## 1. Dataset Overview and Initial Assessment

For this project, we are using the tools that have been introduced as a part of the CS513 curriculum to clean and prepare a sample dirty dataset. Along the way we will document the steps from dirty to clean. The sections that follow describe this process in detail about Data cleaning with OpenRefine, develop Relational Database Schema and create a Workflow Model.

An initial assessment of the New York Public Library dataset (referred to now forward as 'menus' dataset) is over 45,000 historical menus. The majority of these were organized by Frank E. Buttolph (Ref1) around 1900-1921. The dates on the menus range from the 1850s to 2010s.The data contains information on restaurant menus, but also other organizations like railroad or shipping companies. The collection now contains approximately 45,000 items, about quarter of which have so far been digitized and made available in the NYPL Digital Gallery . We are using the Jul 16, 2020 version for this project.

## 2. Structure and content of the dataset

The data is in four files: Menu, MenuPage, MenuItem, and Dish.

- The 'Menu.csv' file has a unique id number, location information, venue, currency used, and other description-based information.
- The 'MenuPage.csv' file contains the id, plus an additional unique mean_id, image_id, height, width, and another with other image related information.
- The 'MenuItem.csv' file contains the id, plus an additional_page_id, dish_id, and other price related information for each dish.
- The 'Dish.csv' file contains the id, name of the dish, description, first/last appearance,and various price information.

A detailed description of each file's contents and structure are as follows:

**Menu.csv**

**id**: unique id for this menu
**name**: name on menu, name of the restaurant, or blank
**sponsor**: sponsor, usually this is the name of the restaurant
**event**: name of the meal or the event the menu was created for eg. Breakfast, dinner etc.
**venue**: location where the food is served whether it is commercial, educational etc.
**place**: often includes city, state, country, address, or name of venue
**physical description**: paper stock, dimensions, colors, design, etc. of menu
**occasion**: special occasion, holiday, daily, or blank
**notes**: additional details about the menu
**call number**: number within the NYPL collection

**keywords:** keywords on menu (mostly blank)
**language**: language the menu is printed in (mostly blank)
**date**: date the menu was collected, formatted as a string as MM/DD/YYYY
**location:** where the menu was used
**location type**: type of the location value
**currency**: money type charged for items on this menu
**currency symbol**: symbol for the currency
**status:** the digitization status of this menu – complete or under review
**page count**: number of pages on the menu
**dish count:** number of dishes on the menu

**MenuPage.csv**
**id**: unique designator for this menu item
**menu id**: specific id for menu
**page number**: page number in the menu
**image id**: a unique id for the scanned image of this menu, accessible on the NYPL site
**full height**: height of menu
**full width**: width of menu
**uuid**: another unique id for this page/image

**MenuItem.csv**
**id:** unique designator for this menu item
**menu page id**: id of the menu page that this item appears
**price**: the cost of the smallest portion of this item
**high price**: cost of the largest portion of this item
**dish id**: designator id of the dish that this menu item refers
**created at**: date/time when this entry was created
**updated at**: the most recent date when the entry was updated
**xpos**: x-axis position of the item on the scanned image
**ypos**: y-axis position of the item on the scanned image

**Dish.csv**
**id**: a unique designator for this dish
**name**: the name of this dish
**description**: a description of this dish, always blank
**menus appeared**: number of menus this dish appears on
**times appeared**: number of times this dish appears (including additional sections)
**first appeared**: year this dish first appeared (also can be: 0, 1 or 2928)
**last appeared**: year this dish last appeared (also can be: 0, 1 or 2928)
**lowest price**: lowest price that this item was sold for
**highest price**: highest price that this item was sold for

## 3. Data quality issues

The following are the main data quality issues encountered in all the files:

- Trailing and leading white spaces.
- Consecutive white spaces.
- Existence of special characters like (%, #, !, /, (, ), [, ], ?)
- Date outliers like 0,1 or 2928.
- Convert all column values to upper case

## 4. Use cases

The data is not really that clean for any practical use case scenario.

- Use cases for which data is clean already:
  - Getting an overall idea of the number of menus and dishes in a particular time frame.
  - Analyzing the dish dataset to gauge the popularity of any dish.
  - The Menu Page and the Menu item together can be used to get the information about when a particular menu item appeared in a menu.

- Use cases once the data is cleaned:
  - The Menu Page and the Menu item together can be used to get the information about when a particular menu item appeared in a menu.
  - Find out the structure of a menu.
  - How the price of a dish has changed over a period of time.

Here we will be discussing how the dish data will be cleaned and how we analyze the price increase of any dish. The data cleaning steps are discussed in the OpenRefine and Tableau Prep section. Once the data is cleaned, we can look at any dish and see how the prices have increased over time.

## 5. Data Cleaning with OpenRefine and Tableau Prep

OpenRefine, formally called Google Refine, is an open source desktop application the has many helpful features for data cleaning. It behaves like a database with rows and cells under columns which are similar to relational database tables. An OpenRefine project itself consists of one table. We cleaned each of the four data file separately, some more than others as needed.The major OpenRefine feature that will be helpful in cleaning our dataset is the clustering feature apart from the standard textual data cleaning operations. The option allows the user to cluster similar text and replace it with a more standardized description. The common problem this solves the many variants of spelling but reference the same object. The following subsections will describe the step-by-step process from input file to output file.

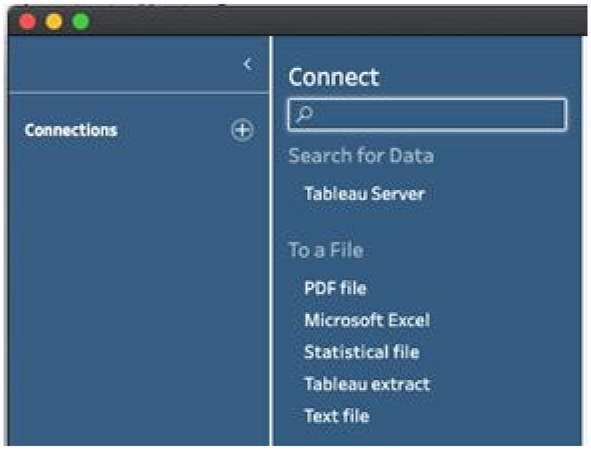
We have also used Tableau Prep for cleaning the Dish.csv since the file was huge. Tableau Prep is one of the most sophisticated and easy to use tools for cleaning, it has got many prebuilt features such as selecting a range of values, removing null values, grouping the values based on multiple options including ‘Pronunciation’, ‘Common Characters’,‘Spelling’ or even ‘Manual Selection’. The operations performed on each column from the dataset can be viewed independently (unlike OpenRefine, where all the operations can be exported into one single '.json' file) making it easy to understand. It has got a very powerful User Interface, an added advantage for people to get an idea by simply glancing at it. Provenance (or sequence of steps to deduce a calculation) is easily understood by using this tool. The major benefit with Tableau Prep, as soon as the data is cleaned it can be published to the server or opened in Tableau Desktop for further analysis (power tool for analyzing the data).

5.1 Menu.csv Cleaning

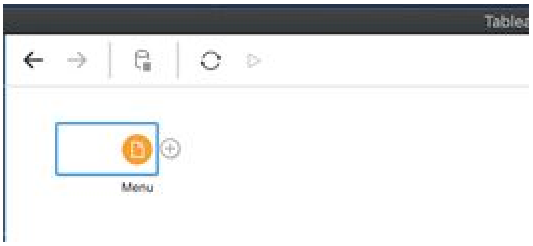
- Convert id, page_count, dish_count to Number format
- Transformations for columns name, sponsor, event, venue, occasion, location are as follows -
    - Remove punctuations by using the regex as shown - 'grel:value.replace(/([%#!\[\]\(\)\'\"\.\?\,-])/,'')'.
    - Trim empty spaces using 'value.replace(/s+/,'')
    - Make the values to Uppercase
- For column date used the following regex to filter valid dates - grel:if(or(datePart(toDate(value, "years") < 1851, datePart(toDate(value), "years") > 2020, '' , value)
- Removed columns 'keywords' and 'language' as these are mostly NULL values.

The following snapshots will describe the steps of cleaning the Menu.csv file in Tableau Prep:

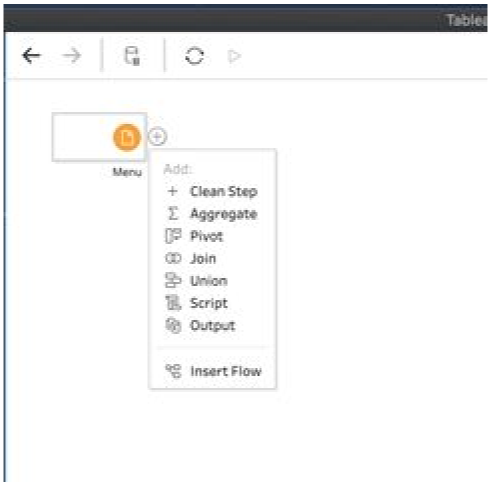
a. Create a connection in Tableau Prep and select 'Text file' to browse Menu.csv file



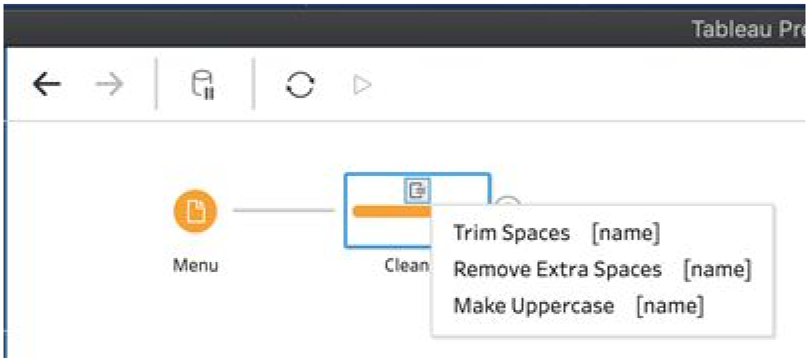
b. A connection is created and 'Menu' dataset is visible as shown below –



c. Click on the '+' sign and start cleaning the data as shown below –

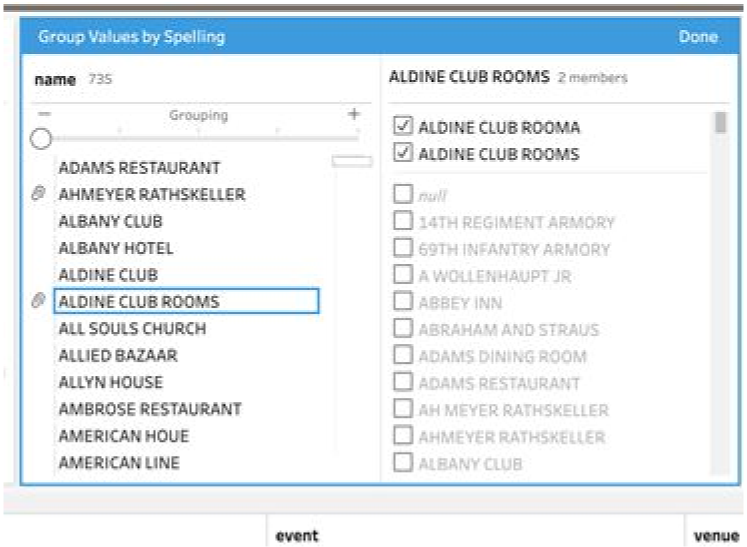


d. 1. Perform different operations and view the changes by hovering on the node as shown below –



Trim Spaces [name]
Remove Extra Spaces [name]
Make Uppercase [name]

e. 1. As a part of cleaning step, values can be grouped based on Spelling as shown below –



Group Values by Spelling    Done

name 735

Grouping

ADAMS RESTAURANT
AHMEYER RATHSKELLER
ALBANY CLUB
ALBANY HOTEL
ALDINE CLUB
ALDINE CLUB ROOMS
ALL SOULS CHURCH
ALLIED BAZAAR
ALLYN HOUSE
AMBROSE RESTAURANT
AMERICAN HOUE
AMERICAN LINE

ALDINE CLUB ROOMS 2 members

☑ ALDINE CLUB ROOMA
☑ ALDINE CLUB ROOMS
□ *null*
□ 14TH REGIMENT ARMORY
□ 69TH INFANTRY ARMORY
□ A WOLLENHAUPT JR
□ ABBEY INN
□ ABRAHAM AND STRAUS
□ ADAMS DINING ROOM
□ ADAMS RESTAURANT
□ AH MEYER RATHSKELLER
□ AHMEYER RATHSKELLER
□ ALBANY CLUB

event    venue

f. Following describes the complete workflow:

Let's understand each node in the above diagram in more details:

| **Node Name** | **Operation Type** | **Sequence of Action** |
|---|---|---|
| **Menu** | Dataset | Considered data sample (default) |
| **clean_menu_name** | Clean | a. Trim Spaces<br>b. Remove Punctuation<br>c. Remove Extra Spaces<br>d. Make Uppercase<br>e. Trim Spaces<br>f. Remove Extra Spaces<br>g. Group values based on Spelling |

| | | |
|---|---|---|
| **clean_menu_sponsor** | Clean | a. Trim Spaces<br>b. Remove Punctuation<br>c. Remove Extra Spaces<br>d. Make Uppercase<br>e. Trim Spaces<br>f. Remove Extra Spaces<br>g. Group values based on Spelling |
| **clean_menu_event** | Clean | a. Trim Spaces<br>b. Remove Extra Spaces<br>c. Remove Punctuation<br>d. Trim Spaces<br>e. Remove Extra Spaces<br>f. Make Uppercase<br>g. Remove Numbers<br>h. Remove Extra Spaces<br>i. Group values based on Spelling |
| **clean_menus_venue** | Clean | a. Remove Punctuation<br>b. Trim Spaces<br>c. Remove Extra Spaces<br>d. Group values based on Spelling |
| **clean_menu_place** | Clean | a. Remove Punctuation<br>b. Trim Spaces<br>c. Remove Extra Spaces<br>d. Group values based on Spelling |

| | | |
|---|---|---|
| **clean_menu_pd (physical_description)** | Clean | a. Trim Spaces<br>b. Remove Extra Spaces<br>c. Make Uppercase |
| **clean_menu_occasion** | Clean | a. Remove Punctuation<br>b. Trim Spaces<br>c. Remove Extra Spaces<br>d. Make Uppercase<br>e. Group values based on Spelling |
| **clean_notes** | Clean | a. Remove Punctuation<br>b. Trim Spaces<br>c. Remove Extra Spaces<br>d. Make Uppercase<br>e. Remove Numbers<br>f. Group values based on Spelling |
| **clean_call_no** | Clean | a. Trim Spaces<br>b. Remove Extra Spaces |
| **clean_keywords** | Clean | Remove field as all these values are null or blanks |
| **clean_lang** | Clean | Remove field as all these values are null or blanks |
| **clean_date** | Clean | a. Filter to exclude Null values<br>b. Filter to select range of dates from 01/01/1851 onwards. |

| | | |
|---|---|---|
| **clean_location** | Clean | a. Remove Punctuation<br>b. Trim Spaces<br>c. Remove Extra Spaces<br>d. Make Uppercase<br>e. Group values based on Spelling |
| **clean_loc_type** | Clean | Remove field as all these values are null or blanks |
| **clean_currency** | Clean | a. Remove Punctuation<br>b. Trim Spaces<br>c. Remove Extra Spaces<br>d. Make Uppercase<br>e. Filter to exclude Null values |
| **clean_currency_symbol** | Clean | a. Trim Spaces<br>b. Remove Extra Spaces<br>c. Filter to exclude Null values |
| **clean_status** | Clean | a. Trim Spaces<br>b. Make Uppercase<br>c. Remove Extra Spaces |
| **clean_page_count** | Clean | a. Filter to exclude Null values<br>b. Filter to select the range of values from 1 and above (as page counts cannot be ‘0’ or less) |
| **clean_dish_count** | Clean | Filter to select values more than ‘0’ |

| | | |
|---|---|---|
| **Menu_Clean_File** | Output | a. Select 'Location'<br>b. Select 'Output type' as 'Comma Separated Values (.csv) |

