Wyoming", "U.S.S. Wyoming”)
	event	Mass edit 4,079 cells in column event Mass edit 6 cells in column event Mass edit 2,257 cells in column event Mass edit3,050 cells in column event Mass edit2,500 cells in column event Mass edit 2,625 cells in column event
	venue	Mass edit 2,068 cells in column venue Mass edit 21 cells in column venue Mass edit 5,051 cells in column venue Mass edit 94 cells in column venue Mass edit 489 cells in column venue Mass edit 162 cells in column venue
	venue	Text transform on 1,122 cells in column venue: value.toTitlecase()

	sponsor	Mass edit 1,016 cells in column sponsor
	place	Mass edit 2,052 cells in column place Mass edit 2,435 cells in column place Mass edit 3,262 cells in column place Mass edit 602 cells in column place Mass edit 3,288 cells in column place Mass edit 16 cells in column place
	occasion	Mass edit 941 cells in column occasion Mass edit 272 cells in column occasion Mass edit 1393 cells in column occasion Mass edit 693 cells in column occasion Mass edit 522 cells in column occasion Mass edit 141 cells in column occasion Mass edit 705 cells in column occasion

Data quality has been improved
I used SQLite to perform integrity constraint checks on the dataset to verify the integrity of actions performed on the dirty dataset using OpenRefine. This process helps identify areas that are still not clean and decide whether further actions are required to achieve Use Case 1

Table - dish_clean (Naming for clean dataset)

- Check count of records contains leading or trailing space:

```
SELECT COUNT(*)
FROM dish_clean
WHERE (LENGTH(id) != LENGTH(TRIM(id))
      OR LENGTH(name) != LENGTH(TRIM(name))
      OR LENGTH(description) != LENGTH(TRIM(description))
      OR LENGTH(menus_appeared) != LENGTH(TRIM(menus_appeared))
      OR LENGTH(times_appeared) != LENGTH(TRIM(times_appeared))
      OR LENGTH(first_appeared) != LENGTH(TRIM(first_appeared))
      OR LENGTH(last_appeared) != LENGTH(TRIM(last_appeared))
      OR LENGTH(lowest_price) != LENGTH(TRIM(lowest_price))
      OR LENGTH(highest_price) != LENGTH(TRIM(highest_price)));
```

Outcome :

Message | Result 1

COUNT(*)
0

Before :

Message | Result 1

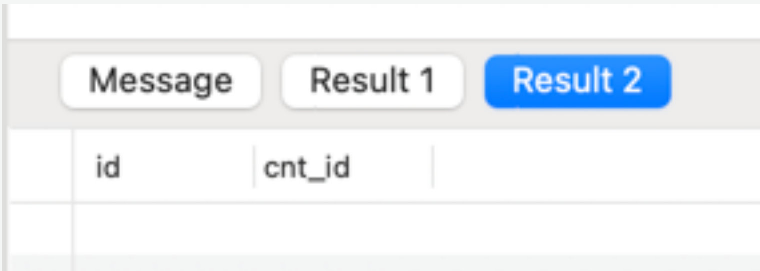
COUNT(*)
9038

Action: No action needed IC check is passed.

- Check if there are any duplicate record with primary key ID

```
SELECT id, COUNT(id) AS cnt_id
 FROM dish_clean
 GROUP BY id
 HAVING COUNT(id) > 1;
```

Outcome :

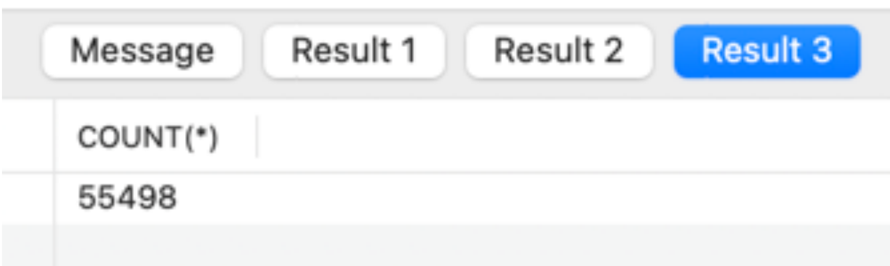


Action : No action needed IC check is passed.

