**Data Clorax  
CS513 Data Cleansing**

Final Project Report

July 5, 2020

Asad Bin Imtiaz, Muhammad Rafay

# Introduction

This report summarizes the methods and tools, as well as the analysis and findings carried out for data wrangling, standardization and provenance workflow for the [1] *The New York Public Library* (NYPL)*, What's on the menu?* Dataset*.* The dataset can be downloaded from the NYPL GitHub website [2]. The analysis and findings are part of Final Project for CS513: Theory and Practice of Data Cleaning Course from the university of Illinois, wherein the NYPL dataset [2] was used to create an end-to-end data wrangling and provenance workflow, together with data landscape analysis and findings, using data provenance and data cleansing techniques learned in the class. The goal of this project was to use several open-source tools and libraries for Data Wrangling and Data Provenance to come up with Data Cleaning Workflow which effectively cleans the selected dataset to high-quality standards with all the lineage and audit tracking available.

## Tools and libraries

Following tools were used in this report:

* Python 3 with Jupyter notebook
* OpenRefine data cleaning tool [3]
* SQLLite [4] with DB-Visualizer Pro 9.2 [5]
* Yes Workflow [4]
* Teradata 14 DWH [7]

## Dataset

The New York Public library (NYPL) maintains a large collection of Menus (~45K) in their 'What's on the Menu' [2] dataset, which is openly available to download [1]. The dataset consists of CSV files with entities such as dish-by-dish menus from a variety of businesses from as early as 1850, and are used by historians, nutritionists and researchers around the globe to understand the patterns and to answer specific questions. The data is collected by taking photographs of menus over several years by volunteers and was digitized in the dataset form in NYPL Digital Gallery [1].

As with all the crowd-sourced gathered data, there are several gaps and inconsistencies in the data, as well as areas with potential for improvement in terms of the data formats, linking & lineage and its schema. The goal of this project is to identify the issues and fix them, keeping the provenance and transformation lineage to understand the cleansing workflow and to later reproduce the cleaned dataset on newer dataset versions.

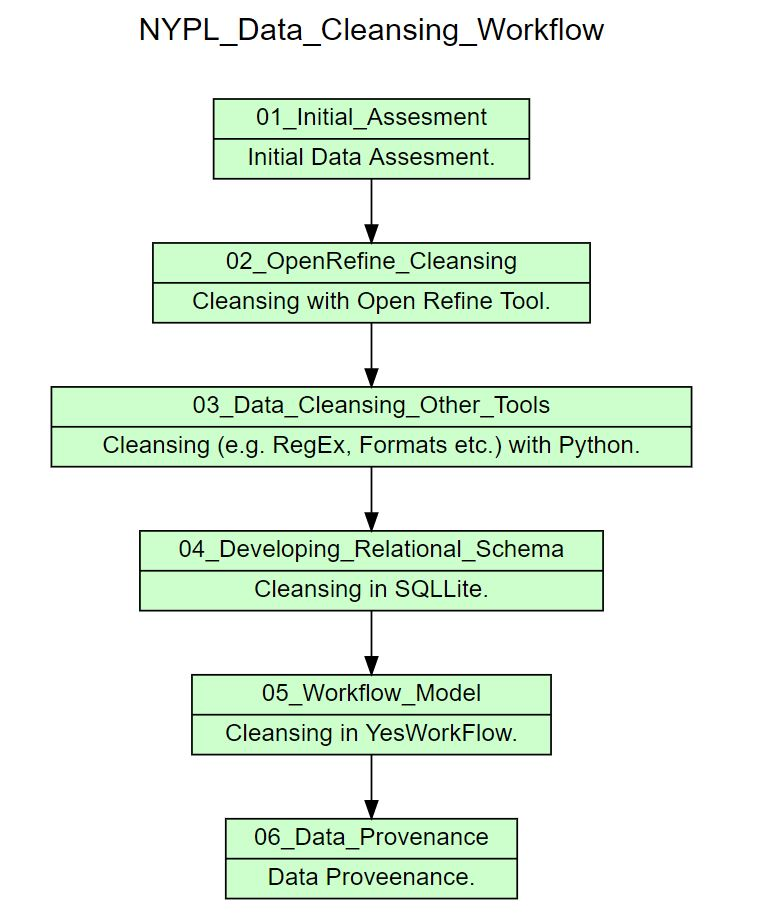
The initial assessment of data quality and respective issues are presented in chapter 2 in detail.

## Approach

The project work was divided into multiple tasks. Below is the task breakdown:

* Overview and initial assessment of the dataset.
* Data cleaning with Open-Refine [3]
* Data cleaning with other tools
* Developing a relational schema
* Creating a workflow model
* Developing provenance

Following is the high-level workflow illustration for the above tasks:



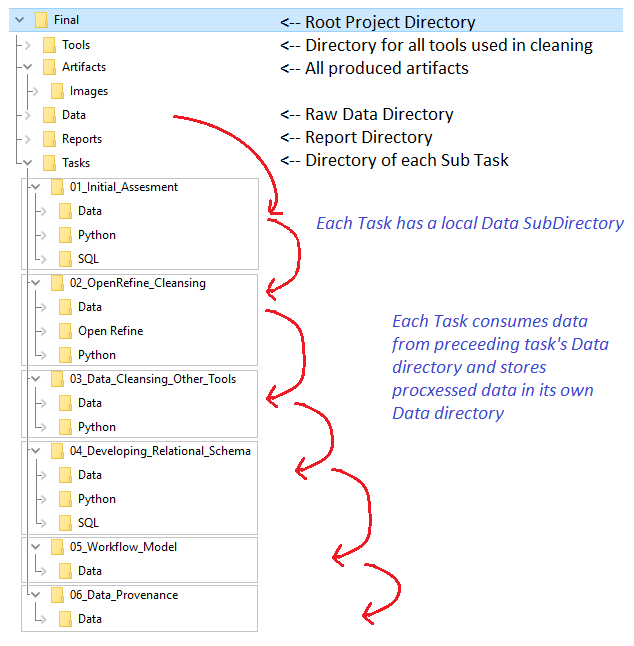
Each of the subtasks is discussed as a separate chapter in the following.

## Project Setup

The project directories are set up hierarchically. The raw data is present in the root project directory in a subdirectory called **Data/**. The **Artifacts/** directory in the root project directory is for any artifacts, such as images generated by the data wrangling tasks. The **Tools/** directory contains executable tools used for data cleaning (such as SQLLite) so that workflow could be started without any installations.

Each of the above 6 tasks has a subdirectory in the **Tasks/** directory. The project is set up in a directory structure where each of the above 6 tasks has their **Data/** directories, under the **Tasks/<Task\_Name>/** directory in the project root directory. The First task, takes the data from the Raw Data directory in the root project folder (**<Root>/Data/)** and stores processed data in its local directory (**<Root>/Tasks/<Task\_Name>/Data/**). Each subsequent task takes the data from the preceding task’s Data directory and stores it in its own Data directory after processing. This directory structure is structured so that retrospective provenance of data flow can be maintained for each step.

Below is how the project directory setup is built.



The **workflow.sh** or **workflow-annotated.sh** are the main workflow scripts that initiate the entire cleaning pipeline.

The project repository is located on Git-Hub [8] at:

<https://github.com/AsadBinImtiaz/CS513_Data_Cleaning_Final>

SHA-1 Key (Read-Only) to access the repository can be found in appendix of this report.

# Overview and initial assessment of the dataset

In the following subsection, the structure and content on the dataset are inspected before starting with the data wrangling and provenance workflow, to get familiarity with data schema and a feel for apparent data quality issues present in the data. There may be more issues in data that would be discussed in subsequent chapters with corresponding tasks.

The initial assessment was performed to get an understanding of the data quality in general and to identify methodology and tools for subsequent tasks. Scoped in this task was also an exploration of the data structures and types, and an understanding of the relationship among these entities.

In the following chapters is the description of the data and data objects, as an outcome of the initial assessment.

## Data Structure

The entire dataset consists of four character-delimited files described below:

1. ***Dish.csv***

This file contains all dishes with their dish names listed on the menu along with their respective pricing and chronology information. Each record represents a specific dish offered by a business and listed on the menu. Each dish has an identifier that uniquely identifies it and is referenced as a foreign key on other entities. The file may be considered as dimension entity for dishes.

1. ***MenuItem.csv***

This file contains menu items that link a menu page entity with dish entities as foreign references. Each record is identified by a unique identifier and carries other information such as associated dish price and x/y position of the image of the menu page. Each menu item refers to a dish in the dish entity and a menu page in a menupage entity, thereby creating a link between a dish and a menu page.

