# Data Clorox
# CS513 Data Cleansing

Final Project Report

August 4, 2020

Asad Bin Imtiaz (Net ID: **aimtiaz2**), Muhammad Rafay (Net ID: **mrafay2**)

## 1 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.

### 1.1 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]

### 1.2 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.

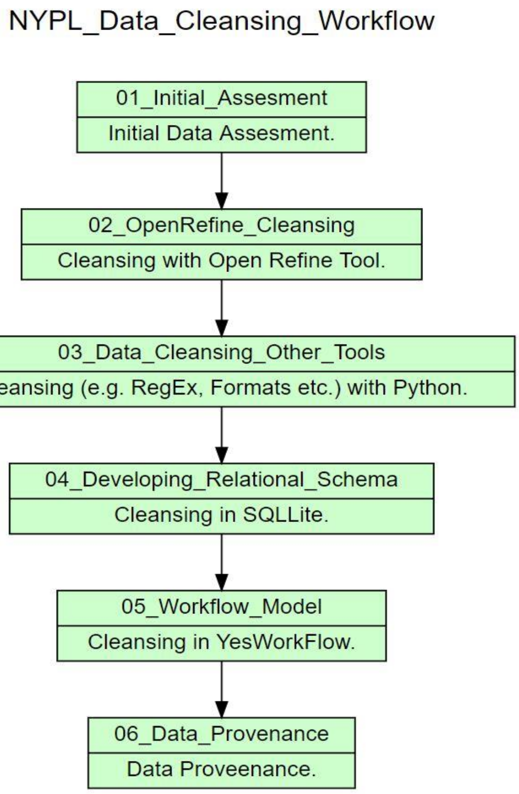
The initial assessment of data quality and respective issues are presented in chapter **Error! Reference source not found.** in detail.

## 1.3 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:


NYPL_Data_Cleansing_Workflow
01_Initial_Assesment
Initial Data Assesment.
02_OpenRefine_Cleansing
Cleansing with Open Refine Tool.
03_Data_Cleansing_Other_Tools
Cleansing (e.g. RegEx, Formats etc.) with Python.
04_Developing_Relational_Schema
Cleansing in SQLLite.
05_Workflow_Model
Cleansing in YesWorkFlow.
06_Data_Provenance
Data Proveenance.

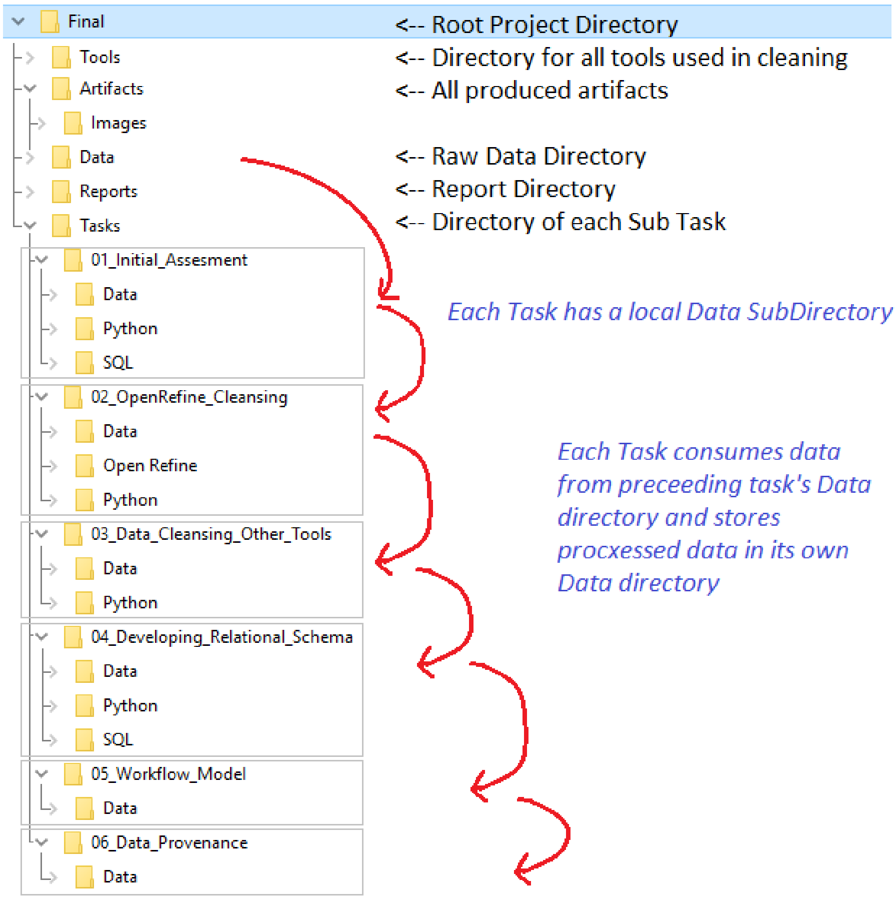
Each of the subtasks is discussed as a separate chapter in the following.

## 1.4 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.



```
Final                                  <-- Root Project Directory
  Tools                                <-- Directory for all tools used in cleaning
  Artifacts                            <-- All produced artifacts
    Images
  Data                                 <-- Raw Data Directory
  Reports                              <-- Report Directory
  Tasks                                <-- Directory of each Sub Task
    01_Initial_Assesment
      Data
      Python
      SQL
    02_OpenRefine_Cleansing
      Data
      Open Refine
      Python
    03_Data_Cleansing_Other_Tools
      Data
      Python
    04_Developing_Relational_Schema
      Data
      Python
      SQL
    05_Workflow_Model
      Data
    06_Data_Provenance
      Data
```

*Each Task has a local Data SubDirectory*

*Each Task consumes data from preceeding task's Data directory and stores procxessed data in its own Data directory*

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.

## 2 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.

## 2.1 Data Structure

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

I. ***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.

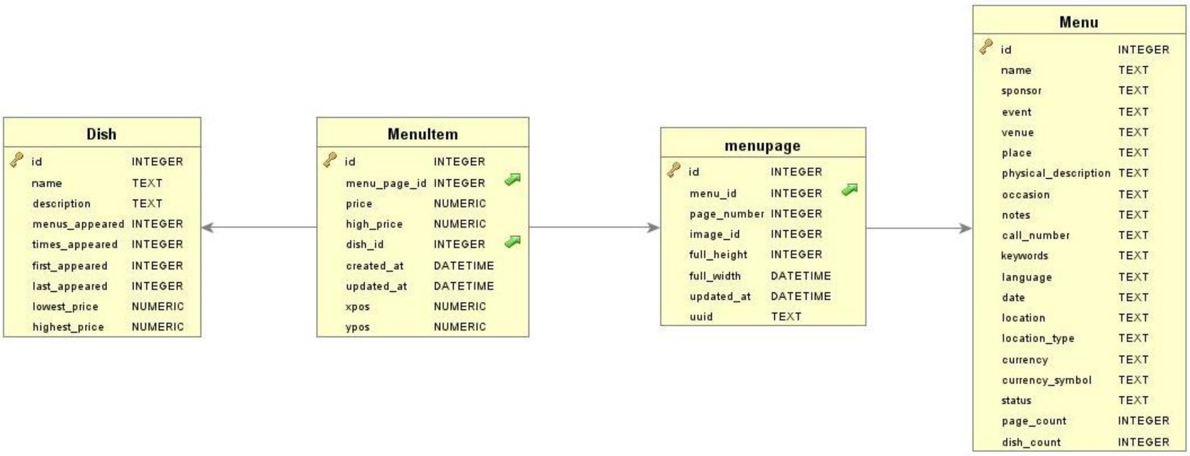
II. ***MenuItem.csv***
This file contains menu items that link a menu page entity with dish entities as foreign references. Each record is identified by a unique identifier and carries other information such as associated dish price and x/y position of the image of the menu page. Each menu item refers 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.

III. ***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.

IV. ***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.

## 2.2 Data Schema

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.

## 2.3 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 |

## 2.4 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 time-stamp is earlier than created_at time-stamp.

**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

## 2.5 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 analytic scenarios and use cases. However once cleaned, further uses can be foreseen, some of them listed below:

### 2.5.1 Fitness for use as is

Although the data in raw form is not clean and has several anomalies, it may still serve several use cases, including but not limited to:

- 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.
- One can analyze how menus have changed across time in terms of their page size and number of dishes.

### 2.5.2 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:

- 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.

### 2.5.3 Unfit for use even after data wrangling

- The data is unfit for rating or performing sentiment analysis of the dishes on its own even after data cleaning without combining it with a review's dataset.
- It is not possible to verify the validity of dish characteristics or their listing, as well as the completeness of the Menus with respect to their images, due to unavailability of this auxiliary data in the raw dataset.

# 3 Data cleaning with Open-Refine (and other tools)

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

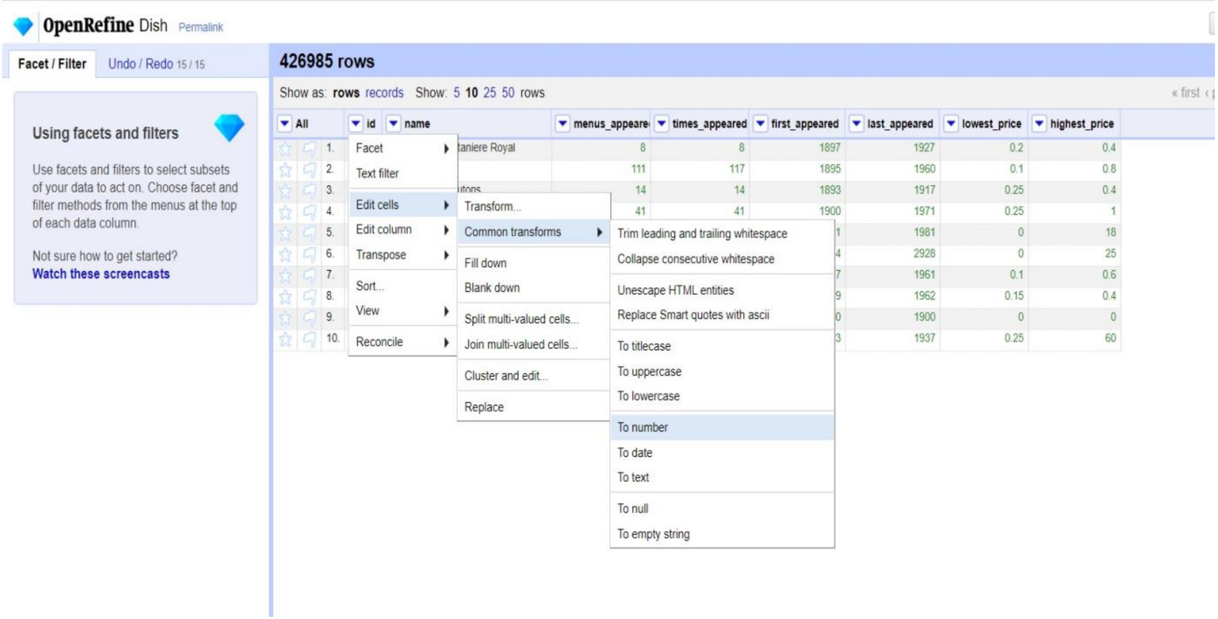
## 3.1 Identifying Data Cleaning Steps for the use cases

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.

## 3.2 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 standardization, for conversions of type and clustering base of same or similar values together in textual data.



Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?
Watch these screencasts

| All | id | name | menus_appeare | times_appeared | first_appeared | last_appeared | lowest_price | highest_price |
|---|---|---|---|---|---|---|---|---|
| 1. | | taniere Royal | 8 | 8 | 1897 | 1927 | 0.2 | 0.4 |
| 2. | | | 111 | 117 | 1895 | 1960 | 0.1 | 0.8 |
| 3. | | itons | 14 | 14 | 1893 | 1917 | 0.25 | 0.4 |
| 4. | | | 41 | 41 | 1900 | 1971 | 0.25 | 1 |
| 5. | | | | | 1 | 1981 | 0 | 18 |
| 6. | | | | | 4 | 2928 | 0 | 25 |
| 7. | | | | | 7 | 1961 | 0.1 | 0.6 |
| 8. | | | | | 9 | 1962 | 0.15 | 0.4 |
| 9. | | | | | 0 | 1900 | 0 | 0 |
| 10. | | | | | 3 | 1937 | 0.25 | 60 |

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

## 3.3 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<br>menus_appeared<br>times_appeared<br>first_appeared<br>last_appeared<br>lowest_price<br>highest_price | Type Cast | Convert to Number | Text transformed with **toNumber()**<br>**426985** Records affected |

| | | | |
|---|---|---|---|
| Description | Completeness | The entire field is empty and unusable | **1** Column removed |
| Name | Extra Spaces | There is extra space<br>● Leading dish names<br>● Training dish names<br>● In between dish names<br>e.g.<br>Dish.id = 131274, Dish.name = 'Consomme printaniere royal' | Text transformed with **trim()**<br>**9226 cells updated** |
| Name | Extra Spaces | There is extra space<br>● Leading dish names<br>● Training dish names<br>● In between dish names<br>e.g.<br>Dish.id = 131274, Dish.name = 'Consomme printaniere royal'<br>Dish.id = 397198, Dish.name = ' "   "   kidneys' | Text transformed with **replace(/\s+/,’ ‘)**<br>**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.<br>● There are 426985 distinct dishes<br>● There are 398443 distinct dishes case insensitive | Text transformed to title case with **toTitlecase()**<br>**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<br>e.g.<br>Dish.id = 2839, Dish.name = 'E. & J. B. ***' | Removed invalid characters with **GREL**<br>**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<br>menu_page_id<br>price<br>high_price<br>dish_id<br>xpos<br>ypos | Type Cast | Convert to Number | Text transformed with **toNumber()**<br>**1334419** Records updated |
| created_at<br>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()**<br>**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<br>menu_id<br>Page_number<br>full_height<br>full_width | Type Cast | Convert to Number | Text transformed with **toNumber()**<br>**66937** Records updated |
| image_id | Correctness transform | remove leading/trailing characters from image number | **Text transform with grel: value.match(/(\w+)*(\d+)/)**<br>**23** cells updated |
| image_id | Type Cast | Convert to Number | Text transformed with **toNumber()**<br>**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<br>Page_Count<br>Dish_Count | Type Cast | Convert to Number | Text transformed with **toNumber()** |
| Keywords<br>Language<br>location_Type | Completeness<br>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+/,’ ‘)**<br>**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+/,’ ‘)**<br>**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+/,’ ‘)**<br>**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()**<br>**45 cells updated** |
| Place | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toUppercase()**<br>**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+/,’ ‘)**<br>**38 cells updated** |
| Occasion | Correctness | Remove Extra spaces | Text transformed to title case with **replace(/\s+/,’ ‘)**<br>**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()**<br>**17 cells updated** |
| Notes | Correctness | Remove Extra spaces | Text transformed to title case with **trim() and replace(/\s+/,’ ‘)**<br>**125+195 cells updated** |
| Notes | Standardization | The values do not appear in the standard case. | Text transformed to title case with **toUppercase()**<br>**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]**<br>**1093+22 cells updated** |
| Location | Correctness | Remove Extra spaces | Text transformed to title case with **trim() and replace(/\s+/,’ ‘)**<br>**14+555 cells updated** |
| Location | Clustering | Similar names to be clustered. | Clustered cells with **MassEdit**<br>**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()**<br>**118 cells updated** |
| Currency | Correctness | Remove invalid characters | Text transformed with **grel: value.split(‘(‘)[0]**<br>**43 cells updated** |
| Location venue | Clustering | Clustering with the mass edit | **1283+1463+5104** cells updated<br>**2243+21** cells updated |

| | | | |
|---|---|---|---|
| name<br>event<br>occasion | | | **609+55** cells updated<br>**2325** cells updated<br>**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**

## 3.4 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:

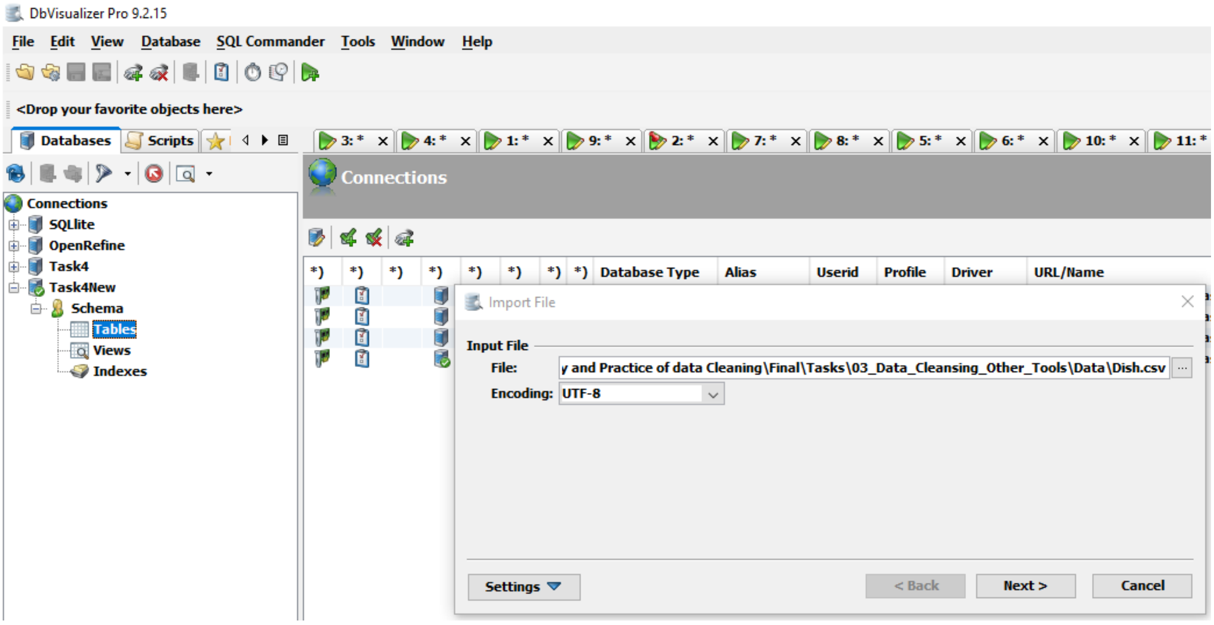
| File | Field(s) | Issue Description | Resolution |
|---|---|---|---|
| Dish.csv | name | Convert To TitleCase | **dish_df.name.str.title()** |
| Dish.csv | name | Replace & with And | **dish_df.name.str.replace(' & ',' And ')** |
| Dish.csv | lowest_price<br>highest_price | Set precision to 2 decimal places | **float_format='%.2f'** |
| Menu.csv | name<br>sponsor<br>event<br>venue<br>place<br>physical_description<br>occasion<br>notes<br>location | Convert To TitleCase | **str.title()** |
| Menu.csv | Date | Correct Date Format | Format = **‘YYYY-MM-DD’** |
| Menu.csv | name<br>sponsor<br>event<br>venue<br>place<br>physical_description<br>occasion<br>notes<br>location | Replace & with And | **dish_df.name.str.replace(' & ',' And ')** |
| MenuItem.csv | xpos<br>ypos | Set precision to 6 decimal places | **float_format='%.6f'** |
| MenuItem.csv | created_at<br>updated_at | Correct Timestamp Format | Format = **‘YYYY-MM-DD MI:HH:SS’** |
| MenuItem.csv | price<br>high_price | Set precision to 2 decimal places | **float_format='%.2f'** |
| MenuPage.csv | image_id | remove suffixes: | **str.replace('psnypl_rbk_',").replace('ps_rbk_',")** |

# 4 Developing a relational schema

In this sub-task, 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 sub-chapters discuss the steps and processes taken to cleanse the data.

## 4.1 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).



Input File

File: y and Practice of data Cleaning\Final\Tasks\03_Data_Cleansing_Other_Tools\Data\Dish.csv

Encoding: UTF-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.

## 4.2 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.<br>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.<br>id =19, menu_appeared = 16, actually appeared = 15 |
| Times_appeared | Plausibility | Several 0 or negative values<br>-- 1 Dishes appeared -10 times ??? [MIN]<br>-- 11900 dishes appeared 0 times !!!<br>-- 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 | Duplication | Same standardized dish name has multiple entries with different context |

**Table MenuItem:**

| Field | Issue type | Description |
|---|---|---|
| dish_id | Null as FK,<br>lineage has broken | 241 menu items have no value for dish id e.g.<br>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.<br>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<br>● The highest price in menu item (58 cases)<br>● Menu Items for the same dish<br>● Dish lowest and highest prices |
| high_price | Completeness | More than 1.2M menu items have no high price. It may be overwritten from<br>● Lowest price in menu item (~800k cases)<br>● Corresponding dish highest prices |
| Price | Correctness | in 1278 cases, High_price is strictly less than the price e.g.<br>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.<br>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**

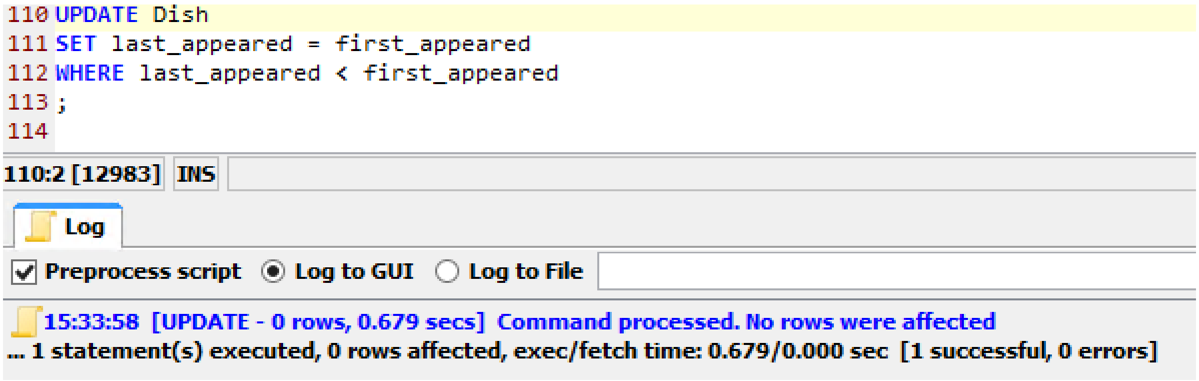
| Field | Issue type | Description |
|---|---|---|
| name | Consistency | There are values such as *[not given]* or *[restaurant name and/or location not given]*. Do these values sound reasonable when nulls/blanks are allowed and vice versa? |
| sponsor | Consistency | 57 records have '?' as value. 30 records have '[restaurant and/or location not given.]'<br>Are these reasonable when nulls/blanks are allowed and vice versa? |
| event | Consistency | Similar or same values e.g<br>dinner, dinner/dinner, [dinner], daily dinner, (dinner)<br>?, <Blank> |
| date | Correctness | Few Invalid Dates e.g 2928-03-26<br>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 |

## 4.3 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.



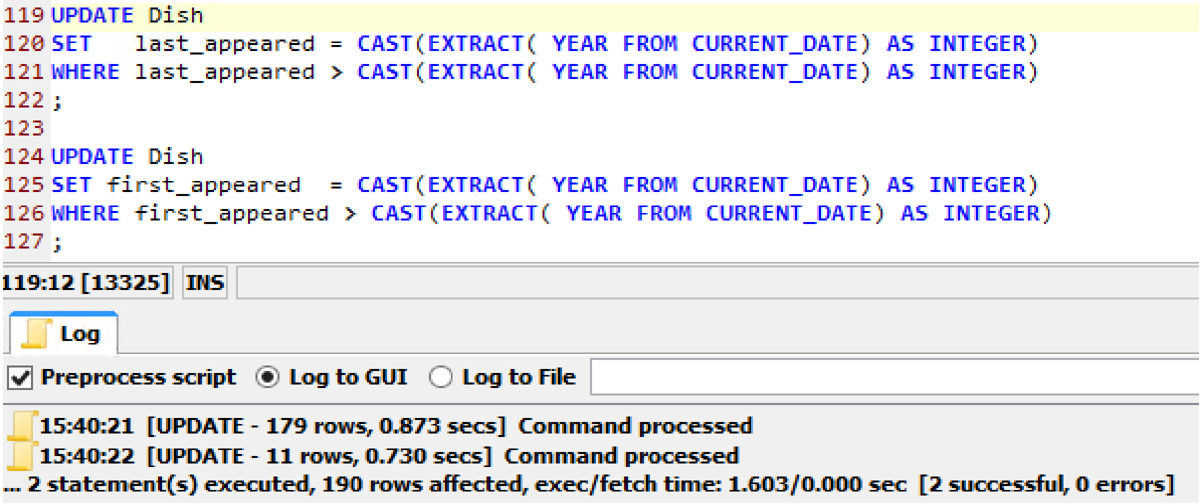
```
UPDATE Dish
SET last_appeared = first_appeared
WHERE last_appeared < first_appeared
;

```

110:2 [12983] INS

☑ Preprocess script ● Log to GUI ○ Log to File

```
15:33:58 [UPDATE - 0 rows, 0.679 secs] Command processed. No rows were affected
... 1 statement(s) executed, 0 rows affected, exec/fetch time: 0.679/0.000 sec [1 successful, 0 errors]
```