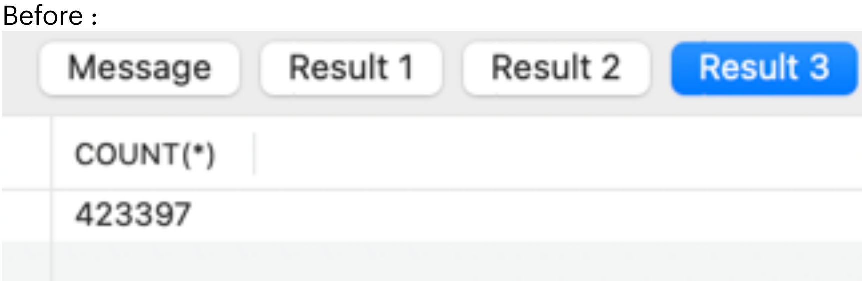
- Check how many records are not in ISO date format

```
SELECT COUNT(*)
FROM dish_clean
WHERE (first_appeared IS NOT NULL
      AND (first_appeared NOT LIKE '____-__-__'
         AND first_appeared NOT LIKE '____-__-__T__:__:%Z'))
  OR (last_appeared IS NOT NULL
      AND (last_appeared NOT LIKE '____-__-__'
         AND last_appeared NOT LIKE ‘____-__-__T__:__:%Z'));
```

Outcome :



Before :



Action : After initial cleaning using OpenRefine on the table 'dish_clean' which contains 423,397 records. However, 55,498 of these records still have non-ISO date formats in the 'first_appeared' and 'last_appeared' columns. My analysis revealed incorrect values such as 0 and 1, which prevent proper date conversion. These remaining records require further cleaning to ensure accurate trend analysis. This additional data cleaning will be implemented in Python as part of the report generation process for trend analysis.

Message | Result 1 | Result 2 | Result 3 | Result 4

id	first_appeared	last_appeared
239	1	1970-01-01T00:00:00Z
247	1	1987-01-01T00:00:00Z
265	1	1918-01-01T00:00:00Z
270	1	1969-01-01T00:00:00Z
280	1	1953-01-01T00:00:00Z
293	1	1989-01-01T00:00:00Z
340	0	0
348	1	1998-01-01T00:00:00Z
352	1	1962-01-01T00:00:00Z

- Check if all special characters are removed successfully from column name.

```
SELECT COUNT(*)
FROM dish_clean
WHERE name LIKE '%(%'
   OR name LIKE '%)%'
   OR name LIKE '%\[%'
   OR name LIKE '%\]%'
   OR name LIKE '%{%'
   OR name LIKE '%}%'
   OR name LIKE '%*%'
   OR name LIKE '%?%'
   OR name LIKE '%"%'
   OR name LIKE '%''%'
   OR name LIKE '%-%';
```

Outcome:

Before :

Action: No action needed IC check is passed.

- Check if lowest_price and highest_price is converted successfully to numeric format.

```
SELECT
    SUM(CASE WHEN CAST(lowest_price AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_lowest_price_count,
    SUM(CASE WHEN CAST(highest_price AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_highest_price_count
FROM dish_clean
WHERE lowest_price IS NOT NULL OR highest_price IS NOT NULL;
```

Outcome:

Message | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6

non_numeric_lowest_price_count	non_numeric_highest_price_count
0	0

Action : No action needed IC check is passed.

Table - item_clean (Naming for clean dataset)

- Check count of records contains leading or trailing space:

```
SELECT COUNT(*)
FROM item_clean
WHERE (LENGTH(id) != LENGTH(TRIM(id))
    OR LENGTH(menu_page_id) != LENGTH(TRIM(menu_page_id))
    OR LENGTH(price) != LENGTH(TRIM(price))
    OR LENGTH(high_price) != LENGTH(TRIM(high_price))
    OR LENGTH(dish_id) != LENGTH(TRIM(dish_id))
    OR LENGTH(created_at) != LENGTH(TRIM(created_at))
    OR LENGTH(updated_at) != LENGTH(TRIM(updated_at))
    OR LENGTH(xpos) != LENGTH(TRIM(xpos))
    OR LENGTH(ypos) != LENGTH(TRIM(ypos)));
```

Outcome:

Message | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 7

COUNT(*)
0

Action : No action needed IC check is passed.