1. ***MenuPage.csv***

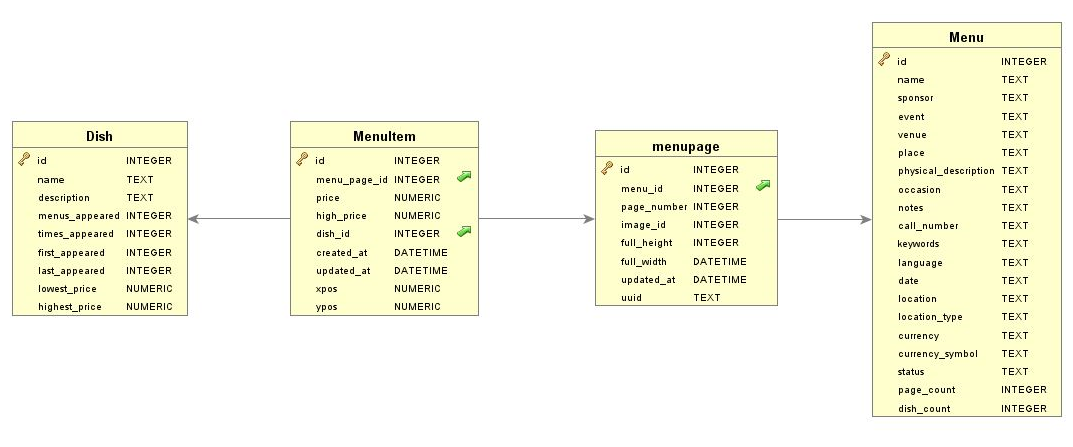
This file contains menu page records. Each item is identified by a unique identifier and links a menu item with a menu. Additional information such as page photo image number and page dimensions also appear here. Every record keeps references for the menu item identifier and menu identifier to link these entities together.

1. ***Menu.csv***

This file contains all individual menus, each associated with a unique id. Each menu has an identifier that uniquely identifies it and is referenced as a foreign key on other entities. Associated data includes the occasion, venue, and event information and chronological information such as created and updated dates and times. Other important fields present in this file include the location where the menu is offered, the associated currency in use for menu items, the language, and the status of the menu.

## Data Structure

The raw data were imported in an SQL-Lite instance and visualized using the DB-Visualizer tool. The ER diagram generated from DB-Visualizer is shown in the figure below:



The cardinalities of objects with respect to one another are found to be (only list for entities among where a direct link is possible):

|  |  |  |
| --- | --- | --- |
| **From Entity** | **To Entity** | **Cardinality of relation** |
| Dish | MenuItem | 1:N |
| MenuItem | Dish | 1:1 |
| ManuItem | MenuPage | 1:1 |
| MenuPage | MenuItem | 1:N |
| MenuPage | Menu | 1:1 |
| Menu | MenuPage | 1:N |

Given the cardinalities above and the Initial Quality Assessment of the dataset in section 2.4, we have assessed that some rows are duplicate in the data and they need to be merged according to certain criteria.

**e.g.** in Dish.csv, the following dish has a duplicate due to difference in cases

Dish.id = 1 , Dish.name = 'Consomme printaniere royal'

Dish.id = 397198, Dish.name = 'Consomme Printaniere Royal'

Although the cardinality from MenuItem to Dish of 1:1 is maintained due to the unique Dish.id but just due to differences in cases for dish names, there should not exist multiple records representing the same dish. These problems can only be analyzed and mitigated after the initial cleansing.

## Data Types

The diagram shows entities and links for data objects present in the data set. Most of the raw data was imported as strings of characters. However, the initial assessment showed the following data types for the fields:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entity: **Dish** | | | | |  |
| Field Name | Type | Precision | Format | Key | Null |
| Id | Integer |  | (10)9 | PK | N |
| Name | String | 1387 | X(1387) Unicode |  | N |
| Description | String | 0 | X(1) |  | Y |
| Menus\_appeared | Integer |  | -(10)9 |  | N |
| Times\_Appeared | Integer |  | -(10)9 |  | N |
| First\_Appeared | Integer |  | (4)9 |  | N |
| Last\_Appeared | Integer |  | (4)9 |  | N |
| Lowest\_Price | Numeric | 2 | --------.99 |  | Y |
| Highest\_Price | Numeric | 2 | --------.99 |  | Y |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entity: **MenuItem** | | | | |  |
| Field Name | Type | Precision | Format | Key | Null |
| Id | Integer |  | (10)9 | PK | N |
| Menu\_Page\_Id | Integer |  | (10)9 | FK | N |
| Price | Numeric | 2 | ----.99 |  | Y |
| High\_price | Numeric | 2 | ----.99 |  | Y |
| Dish\_id | Integer |  | (10)9 | FK | Y |
| Created\_at | Timestamp(0)  With zone |  | YYYY-MM-DD  hh:mm:ss(0) Z |  | N |
| Updated\_at | Timestamp(0)  With zone |  | YYYY-MM-DD  hh:mm:ss(0) Z |  | N |
| Xpos | Numeric | 6 | -.999999 |  | N |
| Ypos | Numeric | 6 | -.999999 |  | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entity: **MenuPage** | | | | |  |
| Field Name | Type | Precision | Format | Key | Null |
| Id | Integer |  | (10)9 | PK | N |
| Menu\_Id | Integer |  | (10)9 | FK | N |
| Page\_Number | Integer |  | ----.99 |  | Y |
| Image\_Id | String | 15 | X(15) |  | N |
| Full\_height | Integer |  | (4)9 |  | Y |
| Full\_width | Integer |  | (4)9 |  | Y |
| Updated\_at | String | 36 | X(36) [UUID] |  | Y |
| Uuid | Numeric | 2 | -.999999 |  | Y |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entity: **Menu** | | | | |  |
| Field Name | Type | Precision | Format | Key | Null |
| Id | Integer |  | (10)9 | PK | N |
| name | String |  | (10)9 | FK | N |
| sponsor | String |  | ----.99 |  | Y |
| event | String | 15 | X(15) |  | Y |
| venue | String |  | (4)9 |  | Y |
| place | String |  | (4)9 |  | Y |
| physical\_description | String | 36 | X(36) [UUID] |  | Y |
| occasion | String | 2 | -.999999 |  | Y |
| notes | String |  | X(260) |  | Y |
| call\_number | String |  | X(40) |  | Y |
| keywords | String |  | X(0) |  | Y |
| language | String |  | X(0) |  | Y |
| date | Date |  | YYYY-MM-HH |  | Y |
| location | String |  | X(0) |  | Y |
| location\_type | String |  | X(127) |  | Y |
| currency | String |  | X(26) |  | Y |
| currency\_symbol | String |  | X(4) |  | Y |
| status | String |  | X(9) |  | Y |
| page\_count | Integer |  | (10)9 |  | Y |
| dish\_count | Integer |  | (10)9 |  | Y |

## Data Quality initial assessment

After an initial assessment of data, various data quality issues were apparent. The following is a small summary of issues discovered at the initial assessment. following data quality checks were performed and respective violations were listed.

**Dish.csv file:**

In the dish file, the names were not stored in the standard case. Some names are upper-case, some lower-case and other mixed-case. The names also contained extra spaces and invalid characters. The description field was empty. Several dishes had menus\_appeared or times\_appeared value as 0. The correctness of these fields was also not correct; several dishes times\_appeared value less than menus appeared. Similar issues were found with first\_appeared and last\_appeared, with many dishes having the first appearance earlier than the menu date or last appearance year earlier than the first appearance year. there were missing values in prices which may be filled or corrected with prices in menu\_item.

**MenuItem.csv file:**

In the menuItem file, several menus had missing references to dishes. These were referential integrity issues such as a dish reference not existing as a dish in the dish file. Many prices were missing and could be completed from the dish file. The same issues were present for the highest price field. In many cases, the Highest price was less than the lowest price. Similarly, in many cases, updated\_at timestamp is earlier than created\_at timestamp.

**MenuPage.csv file:**

In the MenuPage file, there were referential integrity issues. Several menu ids referenced were non-existent in the menu file. The page numbers were also not correct and sometimes had negative values. The Image id field was also not standard and had alphanumeric values in a few cases.

**Menu.csv file:**

In the Menu file, there were several issues with textual fields, such as missing values, non-standard cases, having invalid characters, etc. There were values such as ‘?’ or ‘[restaurant and/or location not given.]’ with the same semantic as missing values. Several fields such as keywords, languages, and location\_type were empty with no value in the entire data file. Other fields such as status contained only a single value all the time. The dates were also not in a standard and consistent format. The dish count value in the menu in a few cases was not correct if associated dishes were counted. The call numbers, mostly numeric, in few cases, had trailing alpha characters

## Use cases

The usefulness of data can be judged by the potential use case it may serve. The raw data has many issues as listed above but may still be valuable for several analytical scenarios and use cases. However once cleaned, further uses can be foreseen, some of them listed below:

## Fitness for use as is

Although the data is not clean enough for many useful use cases, still here are some.

* The data may be used by business owners to generate menus and see historical variations in the dish listings.
* The data may be used to have an estimate on dish popularity based on the number of times a dish is listed on all menus. It may also be used to find dishes previously listed but not offered anymore as a criterion of the unpopularity of the dish.
* The data may be used for building a New York Menus search engine where users can search for restaurants and the dishes associated with them.
* The data may be used to search businesses (with call number) which offer a specific dish
* One can analyze how menus have changed across time in terms of their page size and number of dishes.