**Tableau Desktop Analysis –**

a. Click on the last cleaning step to see the cleaned data in Tableau Desktop as shown below:

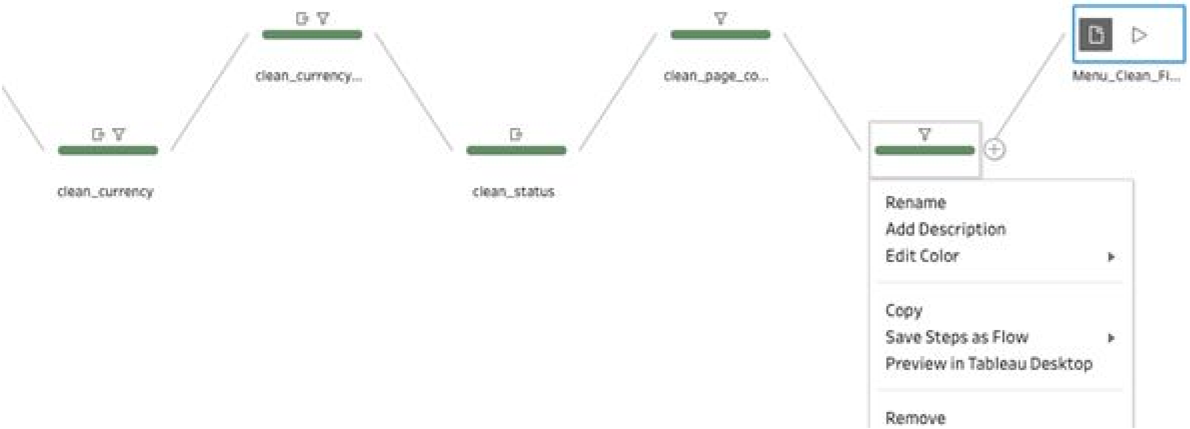
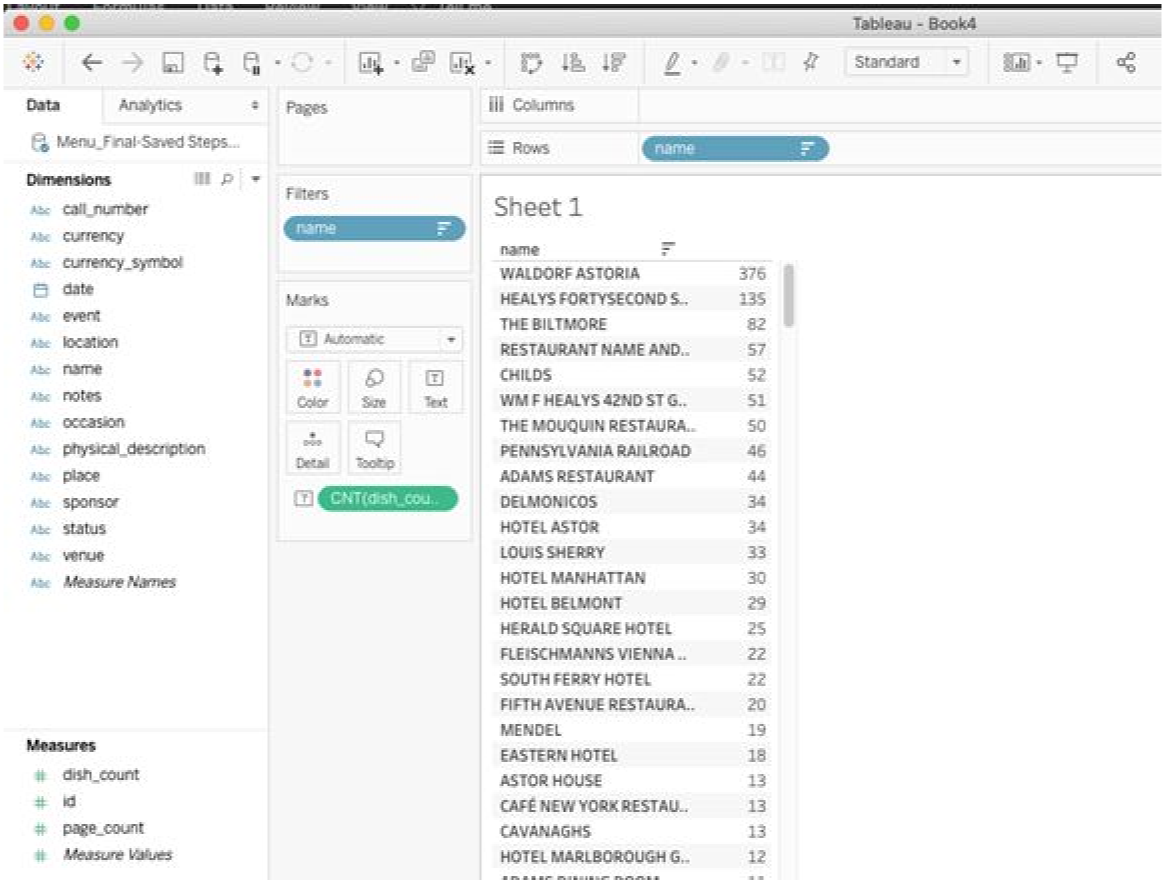


**Tableau Desktop Analysis Results –**

a. We can see the menu names and associated dish counts for each of them in descending order as shown –



Sheet 1

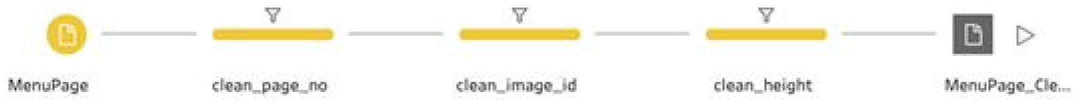
| name | |
|---|---|
| WALDORF ASTORIA | 376 |
| HEALYS FORTYSECOND S.. | 135 |
| THE BILTMORE | 82 |
| RESTAURANT NAME AND.. | 57 |
| CHILDS | 52 |
| WM F HEALYS 42ND ST G.. | 51 |
| THE MOUQUIN RESTAURA.. | 50 |
| PENNSYLVANIA RAILROAD | 46 |
| ADAMS RESTAURANT | 44 |
| DELMONICOS | 34 |
| HOTEL ASTOR | 34 |
| LOUIS SHERRY | 33 |
| HOTEL MANHATTAN | 30 |
| HOTEL BELMONT | 29 |
| HERALD SQUARE HOTEL | 25 |
| FLEISCHMANNS VIENNA .. | 22 |
| SOUTH FERRY HOTEL | 22 |
| FIFTH AVENUE RESTAURA.. | 20 |
| MENDEL | 19 |
| EASTERN HOTEL | 18 |
| ASTOR HOUSE | 13 |
| CAFÉ NEW YORK RESTAU.. | 13 |
| CAVANAGHS | 13 |
| HOTEL MARLBOROUGH G.. | 12 |

## 5.2 MenuPage.csv Cleaning

- Change the data type for id, menu_id,page_number,image_id,full_height,full_width into Number format.
- Used 'grel:if(value>=1, value, '')' to consider page numbers which are equal to or more than 1. As '0' or null values are not valid page_numbers
- Left 'uuid' column as it is, no changes.

The following snapshots will describe the steps of cleaning the Menu.csv file in Tableau Prep:

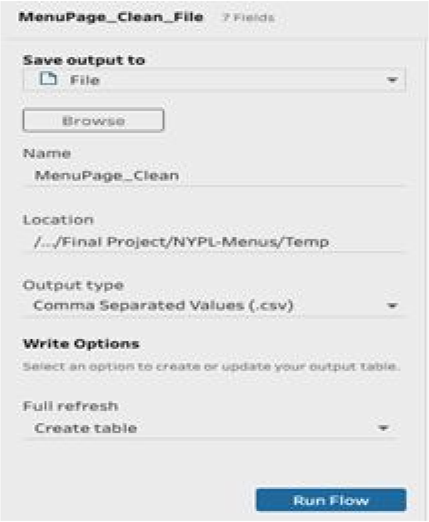
a. A series of cleaning steps are performed to clean ‘MeuPage.csv’ file.
b. Steps to create the flow are the same as discussed previously.
c. We create a connection for ‘MenuPage.csv’ file and let the tool consider a sample of data rather than the entire data.



Let us understand more about each node in this cleaning flow from the below table.

| Node Name | Operation Type | Sequence of Action |
|---|---|---|
| **MenuPage** | Dataset | Considered data sample (default) |
| **clean_page_no** | Clean | Filter to exclude Null values |
| **clean_image_id** | Clean | Filter to exclude Null values |
| **clean_height** | Clean | Filter to exclude Null values |
| **MenuPage_Clean_File** | Output | a. Select 'Location'<br>b. Select 'Output type' as 'Comma Separated Values (.csv) |

We can run the flow, by clicking the 'Run Flow' (refer below screenshot):



MenuPage_Clean_File 7 Fields

Save output to: File

Name: MenuPage_Clean

Location: /.../Final Project/NYPL-Menus/Temp

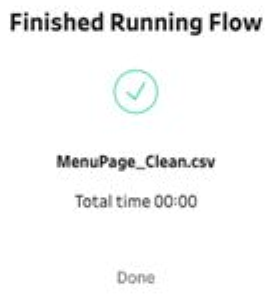
Output type: Comma Separated Values (.csv)

Write Options

Select an option to create or update your output table.

Full refresh: Create table

Following snapshot shows the successful execution:



Finished Running Flow

MenuPage_Clean.csv

Total time 00:00

**Tableau Desktop Analysis**:

a. Click on the last cleaning step to see the cleaned data in Tableau Desktop as shown below:

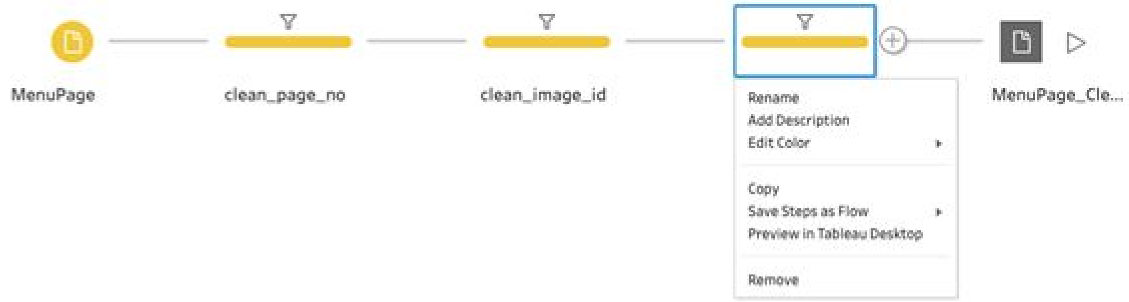
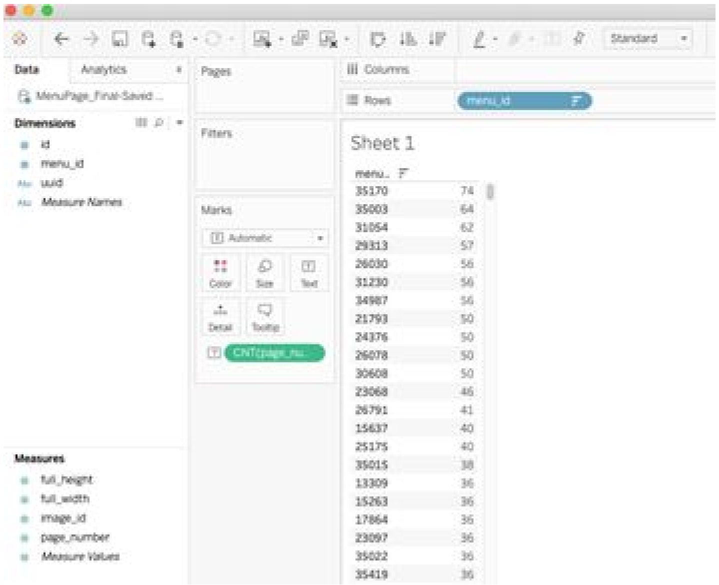


**Tableau Desktop Analysis Results:**

a. We can see the page numbers and menu id’s association in descending order as shown:



Sheet 1

| menu.. | |
|---|---|
| 35170 | 74 |
| 35003 | 64 |
| 31054 | 62 |
| 29313 | 57 |
| 26030 | 56 |
| 31230 | 56 |
| 34987 | 56 |
| 21793 | 50 |
| 24376 | 50 |
| 26078 | 50 |
| 30608 | 50 |
| 23068 | 46 |
| 26791 | 41 |
| 15637 | 40 |
| 25175 | 40 |
| 35015 | 38 |
| 13309 | 36 |
| 15263 | 36 |
| 17864 | 36 |
| 23097 | 36 |
| 35022 | 36 |
| 35419 | 36 |

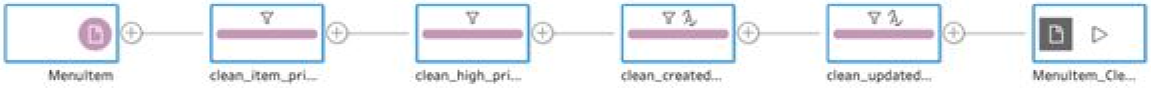
## 5.3 MenuItem.csv Cleaning

- Transform columns id, menu_page_id, price, high_price, dish_id, xpos, ypos to Number format
- For price, high_price columns, used 'grel:if(value>=0.01, value, '') to consider values which are equal to or more than 0.01, eliminating records with '0' or null values.
- For created_at, updated_at performed the following operations -
  a. Replace 'UTC' from the string, to make it a valid time value using 'grel:value.replace('UTC','')'
  b. Convert these values to date format.
  c. Used grel:if(or(datePart(toDate(value, "years") < 1851, datePart(toDate(value), "years") > 2020, '' , value) to filter date values which are less than 1851 or more than 2020, to have a valid selection.