- Check if menu_page,price,high_price,dish_id,xpos and ypos is converted successfully to numeric format.

```
SELECT
   SUM(CASE WHEN CAST(id AS REAL) IS NULL THEN 1 ELSE 0 END) AS non_numeric_id_count,
   SUM(CASE WHEN CAST(menu_page_id AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_menu_page_id_count,
   SUM(CASE WHEN CAST(price AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_price_count,
   SUM(CASE WHEN CAST(high_price AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_high_price_count,
   SUM(CASE WHEN CAST(dish_id AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_dish_id_count,
   SUM(CASE WHEN CAST(xpos AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_xpos_count,
   SUM(CASE WHEN CAST(ypos AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_ypos_count
FROM item_clean
WHERE id IS NOT NULL
   OR menu_page_id IS NOT NULL
   OR price IS NOT NULL
   OR high_price IS NOT NULL
   OR dish_id IS NOT NULL
   OR xpos IS NOT NULL
   OR ypos IS NOT NULL;
```

Outcome:

Action : No action needed IC check is passed.

Table - page_clean (Naming for clean dataset)

- Check count of records contains leading or trailing space:

```
SELECT COUNT(*)
FROM page_clean
WHERE (LENGTH(id) != LENGTH(TRIM(id))
      OR LENGTH(menu_id) != LENGTH(TRIM(menu_id))
      OR LENGTH(page_number) != LENGTH(TRIM(page_number))
      OR LENGTH(image_id) != LENGTH(TRIM(image_id))
      OR LENGTH(full_height) != LENGTH(TRIM(full_height))
      OR LENGTH(full_width) != LENGTH(TRIM(full_width))
      OR LENGTH(uuid) != LENGTH(TRIM(uuid)));
```

Outcome :

Message | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 7 | Result 8 | Result 9

COUNT(*)
0

Action : No action needed IC check is passed.

- Check if id,menu_id,page_number_image_id,full_height and full_widthis converted successfully to numeric format.

```
SELECT
   SUM(CASE WHEN CAST(id AS REAL) IS NULL THEN 1 ELSE 0 END) AS non_numeric_id_count,
   SUM(CASE WHEN CAST(menu_id AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_menu_id,
   SUM(CASE WHEN CAST(page_number AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_page_number,
   SUM(CASE WHEN CAST(image_id AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_image_id,
   SUM(CASE WHEN CAST(full_height AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_full_height,
   SUM(CASE WHEN CAST(full_width AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_full_width
FROM page_clean
WHERE id IS NOT NULL
   OR menu_id IS NOT NULL
   OR page_number IS NOT NULL
   OR image_id IS NOT NULL
   OR full_height IS NOT NULL
   OR full_width IS NOT NULL;
```

Outcome:

Message | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 7 | Result 8 | Result 9 | Result 10

non_numeric_id_count	non_numeric_menu_id	non_numeric_page_number	non_numeric_image_id	non_numeric_full_height	non_numeric_full_width
0	0	0	0	0	0

Action : No action needed IC check is passed.

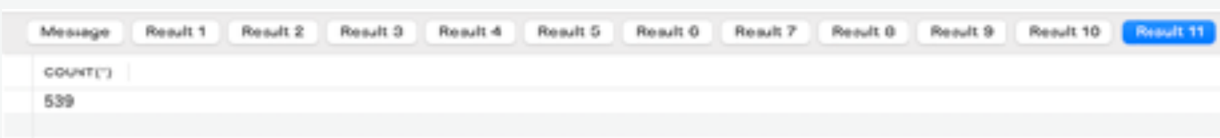
Table - menu_clean (Naming for clean dataset)

- Check count of records contains leading or trailing space:

```
SELECT COUNT(*)
FROM menu_clean
WHERE (LENGTH(id) != LENGTH(TRIM(id))
      OR LENGTH(name) != LENGTH(TRIM(name))
      OR LENGTH(sponsor) != LENGTH(TRIM(sponsor))
      OR LENGTH(event) != LENGTH(TRIM(event))
      OR LENGTH(venue) != LENGTH(TRIM(venue))
      OR LENGTH(place) != LENGTH(TRIM(place))
      OR LENGTH(physical_description) != LENGTH(TRIM(physical_description))
      OR LENGTH(occasion) != LENGTH(TRIM(occasion))
      OR LENGTH(notes) != LENGTH(TRIM(notes))
      OR LENGTH(call_number) != LENGTH(TRIM(call_number))
      OR LENGTH(date) != LENGTH(TRIM(date))
      OR LENGTH(currency) != LENGTH(TRIM(currency))
      OR LENGTH(currency_symbol) != LENGTH(TRIM(currency_symbol))
      OR LENGTH(status) != LENGTH(TRIM(status))
      OR LENGTH(page_count) != LENGTH(TRIM(page_count))
      OR LENGTH(dish_count) != LENGTH(TRIM(dish_count)));
```