## Fitness for use after data wrangling

Once the initial quality issues have been addressed the data will be fit for the following and other similar use cases:

* All use cases for which the data may be used without cleaning may be served with much higher quality after the data is cleansed.
* Once gaps in the data are filled and formats are standardized, Machine Learning models may be applied to uncover patterns in dish offerings, prices, and locations.
* Once the correct chronology of menu listings may be established, the dataset may be used to study how eating preferences evolved by correlating them with the popularity of dishes for a given interval.
* Once duplication in dishes is merged, upselling, cross-selling, and competitive pricing analysis for dishes across locations may be performed.
* If a review dataset (such as yelp reviews) may be combined with this data (based on cleansed business names etc.), rating or sentiment analysis of dishes may be performed per restaurant.
* If the dataset may be mapped with geo-coordinates, location-based dish and menu analysis may be performed.
* A recommender system may be used to recommend dishes in a price range or location.

# Data cleaning with OpenRefine (and other tools)

The data was cleaned with OpenRefine Tool (formerly known as GoogleRefione) and other Programming tools such as python and R. In the following these cleaning tasks are mentioned in detail.

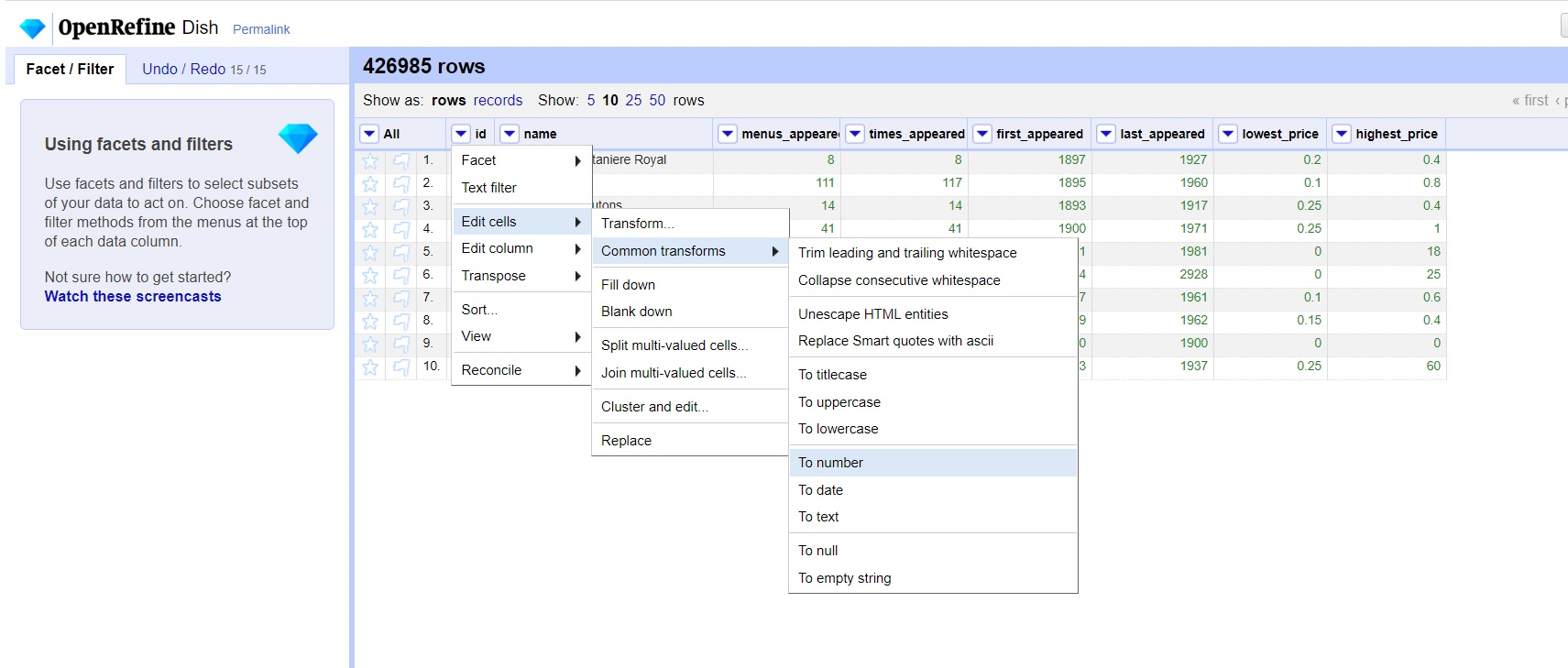
## Identifying Data Cleaning Steps for the use cases

To make the data fit for the above-mentioned use cases, data was needed to be cleaned and standardized. Several cases of missing values in records, non-standard formats, and cases, invalid characters, additional white spaces, inconsistency in spellings of textual, duplicates, and other issues involving completeness and correctness were identified in Initial Assessment. Many of these were fixed with the OpenRefine tool. Moreover, there existed violations such as uniqueness and referential integrity among the data entries but these addressed in Section 4 with a relational database.

OpenRefine allowed us to facet, filter, transform, and cluster data. It also allows operations on data using the GREL language. All features available under a GUI makes it easy to learn and use. It also keeps a track of data provenance. But still, it has its limitations so further cleaning process using completed python language for the tasks that we won’t be able to do using OpenRefine such as regular expression matching. We will be using the pandas library which is a very fast data analysis and manipulation tool in python.

## Data cleaning with OpenRefine

Open Refine is an open-source tool for data wrangling. It reads the CSV files as data tables and performs several data cleaning tasks such as type casting, case standardization, and clustering. OpenRefine was used to cleanse cases of format standardizations, for conversions of type and clustering base of same or similar values together in textual data.



**Note**: All file were read with encoding= ‘**utf-8**’

## Open Refine Data Cleaning Process

**Dish.csv file:**

There were several issues identified which were suitable for cleansing with OpenRefine. The below table contains a summary of these issues and their rectification in OpenRefine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field(s)** | **Issue type** | **Issue Description** | **Resolution** |
| ID  menus\_appeared  times\_appeared  first\_appeared  last\_appeared  lowest\_price  highest\_price | Type Cast | Convert to Number | Text transformed with **toNumber()**  **426985** Records affected |
| Description | Completeness | The entire field is empty and unusable | **1** Column removed |
| Name | Extra Spaces | There is extra space   * Leading dish names * Training dish names * In between dish names   e.g.  Dish.id = 131274, Dish.name = 'Consomme printaniere royal' | Text transformed with **trim()**  **9226 cells updated** |
| Name | Extra Spaces | There is extra space   * Leading dish names * Training dish names * In between dish names   e.g.  Dish.id = 131274, Dish.name = 'Consomme printaniere royal'  Dish.id = 397198, Dish.name = ' " " kidneys' | Text transformed with **replace(/\s+/,’ ‘)**  **6554 cells updated** |
| Name | Case Standardization | The name does not appear in the standard case. Some names are upper-case, some lower-case and other mixed-case.   * There are 426985 distinct dishes * There are 398443 distinct dishes case insensitive | Text transformed to title case with **toTitlecase()**  **284173 cells updated** |
| Name | Clustering | Similar names to be clustered. | Clustered 4006 texts with **MassEdit** |
| Name | Invalid characters | There are invalid characters like (!,@,#,{ etc.) in dish names  e.g.  Dish.id = 2839, Dish.name = 'E. & J. B. \*\*\*' | Removed invalid characters with **GREL**  **6554 cells updated** |

In the project directory, **Tasks/02\_OpenRefine\_Cleansing/Open Refine/** the audit log for Dish table can be found in **Dish.json**

**MenuItem.csv file:**

Menu Items did not have a lot of text fields, therefore much of the cleansing was done later with SQL in a database. The below table contains a summary of issues identified and their rectification in OpenRefine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field(s)** | **Issue type** | **Issue Description** | **Resolution** |
| Id  menu\_page\_id  price  high\_price  dish\_id  xpos  ypos | Type Cast | Convert to Number | Text transformed with  **toNumber()**  **1334419** Records updated |
| created\_at  updated\_at | Type Cast | Convert to Date in format  YYYY-MM-DD HH:MI:SS  without time zone | **grel:value.split(‘UTC’)[0]**and **toDate()**  **1334419** Records updated |

In the project directory, **Tasks/02\_OpenRefine\_Cleansing/Open Refine/** the audit log for Menu Items table can be found in **MenuItem.json**

**MenuPage.csv file:**

Menu page records also did not have a lot of text fields, therefore much of the cleansing was done later with SQL in a database. The below table contains a summary of issues identified and their rectification in OpenRefine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field(s)** | **Issue type** | **Issue Description** | **Resolution** |
| Id  menu\_id  Page\_number  full\_height  full\_width | Type Cast | Convert to Number | Text transformed with  **toNumber()**  **66937** Records updated |
| image\_id | Correctness transform | remove leading/trailing characters from image number | **Text transform with grel:**  **value.match(/(\w+)\*(\d+)/)**  **23** cells updated |
| image\_id | Type Cast | Convert to Number | Text transformed with  **toNumber()**  **66608** Records updated |

In the project directory, **Tasks/02\_OpenRefine\_Cleansing/Open Refine/** the audit log for Menu page table can be found in **MenuPage.json**