The following steps will provide an explanation for cleaning this dataset:

a. A series of cleaning steps are performed to clean the 'MeuItem.csv' file.
b. Steps to create the flow are the same as discussed previously.
c. We create a connection for 'MenuItem.csv' file and let the tool consider a sample of data than the entire data.

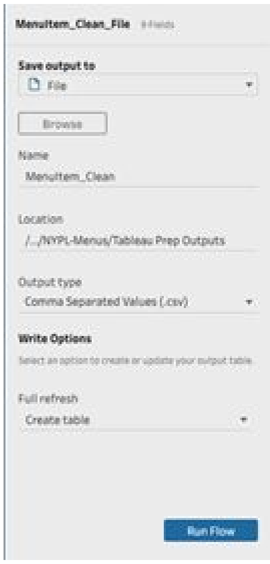
d. Here is the complete process flow:



Let us understand more about each node in this cleaning flow from the below table.

| **Node Name** | **Operation Type** | **Sequence of Action** |
|---|---|---|
| **MenuItem** | Dataset | Considered data sample (default) |
| **clean_item_price** | Clean | a. Filter to exclude Null values<br>b. Filter to select range of values from 0.01 and above (as item price cannot be '0' or less than) |
| **clean_high_price** | Clean | a. Filter to exclude Null values<br>b. Filter to select range of values from 0.01 and above (as item price cannot be '0' or less than) |
| **clean_created_at** | Clean | a. Change type to date & time type<br>b. Filter range of dates for valid dates. |
| **clean_updated_at** | Clean | a. Change type to date & time type<br>b. Filter range of dates for valid dates. |
| **MenuItem_Clean_File** | Output | a. Select 'Location'<br>b. Select 'Output type' as 'Comma Separated Values (.csv) |

Running the flow, by clicking the 'Run Flow', we get the below snapshot:



MenuItem_Clean_File 8 Fields

Save output to: File

Name: MenuItem_Clean

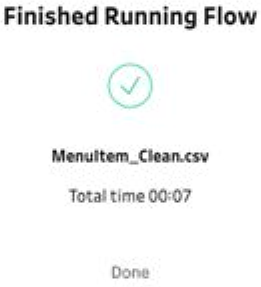
Location: /.../NYPL-Menus/Tableau Prep Outputs

Output type: Comma Separated Values (.csv)

Write Options

Select an option to create or update your output table

Full refresh: Create table



Finished Running Flow

MenuItem_Clean.csv

Total time 00:07

**Tableau Desktop Analysis**:

a. Click on the last cleaning step to see the cleaned data in Tableau Desktop as shown below:

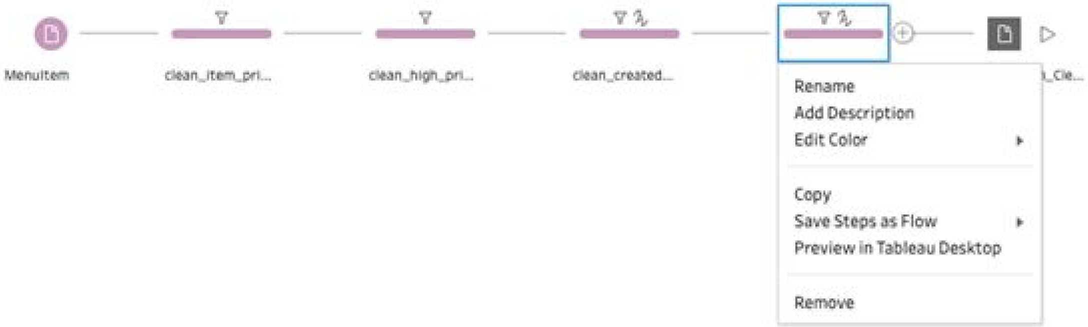
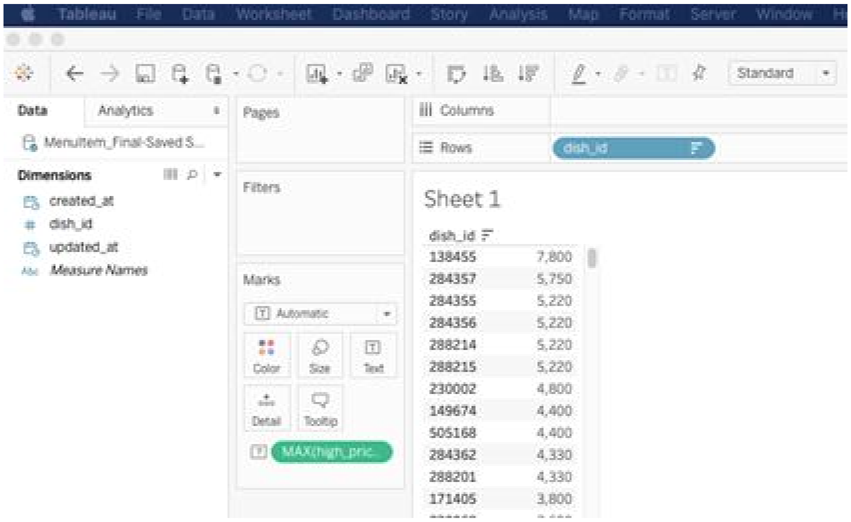


**Tableau Desktop Analysis Results**:

a. We can see the dish id's and maximum high price association with them in descending order as shown below:



| dish_id |  |
|---|---|
| 138455 | 7,800 |
| 284357 | 5,750 |
| 284355 | 5,220 |
| 284356 | 5,220 |
| 288214 | 5,220 |
| 288215 | 5,220 |
| 230002 | 4,800 |
| 149674 | 4,400 |
| 505168 | 4,400 |
| 284362 | 4,330 |
| 288201 | 4,330 |
| 171405 | 3,800 |

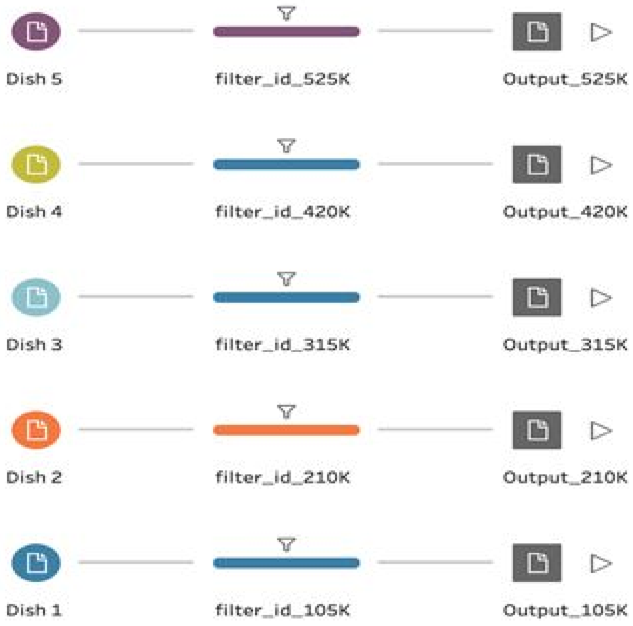
## 5.4 Dish.csv Cleaning (using Tableau Prep)

- We were unable to get it working in OpenRefine as this is very huge data file.
- Thus divided this file into 5 equal parts using Tableau Prep, snapshot as shown below (we can take from my draft)
- For each one of the partitioned Dish files, we performed clean operations.
- For columns name, replaced punctuation by using the regex - grel:value.replace(/([%#!\[\]\(\)\'\"\.\?\,-])/,'')'.
- Trimmed empty spaces using grel:value.replace(/s+/,'')
- Then converted all the values to Uppercase.
- Performed cluster for column 'name'.
- Transform columns id, lowest_price, highest_price to Numbers.
- Keep the values for lowest_price, highest_price which are equal to or more than 0.01, using the grel:if(value >= 0.01, value, '')
- Convert first_appeared, last_appeared to Date format and clean the dates to only keep dates which are in between 1851 to 2020, using the grel:if(or(datePart(value),"years") <= 1851, datePart(toDate(value),"years") > 2020), '', value)

The following steps describe the sequence of data cleaning for this dataset:

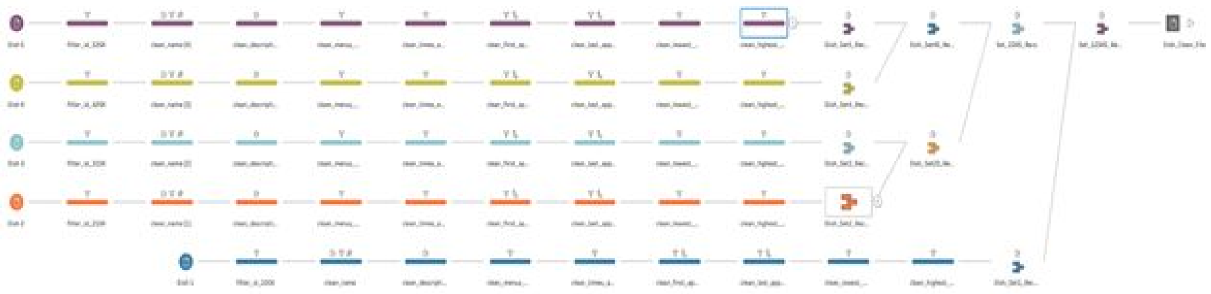
a. A series of cleaning steps are performed to clean Dish.csv' file.
b. Steps to create the flow are same as discussed previously.
c. We create a connection for 'Dish.csv' file and let the tool consider a sample of data rather than the entire data.

d. This being a very huge dataset (unable to clean it through Open Refine right away) we have opted to split this into 5 equal parts of 105K records each. This can be easily done through Tableau Prep as shown below:



- The same data set 'Dish.csv' is used to create 5 outputs by creating a clean step to filter the 'id' values:
    - Dish 1 – Values from 0 to 104,999
    - Dish 2 – Values from 105,000 to 209,999
    - Dish 3 – Values from 210,000 to 314,999
    - Dish 4 – Values from 315,000 to 419,999
    - Dish 5 – Values from 420,000 to 525,000
- An output step to save the filtered file
- In the file output step, the configuration to have a 'comma separated value (.csv)' as output type is chosen.
- The flow can be run using the 'Run Flow' option which triggers the flow and creates the output file.
- The same steps are followed for all the 5 flows to generate 5 'csv' output files for cleaning.

Below are the snapshots:

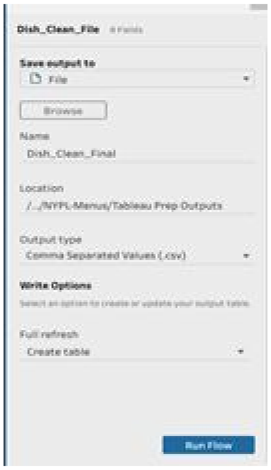


- A series of cleaning steps and unions are used in obtaining the final cleaned Dish.csv file.
- Each flow is designed to work on records in parallel and clean the records accordingly.
- The flow starts with a dataset (eg. Dish 1, Dish 2 etc) followed by a clean step.
- We have put each column clean step independent so that it will be clear for a user looking at the cleaning steps that were performed on each column.
- These clean steps are common for all the 5 data sets of Dish.csv and are as follows:

| Node Name | Operation Type | Sequence of Action |
|---|---|---|
| **Dish** | Dataset | Considered data sample (default) |
| **filter_id** | Clean | Selected range for 105 K records |
| **clean_name** | Clean | a. Trim Spaces<br>b. Remove Punctuation<br>c. Remove Extra Spaces<br>d. Make Uppercase<br>e. Apply Regex ‘NOT REGEXP_MATCH([name],’^\d+$’) to select only string values and not numeric<br>f. Remove Extra Spaces<br>g. Exclude empty spaces or banks<br>h. Group values based on Spelling |

| | | |
|---|---|---|
| **clean_description** | Clean | Remove field as all these values are null or blanks |
| **clean_menus_appeared** | Clean | Filter by range of values from 1 and above (menus appeared cannot be 0 or less) |
| **clean_times_appeared** | Clean | Filter by range of values from 1 and above (times appeared cannot be 0 or less) |
| **clean_first_appeared** | Clean | a. Filter by excluding values from 0 to 100 (these are not valid date values)<br>b. Change Type to Date type<br>c. Filter by selecting a range of dates from 01/01/1850 and above |
| **clean_last_appeared** | Clean | a. Change Type to Date type<br>b. Filter by selecting a range of dates from 01/01/1850 and above |
| **clean_lowest_price** | Clean | a. Filter null values (Exclude Null values)<br>b. Filter range of values from 0.01 and above (lowest price cannot be ‘0’ practically the item cannot be free) |
| **clean_highest_price** | Clean | a. Filter null values (Exclude Null values)<br>b. Filter range of values from 0.01 and above (highest price cannot be ‘0’) |
| **After the above steps are performed there are multiple Union operations performed to merge the cleaned data records to a single file, ‘Union’ operation can be performed between two values only, so use multiple ‘Union’ steps to obtain the merged file.** | | |
| **Dish_Clean_File** | Output | a. Select ‘Location’<br>b. Select ‘Output type’ as ‘Comma Separated Values (.csv) |

Run the flow by clicking the ‘Run Flow’ button (refer below screenshot):



Dish_Clean_File 8 Fields

Save output to
File

Name
Dish_Clean_Final

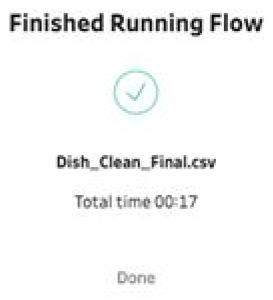
Location
/.../NYPL-Menus/Tableau Prep Outputs

Output type
Comma Separated Values (.csv)

Write Options
Select an option to create or update your output table.