Outcome:

Before :



Action : No action needed IC check is passed.

- Check if id, page_count and dish_count converted successfully to numeric format.

```
SELECT
   SUM(CASE WHEN CAST(id AS REAL) IS NULL THEN 1 ELSE 0 END) AS non_numeric_id_count,
   SUM(CASE WHEN CAST(page_count AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_page_count,
   SUM(CASE WHEN CAST(dish_count AS REAL) IS NULL THEN 1 ELSE 0 END) AS
non_numeric_dish_count
FROM menu_clean
WHERE id IS NOT NULL
   OR page_count IS NOT NULL
   OR dish_count IS NOT NULL;
```

Action : No action needed IC check is passed.

- Check how many records are not in ISO date format

```
SELECT COUNT(*)
FROM menu_clean
WHERE (date IS NOT NULL
      AND (date NOT LIKE '____-__-__'
         AND date NOT LIKE '____-__-__T__:__:%Z'));
```

Outcome:

Message | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 7 | Result 8 | Result 9 | Result 10 | Result 11 | Result 12 | Result 13

COUNT(*)
586

Action : After initial cleaning using OpenRefine on the table 'menu_clean' which contains 1332726 records. However,586 of these records still have non-ISO date formats in the date column. My analysis revealed that date fields are blank which prevent proper date conversion. These remaining records require further cleaning to ensure accurate trend analysis. This additional data cleaning will be implemented in Python as part of the report generation process for trend analysis.

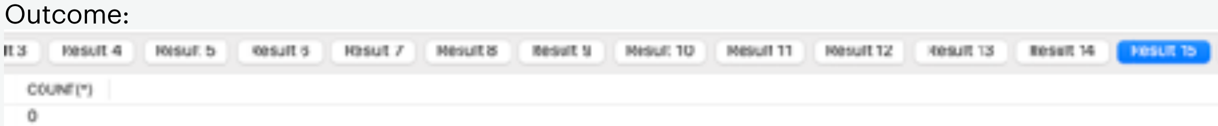
lt 3 | Result 4 | Result 5 | Result 6 | Result 7 | Result 8 | Result 9 | Result 10 | Result 11 | Result 12 | Result 13 | Result 14

id	date
13042	
17603	
19487	
20978	
21467	
21969	
25998	
26119	
26347	

- Check if all special characters are removed successfully from column name,sponsor,event,venue and place.

```
SELECT COUNT(*)
FROM menu_clean
WHERE name LIKE '%(%' OR name LIKE '%)%' OR name LIKE '%\[%' OR name LIKE '%\]%' OR name
LIKE '%{%'
   OR name LIKE '%}%' OR name LIKE '%*%' OR name LIKE '%?%' OR name LIKE '%"%' OR name LIKE
'%''%'
   OR name LIKE '%-%' OR name LIKE '%;%'
   OR sponsor LIKE '%(%' OR sponsor LIKE '%)%' OR sponsor LIKE '%\[%' OR sponsor LIKE '%\]%'
   OR sponsor LIKE '%{%' OR sponsor LIKE '%}%' OR sponsor LIKE '%*%' OR sponsor LIKE '%?%'
   OR sponsor LIKE '%"%' OR sponsor LIKE '%''%' OR sponsor LIKE '%-%' OR sponsor LIKE '%;%'
   OR event LIKE '%(%' OR event LIKE '%)%' OR event LIKE '%\[%' OR event LIKE '%\]%' OR event
LIKE '%{%'
   OR event LIKE '%}%' OR event LIKE '%*%' OR event LIKE '%?%' OR event LIKE '%"%' OR event LIKE
'%''%'
   OR event LIKE '%-%' OR event LIKE '%;%'
   OR venue LIKE '%(%' OR venue LIKE '%)%' OR venue LIKE '%\[%' OR venue LIKE '%\]%' OR venue
LIKE '%{%'
   OR venue LIKE '%}%' OR venue LIKE '%*%' OR venue LIKE '%?%' OR venue LIKE '%"%' OR venue
LIKE '%''%'
   OR venue LIKE '%-%' OR venue LIKE '%;%'
   OR place LIKE '%(%' OR place LIKE '%)%' OR place LIKE '%\[%' OR place LIKE '%\]%' OR place
LIKE '%{%'
   OR place LIKE '%}%' OR place LIKE '%*%' OR place LIKE '%?%' OR place LIKE '%"%' OR place LIKE
'%''%'
   OR place LIKE '%-%' OR place LIKE ‘%;%’;
```