**Menu.csv file:**

The menu file contained a lot of text fields, ideal for cleansing with the OpenRefine tool. It had a lot of potential for clustering and pattern matching for the cleansing of fields. The below table contains a summary of issues identified and their rectification in OpenRefine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field(s)** | **Issue type** | **Issue Description** | **Resolution** |
| Id  Page\_Count  Dish\_Count | Type Cast | Convert to Number | Text transformed with  **toNumber()** |
| Keywords  Language  location\_Type | Completeness  Empty fields | The entire fields are empty and unusable | 3 Columns removed |
| Name | Standardization | The values do not appear in the standard case. Some values are upper-case, some lower-case, and other mixed-case. | Text transformed to title case with **toUppercase()** |
| Name | Correctness | Remove Extra spaces | Text transformed to title case with **trim() and replace(/\s+/,’ ‘)**  **9+1 cells updated** |
| Name | Clustering | Similar names to be clustered. | Clustered 588 texts with **MassEdit** |
| Sponsor | Correctness | Remove Extra spaces | Text transformed to title case with **trim() and replace(/\s+/,’ ‘)**  **3+6 cells updated** |
| Sponsor | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toUppercase()** |
| Sponsor | Clustering | Similar names to be clustered. | Clustered 899 cells with **MassEdit** |
| Event | Correctness | Remove Extra spaces | Text transformed to title case with **trim() and replace(/\s+/,’ ‘)**  **3+6 cells updated** |
| Event | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toUppercase()** |
| Event | Clustering | Similar names to be clustered. | Clustered 5314 cells with **MassEdit** |
| Place | Correctness | Remove Extra spaces | Text transformed to title case with **trim()**  **45 cells updated** |
| Place | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toUppercase()**  **899 cells updated** |
| Place | Clustering | Similar names to be clustered. | Clustered 3184 cells with **MassEdit** |
| Physical\_Description | Correctness | Remove Extra spaces | Text transformed to title case with **replace(/\s+/,’ ‘)**  **38 cells updated** |
| Occasion | Correctness | Remove Extra spaces | Text transformed to title case with **replace(/\s+/,’ ‘)**  **3 cells updated** |
| Occasion | Clustering | Similar names to be clustered. | Clustered 2779 cells with **MassEdit** |
| Occasion | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toUppercase()**  **17 cells updated** |
| Notes | Correctness | Remove Extra spaces | Text transformed to title case with **trim() and replace(/\s+/,’ ‘)**  **125+195 cells updated** |
| Notes | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toUppercase()**  **3873 cells updated** |
| Notes | Clustering | Similar names to be clustered. | Clustered 1355 cells with **MassEdit** |
| Call\_Number | Truncation | remove extra suffix characters | Text transformed with **grel:**  **value.split(‘ ’)[0] and value.match(/(\d+-\d+/))[0]**  **1093+22 cells updated** |
| Location | Correctness | Remove Extra spaces | Text transformed to title case with **trim() and replace(/\s+/,’ ‘)**  **14+555 cells updated** |
| Location | Clustering | Similar names to be clustered. | Clustered cells with **MassEdit**  **4205+5+48+24+36 cells updated** |
| Currency | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toTitlecase()**  **118 cells updated** |
| Currency | Correctness | Remove invalid characters | Text transformed with **grel:**  **value.split(‘(‘)[0]**  **43 cells updated** |
| Location  venue  name  event  occasion | Clustering | Clustering with the mass edit | **1283+1463+5104** cells updated  **2243+21** cells updated  **609+55** cells updated  **2325** cells updated  **270+2** cells updated |

In the project directory, **Tasks/02\_OpenRefine\_Cleansing/Open Refine/** the audit log for Menu page table can be found in **Menu.json**

## Data cleaning with Pandas (Python)

Since the title case in OpenRefine does not ignore brackets, e.g for text **“hello (world)”,** OpenRefine would TitleCase it to **“Hello (world)”** and not to **“Hello (World)”**, most case conversions in OpenRefine were limited to UpperCase. Python is intelligent in such conversions, and reconverted appropriately, in our example’s case to **“Hello (World)”**.

With python, the following corrections were performed:

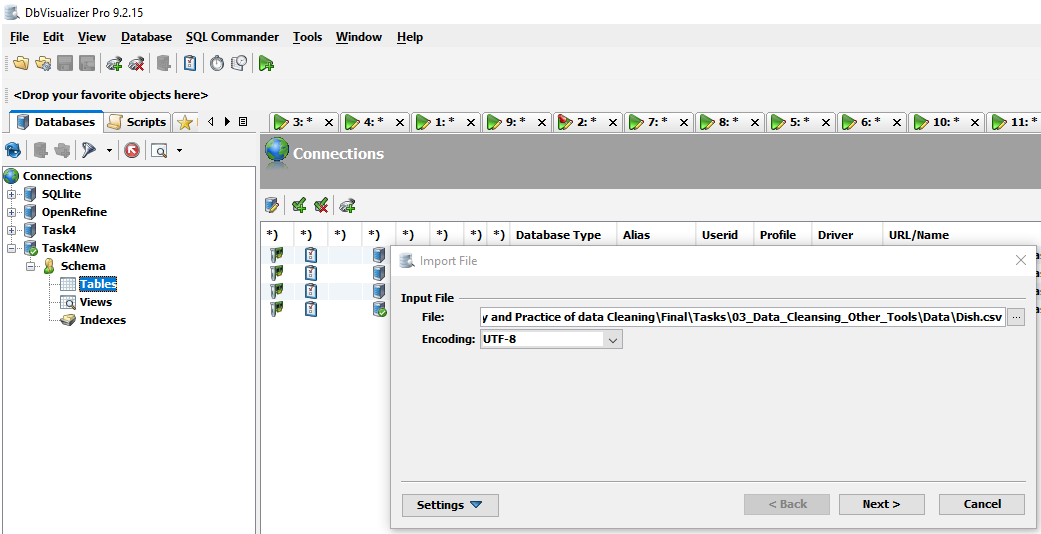
|  |  |  |  |
| --- | --- | --- | --- |
| **File** | **Field(s)** | **Issue Description** | **Resolution** |
| Dish.csv | name | Convert To TitleCase | **dish\_df.name.str.title()** |
| Dish.csv | name | Replace & with And | **dish\_df.name.str.replace(' & ',' And ')** |
| Dish.csv | lowest\_price  highest\_price | Set precision to 2 decimal places | **float\_format='%.2f'** |
| Menu.csv | name  sponsor  event  venue  place  physical\_description  occasion  notes  location | Convert To TitleCase | **str.title()** |
| Menu.csv | Date | Correct Date Format | Format = **‘YYYY-MM-DD’** |
| Menu.csv | name  sponsor  event  venue  place  physical\_description  occasion  notes  location | Replace & with And | **dish\_df.name.str.replace(' & ',' And ')** |
| MenuItem.csv | xpos  ypos | Set precision to 6 decimal places | **float\_format='%.6f'** |
| MenuItem.csv | created\_at  updated\_at | Correct Timestamp Format | Format = **‘YYYY-MM-DD MI:HH:SS’** |
| MenuItem.csv | price  high\_price | Set precision to 2 decimal places | **float\_format='%.2f'** |
| MenuPage.csv | image\_id | remove suffixes: | **str.replace('psnypl\_rbk\_','').replace('ps\_rbk\_','')** |

# Developing a relational schema

In this subtask, the data cleansed with Openrefine and Python was loaded into an SQL-Lite DB and further cleansing was performed with Structured Query Language (SQL). The following subchapters discuss the steps and processes taken to cleanse the data.

## Loading data in SQL-Lite Database

The four CSV files were loaded into an SQL-lite instance via DB visualizer (v9.2.15 pro) using its internal SQL-Lite connector (v3.8).



Once the data was inside the SQL-Lie DB, SQL queries were written to identify issues relating to missing values, domain type conversion, and integrity constraints. These subtasks are discussed in the following chapters.

## Identifying Remaining issues & referential integrity violations

Within the imported data in SQL-Tables, the following issues were identified.

**Table Dish**

|  |  |  |
| --- | --- | --- |
| **Field** | **Issue type** | **Description** |
| Name | Invalid characters | There are invalid characters like (!,@,#,{ etc.) in dish names e.g.  Dish.id = 2839, Dish.name = 'E. & J. B. \*\*\*' |
| Menus\_appeared | Plausibility | There are 2412 Dishes with menus\_appeared = 0 |
| Menus\_appeared | Correctness | There are differences in menus appeared and actual menu count for dish e.g.  id =19, menu\_appeared = 16, actually appeared = 15 |
| Times\_appeared | Plausibility | Several 0 or negative values  -- 1 Dishes appeared -10 times ??? [MIN]  -- 11900 dishes appeared 0 times !!!  -- 372 dishes appeared 19 Menus [MAX] |
| Times\_appeared | Correctness | There are differences in times appeared and actual count for the dish in menus e.g.  id =17, times\_appeared = 535, actually appeared = 536 |
| First\_appreaed | Plausibility | Many dishes have first appearance earlier than menu date or later than last\_appreared year |
| Lowest\_price | Correctness | Dish lowest price should not be negative. Several dishes have 0 lowest prices. It may not be an issue but worth analyzing especially when nulls are allowed. |
| Highest\_price | Correctness | Dish’s highest price should not be negative. |
| Name | Duplications | Same standardized dish name has multiple entries with different context |