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

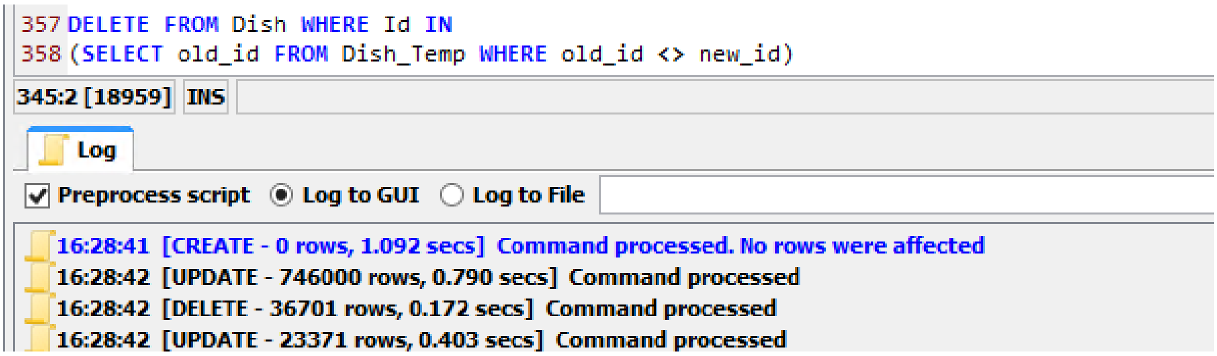


```
UPDATE Dish
SET    last_appeared = CAST(EXTRACT( YEAR FROM CURRENT_DATE) AS INTEGER)
WHERE last_appeared > CAST(EXTRACT( YEAR FROM CURRENT_DATE) AS INTEGER)
;

UPDATE Dish
SET first_appeared  = CAST(EXTRACT( YEAR FROM CURRENT_DATE) AS INTEGER)
WHERE first_appeared > CAST(EXTRACT( YEAR FROM CURRENT_DATE) AS INTEGER)
;
```

☑ Preprocess script ● Log to GUI ○ Log to File

```
15:40:21  [UPDATE - 179 rows, 0.873 secs]  Command processed
15:40:22  [UPDATE - 11 rows, 0.730 secs]  Command processed
... 2 statement(s) executed, 190 rows affected, exec/fetch time: 1.603/0.000 sec  [2 successful, 0 errors]
```

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



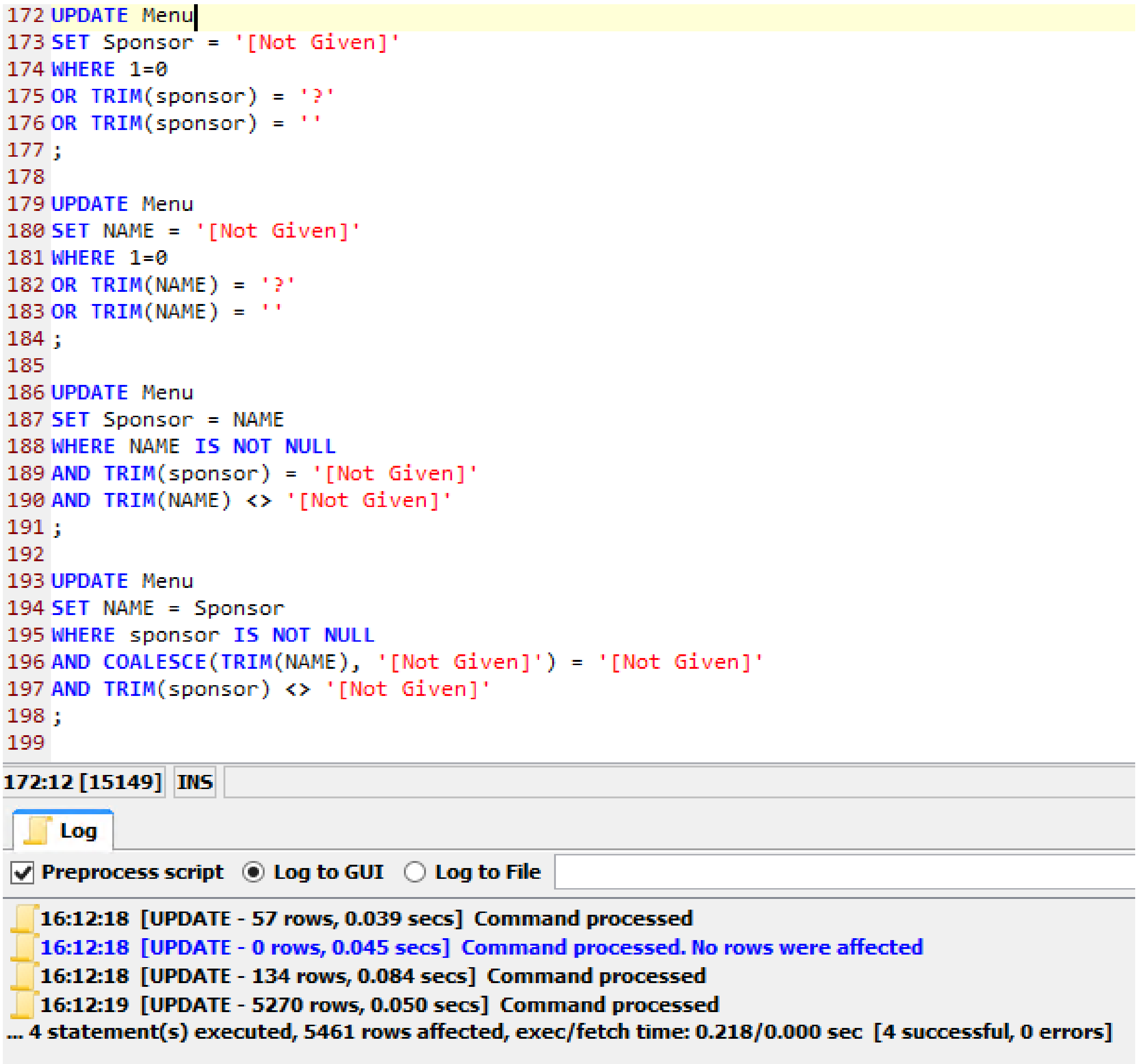
```
DELETE FROM Dish WHERE Id IN
(SELECT old_id FROM Dish_Temp WHERE old_id <> new_id)
```

☑ Preprocess script ● Log to GUI ○ Log to File

```
16:28:41  [CREATE - 0 rows, 1.092 secs]  Command processed. No rows were affected
16:28:42  [UPDATE - 746000 rows, 0.790 secs]  Command processed
16:28:42  [DELETE - 36701 rows, 0.172 secs]  Command processed
16:28:42  [UPDATE - 23371 rows, 0.403 secs]  Command processed
```

## Table Menu

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



```
UPDATE Menu
SET Sponsor = '[Not Given]'
WHERE 1=0
OR TRIM(sponsor) = '?'
OR TRIM(sponsor) = ''
;

UPDATE Menu
SET NAME = '[Not Given]'
WHERE 1=0
OR TRIM(NAME) = '?'
OR TRIM(NAME) = ''
;

UPDATE Menu
SET Sponsor = NAME
WHERE NAME IS NOT NULL
AND TRIM(sponsor) = '[Not Given]'
AND TRIM(NAME) <> '[Not Given]'
;

UPDATE Menu
SET NAME = Sponsor
WHERE sponsor IS NOT NULL
AND COALESCE(TRIM(NAME), '[Not Given]') = '[Not Given]'
AND TRIM(sponsor) <> '[Not Given]'
;

```

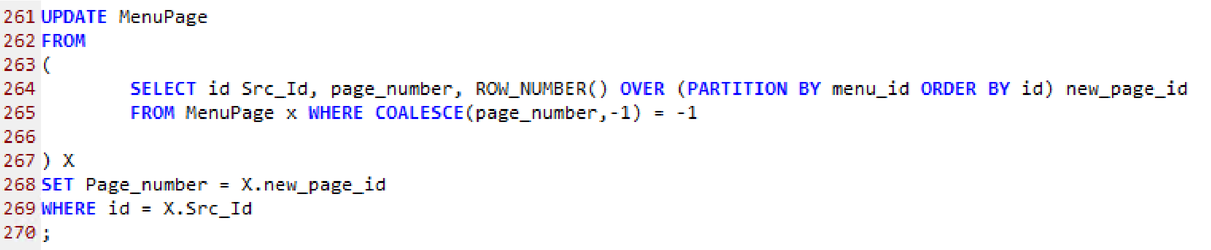
172:12 [15149] INS

☑ Preprocess script ● Log to GUI ○ Log to File

```
16:12:18 [UPDATE - 57 rows, 0.039 secs] Command processed
16:12:18 [UPDATE - 0 rows, 0.045 secs] Command processed. No rows were affected
16:12:18 [UPDATE - 134 rows, 0.084 secs] Command processed
16:12:19 [UPDATE - 5270 rows, 0.050 secs] Command processed
... 4 statement(s) executed, 5461 rows affected, exec/fetch time: 0.218/0.000 sec [4 successful, 0 errors]
```

## Table MenuPage

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

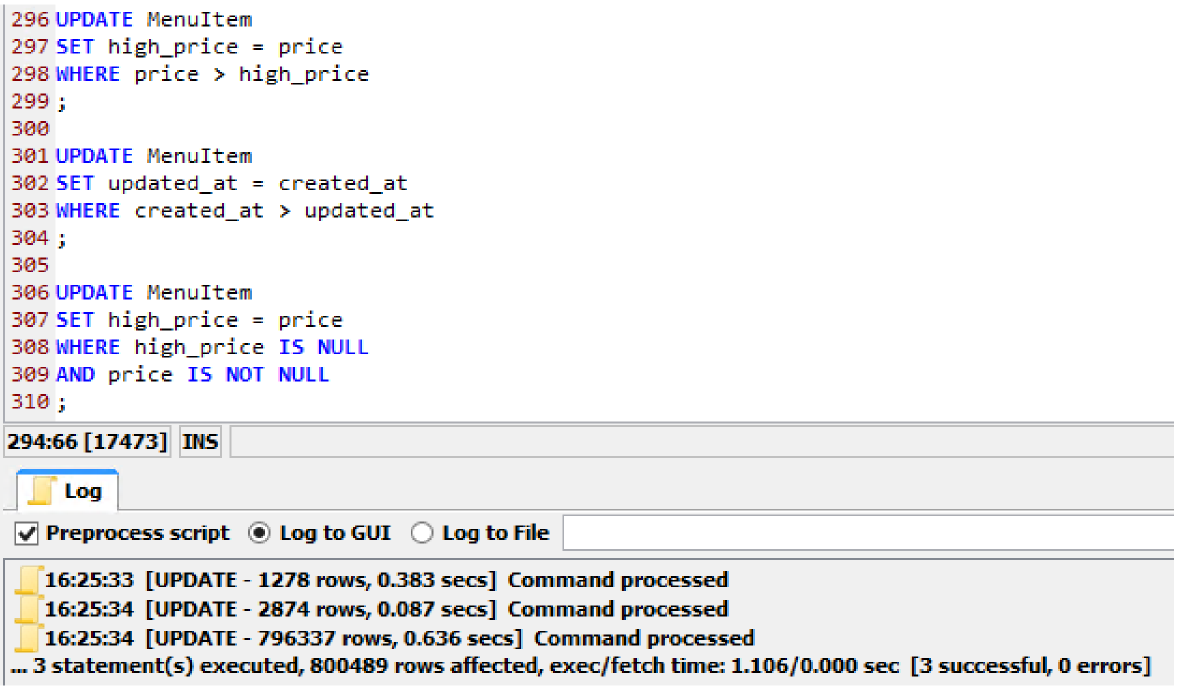


```
UPDATE MenuPage
FROM
(
	SELECT id Src_Id, page_number, ROW_NUMBER() OVER (PARTITION BY menu_id ORDER BY id) new_page_id
	FROM MenuPage x WHERE COALESCE(page_number,-1) = -1

) X
SET Page_number = X.new_page_id
WHERE id = X.Src_Id
;
```

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.



```
UPDATE MenuItem
SET high_price = price
WHERE price > high_price
;

UPDATE MenuItem
SET updated_at = created_at
WHERE created_at > updated_at
;

UPDATE MenuItem
SET high_price = price
WHERE high_price IS NULL
AND price IS NOT NULL
;
```

294:66 [17473]  INS

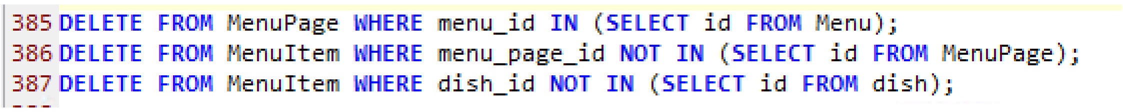
Log

☑ Preprocess script  ● Log to GUI  ○ Log to File

```
16:25:33  [UPDATE - 1278 rows, 0.383 secs]  Command processed
16:25:34  [UPDATE - 2874 rows, 0.087 secs]  Command processed
16:25:34  [UPDATE - 796337 rows, 0.636 secs]  Command processed
... 3 statement(s) executed, 800489 rows affected, exec/fetch time: 1.106/0.000 sec  [3 successful, 0 errors]
```