Outcome:



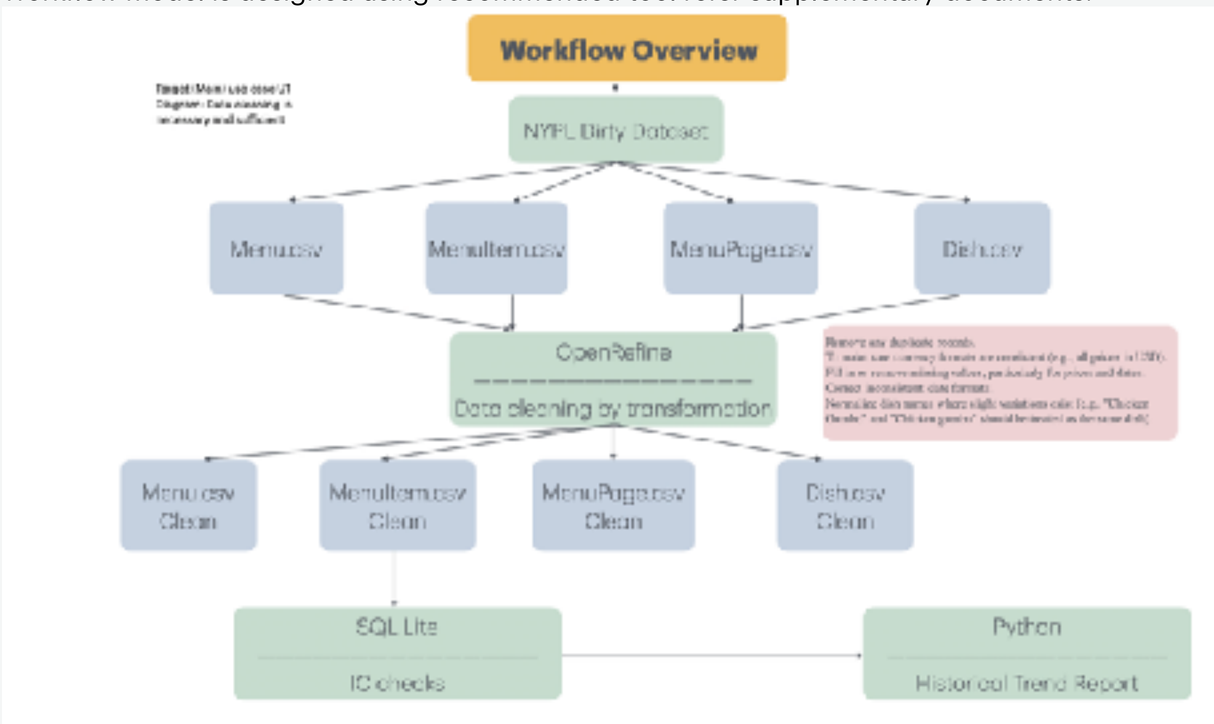
Before :

Action : No action needed IC check is passed.

Note: Not all before screenshots are added here, however to view all before results please check uncleanedSQL.txt

Workflow model

Workflow model is designed using recommended tool refer supplementary documents.



Conclusions & Summary

This project helped me in successfully demonstrating the process of cleaning a dataset and performing integrity checks using OpenRefine, SQLite, and Python. By following a structured approach to data cleaning and verification, All the requested elements are included in my project that ensured the dataset's reliability for analyzing my Use case 1 historical pricing trends. The lessons I learned highlights the importance of meticulous data preparation and the utility of combining different tools to achieve a high-quality dataset ready for analysis.