**Table MenuItem:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Issue type** | **Description** |
| dish\_id | Null as FK,  lineage has broken | 241 menu items have no value for dish id e.g.  menu\_item.id = 19171 , Menu\_item.dish\_id = NULL |
| dish\_id | Referential integrity | 3 dish ids in menu item which do not exist in the dish.csv e.g.  menu\_item.id = 619133 , Menu\_item.dish\_id = 220797 |
| Price | Completeness | More than 446K menus have null in price. It may be overwritten by an average dish price from   * The highest price in menu item (58 cases) * Menu Items for the same dish * Dish lowest and highest prices |
| high\_price | Completeness | More than 1.2M menu items have no high price. It may be overwritten from   * Lowest price in menu item (~800k cases) * Corresponding dish highest prices |
| Price | Correctness | in 1278 cases, High\_price is strictly less than the price e.g.  MenuItem.id = 1455, price = 40, high\_price = 0.4 |
| Created\_at | plausibility | in 2874 cases updated\_at timestamp is earlier than created\_at timestamp |

**Table MenuPage**

|  |  |  |
| --- | --- | --- |
| **Field** | **Issue type** | **Description** |
| id | Referential integrity | 40334 Menu pages are not referred to by any menu items. This may not be a problem (e.g. title page etc.) but worth analyzing. |
| menu\_id | Referential integrity | 5803 Menu ids in menu page which do not exist in Menu file e.g.  menu\_page.id = 119 , Menu\_Page.Menu\_id = 12460 |
| page\_number | Plausibility | Range of Page\_numberis (1,74) Is 74 pages long menu plausible? |
| page\_number | Completeness | Missing 1202 values. Can be constructed from image\_id and menu\_id |
| Image\_Id | Correctness | in 92 cases, menupage refers to the same menu\_id and same page\_number but different image\_id |
| full\_height | Completeness | in 329 cases, full\_height is null nut image\_id is known |
| full\_width | Completeness | in 329 cases, full\_width is null nut image\_id is known |
| uuid | Uniqueness | Having UUID in it, this field has 2922 duplicated ids, several ids repeating as many as 10 times. |

**Table Menu**

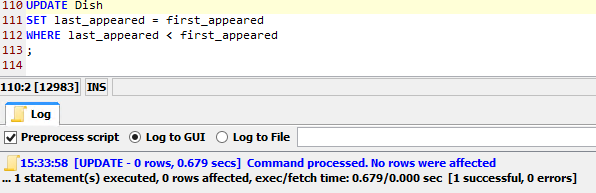
|  |  |  |
| --- | --- | --- |
| **Field** | **Issue type** | **Description** |
| name | Consistency | There are values such as *[not given]* or *[restaurant name and/or location not given]*. Do these values sound reasonable when nulls/blanks are allowed and vice versa? |
| sponsor | Consistency | 57 records have ‘?’ as value. 30 records have ‘[restaurant and/or location not given.]’  Are these reasonable when nulls/blanks are allowed and vice versa? |
| event | Consistency | Similar or same values e.g  dinner, dinner/dinner, [dinner], daily dinner, (dinner)  ?, <Blank> |
| date | Correctness | Few Invalid Dates e.g 2928-03-26  586/17545 Missing Values |
| dish\_count | Correctness | in 214 cases, the value of dish count is different to distinct dishes the menu can be connected to |

## SQL queries to check and meet the integrity constraints

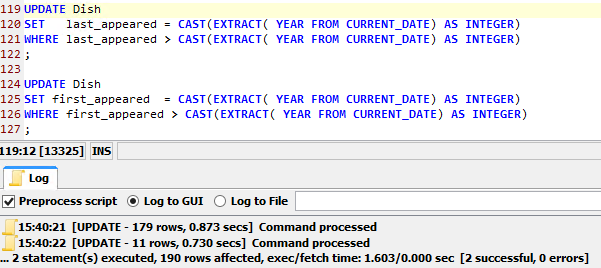
**Table Dish**

The first correction which was performed was a truncation of dish names longer than 500 characters. 104 such dishes were identified and realized to have descriptions along with names. These were manually corrected.

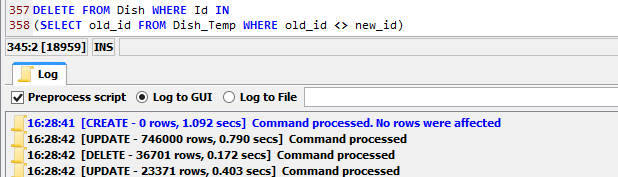
All dish records where last\_appeared as earlier than first\_appeared were corrected.



Moreover, any first or last appearance of a dish later than the current year was adjusted to the current year:

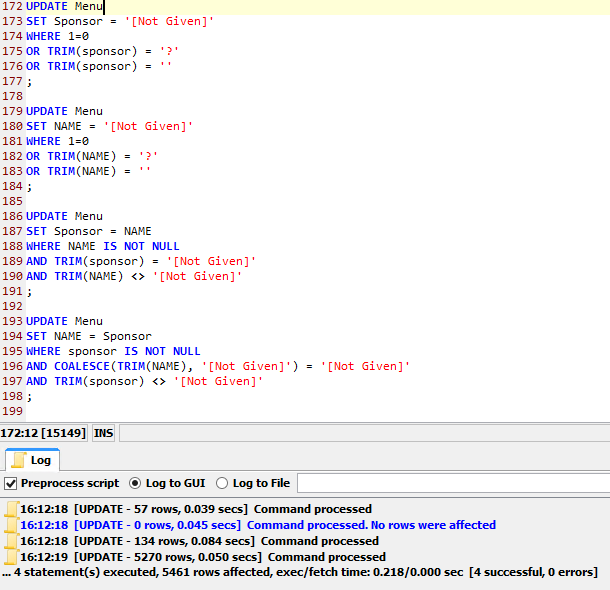


All values for dishes that have the same name were consolidated into single records and extra records were deleted.



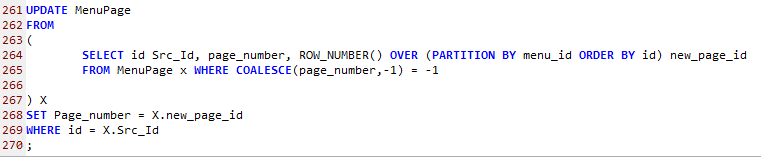
**Table Menu**

The text which represents missing values such as ?,[Not Given], unknown, etc were converted to unique singletons.



**Table MenuPage**

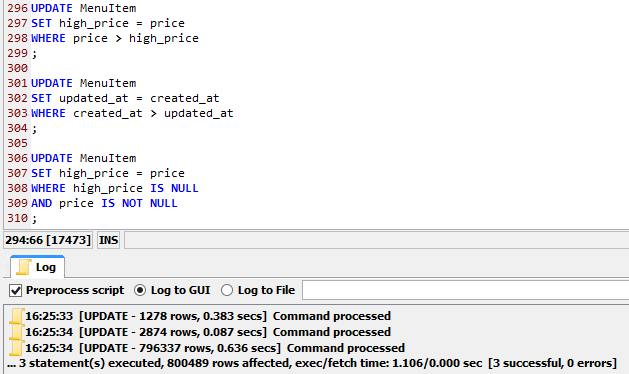
All page numbers in the Menu page which were negative or had gaps or overlaps were resequenced again with image\_ids.



All menu-pages not referencing menus, and all menu-items not referencing dishes or menus were iteratively removed.

**Table MenuItem:**

Records, where high\_price was less than price or created timestamp, was later than updated timestamp were corrected.

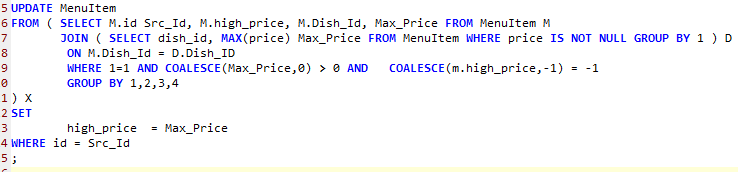


**Fixing RI:**

All records were removed where referenced id did not exist in the referenced table.



The values of times\_appeared in the dish table were correct by counting the number of its appearances in menus. Moreover, the value for menus\_appeared was corrected by associating distinct menus.