Full refresh
Create table

Successful execution of ‘Dish_Clean_File’ –



Finished Running Flow

Dish_Clean_Final.csv

Total time 00:17

**Tableau Desktop Analysis:**

a. Click on the last cleaning step to see the cleaned data in Tableau Desktop as shown below

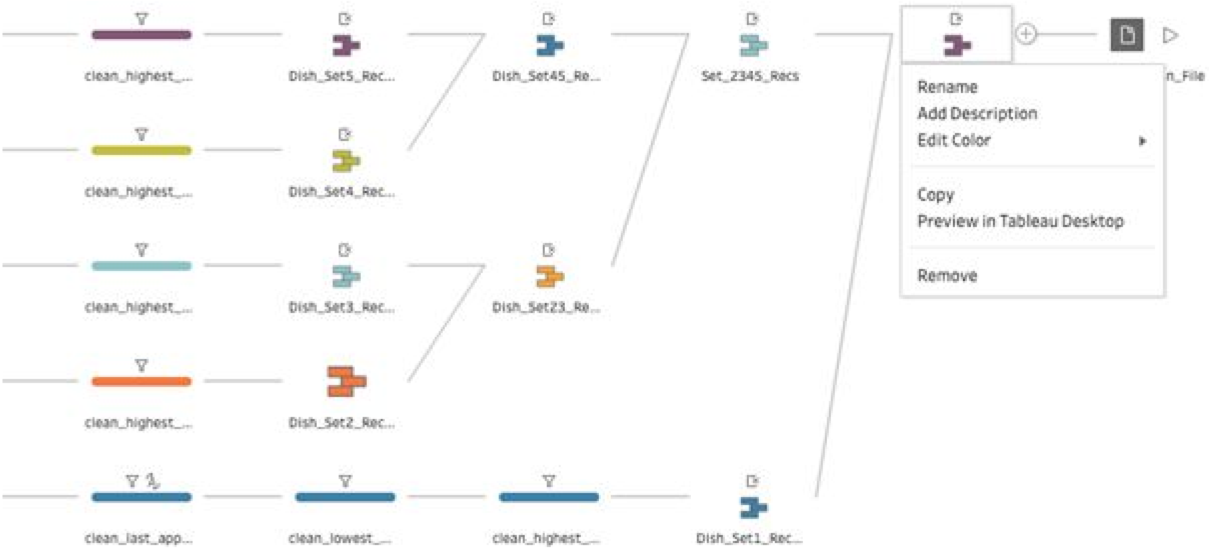
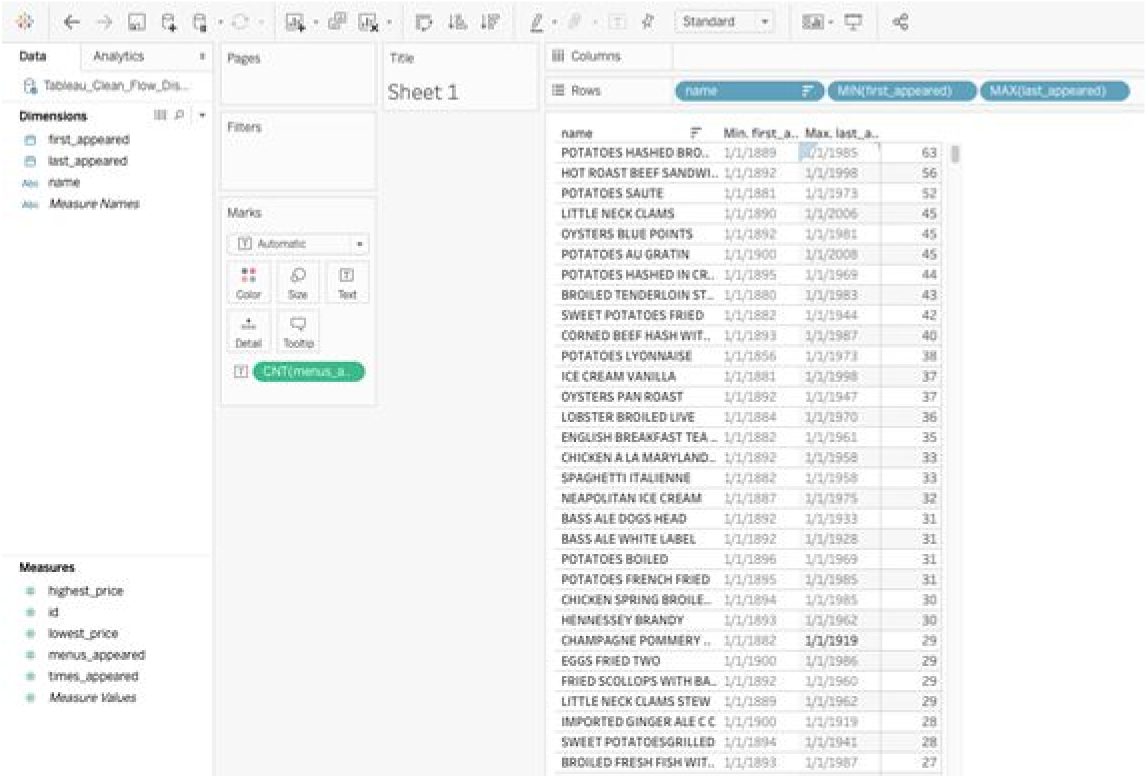


**Tableau Desktop Analysis Results:**

a. We can see the relation between dish names, when they first appeared, last appeared and the count of menu's they have appeared in descending order as shown below:



| name | Min. first_a.. | Max. last_a.. | |
|---|---|---|---|
| POTATOES HASHED BRO.. | 1/1/1889 | 1/1/1985 | 63 |
| HOT ROAST BEEF SANDWI.. | 1/1/1892 | 1/1/1998 | 56 |
| POTATOES SAUTE | 1/1/1881 | 1/1/1973 | 52 |
| LITTLE NECK CLAMS | 1/1/1890 | 1/1/2006 | 45 |
| OYSTERS BLUE POINTS | 1/1/1892 | 1/1/1981 | 45 |
| POTATOES AU GRATIN | 1/1/1900 | 1/1/2008 | 45 |
| POTATOES HASHED IN CR.. | 1/1/1895 | 1/1/1969 | 44 |
| BROILED TENDERLOIN ST.. | 1/1/1880 | 1/1/1983 | 43 |
| SWEET POTATOES FRIED | 1/1/1882 | 1/1/1944 | 42 |
| CORNED BEEF HASH WIT.. | 1/1/1893 | 1/1/1987 | 40 |
| POTATOES LYONNAISE | 1/1/1856 | 1/1/1973 | 38 |
| ICE CREAM VANILLA | 1/1/1881 | 1/1/1998 | 37 |
| OYSTERS PAN ROAST | 1/1/1892 | 1/1/1947 | 37 |
| LOBSTER BROILED LIVE | 1/1/1884 | 1/1/1970 | 36 |
| ENGLISH BREAKFAST TEA .. | 1/1/1882 | 1/1/1961 | 35 |
| CHICKEN A LA MARYLAND.. | 1/1/1892 | 1/1/1958 | 33 |
| SPAGHETTI ITALIENNE | 1/1/1882 | 1/1/1958 | 33 |
| NEAPOLITAN ICE CREAM | 1/1/1887 | 1/1/1975 | 32 |
| BASS ALE DOGS HEAD | 1/1/1892 | 1/1/1933 | 31 |
| BASS ALE WHITE LABEL | 1/1/1892 | 1/1/1928 | 31 |
| POTATOES BOILED | 1/1/1896 | 1/1/1969 | 31 |
| POTATOES FRENCH FRIED | 1/1/1895 | 1/1/1985 | 31 |
| CHICKEN SPRING BROILE.. | 1/1/1894 | 1/1/1985 | 30 |
| HENNESSEY BRANDY | 1/1/1893 | 1/1/1962 | 30 |
| CHAMPAGNE POMMERY .. | 1/1/1882 | 1/1/1919 | 29 |
| EGGS FRIED TWO | 1/1/1900 | 1/1/1986 | 29 |
| FRIED SCOLLOPS WITH BA.. | 1/1/1892 | 1/1/1960 | 29 |
| LITTLE NECK CLAMS STEW | 1/1/1889 | 1/1/1962 | 29 |
| IMPORTED GINGER ALE C C | 1/1/1900 | 1/1/1919 | 28 |
| SWEET POTATOESGRILLED | 1/1/1894 | 1/1/1941 | 28 |
| BROILED FRESH FISH WIT.. | 1/1/1893 | 1/1/1987 | 27 |

## 6. Data Cleaning Steps for Use Case

Here we discuss the use case, how the dish data will be cleaned and how we analyze the price increase of any dish. Once we have cleaned the Dish data as described in 5.4, we can take a look at the data to see how the prices have increased for a given dish over a period of time.

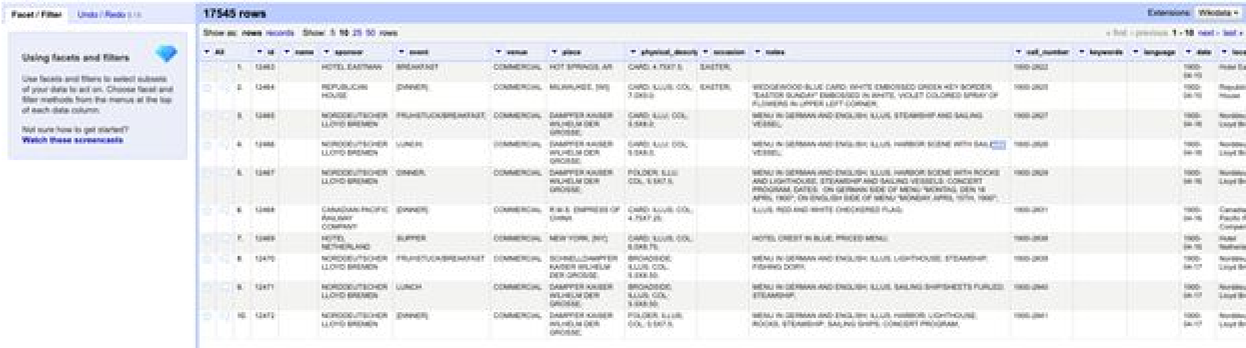
## 7. Description of the data cleaning steps with supplemental information

Here we discuss the various steps taken to clean the data using OpenRefine and also provide the sequence of steps that have been followed.

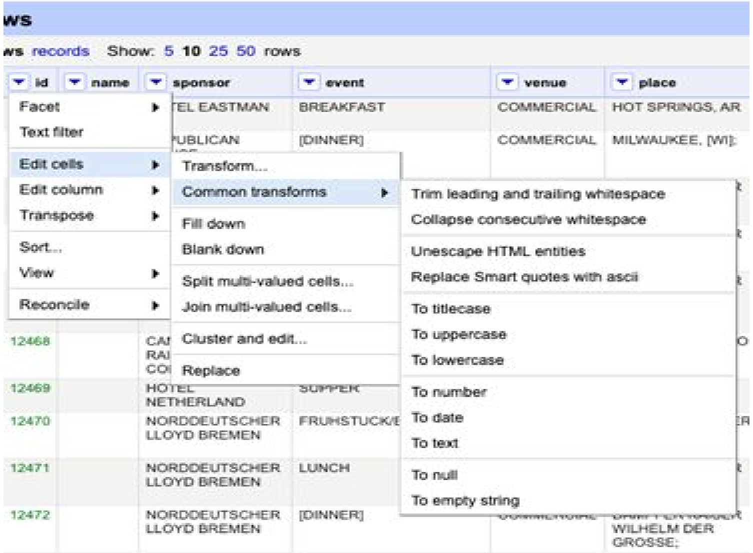
### 7.1 Menu.csv

The following snapshots will show the sequence of steps for cleaning the Menu.csv file.

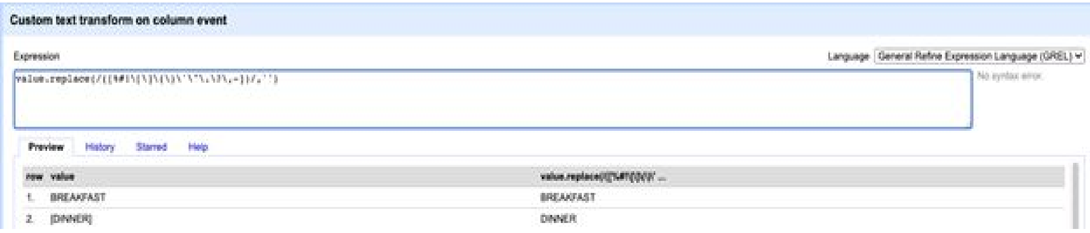
a. Import the dataset into OpenRefine.



b. Convert the columns id, dish_count, page_count to number.



c. Remove the special characters using regex pattern in grel for columns ‘name’, ‘sponsor’ , ‘event’, ‘venue’, place’ , ‘physical_description’, ‘occasiion’, ‘notes’, ‘cell_number’, ‘location’, ‘currency’ and ‘status’.



**Custom text transform on column event**

Expression

```
value.replace(/[[%#!\[\]\(\)\'\"\.\?\,-]]/,'')
```

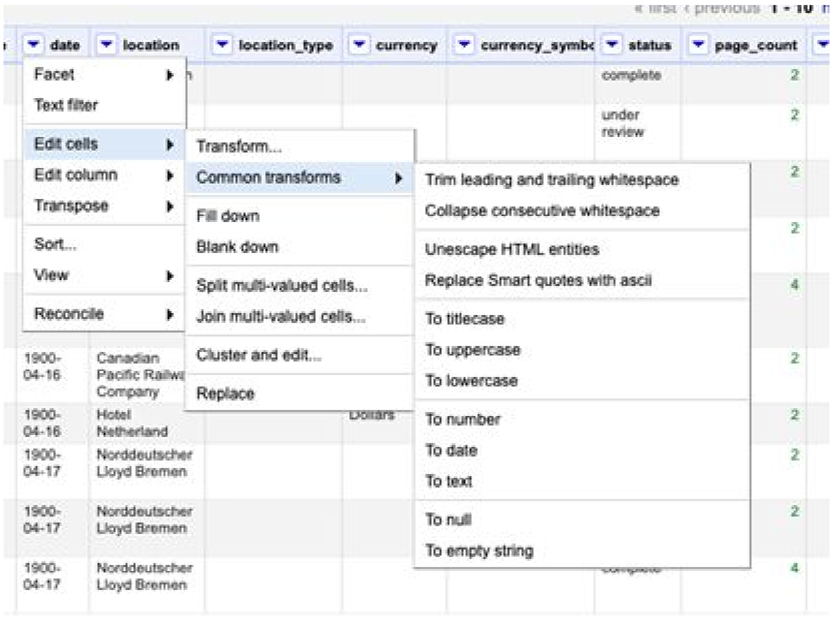
Language: General Refine Expression Language (GREL)

No syntax error.

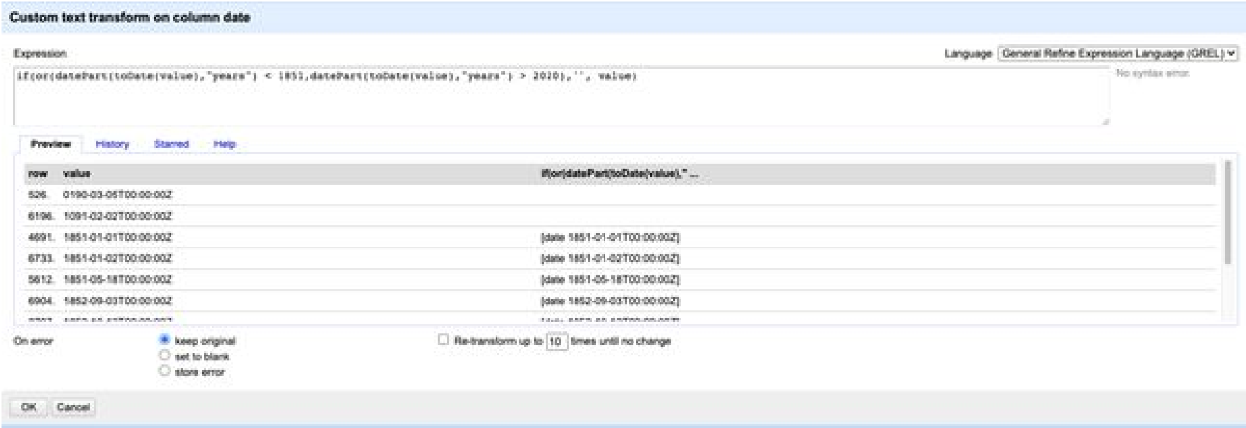
Preview | History | Starred | Help

| row | value | value.replace(/[[%#!\[\]\(\)\' ... |
|---|---|---|
| 1. | BREAKFAST | BREAKFAST |
| 2. | [DINNER] | DINNER |

d. Convert the column ‘date’ to date format.



e. Check for valid dates in column ‘date’.