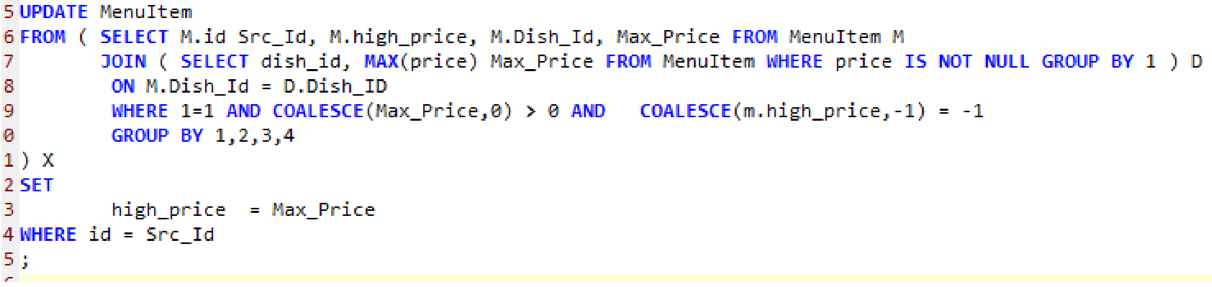
**<u>Fixing RI:</u>**

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

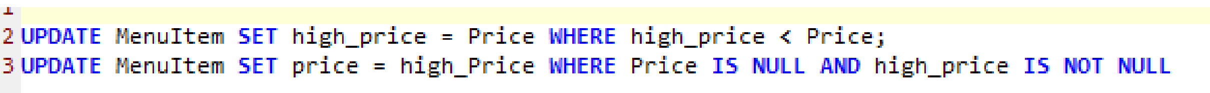


```
DELETE FROM MenuPage WHERE menu_id IN (SELECT id FROM Menu);
DELETE FROM MenuItem WHERE menu_page_id NOT IN (SELECT id FROM MenuPage);
DELETE FROM MenuItem WHERE dish_id NOT IN (SELECT id FROM dish);
```

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.



```
UPDATE MenuItem
FROM ( SELECT M.id Src_Id, M.high_price, M.Dish_Id, Max_Price FROM MenuItem M
       JOIN ( SELECT dish_id, MAX(price) Max_Price FROM MenuItem WHERE price IS NOT NULL GROUP BY 1 ) D
        ON M.Dish_Id = D.Dish_ID
        WHERE 1=1 AND COALESCE(Max_Price,0) > 0 AND   COALESCE(m.high_price,-1) = -1
        GROUP BY 1,2,3,4
) X
SET
        high_price  = Max_Price
WHERE id = Src_Id
;
```

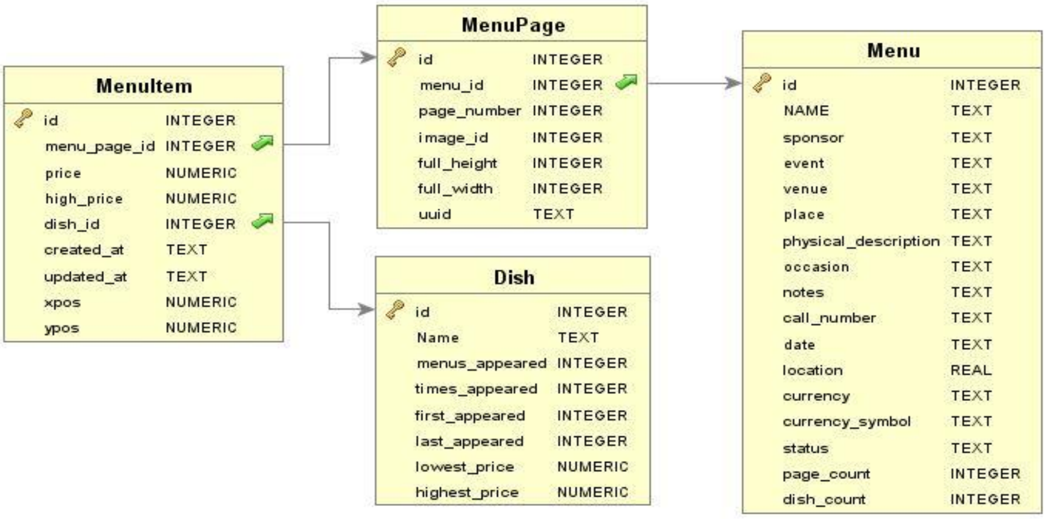


```
UPDATE MenuItem SET high_price = Price WHERE high_price < Price;
UPDATE MenuItem SET price = high_Price WHERE Price IS NULL AND high_price IS NOT NULL
```

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.

## 4.4 Creating a New SQL Schema for cleaned data

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



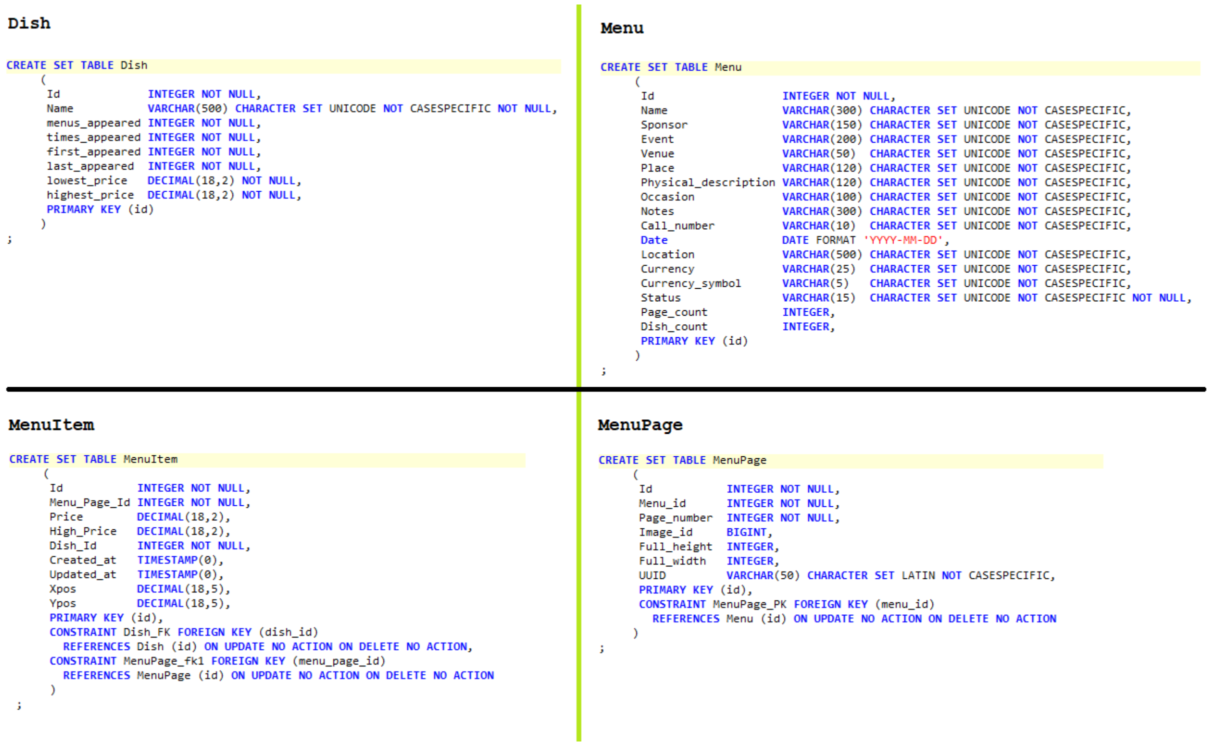
| MenuItem | |
|---|---|
| id | INTEGER |
| menu_page_id | INTEGER |
| price | NUMERIC |
| high_price | NUMERIC |
| dish_id | INTEGER |
| created_at | TEXT |
| updated_at | TEXT |
| xpos | NUMERIC |
| ypos | NUMERIC |

| MenuPage | |
|---|---|
| id | INTEGER |
| menu_id | INTEGER |
| page_number | INTEGER |
| image_id | INTEGER |
| full_height | INTEGER |
| full_width | INTEGER |
| uuid | TEXT |

| Dish | |
|---|---|
| id | INTEGER |
| Name | TEXT |
| menus_appeared | INTEGER |
| times_appeared | INTEGER |
| first_appeared | INTEGER |
| last_appeared | INTEGER |
| lowest_price | NUMERIC |
| highest_price | NUMERIC |

| Menu | |
|---|---|
| id | INTEGER |
| NAME | TEXT |
| sponsor | TEXT |
| event | TEXT |
| venue | TEXT |
| place | TEXT |
| physical_description | TEXT |
| occasion | TEXT |
| notes | TEXT |
| call_number | TEXT |
| date | TEXT |
| location | REAL |
| currency | TEXT |
| currency_symbol | TEXT |
| status | TEXT |
| page_count | INTEGER |
| dish_count | INTEGER |

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



**Dish**

```
CREATE SET TABLE Dish
     (
      Id             INTEGER NOT NULL,
      Name           VARCHAR(500) CHARACTER SET UNICODE NOT CASESPECIFIC NOT NULL,
      menus_appeared INTEGER NOT NULL,
      times_appeared INTEGER NOT NULL,
      first_appeared INTEGER NOT NULL,
      last_appeared  INTEGER NOT NULL,
      lowest_price   DECIMAL(18,2) NOT NULL,
      highest_price  DECIMAL(18,2) NOT NULL,
      PRIMARY KEY (id)
     )
;
```

**Menu**

```
CREATE SET TABLE Menu
     (
      Id                   INTEGER NOT NULL,
      Name                 VARCHAR(300) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Sponsor              VARCHAR(150) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Event                VARCHAR(200) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Venue                VARCHAR(50)  CHARACTER SET UNICODE NOT CASESPECIFIC,
      Place                VARCHAR(120) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Physical_description VARCHAR(120) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Occasion             VARCHAR(100) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Notes                VARCHAR(300) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Call_number          VARCHAR(10)  CHARACTER SET UNICODE NOT CASESPECIFIC,
      Date                 DATE FORMAT 'YYYY-MM-DD',
      Location             VARCHAR(500) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Currency             VARCHAR(25)  CHARACTER SET UNICODE NOT CASESPECIFIC,
      Currency_symbol      VARCHAR(5)   CHARACTER SET UNICODE NOT CASESPECIFIC,
      Status               VARCHAR(15)  CHARACTER SET UNICODE NOT CASESPECIFIC NOT NULL,
      Page_count           INTEGER,
      Dish_count           INTEGER,
      PRIMARY KEY (id)
     )
;
```

**MenuItem**

```
CREATE SET TABLE MenuItem
     (
      Id           INTEGER NOT NULL,
      Menu_Page_Id INTEGER NOT NULL,
      Price        DECIMAL(18,2),
      High_Price   DECIMAL(18,2),
      Dish_Id      INTEGER NOT NULL,
      Created_at   TIMESTAMP(0),
      Updated_at   TIMESTAMP(0),
      Xpos         DECIMAL(18,5),
      Ypos         DECIMAL(18,5),
      PRIMARY KEY (id),
      CONSTRAINT Dish_FK FOREIGN KEY (dish_id)
        REFERENCES Dish (id) ON UPDATE NO ACTION ON DELETE NO ACTION,
      CONSTRAINT MenuPage_fk1 FOREIGN KEY (menu_page_id)
        REFERENCES MenuPage (id) ON UPDATE NO ACTION ON DELETE NO ACTION
      )
 ;
```

**MenuPage**

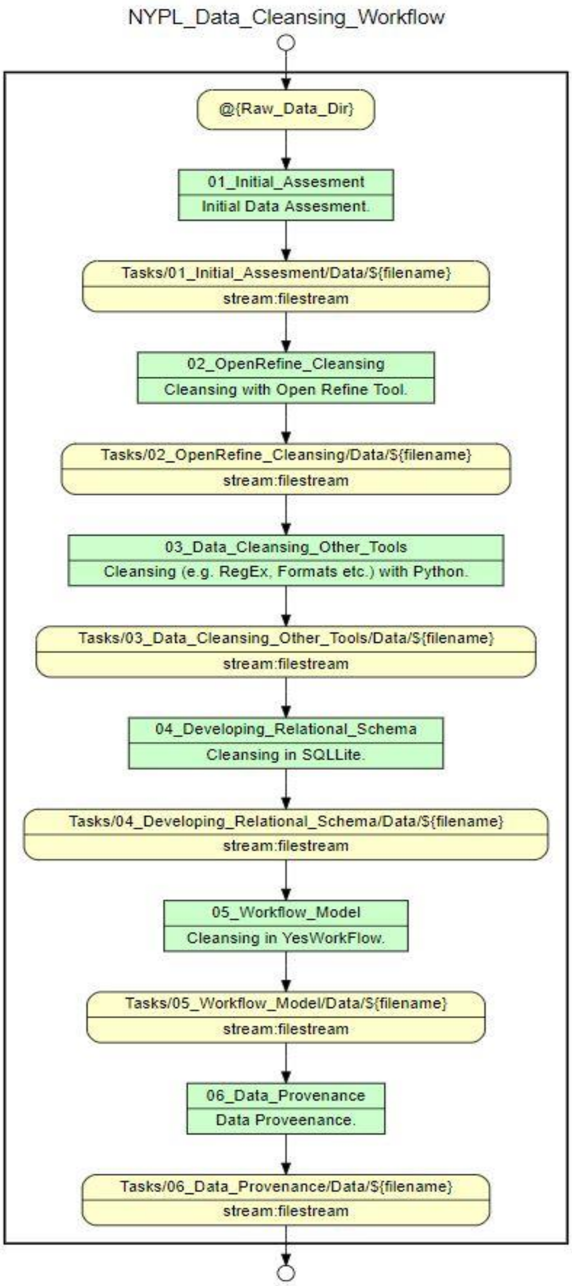
```
CREATE SET TABLE MenuPage
     (
      Id           INTEGER NOT NULL,
      Menu_id      INTEGER NOT NULL,
      Page_number  INTEGER NOT NULL,
      Image_id     BIGINT,
      Full_height  INTEGER,
      Full_width   INTEGER,
      UUID         VARCHAR(50) CHARACTER SET LATIN NOT CASESPECIFIC,
      PRIMARY KEY (id),
      CONSTRAINT MenuPage_PK FOREIGN KEY (menu_id)
        REFERENCES Menu (id) ON UPDATE NO ACTION ON DELETE NO ACTION
     )
;
```