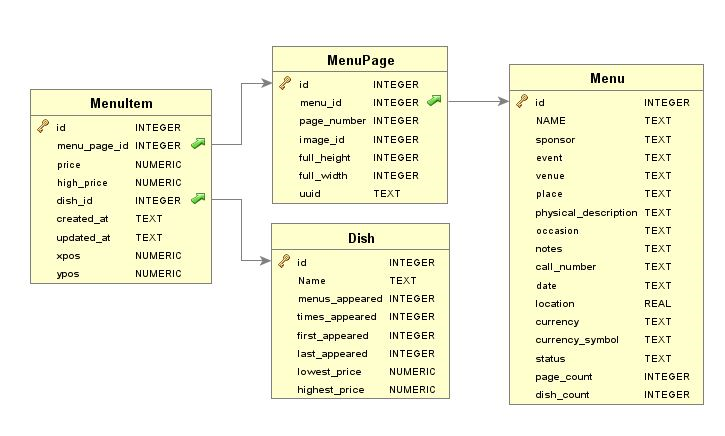




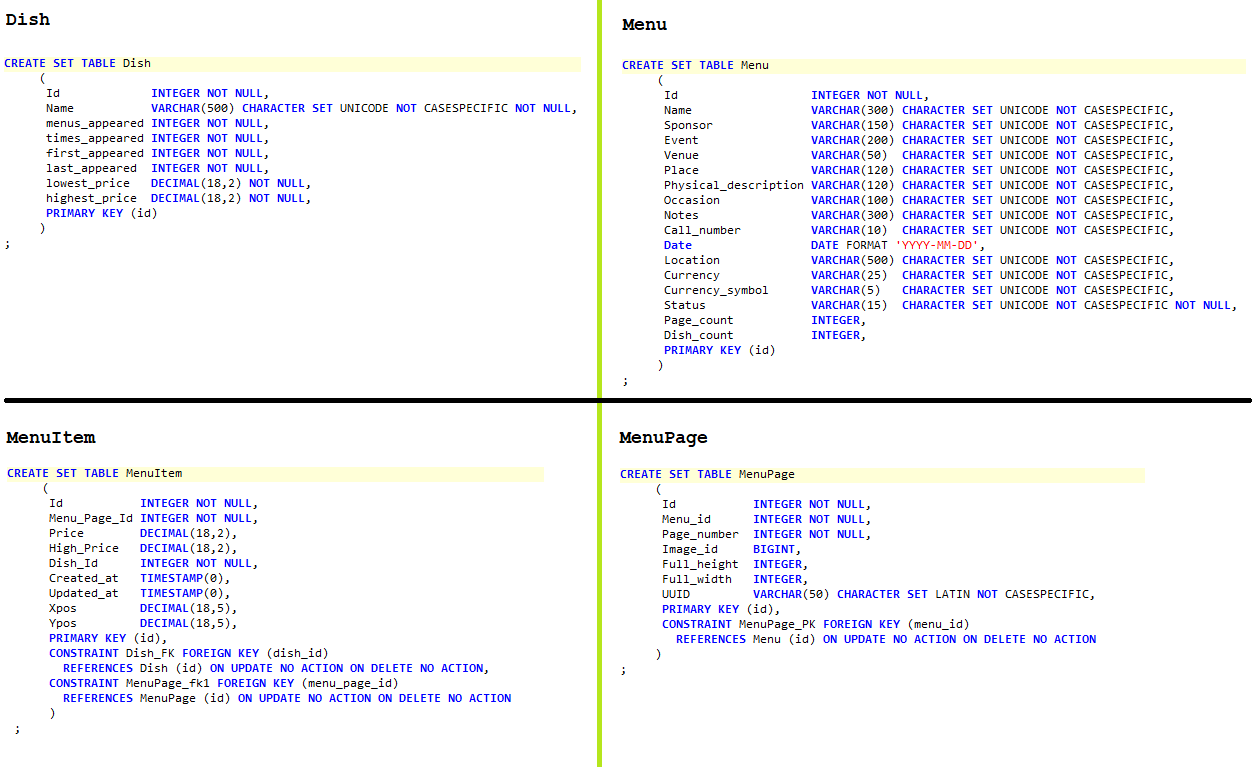
Similarly first\_appeared and last\_appeared in the dish table was corrected from the year of the menu. The Dish prices were also adjusted from MenuItem and vice versa where missing.

## Creating a New SQL Schema for cleaned data

After cleaning the data, a new schema was generated, as shown in the illustration below:



Below are the final Data Definition Language (DDL) definitions for RDBMS. The definitions show each field and its appropriate types and nullability.

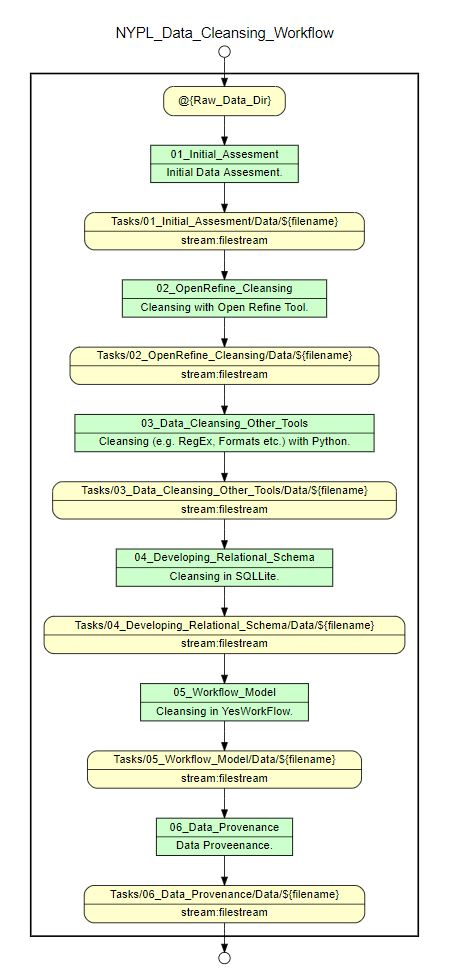


The data was finally exported back into CSV files in the data directory of the respective task.

# Creating a workflow model

The data wrangling pipeline was automated in a workflow wherein input files atr transformed through individual cleaning tasks. For each of these tasks, the data lineage and processing information is retained as provenance information, so that the entire cleanup workflow can be reproduced and audited. In the following chapters, we will discuss several of these workflow components and their provenance model.

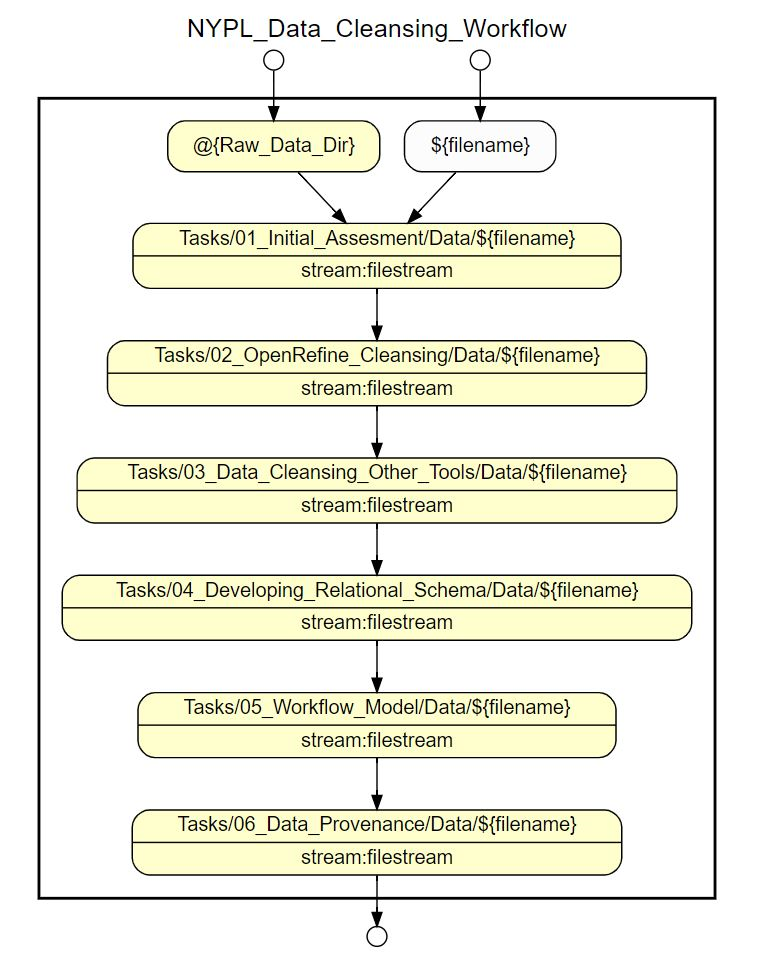
The main tool used for generating workflow provenance diagrams was YesWorkflow Online Editor. Respective workflow jobs were annotated and used to generate the respective workflow diagrams.