**Custom text transform on column date**

Expression

```
if(or(datePart(toDate(value),"years") < 1851,datePart(toDate(value),"years") > 2020),'', value)
```

Language: General Refine Expression Language (GREL)

No syntax error.

Preview | History | Starred | Help

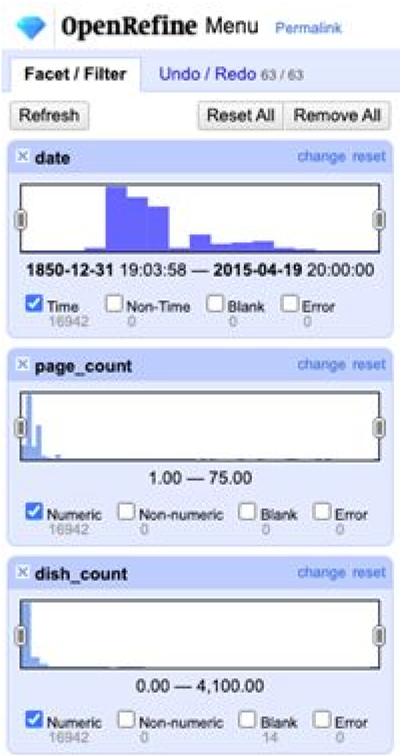
| row | value | if(or(datePart(toDate(value),"... |
|---|---|---|
| 526. | 0190-03-05T00:00:00Z | |
| 6196. | 1091-02-02T00:00:00Z | |
| 4691. | 1851-01-01T00:00:00Z | [date 1851-01-01T00:00:00Z] |
| 6733. | 1851-01-02T00:00:00Z | [date 1851-01-02T00:00:00Z] |
| 5612. | 1851-05-18T00:00:00Z | [date 1851-05-18T00:00:00Z] |
| 6904. | 1852-09-03T00:00:00Z | [date 1852-09-03T00:00:00Z] |

On error ● keep original ○ set to blank ○ store error

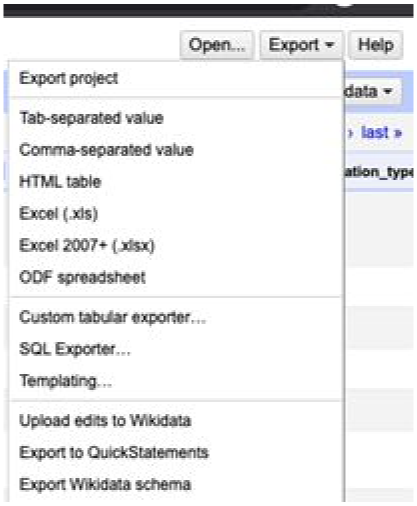
□ Re-transform up to 10 times until no change

OK Cancel

f. Then we use the facet to filter the results.



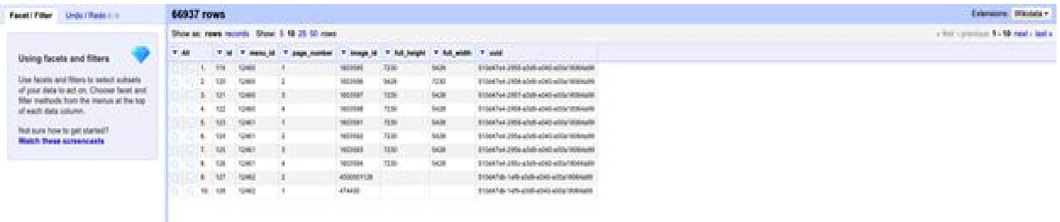
g. Then we export the results to a clean .csv file as shown below.



7.2 **MenuPage.csv**

The following snapshots will show the sequence of steps for cleaning the MenuPage.csv file.

- Import the dataset into OpenRefine.



Facet / Filter | Undo / Redo 0 / 0

**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

**64937 rows**

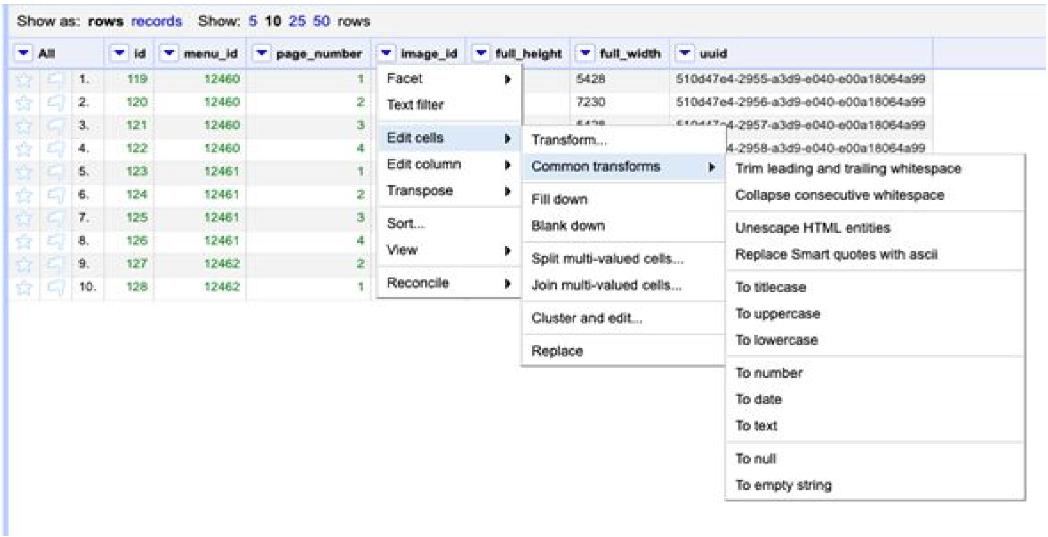
Extensions: Wikidata ▾

Show as: rows records Show: 5 10 25 50 rows

« first ‹ previous 1 - 10 next › last »

| All | id | menu_id | page_number | image_id | full_height | full_width | uuid |
|---|---|---|---|---|---|---|---|
| 1. | 119 | 12460 | 1 | 1603595 | 7230 | 5428 | 510d47e4-2955-a3d9-e040-e00a18064a99 |
| 2. | 120 | 12460 | 2 | 1603596 | 5428 | 7230 | 510d47e4-2956-a3d9-e040-e00a18064a99 |
| 3. | 121 | 12460 | 3 | 1603597 | 7230 | 5428 | 510d47e4-2957-a3d9-e040-e00a18064a99 |
| 4. | 122 | 12460 | 4 | 1603598 | 7230 | 5428 | 510d47e4-2958-a3d9-e040-e00a18064a99 |
| 5. | 123 | 12461 | 1 | 1603591 | 7230 | 5428 | 510d47e4-2959-a3d9-e040-e00a18064a99 |
| 6. | 124 | 12461 | 2 | 1603592 | 7230 | 5428 | 510d47e4-295a-a3d9-e040-e00a18064a99 |
| 7. | 125 | 12461 | 3 | 1603593 | 7230 | 5428 | 510d47e4-295b-a3d9-e040-e00a18064a99 |
| 8. | 126 | 12461 | 4 | 1603594 | 7230 | 5428 | 510d47e4-295c-a3d9-e040-e00a18064a99 |
| 9. | 127 | 12462 | 2 | 4000001128 | | | 510d47db-1af8-a3d9-e040-e00a18064a99 |
| 10. | 128 | 12462 | 1 | 474430 | | | 510d47db-1af9-a3d9-e040-e00a18064a99 |

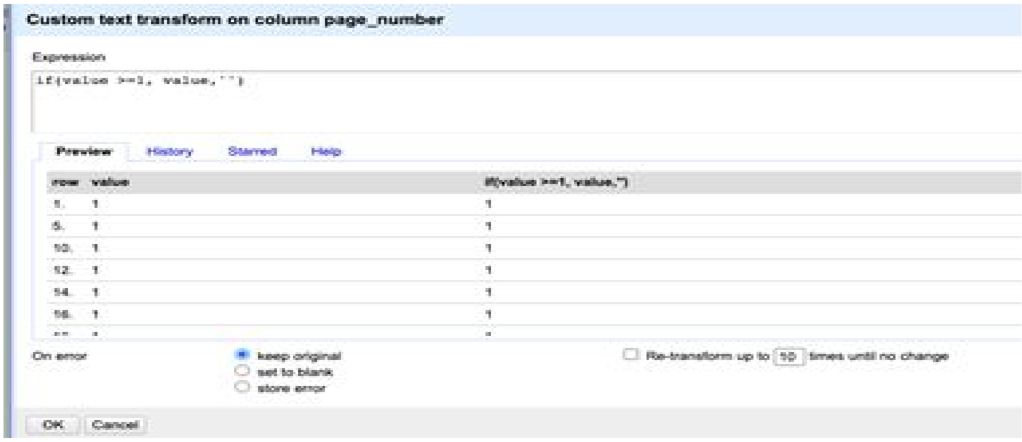
- Convert the columns columns ‘id’, ‘menu_id’,’page_number’, ‘image_id’, ‘full_height’, ‘full_width’ to number format.



Show as: rows records Show: 5 10 25 50 rows

| All | id | menu_id | page_number | image_id | full_height | full_width | uuid |
|---|---|---|---|---|---|---|---|
| 1. | 119 | 12460 | 1 | | | 5428 | 510d47e4-2955-a3d9-e040-e00a18064a99 |
| 2. | 120 | 12460 | 2 | | | 7230 | 510d47e4-2956-a3d9-e040-e00a18064a99 |
| 3. | 121 | 12460 | 3 | | | 5428 | 510d47e4-2957-a3d9-e040-e00a18064a99 |
| 4. | 122 | 12460 | 4 | | | | 4-2958-a3d9-e040-e00a18064a99 |
| 5. | 123 | 12461 | 1 | | | | |
| 6. | 124 | 12461 | 2 | | | | |
| 7. | 125 | 12461 | 3 | | | | |
| 8. | 126 | 12461 | 4 | | | | |
| 9. | 127 | 12462 | 2 | | | | |
| 10. | 128 | 12462 | 1 | | | | |

- Use regular expressions to remove page numbers less than ‘1’.



**Custom text transform on column page_number**

Expression

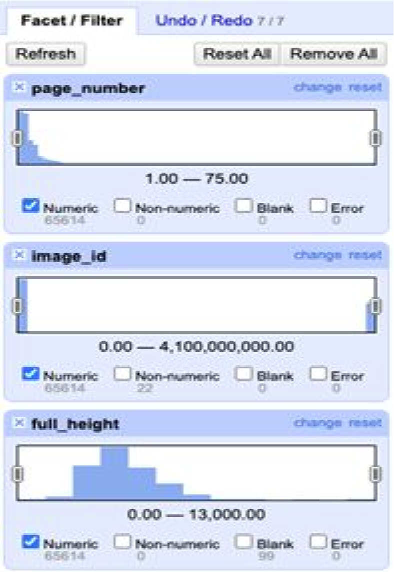
```
if(value >=1, value,'')
```

| row | value | if(value >=1, value,'') |
|---|---|---|
| 1. | 1 | 1 |
| 5. | 1 | 1 |
| 10. | 1 | 1 |
| 12. | 1 | 1 |
| 14. | 1 | 1 |
| 16. | 1 | 1 |

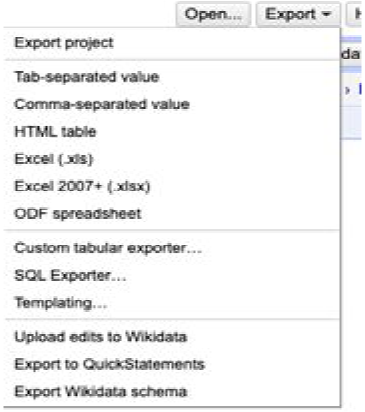
On error ● keep original ○ set to blank ○ store error

□ Re-transform up to 10 times until no change

- Use facet to filter the results.



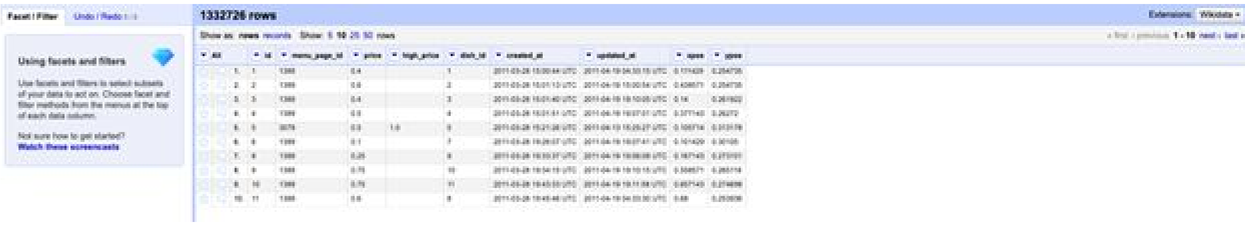
- Export the cleaned data to csv file.



### 7.3 **MenuItem.csv**

The following snapshots will show the sequence of steps for cleaning the MenuItem.csv file.

a. Import the dataset to Open Refine.