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

## 5 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.

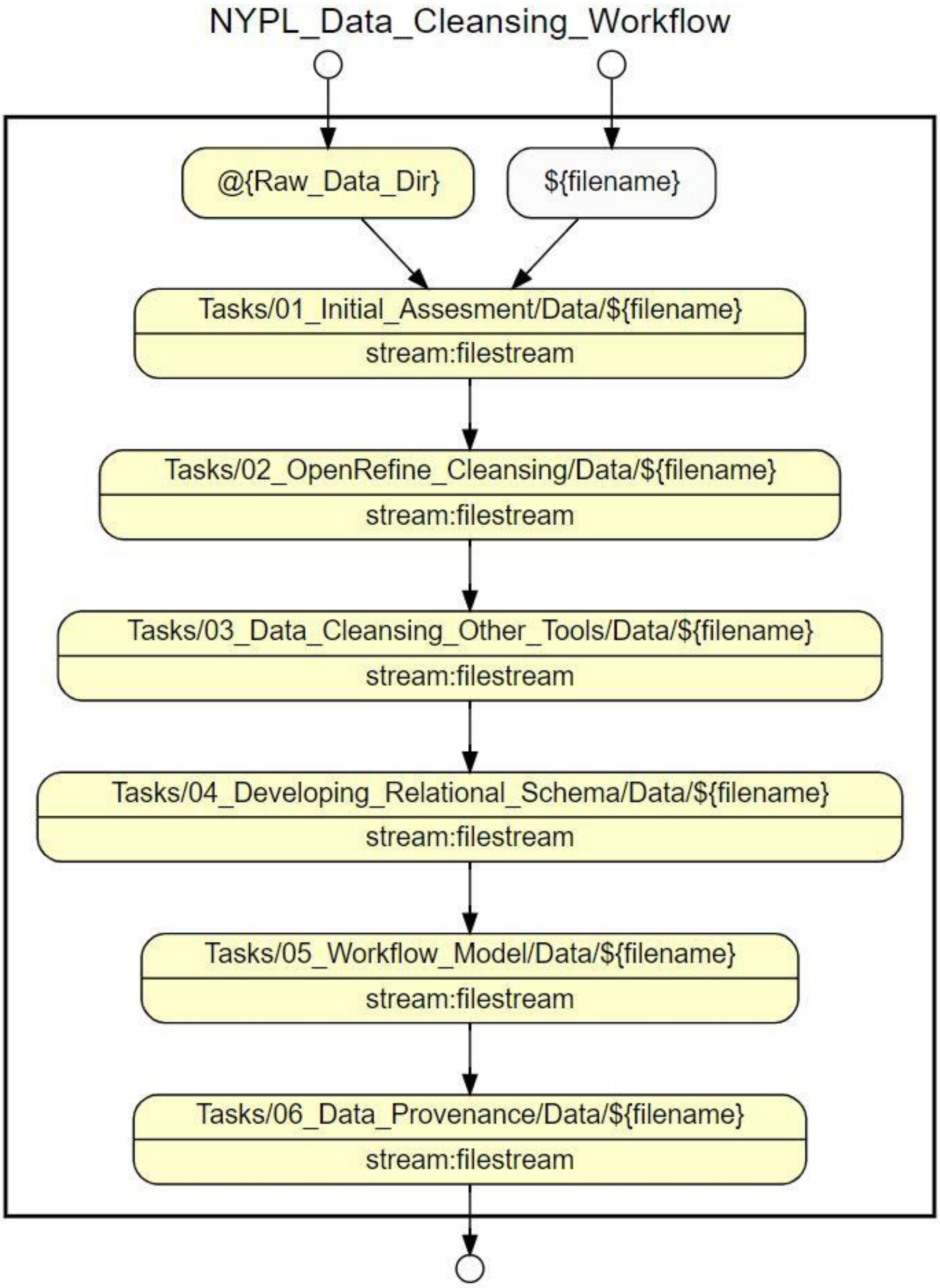
The main tool used for generating workflow provenance diagrams was YesWorkflow Online Editor. Respective workflow jobs were annotated and used to generate the respective workflow diagrams.


NYPL_Data_Cleansing_Workflow
@{Raw_Data_Dir}
01_Initial_Assesment
Initial Data Assesment.
Tasks/01_Initial_Assesment/Data/${filename}
stream:filestream
02_OpenRefine_Cleansing
Cleansing with Open Refine Tool.
Tasks/02_OpenRefine_Cleansing/Data/${filename}
stream:filestream
03_Data_Cleansing_Other_Tools
Cleansing (e.g. RegEx, Formats etc.) with Python.
Tasks/03_Data_Cleansing_Other_Tools/Data/${filename}
stream:filestream
04_Developing_Relational_Schema
Cleansing in SQLLite.
Tasks/04_Developing_Relational_Schema/Data/${filename}
stream:filestream
05_Workflow_Model
Cleansing in YesWorkFlow.
Tasks/05_Workflow_Model/Data/${filename}
stream:filestream
06_Data_Provenance
Data Proveenance.
Tasks/06_Data_Provenance/Data/${filename}
stream:filestream

## 5.1 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.


NYPL_Data_Cleansing_Workflow
@{Raw_Data_Dir}
${filename}
Tasks/01_Initial_Assesment/Data/${filename}
stream:filestream
Tasks/02_OpenRefine_Cleansing/Data/${filename}
stream:filestream
Tasks/03_Data_Cleansing_Other_Tools/Data/${filename}
stream:filestream
Tasks/04_Developing_Relational_Schema/Data/${filename}
stream:filestream
Tasks/05_Workflow_Model/Data/${filename}
stream:filestream
Tasks/06_Data_Provenance/Data/${filename}
stream:filestream

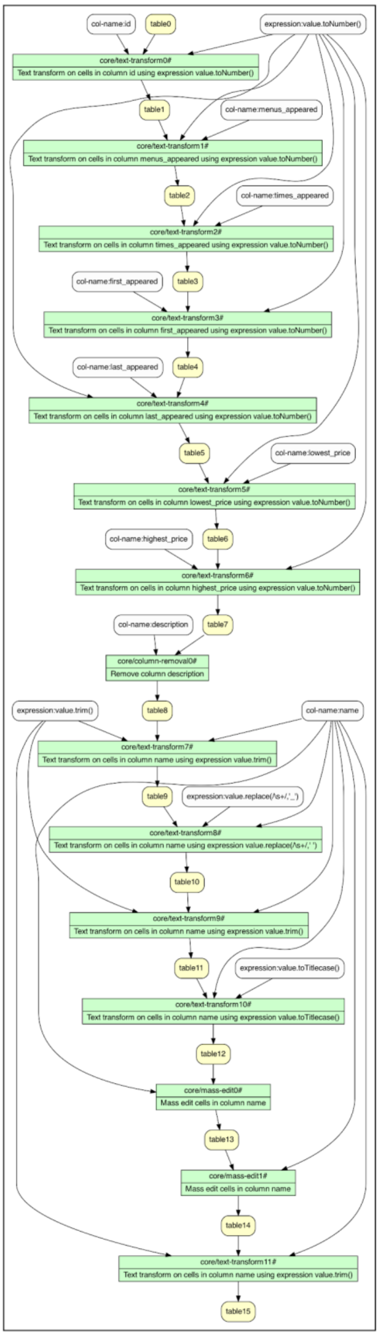
Note: To clean up the project and start fresh (delete all processed data), run **cleanup.sh**

## 5.2 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:

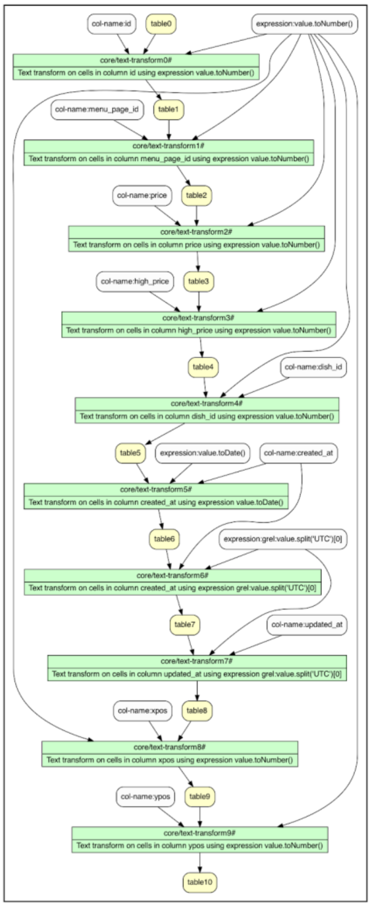
**Menu.csv**

Linear_OR


col-name:id
table0
expression:value.toNumber()
core/text-transform0#
Text transform on cells in column id using expression value.toNumber()
table1
col-name:menus_appeared
core/text-transform1#
Text transform on cells in column menus_appeared using expression value.toNumber()
table2
col-name:times_appeared
core/text-transform2#
Text transform on cells in column times_appeared using expression value.toNumber()
col-name:first_appeared
table3
core/text-transform3#
Text transform on cells in column first_appeared using expression value.toNumber()
col-name:last_appeared
table4
core/text-transform4#
Text transform on cells in column last_appeared using expression value.toNumber()
table5
col-name:lowest_price
core/text-transform5#
Text transform on cells in column lowest_price using expression value.toNumber()
col-name:highest_price
table6
core/text-transform6#
Text transform on cells in column highest_price using expression value.toNumber()
col-name:description
table7
core/column-removal0#
Remove column description
expression:value.trim()
table8
col-name:name
core/text-transform7#
Text transform on cells in column name using expression value.trim()
table9
expression:value.replace(/\s+/,'_')
core/text-transform8#
Text transform on cells in column name using expression value.replace(/\s+/,' ')
table10
core/text-transform9#
Text transform on cells in column name using expression value.trim()
table11
expression:value.toTitlecase()
core/text-transform10#
Text transform on cells in column name using expression value.toTitlecase()
table12
core/mass-edit0#
Mass edit cells in column name
table13
core/mass-edit1#
Mass edit cells in column name
table14
core/text-transform11#
Text transform on cells in column name using expression value.trim()
table15