## Overall Project workflow

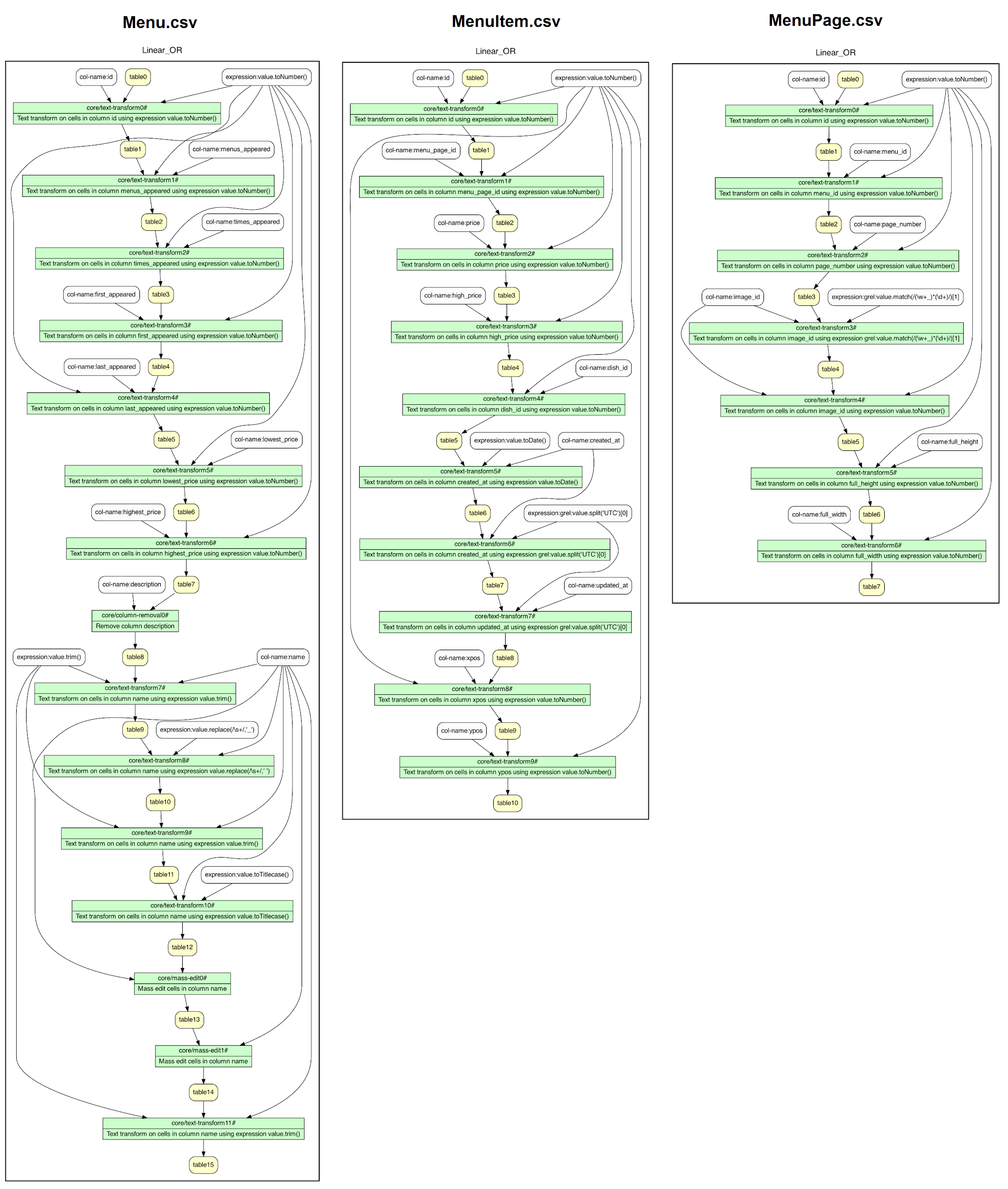
The project data processing pipeline was set up such that each task takes the data from the preceding task’s data directory, processes this data and stores it in its own data directory for the subsequent task. The first task takes the data from the project main raw data directory and the final task produces the cleansed data in its directory as the output of this entire workflow.

The main project workflow (workflow.sh) was annotated with yes workflow annotations and processes through an online editor to have the workflow graph generated for the whole project. For each individual workflow step, the flow of data from raw files to the final output file can be seen in the following graph.

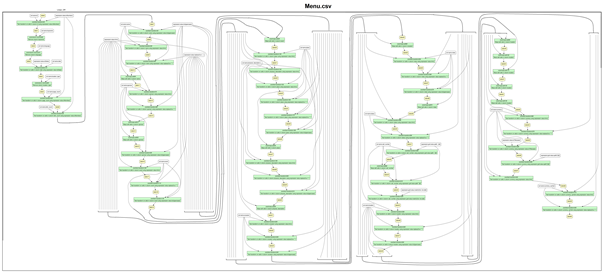


## Open-Refine workflow

In the following are workflow graphs for each of the four input files cleansed with Open-Refine. The OR2YW tool was used to get a visual representation of the OpenRefine workflow. The JSON provenance file from OpenRefine was used to generate graphs for the workflow. The workflow was linear so changing the sequence of steps would not make much of a difference. Since the data was already in four different files in the form of a relational schema we had to clean them separately in OpenRefine. So four workflows were generated using OR2YW. The graphs for three of the input file files (**Dish.csv**, **MenuPage.csv** & **MenuItem.csv**) are as below:

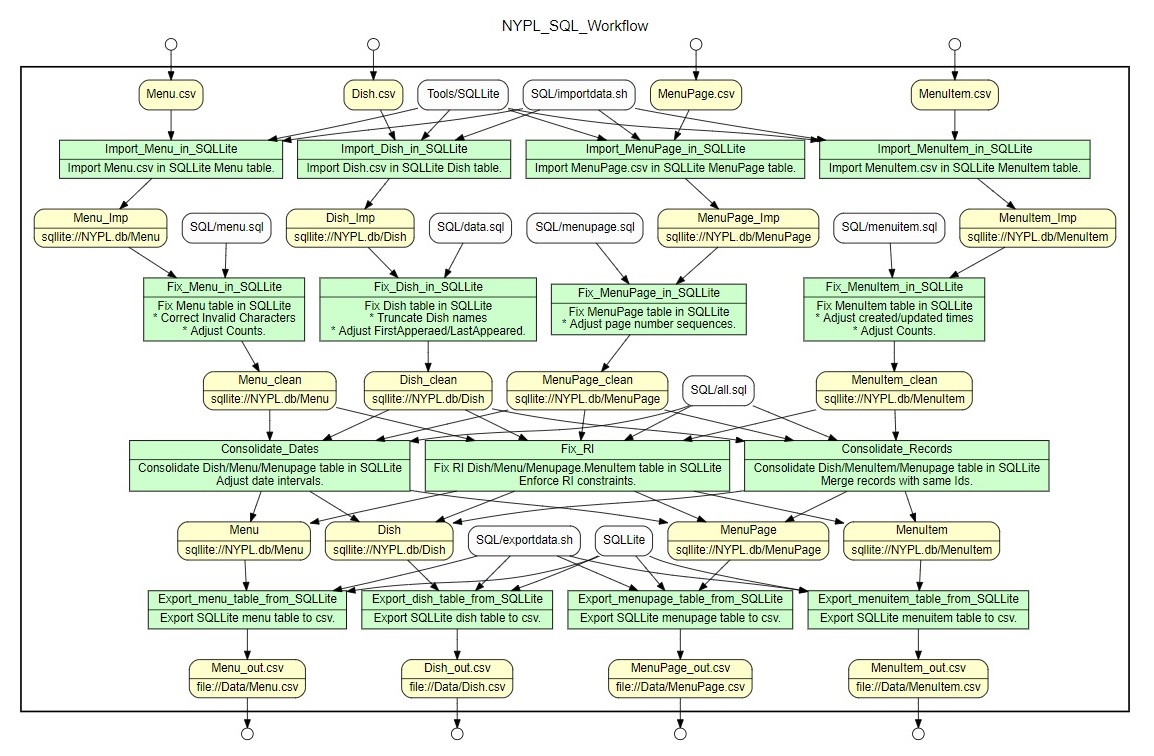


The graph of the **Menu.csv** was a bit longer and is modified to fit this report width and can be seen as below:



## SQL-Lite workflow

Below is the work flow graph of cleansing with SQL-Lite. The input files are the pre-processed CSV file. The output files are exported CSV files after cleansing in SQL-Lite.



Please not that for certain updates, Teradata RDBMs was used to gain performace. The SQL scripts can be fount in respective task's SQL directory.

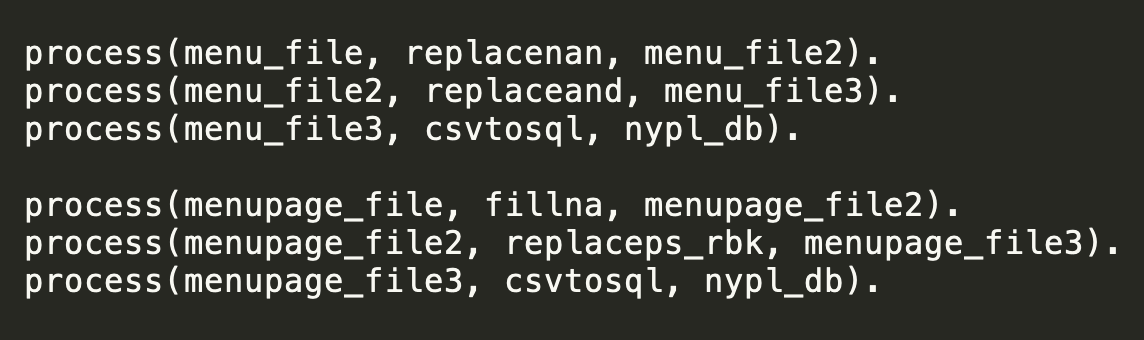
## Workflow inputs, Outputs and dependencies

The following workflow graph identifies key inputs, outputs, and dependencies of the overall workflow.