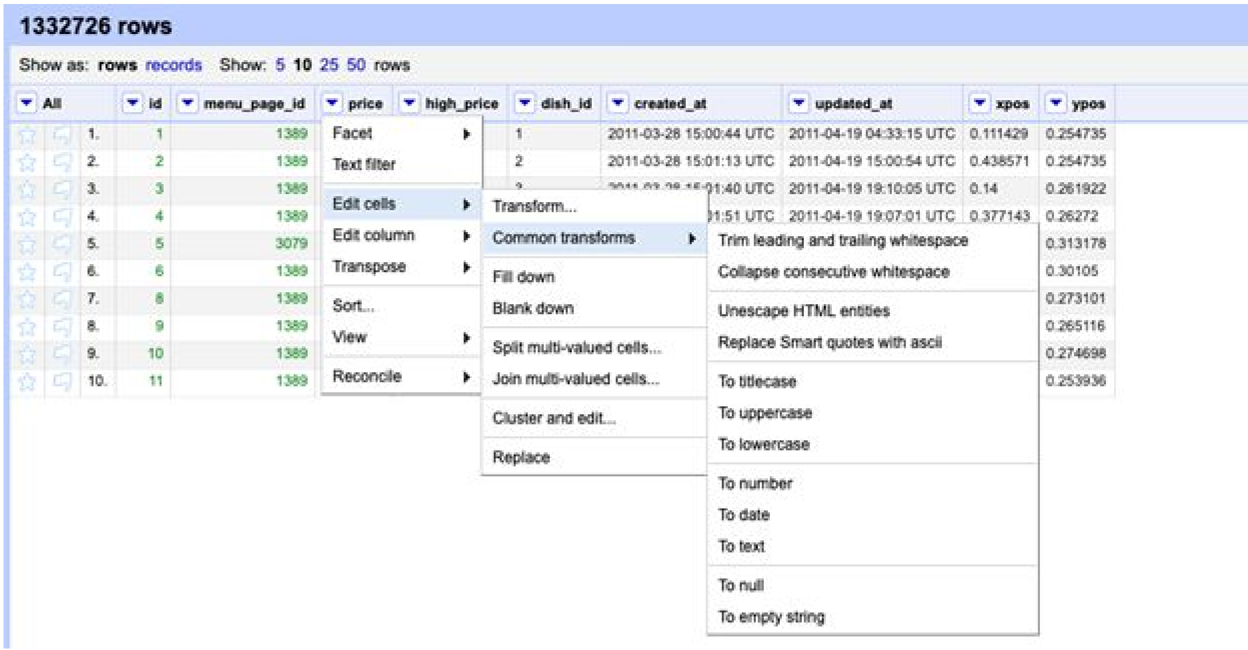

Facet / Filter
Undo / Redo 0 / 0
Using facets and filters
Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.
Not sure how to get started?
Watch these screencasts
Extensions: Wikidata ▾
« first ‹ previous 1 - 10 next › last »

1332726 rows

Show as: rows records Show: 5 10 25 50 rows

| All | id | menu_page_id | price | high_price | dish_id | created_at | updated_at | xpos | ypos |
|---|---|---|---|---|---|---|---|---|---|
| 1. | 1 | 1389 | 0.4 |  | 1 | 2011-03-28 15:00:44 UTC | 2011-04-19 04:33:15 UTC | 0.111429 | 0.254735 |
| 2. | 2 | 1389 | 0.6 |  | 2 | 2011-03-28 15:01:13 UTC | 2011-04-19 15:00:54 UTC | 0.438571 | 0.254735 |
| 3. | 3 | 1389 | 0.4 |  | 3 | 2011-03-28 15:01:40 UTC | 2011-04-19 19:10:05 UTC | 0.14 | 0.261922 |
| 4. | 4 | 1389 | 0.5 |  | 4 | 2011-03-28 15:01:51 UTC | 2011-04-19 19:07:01 UTC | 0.377143 | 0.26272 |
| 5. | 5 | 3079 | 0.5 | 1.0 | 5 | 2011-03-28 15:21:26 UTC | 2011-04-13 15:26:27 UTC | 0.105714 | 0.313178 |
| 6. | 6 | 1389 | 0.1 |  | 7 | 2011-03-28 19:26:07 UTC | 2011-04-19 19:07:41 UTC | 0.101429 | 0.30105 |
| 7. | 8 | 1389 | 0.25 |  | 8 | 2011-03-28 19:33:37 UTC | 2011-04-19 19:08:08 UTC | 0.187143 | 0.273101 |
| 8. | 9 | 1389 | 0.75 |  | 10 | 2011-03-28 19:34:15 UTC | 2011-04-19 19:10:15 UTC | 0.558571 | 0.265114 |
| 9. | 10 | 1389 | 0.75 |  | 11 | 2011-03-28 19:43:33 UTC | 2011-04-19 19:11:58 UTC | 0.657143 | 0.274698 |
| 10. | 11 | 1389 | 0.6 |  | 8 | 2011-03-28 19:45:46 UTC | 2011-04-19 04:33:30 UTC | 0.68 | 0.253938 |

b. Convert the columns ‘id’, ‘menu_page_id’,’price’, ‘high_price’, ‘dish_id’, ‘xpos’ and ‘ypos’ to number format.

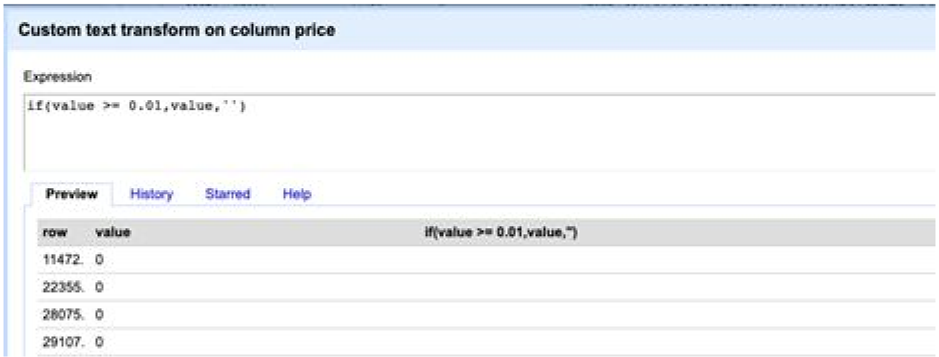


1332726 rows

Show as: rows records Show: 5 10 25 50 rows

| All | | | id | menu_page_id | price | high_price | dish_id | created_at | updated_at | xpos | ypos |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ☆ | | 1. | 1 | 1389 | | | 1 | 2011-03-28 15:00:44 UTC | 2011-04-19 04:33:15 UTC | 0.111429 | 0.254735 |
| ☆ | | 2. | 2 | 1389 | | | 2 | 2011-03-28 15:01:13 UTC | 2011-04-19 15:00:54 UTC | 0.438571 | 0.254735 |
| ☆ | | 3. | 3 | 1389 | | | 3 | 2011-03-28 15:01:40 UTC | 2011-04-19 19:10:05 UTC | 0.14 | 0.261922 |
| ☆ | | 4. | 4 | 1389 | | | | 01:51 UTC | 2011-04-19 19:07:01 UTC | 0.377143 | 0.26272 |
| ☆ | | 5. | 5 | 3079 | | | | | | | 0.313178 |
| ☆ | | 6. | 6 | 1389 | | | | | | | 0.30105 |
| ☆ | | 7. | 8 | 1389 | | | | | | | 0.273101 |
| ☆ | | 8. | 9 | 1389 | | | | | | | 0.265116 |
| ☆ | | 9. | 10 | 1389 | | | | | | | 0.274698 |
| ☆ | | 10. | 11 | 1389 | | | | | | | 0.253936 |

c. We use regular expressions to remove price values from ‘price’ and ‘high_price’ columns less than ‘0.01’.



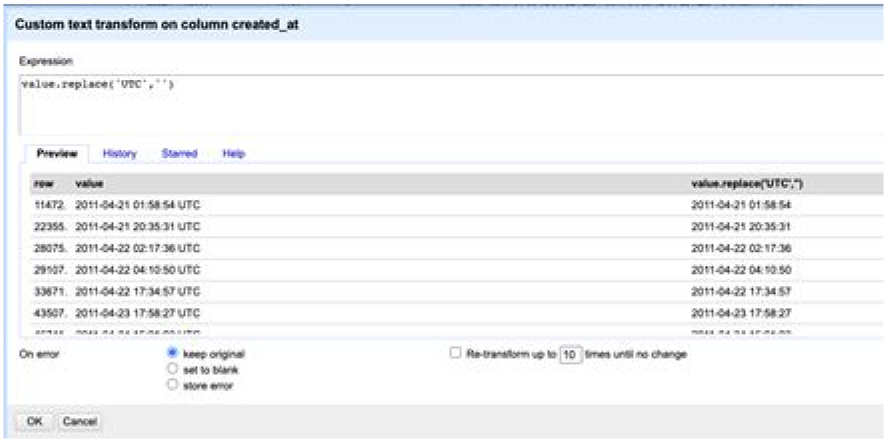
Custom text transform on column price

Expression

```
if(value >= 0.01,value,'')
```

| row | value | if(value >= 0.01,value,'') |
|---|---|---|
| 11472. | 0 | |
| 22355. | 0 | |
| 28075. | 0 | |
| 29107. | 0 | |

d. Replace ‘UTC’ from date values for ‘created_at’ and ‘updated_at’ values to convert them into valid date values.


Custom text transform on column created_at

Expression

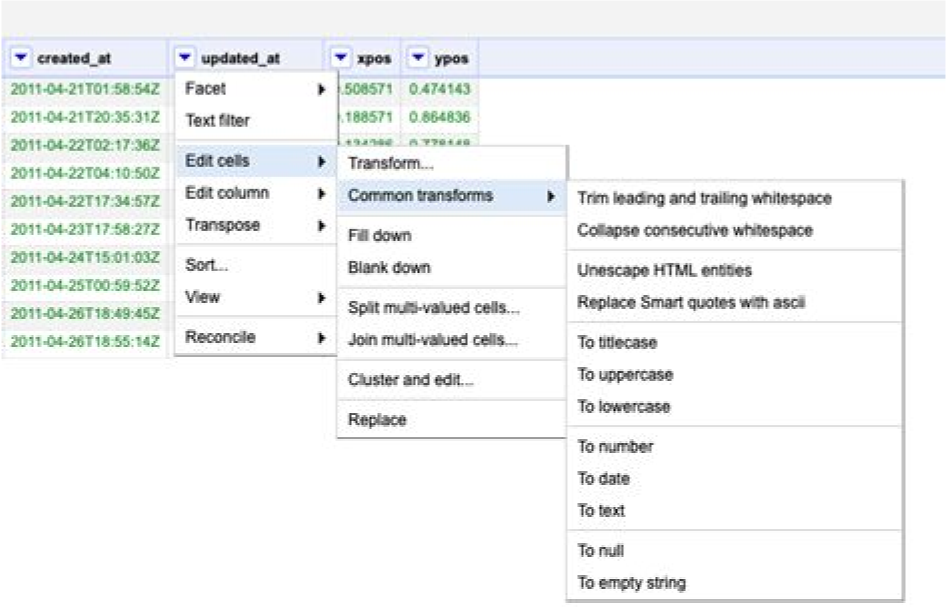
```
value.replace('UTC','')
```

| row | value | value.replace('UTC','') |
|---|---|---|
| 11472. | 2011-04-21 01:58:54 UTC | 2011-04-21 01:58:54 |
| 22355. | 2011-04-21 20:35:31 UTC | 2011-04-21 20:35:31 |
| 28075. | 2011-04-22 02:17:36 UTC | 2011-04-22 02:17:36 |
| 29107. | 2011-04-22 04:10:50 UTC | 2011-04-22 04:10:50 |
| 33671. | 2011-04-22 17:34:57 UTC | 2011-04-22 17:34:57 |
| 43507. | 2011-04-23 17:58:27 UTC | 2011-04-23 17:58:27 |

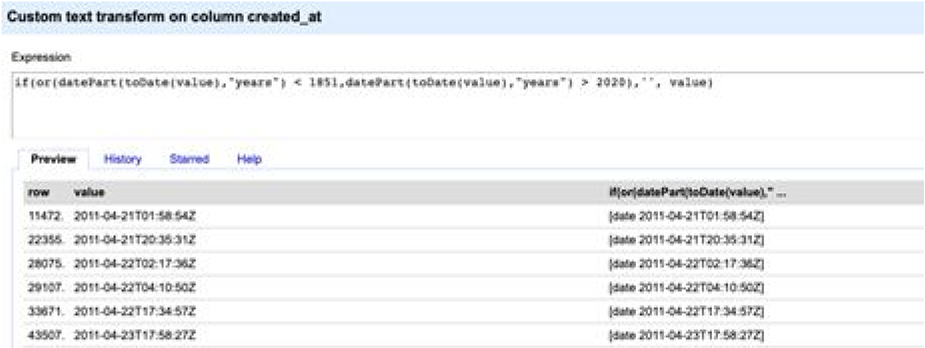
On error ● keep original ○ set to blank ○ store error

□ Re-transform up to 10 times until no change

e. Convert the 'created_at' and 'updated_at' columns to date format.


| created_at | updated_at | xpos | ypos |
|---|---|---|---|
| 2011-04-21T01:58:54Z | | .508571 | 0.474143 |
| 2011-04-21T20:35:31Z | | .188571 | 0.864836 |
| 2011-04-22T02:17:36Z | | | |
| 2011-04-22T04:10:50Z | | | |
| 2011-04-22T17:34:57Z | | | |
| 2011-04-23T17:58:27Z | | | |
| 2011-04-24T15:01:03Z | | | |
| 2011-04-25T00:59:52Z | | | |
| 2011-04-26T18:49:45Z | | | |
| 2011-04-26T18:55:14Z | | | |

f. Filter the range of date values in 'created_at' and 'updated_at' columns for values equal to or more than 1851.

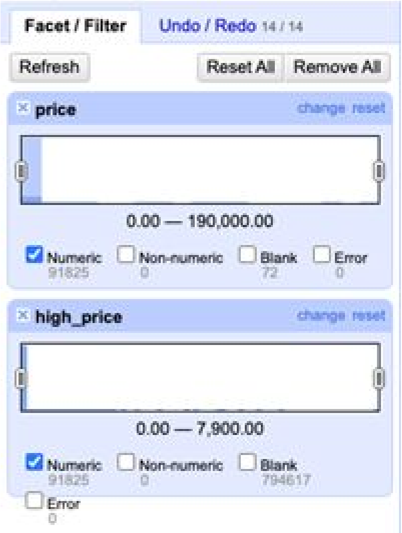

Custom text transform on column created_at

Expression

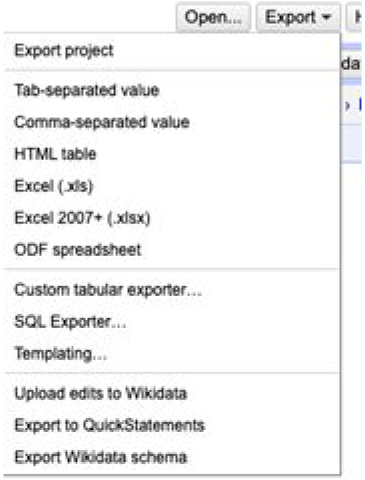
```
if(or(datePart(toDate(value),"years") < 1851,datePart(toDate(value),"years") > 2020),'', value)
```

| row | value | if(or(datePart(toDate(value)," ... |
|---|---|---|
| 11472. | 2011-04-21T01:58:54Z | [date 2011-04-21T01:58:54Z] |
| 22355. | 2011-04-21T20:35:31Z | [date 2011-04-21T20:35:31Z] |
| 28075. | 2011-04-22T02:17:36Z | [date 2011-04-22T02:17:36Z] |
| 29107. | 2011-04-22T04:10:50Z | [date 2011-04-22T04:10:50Z] |
| 33671. | 2011-04-22T17:34:57Z | [date 2011-04-22T17:34:57Z] |
| 43507. | 2011-04-23T17:58:27Z | [date 2011-04-23T17:58:27Z] |

g. We use facets to filter the results.