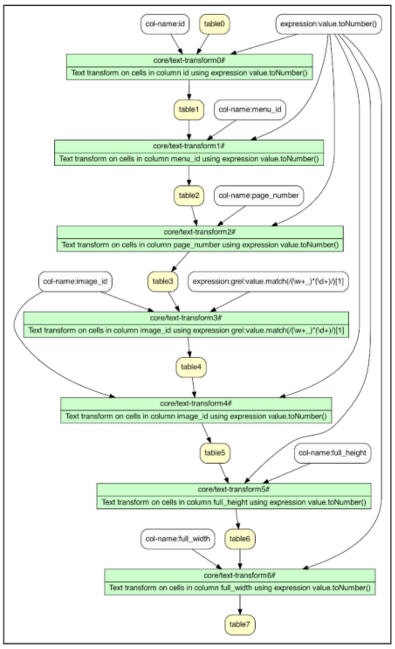
**MenuItem.csv**

Linear_OR

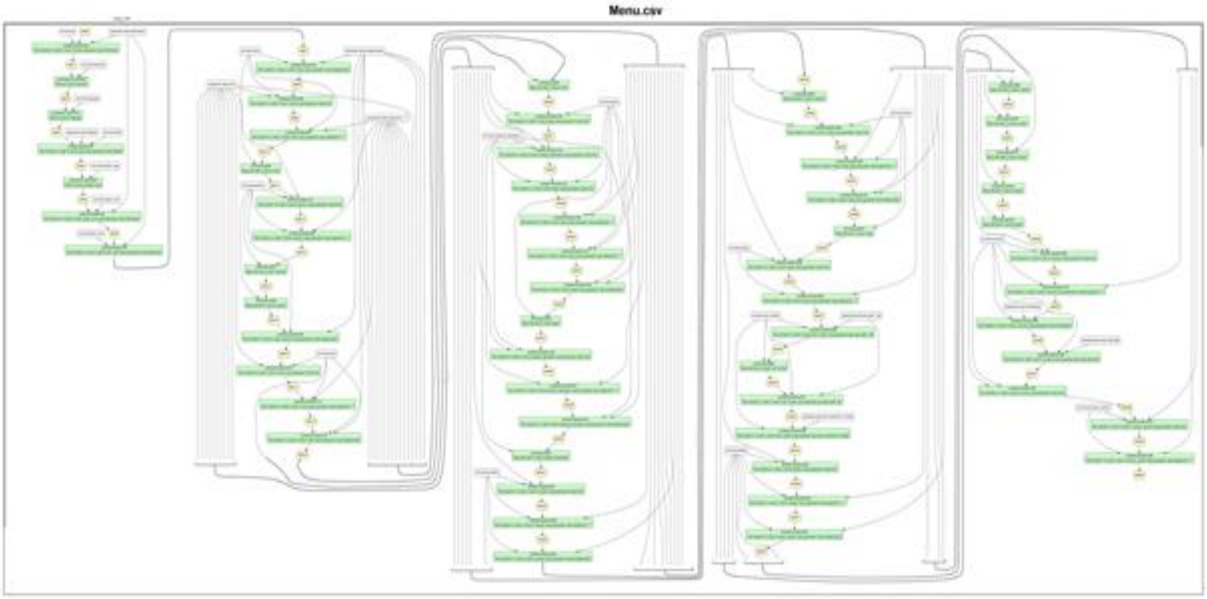

col-name:id
table0
expression:value.toNumber()
core/text-transform0#
Text transform on cells in column id using expression value.toNumber()
col-name:menu_page_id
table1
core/text-transform1#
Text transform on cells in column menu_page_id using expression value.toNumber()
col-name:price
table2
core/text-transform2#
Text transform on cells in column price using expression value.toNumber()
col-name:high_price
table3
core/text-transform3#
Text transform on cells in column high_price using expression value.toNumber()
table4
col-name:dish_id
core/text-transform4#
Text transform on cells in column dish_id using expression value.toNumber()
table5
expression:value.toDate()
col-name:created_at
core/text-transform5#
Text transform on cells in column created_at using expression value.toDate()
table6
expression:grel:value.split('UTC')[0]
core/text-transform6#
Text transform on cells in column created_at using expression grel:value.split('UTC')[0]
table7
col-name:updated_at
core/text-transform7#
Text transform on cells in column updated_at using expression grel:value.split('UTC')[0]
col-name:xpos
table8
core/text-transform8#
Text transform on cells in column xpos using expression value.toNumber()
col-name:ypos
table9
core/text-transform9#
Text transform on cells in column ypos using expression value.toNumber()
table10

**MenuPage.csv**

Linear_OR

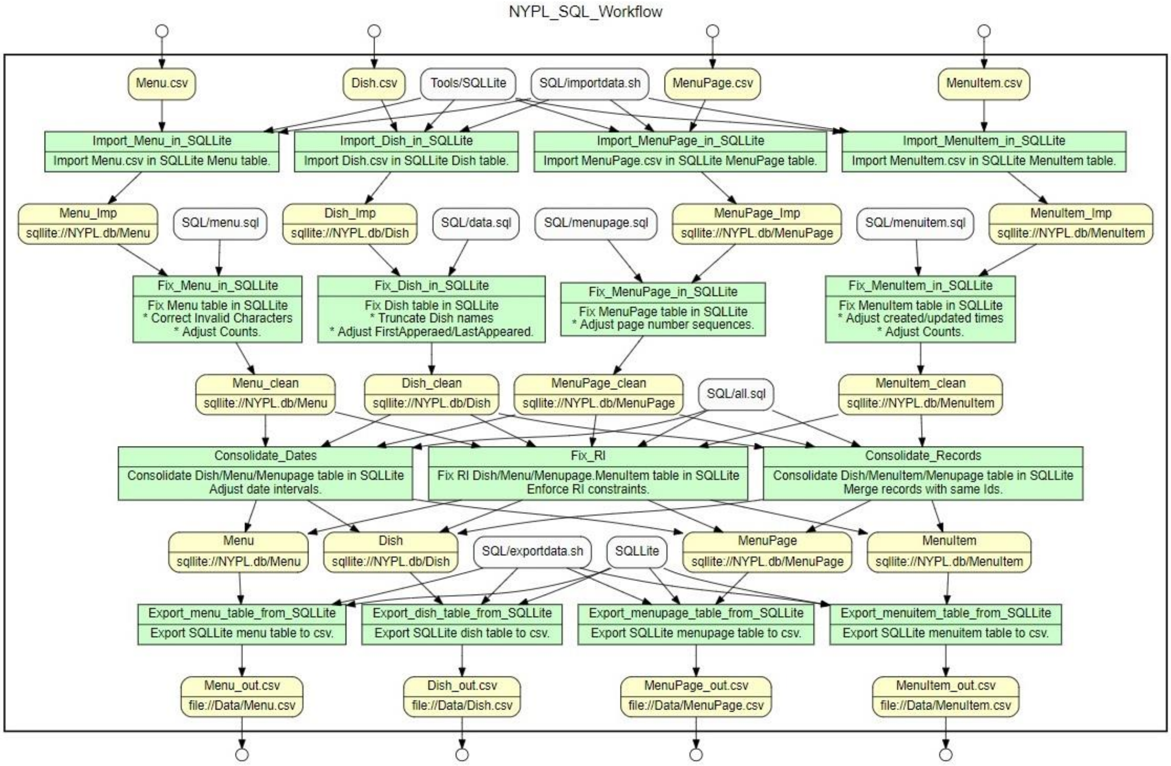

col-name:id
table0
expression:value.toNumber()
core/text-transform0#
Text transform on cells in column id using expression value.toNumber()
table1
col-name:menu_id
core/text-transform1#
Text transform on cells in column menu_id using expression value.toNumber()
table2
col-name:page_number
core/text-transform2#
Text transform on cells in column page_number using expression value.toNumber()
col-name:image_id
table3
expression:grel:value.match(/(\w+_)*(\d+)/)[1]
core/text-transform3#
Text transform on cells in column image_id using expression grel:value.match(/(\w+_)*(\d+)/)[1]
table4
core/text-transform4#
Text transform on cells in column image_id using expression value.toNumber()
table5
col-name:full_height
core/text-transform5#
Text transform on cells in column full_height using expression value.toNumber()
col-name:full_width
table6
core/text-transform6#
Text transform on cells in column full_width using expression value.toNumber()
table7

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



## 5.3 SQL-Lite workflow

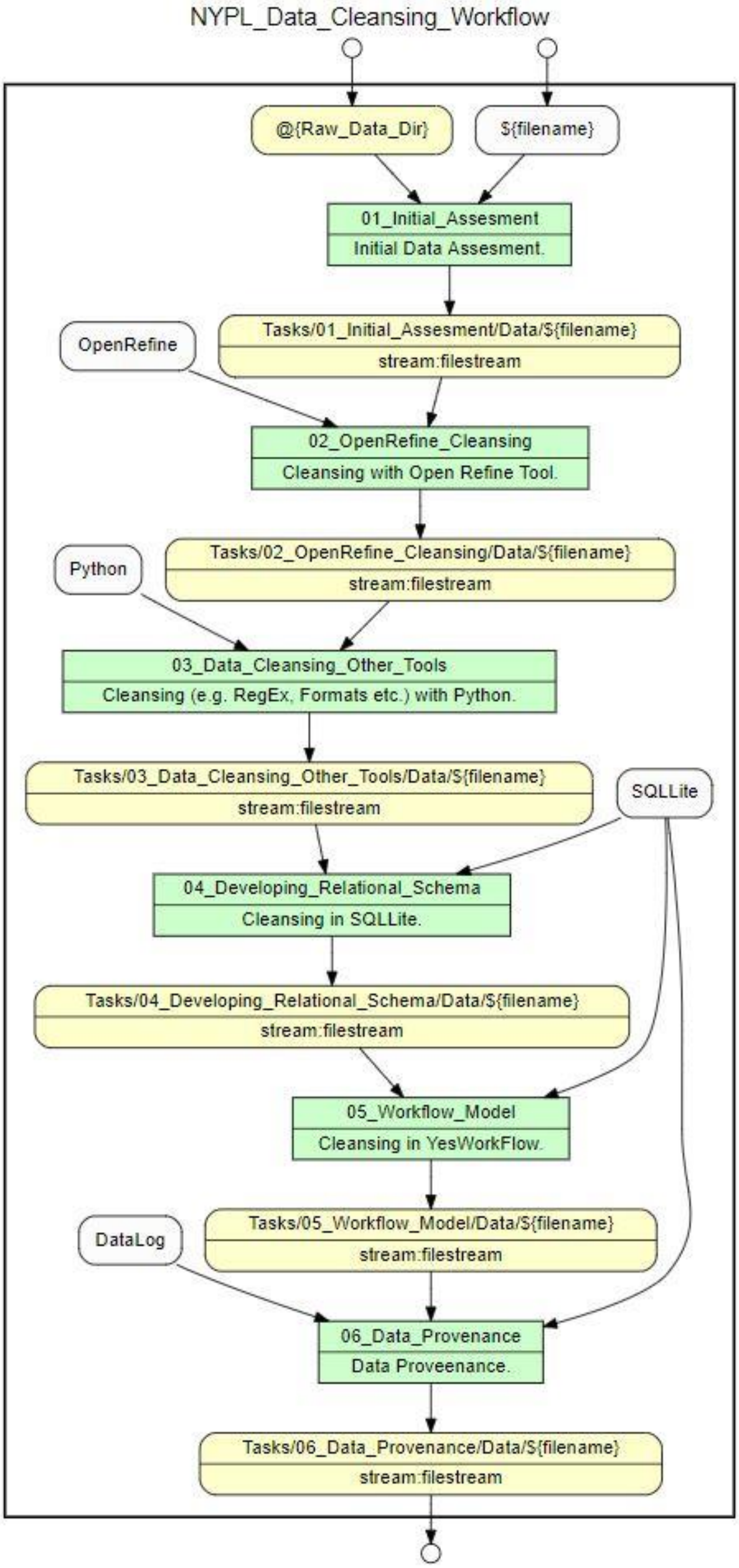
Below is the work flow graph of cleansing with SQL-Lite. The input files are the pre-processed CSV file. The output files are exported CSV files after cleansing in SQL-Lite.


NYPL_SQL_Workflow
Menu.csv
Dish.csv
Tools/SQLLite
SQL/importdata.sh
MenuPage.csv
MenuItem.csv
Import_Menu_in_SQLLite
Import Menu.csv in SQLLite Menu table.
Import_Dish_in_SQLLite
Import Dish.csv in SQLLite Dish table.
Import_MenuPage_in_SQLLite
Import MenuPage.csv in SQLLite MenuPage table.
Import_MenuItem_in_SQLLite
Import MenuItem.csv in SQLLite MenuItem table.
Menu_Imp
sqllite://NYPL.db/Menu
SQL/menu.sql
Dish_Imp
sqllite://NYPL.db/Dish
SQL/data.sql
SQL/menupage.sql
MenuPage_Imp
sqllite://NYPL.db/MenuPage
SQL/menuitem.sql
MenuItem_Imp
sqllite://NYPL.db/MenuItem
Fix_Menu_in_SQLLite
Fix Menu table in SQLLite
* Correct Invalid Characters
* Adjust Counts.
Fix_Dish_in_SQLLite
Fix Dish table in SQLLite
* Truncate Dish names
* Adjust FirstApperaed/LastAppeared.
Fix_MenuPage_in_SQLLite
Fix MenuPage table in SQLLite
* Adjust page number sequences.
Fix_MenuItem_in_SQLLite
Fix MenuItem table in SQLLite
* Adjust created/updated times
* Adjust Counts.
Menu_clean
sqllite://NYPL.db/Menu
Dish_clean
sqllite://NYPL.db/Dish
MenuPage_clean
sqllite://NYPL.db/MenuPage
SQL/all.sql
MenuItem_clean
sqllite://NYPL.db/MenuItem
Consolidate_Dates
Consolidate Dish/Menu/Menupage table in SQLLite
Adjust date intervals.
Fix_RI
Fix RI Dish/Menu/Menupage.MenuItem table in SQLLite
Enforce RI constraints.
Consolidate_Records
Consolidate Dish/MenuItem/Menupage table in SQLLite
Merge records with same Ids.
Menu
sqllite://NYPL.db/Menu
Dish
sqllite://NYPL.db/Dish
SQL/exportdata.sh
SQLLite
MenuPage
sqllite://NYPL.db/MenuPage
MenuItem
sqllite://NYPL.db/MenuItem
Export_menu_table_from_SQLLite
Export SQLLite menu table to csv.
Export_dish_table_from_SQLLite
Export SQLLite dish table to csv.
Export_menupage_table_from_SQLLite
Export SQLLite menupage table to csv.
Export_menuitem_table_from_SQLLite
Export SQLLite menuitem table to csv.
Menu_out.csv
file://Data/Menu.csv
Dish_out.csv
file://Data/Dish.csv
MenuPage_out.csv
file://Data/MenuPage.csv
MenuItem_out.csv
file://Data/MenuItem.csv