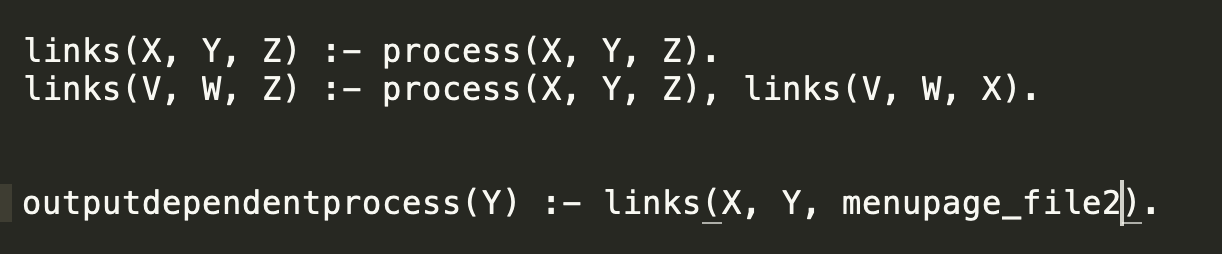


# Developing Provenance

A provenance model was developed using clingo which is an answer set programming language used to solve combinatorial problems. Using clingo a workflow model was defined by defining input-process-output links for each task that was performed input and output being the data involved in the process. An example is shown below:



Then a datalog rule was made to identify all the processes required to obtain a certain output file within the workflow, e.g:



This query outputs the dependent process ***outputdependentprocess(fillna)*** since this output file is dependent on only one process.

Similarity another rule was made to identify all the files required to obtain a certain output file within our workflow:



This query outputs the dependent file ***outputdependentfiles(menupage\_file)*** since this output file is dependent on only one file.

# Conclusions

The data wrangling is a very crucial step in data analysis and comes between data gathering and analysis. Unfortunately, this step is often ignored, and many go directly to analyze or process the data after its sourcing. The effect on analysis results can be drastic depending on whether the data was cleansed or not. Therefore, Data cleaning and wrangling is extremely effective not only in fixing data types and formats or filling in gaps, but also for better analysis results.

In this project, an attempt was made to clean the NYPL "What's on the menu dataset" [4] to make more analysis ready. For this purpose, open source tools such as Open-Refine, SQL-Lite and python were used. The provenance graphs were generated with Yes-Workflow Online editor, as well as with OR2YW tool. For certain tasks for better visualization or performance, closed source tools such as DB-Visualizer and Teradata RDBMs were used, however, these are not required and used by the main project workflow.

# Appendix

1. SSH-1 GitHub Hey:

-----BEGIN OPENSSH PRIVATE KEY-----

b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn

NhAAAAAwEAAQAAAYEAviThY9kFTDex5+W4bZm1sQfDphFNeozLIFH3Q43sym94AXsnnHsu

m+2iLkClnYkKbCzR8bCTP0dQtkdbD9oJIx7wkdcdyM2ieoUenrSMNcVcqG/12sqrPm6SSr

jb28pBFQXj5gIUgkmddxtdbYoj8fKBx+6oMjJg0+6fepIgkArYgPFFh64S8jFy6oqm6rUV

Tfvy+NQYsI4xDsS4nvbbFQfzujCDjd4bh5yhQu3uhXqJ+tsWoDGphIxU1WQx2eMnnCiAZg

1zce6bL5pZfqxBpr4j4u6XJs4WZN3YIbiUleKLZPMPfCibwg9/1m7OXOVzn/yqbNdCA4pJ

TdJ5tpAr7A9IW448SUdhTdduIfVmlHe7K9Psr+KgQMRRF0NLouv4SaKo7Z69LniSILnf9T

Ni4ojQGpI32k2zju3jjBaCcet0IIYy5Jj5pz3HMm+zLfx8MWhyYSrBh9q9mu1s4OLnGYo6

386V0DwE7BK4kDf2WxSUpZ4xcwftK8RyTZEzksbBAAAFkNaNgTHWjYExAAAAB3NzaC1yc2

EAAAGBAL4k4WPZBUw3sefluG2ZtbEHw6YRTXqMyyBR90ON7MpveAF7J5x7Lpvtoi5ApZ2J

Cmws0fGwkz9HULZHWw/aCSMe8JHXHcjNonqFHp60jDXFXKhv9drKqz5ukkq429vKQRUF4+

YCFIJJnXcbXW2KI/HygcfuqDIyYNPun3qSIJAK2IDxRYeuEvIxcuqKpuq1FU378vjUGLCO

MQ7EuJ722xUH87owg43eG4ecoULt7oV6ifrbFqAxqYSMVNVkMdnjJ5wogGYNc3Humy+aWX

6sQaa+I+LulybOFmTd2CG4lJXii2TzD3wom8IPf9Zuzlzlc5/8qmzXQgOKSU3SebaQK+wP

SFuOPElHYU3XbiH1ZpR3uyvT7K/ioEDEURdDS6Lr+EmiqO2evS54kiC53/UzYuKI0BqSN9

pNs47t44wWgnHrdCCGMuSY+ac9xzJvsy38fDFocmEqwYfavZrtbODi5xmKOt/OldA8BOwS

uJA39lsUlKWeMXMH7SvEck2RM5LGwQAAAAMBAAEAAAGAJA1GFvJtCLh+qONrhoxgYMKCHB

inwhaz5NrlsQ9V2EQuUsmRByBsA/CYkYB8ZlQ3f68310WNha/147m/0E+c7+XL4zsQUKR8

rBJVbe35r7BOEzm3Oda3hUzdvAE03oWItX26aj/2t4VRr+WmX0CT9Cnw5YGgrnPS3BwgbN

MMDYOg7UHR1xnwSa+KhGHpqBLRiUm+FePREpjDQHOYNbvEakZpteHA78MvIRVXliMa2IAV

bVThA1qnM5SP+S3YLeHdqtXk3Rep8j8ziFLRxB/sKiCMjLtwYpeMlc6L3Hae4ruoII+EVc

mXHNzxkIatE0es7G/xE+riwJ6+kY1Pd+LBcDQfgyp3LBkKAIz5HT1QU/vwvAVZxLFU0ShK

q078VJyTC5Tb8xG6Iq3AWgH20x6UigZPYPj4A+3Ya9lbbIOy74GzmHufev6jp4sBoZr5n2

iH1t6tP/sle9PwLJ9OoulCzs0qED5N1f68glANZnIwVKgjfL3CerlcZJBrWuiYOpGBAAAA

wQCnblIV92ua8gZWITiYW6Qe1OYtVQxJyZkTRAFhYUDsY69u1e67ZWBrHAD4WUpJYlxxC4

TG/nwRjzsTjQbCi019PDcxTa6a0cCqIOVOPg4ZQnsabDKRq41oWxaiEg0ZhUsGgu1QLwnG

fF8MxfsoGF36bHy31M2+PviI9P779EoDg6zBGp7de6CXe09xXcl1X/q3SQHf2JI4eyTfcQ

qFpzZ7ltEVmmxAK1M7jWuqML6V4iEYqJ68nKEbbU16XbE1TScAAADBAO7tQl7PUGKI3jeQ

yhTfTDU2S09ooB8kLZaJludwppfHgWGNq82ATowEynxEUZviQ2PcR8NWQANAAgopFPI+m0

ya7CmoLlBCizpzNCwsOVjWHXp8JteJqfVvEg9HgSJQ8QasNDO7f4JxJZXh+c8RPqQ9ILTT

WyV2khl5mQu8uWncoMq7y5WOmKxbeFunnvDo8lJ1jFaATBzAJ4XKqiKMt5Fdx/XravhKkP

o97Aal/ezFVxvKlnrw+kfhI57wDdLd2QAAAMEAy7s6Vz/4QEhnz2csdpSLbLx9nOB3i1pS

A0tHeEhwARoygyKz9jrthrfMG5J+htEpo9GnmWTaCyMcMzlMRGA0zP5wuqeqNNe3Q39i/q

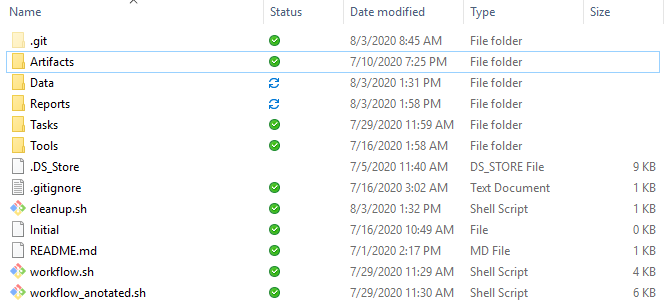
1U7UGDfON1X3lGrE3f0TB/j4tBG9IqOxweY80ETSeR/PufeMmoaFWQ1NvSGzjj0uZ3gVlh

xfa9XvvnIuEhS8nIyUpF2wYY366XD9lcQwwyPKD6AbnNPq1vkMWvF4VyNUaCkfuC00R6nT

JzM2UPaCFu3NcpAAAAFWFzYWRfQERFU0tUT1AtN1E5UE5NOAECAwQF

-----END OPENSSH PRIVATE KEY-----

1. Project Structure:



1. Main workflow script (not annotated):



1. Workflow log:



# Bibliography

|  |  |
| --- | --- |
| [1] | David Huynh, Stefano Mazzocchi, Metaweb Technologies, Inc, "OpenRefine," October 2012. [Online]. Available: https://openrefine.org/. |
| [2] | SQLite Consortium, "SQLite," [Online]. Available: https://www.sqlite.org/index.html. |
| [3] | DBVis Software, "DbVisualizer: A universal database tool," DbVis Software AB, [Online]. Available: https://www.dbvis.com/. |
| [4] | NYPL Labs, "What's on the menu Dataset," 16 June 2020. [Online]. Available: http://menus.nypl.org/data. |
| [5] | NYPL Labs, "Whats-On-The-Menu," June 2020. [Online]. Available: http://nypl.github.io/menus-api/. |