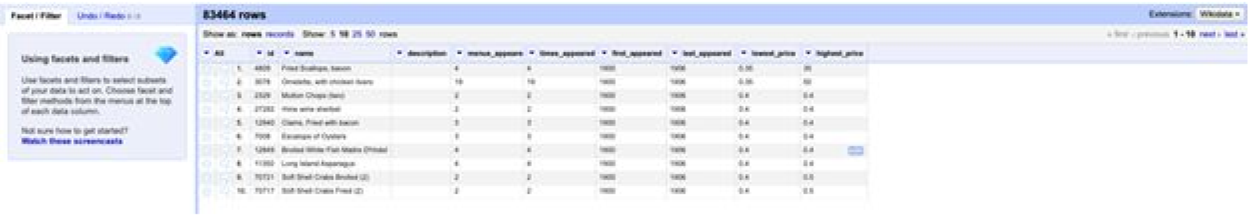
h. Export the cleaned data to a csv file.



7.4 **Dish.csv**

The following snapshots will show the sequence of steps for cleaning the Dish.csv file.

a. Import the dataset to Open Refine.

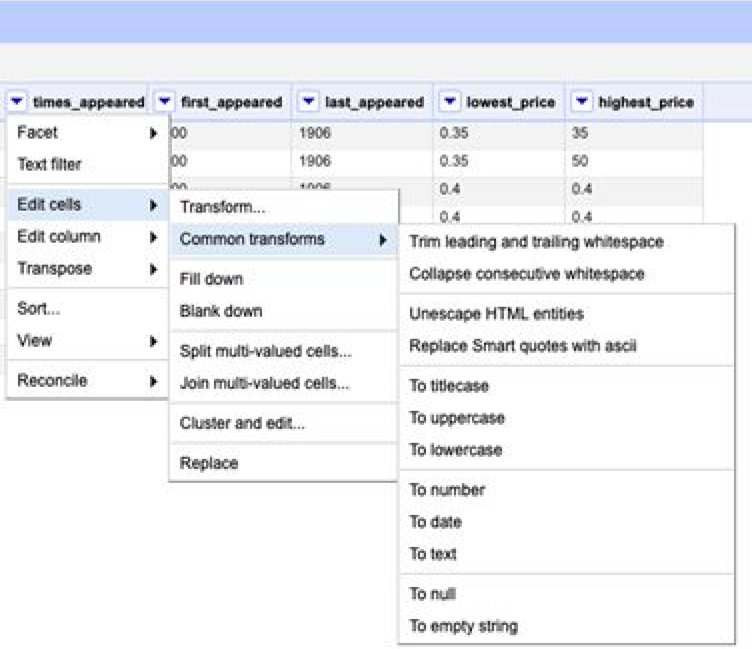

Facet / Filter
Undo / Redo
Using facets and filters
Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.
Not sure how to get started?
Watch these screencasts
Extensions: Wikidata ▾
« first ‹ previous 1 - 10 next › last »

83464 rows

Show as: rows records Show: 5 10 25 50 rows

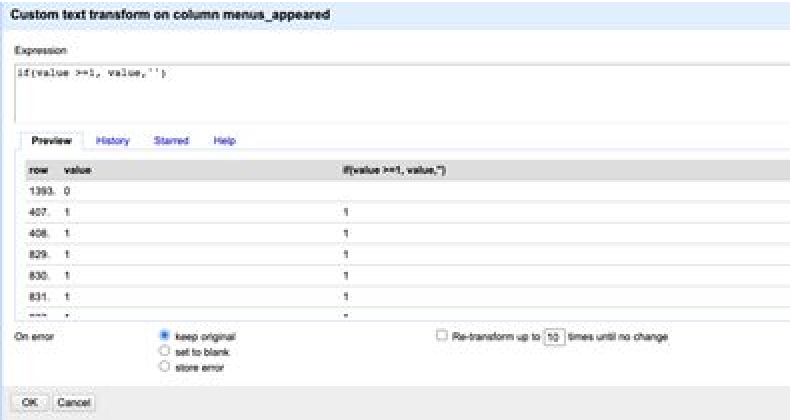
| All | id | name | description | menus_appeared | times_appeared | first_appeared | last_appeared | lowest_price | highest_price |
|---|---|---|---|---|---|---|---|---|---|
| 1. | 4809 | Fried Scallops, bacon | | 4 | 4 | 1900 | 1906 | 0.35 | 35 |
| 2. | 3076 | Omelette, with chicken livers | | 19 | 19 | 1900 | 1906 | 0.35 | 50 |
| 3. | 2529 | Mutton Chops (two) | | 2 | 2 | 1900 | 1906 | 0.4 | 0.4 |
| 4. | 27282 | Hine wine sherbet | | 2 | 2 | 1900 | 1906 | 0.4 | 0.4 |
| 5. | 12640 | Clams, Fried with bacon | | 3 | 3 | 1900 | 1906 | 0.4 | 0.4 |
| 6. | 7008 | Escallops of Oysters | | 3 | 3 | 1900 | 1906 | 0.4 | 0.4 |
| 7. | 12649 | Broiled White Fish Maitre D'Hotel | | 4 | 4 | 1900 | 1906 | 0.4 | 0.4 |
| 8. | 11350 | Long Island Asparagus | | 4 | 4 | 1900 | 1906 | 0.4 | 0.4 |
| 9. | 70721 | Soft Shell Crabs Broiled (2) | | 2 | 2 | 1900 | 1906 | 0.4 | 0.5 |
| 10. | 70717 | Soft Shell Crabs Fried (2) | | 2 | 2 | 1900 | 1906 | 0.4 | 0.5 |

b. Convert the columns ‘id’, ‘menus_appeared’,’times_appeared’, ‘lowest_price’ and ‘highest_price’ to number.



| times_appeared | first_appeared | last_appeared | lowest_price | highest_price |
|---|---|---|---|---|
| | 00 | 1906 | 0.35 | 35 |
| | 00 | 1906 | 0.35 | 50 |
| | | | 0.4 | 0.4 |
| | | | 0.4 | 0.4 |

c. We use regular expressions to remove price values from ‘menu_appeared’ and ‘times_appeared’ columns less than ‘ 1’.



Custom text transform on column menus_appeared

Expression

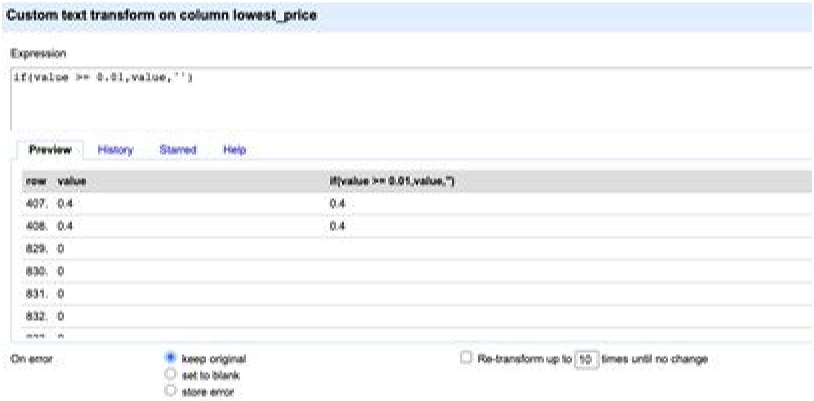
```
if(value >=1, value,'')
```

| row | value | if(value >=1, value,'') |
|---|---|---|
| 1393. | 0 | |
| 407. | 1 | 1 |
| 408. | 1 | 1 |
| 829. | 1 | 1 |
| 830. | 1 | 1 |
| 831. | 1 | 1 |

On error ● keep original ○ set to blank ○ store error

□ Re-transform up to 10 times until no change

d. We use regular expressions to remove price values from ‘lowest_price’ and ‘highest_price’ columns less than ‘0.01’ .



Custom text transform on column lowest_price

Expression

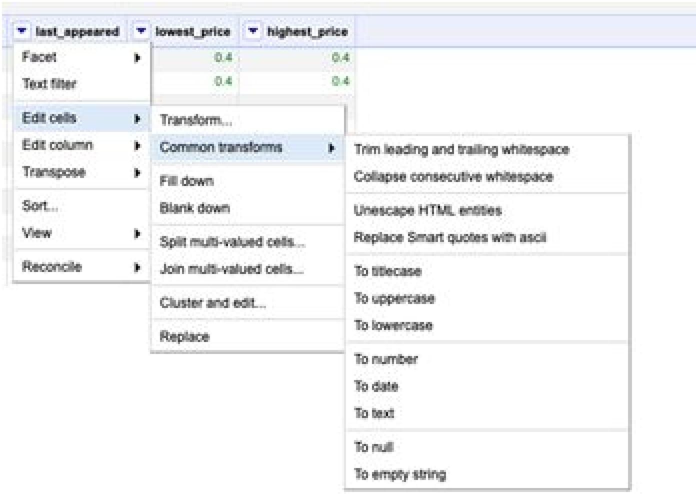
```
if(value >= 0.01,value,'')
```

| row | value | if(value >= 0.01,value,'') |
|---|---|---|
| 407. | 0.4 | 0.4 |
| 408. | 0.4 | 0.4 |
| 829. | 0 | |
| 830. | 0 | |
| 831. | 0 | |
| 832. | 0 | |

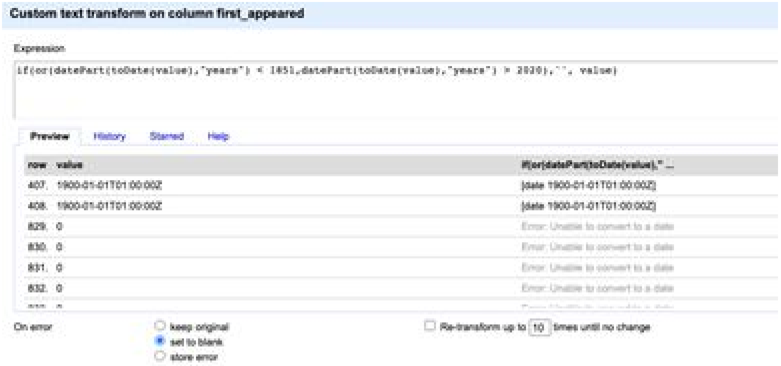
On error ● keep original ○ set to blank ○ store error

□ Re-transform up to 10 times until no change

e. Convert the columns ‘first_appeared’ and ‘last_appeared’ to date format.



f. Filter the range of date values in ‘first_appeared’ and ‘last_appeared’ columns for values equal to or more than 1851. The ‘On error’ value is set to ‘blank’.



Custom text transform on column first_appeared

Expression

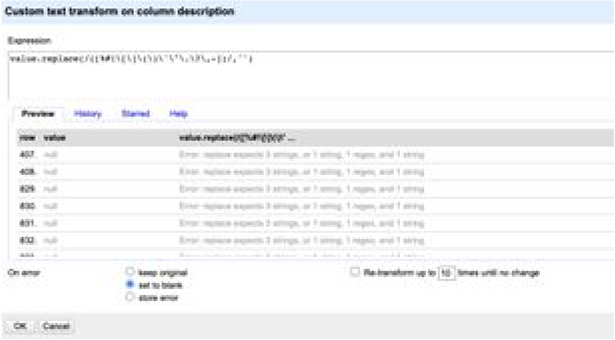
```
if(or(datePart(toDate(value),"years") < 1851,datePart(toDate(value),"years") > 2020),'', value)
```

| row | value | if(or(datePart(toDate(value),"... |
|---|---|---|
| 407. | 1900-01-01T01:00:00Z | [date 1900-01-01T01:00:00Z] |
| 408. | 1900-01-01T01:00:00Z | [date 1900-01-01T01:00:00Z] |
| 829. | 0 | Error: Unable to convert to a date |
| 830. | 0 | Error: Unable to convert to a date |
| 831. | 0 | Error: Unable to convert to a date |
| 832. | 0 | Error: Unable to convert to a date |

On error ○ keep original ● set to blank ○ store error

□ Re-transform up to 10 times until no change

g. Remove punctuations using regex pattern in grel for columns, ‘name’ and ‘description’. Trim spaces, to uppercase. The ‘On error’ value is set to ‘blank’.



Custom text transform on column description

Expression

```
value.replace(/([%#!?[\]\(\)\'\"\`\\,\?%,-])/,'')
```

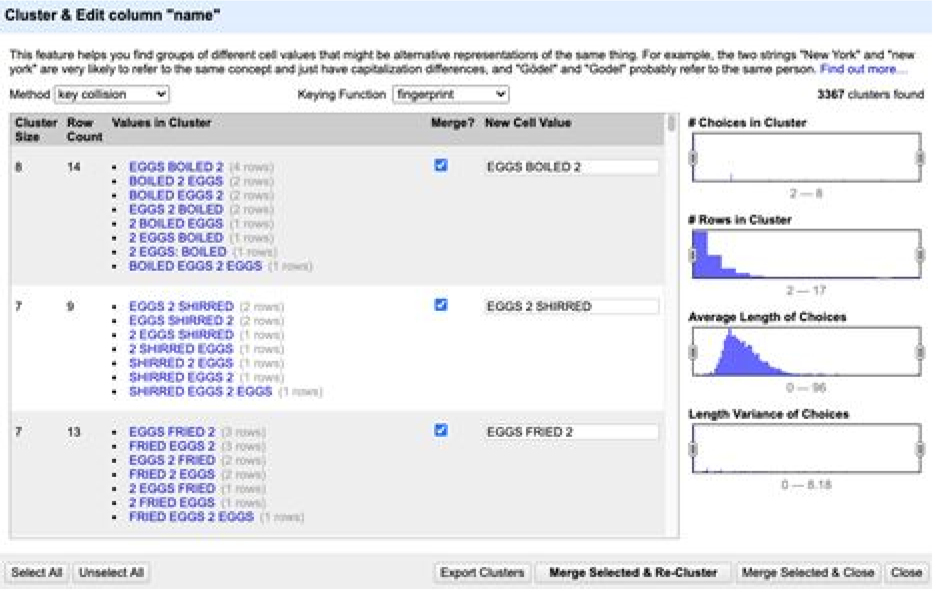
Preview | History | Starred | Help

```
row   value   value.replace(/([%#!?[\]\(\)\' ...
407.  null    Error: replace expects 3 strings, or 1 string, 1 regex, and 1 string
408.  null    Error: replace expects 3 strings, or 1 string, 1 regex, and 1 string
829.  null    Error: replace expects 3 strings, or 1 string, 1 regex, and 1 string
830.  null    Error: replace expects 3 strings, or 1 string, 1 regex, and 1 string
831.  null    Error: replace expects 3 strings, or 1 string, 1 regex, and 1 string
832.  null    Error: replace expects 3 strings, or 1 string, 1 regex, and 1 string
```

On error ○ keep original ● set to blank ○ store error

□ Re-transform up to 10 times until no change

h. Appy cluster and edit for values in column ‘name’. Click on ‘Select All’ and ‘Merge Selected & Re-Cluster’. Perform this operation until no more clusters are found.