Highlights of my learning :

- During the project phase I understood that Data preparation is critical & the importance of thoroughly checking for and cleaning special characters, leading/trailing spaces, and ensuring consistent formatting cannot be overstated. These steps are crucial for reliable data analysis.

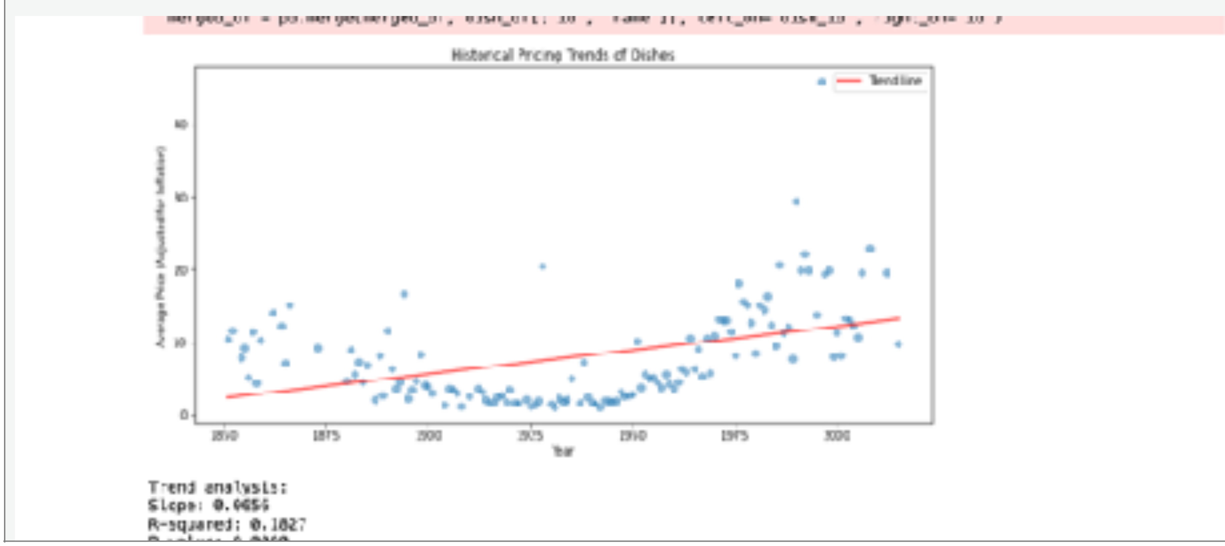
- Writing efficient and accurate SQL queries is essential for database integrity checks & Understanding how to use SQL functions to clean and verify data proved to be invaluable.

- OpenRefine's ability to clean and transform data was leveraged significantly. Understanding its functions and how to replicate them in SQL was an important skill learned during the project.

- Last but not least data cleaning and integrity verification is an iterative process. I invested my time in doing multiple rounds of checks and validations were necessary to ensure data quality.

Reflecting on the completion of the work, as a single-member team, time management was a crucial factor for me. Therefore, for the "Description of Data Cleaning Performed" and "Document Data Quality Changes" parts of Phase 2, I adopted a parallel processing approach. I picked each dataset, performed data cleaning, and documented the changes simultaneously. This method helped avoid recall time at the end and ensured that I stayed on track to finish all my tasks on time.In the end, I used Python to generate a Historical Pricing Trend to show the outcome for Use Case

For the code, refer to my UseCase1_HistoricalTrend_Clean.py file zipped in the supplementary folder.



Checklist of supplementary materials in a single ZIP file

Here are list of files I saved in my supplementary folder - SupplementaryFolder_anjali and submitted on Coursera.

- Workflow Model -attached pdf & .yw in folder.
- Operation History : 4 OpenRefine Recipe Dish.json,MenuItem.json,MenuPage.json and Menu.json.
- UseCase1_HistoricalTrend_Clean.py file to show code outcome after cleaning dataset.
- Querries :queries.txt contains SQL validating Integrity Constraints Checks
- Original (“dirty”) and Cleaned datasets : DataLinks.txt contains box folder link to Parent folder NYPL-menus and it consist of NYPL-menus_clean and NYPL-menus_dirty.