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

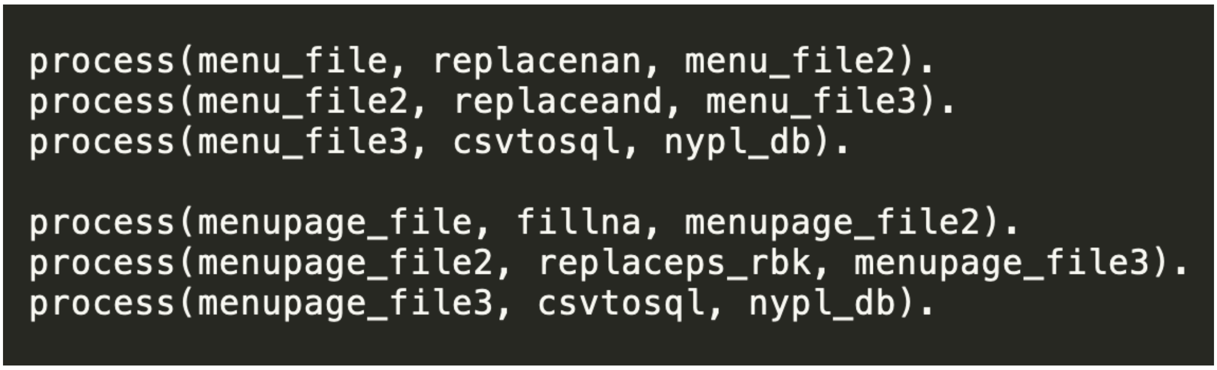
## 5.4 Workflow inputs, Outputs and dependencies

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


NYPL_Data_Cleansing_Workflow
@{Raw_Data_Dir}
${filename}
01_Initial_Assesment
Initial Data Assesment.
OpenRefine
Tasks/01_Initial_Assesment/Data/${filename}
stream:filestream
02_OpenRefine_Cleansing
Cleansing with Open Refine Tool.
Python
Tasks/02_OpenRefine_Cleansing/Data/${filename}
stream:filestream
03_Data_Cleansing_Other_Tools
Cleansing (e.g. RegEx, Formats etc.) with Python.
Tasks/03_Data_Cleansing_Other_Tools/Data/${filename}
stream:filestream
SQLLite
04_Developing_Relational_Schema
Cleansing in SQLLite.
Tasks/04_Developing_Relational_Schema/Data/${filename}
stream:filestream
05_Workflow_Model
Cleansing in YesWorkFlow.
DataLog
Tasks/05_Workflow_Model/Data/${filename}
stream:filestream
06_Data_Provenance
Data Proveenance.
Tasks/06_Data_Provenance/Data/${filename}
stream:filestream

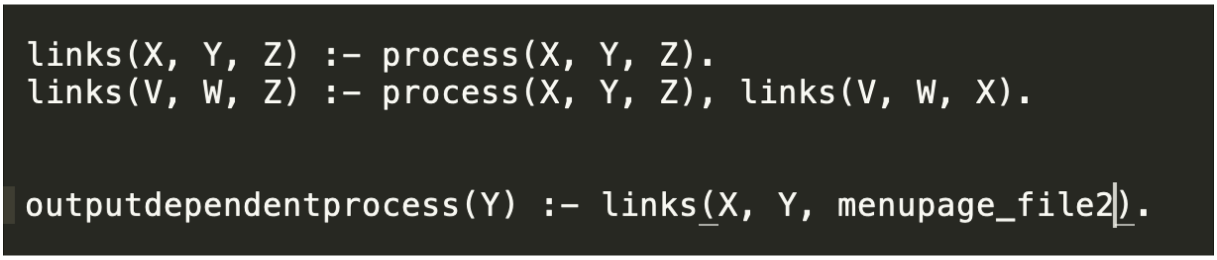
# 6 Developing Provenance

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:



```
process(menu_file, replacenan, menu_file2).
process(menu_file2, replaceand, menu_file3).
process(menu_file3, csvtosql, nypl_db).

process(menupage_file, fillna, menupage_file2).
process(menupage_file2, replaceps_rbk, menupage_file3).
process(menupage_file3, csvtosql, nypl_db).
```

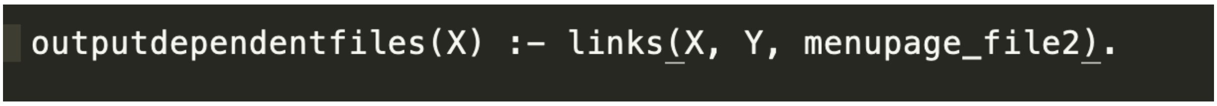
Then a datalog rule was made to identify all the processes required to obtain a certain output file within the workflow, e.g:



```
links(X, Y, Z) :- process(X, Y, Z).
links(V, W, Z) :- process(X, Y, Z), links(V, W, X).


outputdependentprocess(Y) :- links(X, Y, menupage_file2).
```

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:



```
outputdependentfiles(X) :- links(X, Y, menupage_file2).
```

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

# 7 Conclusions and Future Work

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 the 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.

Since several cleaning tasks were identifies at the time of initial assessment which were interdependent on one other, especially in the case of integrity violations, the division if project work and time was in itself a challenge, which was managed by taking advantage of time zone difference and daily standup calls, to efficiently and effectively distribute tasks.

All workflows were created keeping the provenance of data in mind from the inception. The architecture of processing pipeline was designed such that any deliverable or data is easily traceable, reproducible and audited. A provenance model was generated in cling so that it easier to query and identify dependencies in the workflow.

Since duplicated dishes with respect to their names were consolidated and redundancies were removed, the final dataset is much more concise and accurate and can serve multiple new use cases which was difficult if not possible in its raw form. The counts of records, respective entity prices and year of first and last appeared have been consolidated and corrected. The processed data would now be fit for analyzing how eating preferences evolved and evaluating the popularity of dishes for a given interval based on their consolidated counts or appearance. Removal of redundancies and standardization of entity names enabled easy and consistent integration of this data with another review dataset (such as Yelp reviews dataset) for rating analysis of dishes or its use in recommender systems.

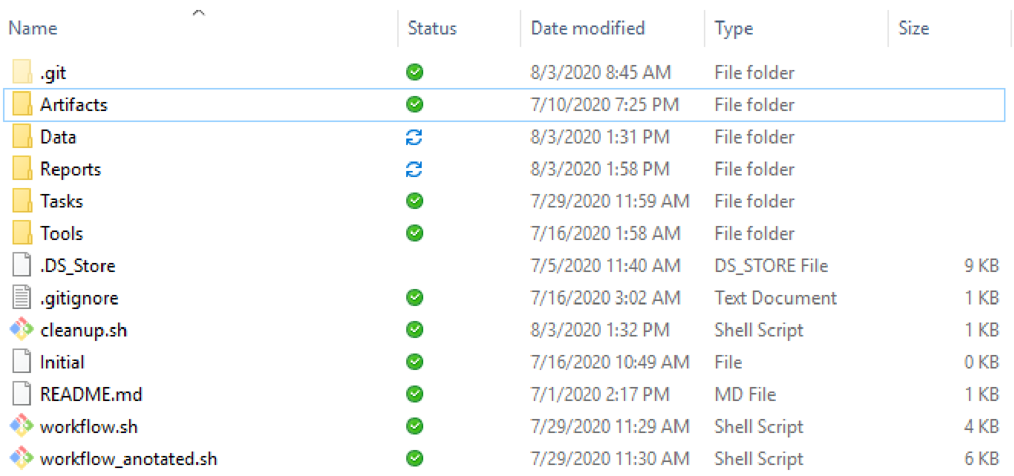
In the future, the processed data could be beneficial to anyone willing to consume it for analytical work or for further wrangling tasks.

# 8 Appendix

## 1 SSH-1 GitHub Hey:

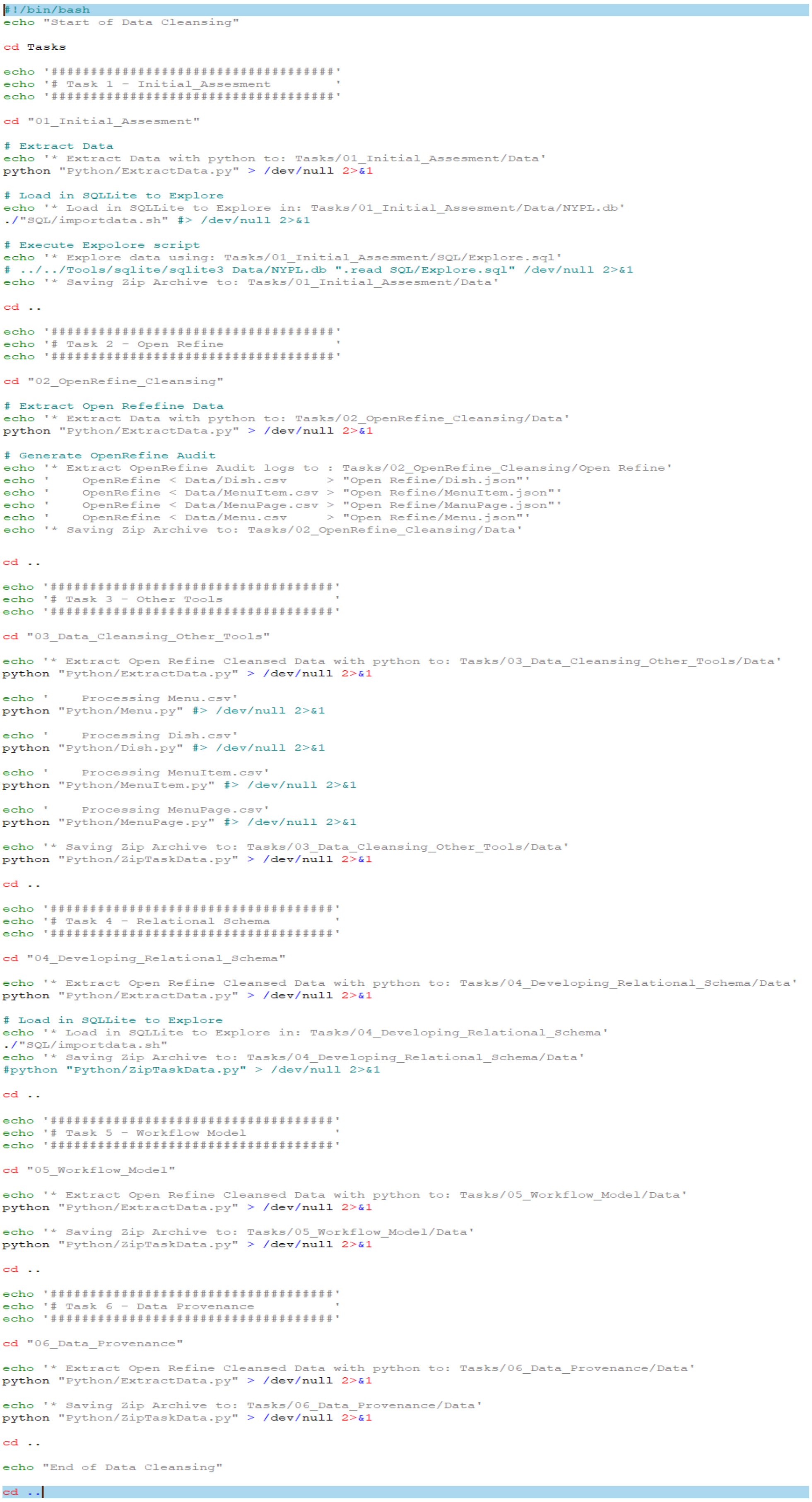
```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAviThY9kFTDex5+W4bZm1sQfDphFNeozLIFH3Q43sym94AXsnnHsu
m+2iLkClnYkKbCzR8bCTP0dQtkdbD9oJIx7wkdcdyM2ieoUenrSMNcVcqG/12sqrPm6SSr
jb28pBFQXj5gIUgkmddxtdbYoj8fKBx+6oMjJg0+6fepIgkArYgPFFh64S8jFy6oqm6rUV
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-----
```

## 2 Project Structure:



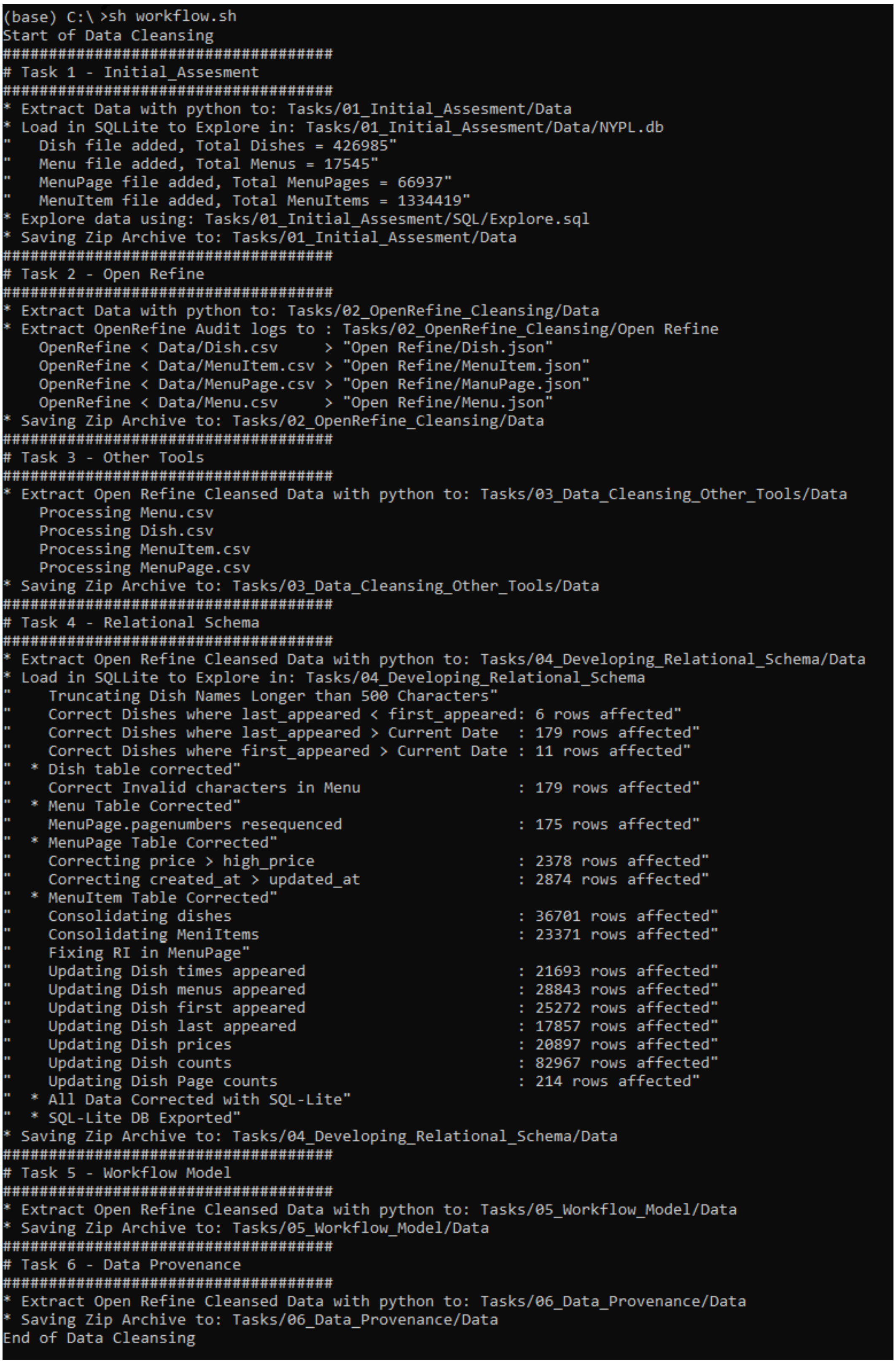
| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| .git | ✓ | 8/3/2020 8:45 AM | File folder | |
| Artifacts | ✓ | 7/10/2020 7:25 PM | File folder | |
| Data | ⟳ | 8/3/2020 1:31 PM | File folder | |
| Reports | ⟳ | 8/3/2020 1:58 PM | File folder | |
| Tasks | ✓ | 7/29/2020 11:59 AM | File folder | |
| Tools | ✓ | 7/16/2020 1:58 AM | File folder | |
| .DS_Store | | 7/5/2020 11:40 AM | DS_STORE File | 9 KB |
| .gitignore | ✓ | 7/16/2020 3:02 AM | Text Document | 1 KB |
| cleanup.sh | ✓ | 8/3/2020 1:32 PM | Shell Script | 1 KB |
| Initial | ✓ | 7/16/2020 10:49 AM | File | 0 KB |
| README.md | ✓ | 7/1/2020 2:17 PM | MD File | 1 KB |
| workflow.sh | ✓ | 7/29/2020 11:29 AM | Shell Script | 4 KB |
| workflow_anotated.sh | ✓ | 7/29/2020 11:30 AM | Shell Script | 6 KB |

## 3 Main workflow script (not annotated):



```
#!/bin/bash
echo "Start of Data Cleansing"

cd Tasks

echo '####################################'
echo '# Task 1 - Initial_Assesment        '
echo '####################################'

cd "01_Initial_Assesment"

# Extract Data
echo '* Extract Data with python to: Tasks/01_Initial_Assesment/Data'
python "Python/ExtractData.py" > /dev/null 2>&1

# Load in SQLLite to Explore
echo '* Load in SQLLite to Explore in: Tasks/01_Initial_Assesment/Data/NYPL.db'
./"SQL/importdata.sh" #> /dev/null 2>&1

# Execute Expolore script
echo '* Explore data using: Tasks/01_Initial_Assesment/SQL/Explore.sql'
# ../../Tools/sqlite/sqlite3 Data/NYPL.db ".read SQL/Explore.sql" /dev/null 2>&1
echo '* Saving Zip Archive to: Tasks/01_Initial_Assesment/Data'

cd ..

echo '####################################'
echo '# Task 2 - Open Refine              '
echo '####################################'

cd "02_OpenRefine_Cleansing"

# Extract Open Refefine Data
echo '* Extract Data with python to: Tasks/02_OpenRefine_Cleansing/Data'
python "Python/ExtractData.py" > /dev/null 2>&1

# Generate OpenRefine Audit
echo '* Extract OpenRefine Audit logs to : Tasks/02_OpenRefine_Cleansing/Open Refine'
echo '    OpenRefine < Data/Dish.csv     > "Open Refine/Dish.json"'
echo '    OpenRefine < Data/MenuItem.csv > "Open Refine/MenuItem.json"'
echo '    OpenRefine < Data/MenuPage.csv > "Open Refine/ManuPage.json"'
echo '    OpenRefine < Data/Menu.csv     > "Open Refine/Menu.json"'
echo '* Saving Zip Archive to: Tasks/02_OpenRefine_Cleansing/Data'


cd ..

echo '####################################'
echo '# Task 3 - Other Tools              '
echo '####################################'

cd "03_Data_Cleansing_Other_Tools"

echo '* Extract Open Refine Cleansed Data with python to: Tasks/03_Data_Cleansing_Other_Tools/Data'
python "Python/ExtractData.py" > /dev/null 2>&1

echo '    Processing Menu.csv'
python "Python/Menu.py" #> /dev/null 2>&1

echo '    Processing Dish.csv'
python "Python/Dish.py" #> /dev/null 2>&1

echo '    Processing MenuItem.csv'
python "Python/MenuItem.py" #> /dev/null 2>&1

echo '    Processing MenuPage.csv'
python "Python/MenuPage.py" #> /dev/null 2>&1

echo '* Saving Zip Archive to: Tasks/03_Data_Cleansing_Other_Tools/Data'
python "Python/ZipTaskData.py" > /dev/null 2>&1

cd ..

echo '####################################'
echo '# Task 4 - Relational Schema        '
echo '####################################'

cd "04_Developing_Relational_Schema"

echo '* Extract Open Refine Cleansed Data with python to: Tasks/04_Developing_Relational_Schema/Data'
python "Python/ExtractData.py" > /dev/null 2>&1

# Load in SQLLite to Explore
echo '* Load in SQLLite to Explore in: Tasks/04_Developing_Relational_Schema'
./"SQL/importdata.sh"
echo '* Saving Zip Archive to: Tasks/04_Developing_Relational_Schema/Data'
#python "Python/ZipTaskData.py" > /dev/null 2>&1

cd ..

echo '####################################'
echo '# Task 5 - Workflow Model           '
echo '####################################'

cd "05_Workflow_Model"

echo '* Extract Open Refine Cleansed Data with python to: Tasks/05_Workflow_Model/Data'
python "Python/ExtractData.py" > /dev/null 2>&1

echo '* Saving Zip Archive to: Tasks/05_Workflow_Model/Data'
python "Python/ZipTaskData.py" > /dev/null 2>&1

cd ..

echo '####################################'
echo '# Task 6 - Data Provenance          '
echo '####################################'

cd "06_Data_Provenance"

echo '* Extract Open Refine Cleansed Data with python to: Tasks/06_Data_Provenance/Data'
python "Python/ExtractData.py" > /dev/null 2>&1

echo '* Saving Zip Archive to: Tasks/06_Data_Provenance/Data'
python "Python/ZipTaskData.py" > /dev/null 2>&1

cd ..

echo "End of Data Cleansing"

cd ..
```

# 4 Workflow log:



```
(base) C:\ >sh workflow.sh
Start of Data Cleansing
####################################
# Task 1 - Initial_Assesment
####################################
* Extract Data with python to: Tasks/01_Initial_Assesment/Data
* Load in SQLLite to Explore in: Tasks/01_Initial_Assesment/Data/NYPL.db
"   Dish file added, Total Dishes = 426985"
"   Menu file added, Total Menus = 17545"
"   MenuPage file added, Total MenuPages = 66937"
"   MenuItem file added, Total MenuItems = 1334419"
* Explore data using: Tasks/01_Initial_Assesment/SQL/Explore.sql
* Saving Zip Archive to: Tasks/01_Initial_Assesment/Data
####################################
# Task 2 - Open Refine
####################################
* Extract Data with python to: Tasks/02_OpenRefine_Cleansing/Data
* Extract OpenRefine Audit logs to : Tasks/02_OpenRefine_Cleansing/Open Refine
    OpenRefine < Data/Dish.csv     > "Open Refine/Dish.json"
    OpenRefine < Data/MenuItem.csv > "Open Refine/MenuItem.json"
    OpenRefine < Data/MenuPage.csv > "Open Refine/ManuPage.json"
    OpenRefine < Data/Menu.csv     > "Open Refine/Menu.json"
* Saving Zip Archive to: Tasks/02_OpenRefine_Cleansing/Data
####################################
# Task 3 - Other Tools
####################################
* Extract Open Refine Cleansed Data with python to: Tasks/03_Data_Cleansing_Other_Tools/Data
    Processing Menu.csv
    Processing Dish.csv
    Processing MenuItem.csv
    Processing MenuPage.csv
* Saving Zip Archive to: Tasks/03_Data_Cleansing_Other_Tools/Data
####################################
# Task 4 - Relational Schema
####################################
* Extract Open Refine Cleansed Data with python to: Tasks/04_Developing_Relational_Schema/Data
* Load in SQLLite to Explore in: Tasks/04_Developing_Relational_Schema
"    Truncating Dish Names Longer than 500 Characters"
"    Correct Dishes where last_appeared < first_appeared: 6 rows affected"
"    Correct Dishes where last_appeared > Current Date  : 179 rows affected"
"    Correct Dishes where first_appeared > Current Date : 11 rows affected"
"  * Dish table corrected"
"    Correct Invalid characters in Menu                 : 179 rows affected"
"  * Menu Table Corrected"
"    MenuPage.pagenumbers resequenced                   : 175 rows affected"
"  * MenuPage Table Corrected"
"    Correcting price > high_price                      : 2378 rows affected"
"    Correcting created_at > updated_at                 : 2874 rows affected"
"  * MenuItem Table Corrected"
"    Consolidating dishes                               : 36701 rows affected"
"    Consolidating MeniItems                            : 23371 rows affected"
"    Fixing RI in MenuPage"
"    Updating Dish times appeared                       : 21693 rows affected"
"    Updating Dish menus appeared                       : 28843 rows affected"
"    Updating Dish first appeared                       : 25272 rows affected"
"    Updating Dish last appeared                        : 17857 rows affected"
"    Updating Dish prices                               : 20897 rows affected"
"    Updating Dish counts                               : 82967 rows affected"
"    Updating Dish Page counts                          : 214 rows affected"
"  * All Data Corrected with SQL-Lite"
"  * SQL-Lite DB Exported"
* Saving Zip Archive to: Tasks/04_Developing_Relational_Schema/Data
####################################
# Task 5 - Workflow Model
####################################
* Extract Open Refine Cleansed Data with python to: Tasks/05_Workflow_Model/Data
* Saving Zip Archive to: Tasks/05_Workflow_Model/Data
####################################
# Task 6 - Data Provenance
####################################
* Extract Open Refine Cleansed Data with python to: Tasks/06_Data_Provenance/Data
* Saving Zip Archive to: Tasks/06_Data_Provenance/Data
End of Data Cleansing
```

## 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/.

## Contributors:

**Asad Bin Imtiaz** [Net ID: **aimtiaz2**] worked on:

1> Initial assessment (Task 1) in identifying the potential issues, writing queries to identify issues and preparing directory structure
2> Data Cleansing with Open Refine (Task 2) in clustering records and case standardization
3> Data Cleansing with Other tools (Task3) in writing python scripts
4> Developing relational schema (Task 4) in writing SQL queries in Teradata and developing SQL Schema
5> Creating workflow model (Task 5) in creating overall workflow and directory structure, created workflow for SQLs and generating provenance graphs
6> Writing of this project report.

**Mohammad Rafay** [Net ID: **mrafay2**] worked on:

1> Initial assessment (Task 1) in identifying the defining use cases, data set structure and data quality assessment.
2> Data Cleansing with Open Refine (Task 2) in fixing data type and standardization issues.
3> Developing relational schema (Task 4) in writing SQL queries in SQL Lite and fixing RI issues.
4> Creating workflow model (Task 5) in generating workflow for OpenRefine logs and their respective provenance graphs.
5> Developing provenance (Task 6) in developing data log statements using clingo.
6> Writing of this project report.