Cluster & Edit column "name"

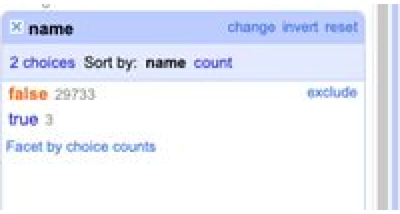
This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

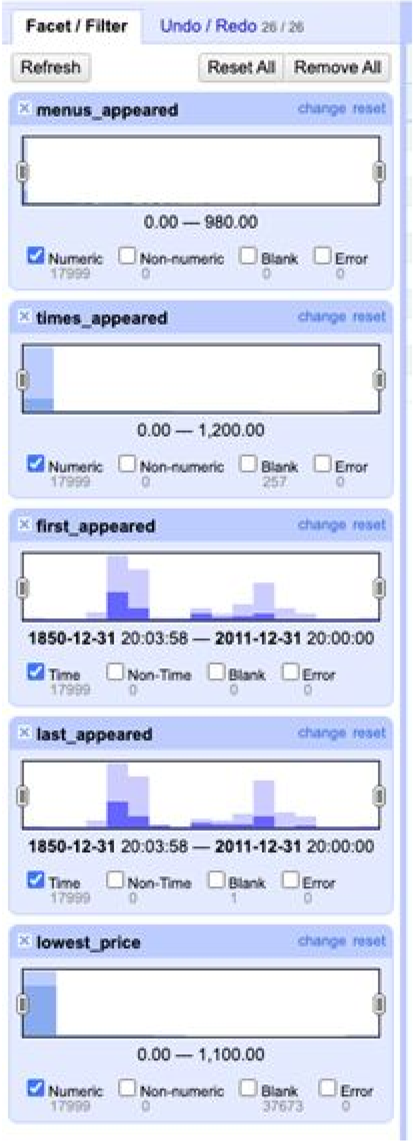
Method: key collision | Keying Function: fingerprint | 3367 clusters found

| Cluster Size | Row Count | Values in Cluster | Merge? | New Cell Value |
|---|---|---|---|---|
| 8 | 14 | EGGS BOILED 2 (4 rows)<br>BOILED 2 EGGS (2 rows)<br>BOILED EGGS 2 (2 rows)<br>EGGS 2 BOILED (2 rows)<br>2 BOILED EGGS (1 rows)<br>2 EGGS BOILED (1 rows)<br>2 EGGS: BOILED (1 rows)<br>BOILED EGGS 2 EGGS (1 rows) | ☑ | EGGS BOILED 2 |
| 7 | 9 | EGGS 2 SHIRRED (2 rows)<br>EGGS SHIRRED 2 (2 rows)<br>2 EGGS SHIRRED (1 rows)<br>2 SHIRRED EGGS (1 rows)<br>SHIRRED 2 EGGS (1 rows)<br>SHIRRED EGGS 2 (1 rows)<br>SHIRRED EGGS 2 EGGS (1 rows) | ☑ | EGGS 2 SHIRRED |
| 7 | 13 | EGGS FRIED 2 (3 rows)<br>FRIED EGGS 2 (3 rows)<br>EGGS 2 FRIED (2 rows)<br>FRIED 2 EGGS (2 rows)<br>2 EGGS FRIED (1 rows)<br>2 FRIED EGGS (1 rows)<br>FRIED EGGS 2 EGGS (1 rows) | ☑ | EGGS FRIED 2 |

i. We use facets to filter the results.

Facet for blank and nulls.





Facet / Filter | Undo / Redo 26 / 26

Refresh | Reset All | Remove All

**menus_appeared** change reset

0.00 — 980.00

☑ Numeric 17999 □ Non-numeric 0 □ Blank 0 □ Error 0

**times_appeared** change reset

0.00 — 1,200.00

☑ Numeric 17999 □ Non-numeric 0 □ Blank 257 □ Error 0

**first_appeared** change reset

1850-12-31 20:03:58 — 2011-12-31 20:00:00

☑ Time 17999 □ Non-Time 0 □ Blank 0 □ Error 0

**last_appeared** change reset

1850-12-31 20:03:58 — 2011-12-31 20:00:00

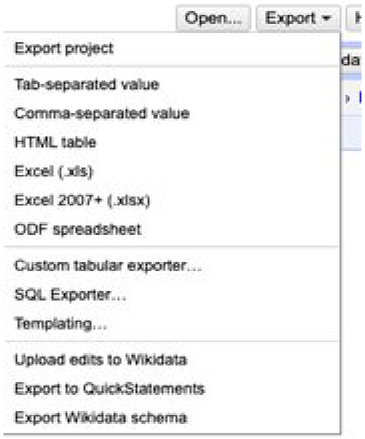
☑ Time 17999 □ Non-Time 0 □ Blank 1 □ Error 0

**lowest_price** change reset

0.00 — 1,100.00

☑ Numeric 17999 □ Non-numeric 0 □ Blank 37673 □ Error 0

j. Export the cleaned dataset to a csv file.



## 8. Results of Data Cleaning

The following shows the results of the data cleaning steps in Tableau Prep.

| Data Set Name | No of Records After Cleaning | |
|---|---|---|
| | Open Refine | Tableau Prep |
| **Menu.csv** | 5960 | 5960 |
| **MenuPage.csv** | 65614 | 65614 |
| **MenuItem.csv** | 91825 | 91825 |
| **Dish.csv** | 163292 | 163292 |

The records match after cleaning by both Open Refine and Tableau Prep.

## 9. Provenance information from OpenRefine

For our provenance queries we took the workflow output from YesWorkflow, and converted it into Datalog queries.

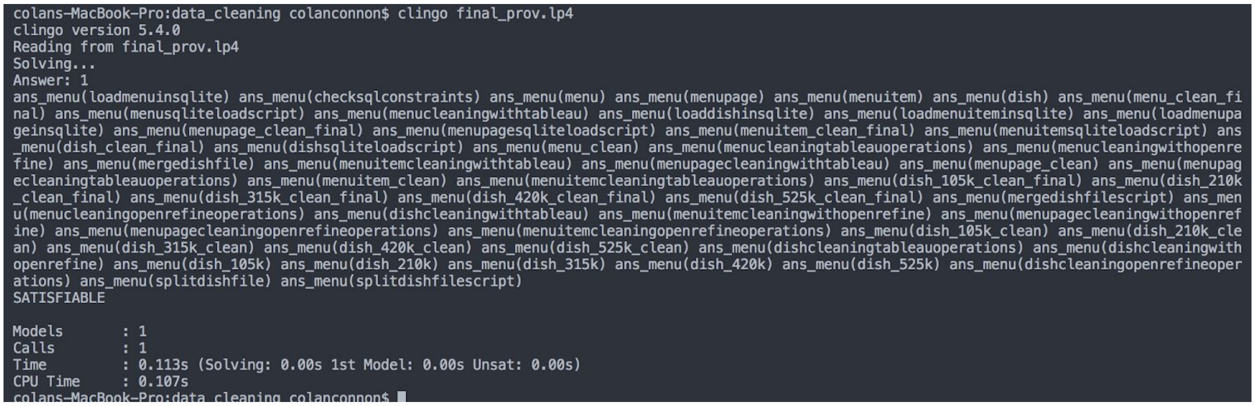
e(menucleaningwithopenrefine , menu_clean).

```
e(menu , menucleaningwithopenrefine).
e(menucleaningopenrefineoperations , menucleaningwithopenrefine).
e(menucleaningwithtableau , menu_clean_final).
e(menu_clean , menucleaningwithtableau).
e(menucleaningtableauoperations , menucleaningwithtableau).
e(menupagecleaningwithopenrefine , menupage_clean).
e(menupage , menupagecleaningwithopenrefine).
e(menupagecleaningopenrefineoperations , menupagecleaningwithopenrefine).
e(menupagecleaningwithtableau , menupage_clean_final).
e(menupage_clean , menupagecleaningwithtableau).
e(menupagecleaningtableauoperations , menupagecleaningwithtableau).
e(menuitemcleaningwithopenrefine , menuitem_clean).
e(menuitem , menuitemcleaningwithopenrefine).
e(menuitemcleaningopenrefineoperations , menuitemcleaningwithopenrefine).
e(menuitemcleaningwithtableau , menuitem_clean_final).
e(menuitem_clean , menuitemcleaningwithtableau).
e(menuitemcleaningtableauoperations , menuitemcleaningwithtableau).
e(splitdishfile , dish_105k).
e(splitdishfile , dish_210k).
e(splitdishfile , dish_315k).
e(splitdishfile , dish_420k).
e(splitdishfile , dish_525k).
e(dish , splitdishfile).
e(splitdishfilescript , splitdishfile).
e(dishcleaningwithopenrefine , dish_105k_clean).
e(dishcleaningwithopenrefine , dish_210k_clean).
e(dishcleaningwithopenrefine , dish_315k_clean).
e(dishcleaningwithopenrefine , dish_420k_clean).
e(dishcleaningwithopenrefine , dish_525k_clean).
e(dish_105k , dishcleaningwithopenrefine).
e(dish_210k , dishcleaningwithopenrefine).
e(dish_315k , dishcleaningwithopenrefine).
e(dish_420k , dishcleaningwithopenrefine).
e(dish_525k , dishcleaningwithopenrefine).
e(dishcleaningopenrefineoperations , dishcleaningwithopenrefine).
e(dishcleaningwithtableau , dish_105k_clean_final).
e(dishcleaningwithtableau , dish_210k_clean_final).
e(dishcleaningwithtableau , dish_315k_clean_final).
e(dishcleaningwithtableau , dish_420k_clean_final).
e(dishcleaningwithtableau , dish_525k_clean_final).
e(dish_105k_clean , dishcleaningwithtableau).
e(dish_210k_clean , dishcleaningwithtableau).
e(dish_315k_clean , dishcleaningwithtableau).
e(dish_420k_clean , dishcleaningwithtableau).
e(dish_525k_clean , dishcleaningwithtableau).
```

```
e(dishcleaningtableauoperations , dishcleaningwithtableau).
e(mergedishfile , dish_clean_final).
e(dish_105k_clean_final , mergedishfile).
e(dish_210k_clean_final , mergedishfile).
e(dish_315k_clean_final , mergedishfile).
e(dish_420k_clean_final , mergedishfile).
e(dish_525k_clean_final , mergedishfile).
e(mergedishfilescript , mergedishfile).
e(loadmenuinsqlite , menu).
e(menu_clean_final , loadmenuinsqlite).
e(menusqliteloadscript , loadmenuinsqlite).
e(loadmenupageinsqlite , menupage).
e(menupage_clean_final , loadmenupageinsqlite).
e(menupagesqliteloadscript , loadmenupageinsqlite).
e(loadmenuiteminsqlite , menuitem).
e(menuitem_clean_final , loadmenuiteminsqlite).
e(menuitemsqliteloadscript , loadmenuiteminsqlite).
e(loaddishinsqlite , dish).
e(dish_clean_final , loaddishinsqlite).
e(dishsqliteloadscript , loaddishinsqlite).
e(checksqlconstraints , menu).
e(checksqlconstraints , menupage).
e(checksqlconstraints , menuitem).
e(checksqlconstraints , dish).
e(menu , checksqlconstraints).
e(menupage , checksqlconstraints).
e(menuitem , checksqlconstraints).
e(dish , checksqlconstraints).
```

We ran three different queries. First we wanted to check the ancestors of the menu file.

```
tc(X, Y) :- e(X, Y).
tc(X, Y) :- e(X, Z), tc(Z, Y).
ans_menu(X) :- tc(X, menu).
```

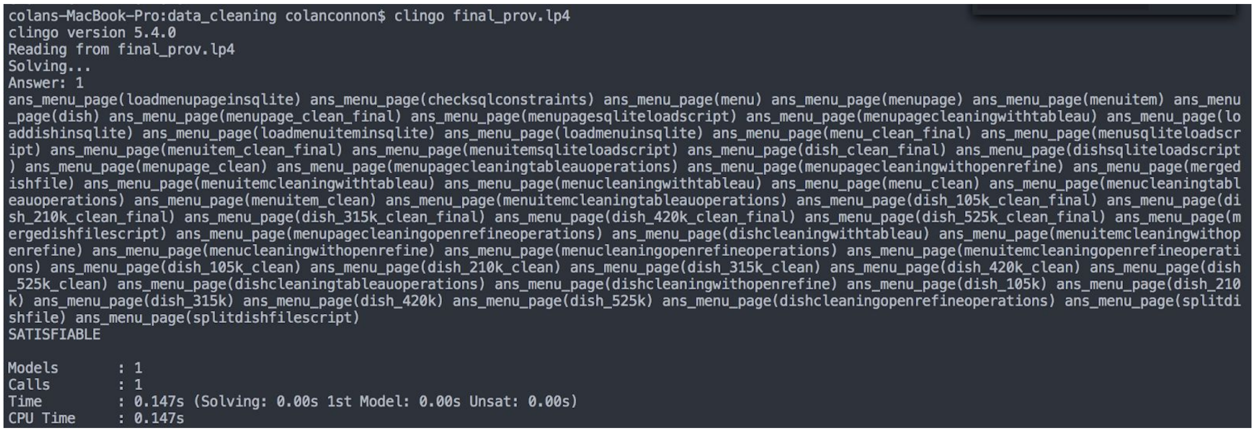


```
colans-MacBook-Pro:data_cleaning colanconnon$ clingo final_prov.lp4
clingo version 5.4.0
Reading from final_prov.lp4
Solving...
Answer: 1
ans_menu(loadmenuinsqlite) ans_menu(checksqlconstraints) ans_menu(menu) ans_menu(menupage) ans_menu(menuitem) ans_menu(dish) ans_menu(menu_clean_fi
nal) ans_menu(menusqliteloadscript) ans_menu(menucleaningwithtableau) ans_menu(loaddishinsqlite) ans_menu(loadmenuiteminsqlite) ans_menu(loadmenupa
geinsqlite) ans_menu(menupage_clean_final) ans_menu(menupagesqliteloadscript) ans_menu(menuitem_clean_final) ans_menu(menuitemsqliteloadscript) ans
_menu(dish_clean_final) ans_menu(dishsqliteloadscript) ans_menu(menu_clean) ans_menu(menucleaningtableauoperations) ans_menu(menucleaningwithopenre
fine) ans_menu(mergedishfile) ans_menu(menuitemcleaningwithtableau) ans_menu(menupagecleaningwithtableau) ans_menu(menupage_clean) ans_menu(menupag
ecleaningtableauoperations) ans_menu(menuitem_clean) ans_menu(menuitemcleaningtableauoperations) ans_menu(dish_105k_clean_final) ans_menu(dish_210k
_clean_final) ans_menu(dish_315k_clean_final) ans_menu(dish_420k_clean_final) ans_menu(dish_525k_clean_final) ans_menu(mergedishfilescript) ans_men
u(menucleaningopenrefineoperations) ans_menu(dishcleaningwithtableau) ans_menu(menuitemcleaningwithopenrefine) ans_menu(menupagecleaningwithopenref
ine) ans_menu(menupagecleaningopenrefineoperations) ans_menu(menuitemcleaningopenrefineoperations) ans_menu(dish_105k_clean) ans_menu(dish_210k_cle
an) ans_menu(dish_315k_clean) ans_menu(dish_420k_clean) ans_menu(dish_525k_clean) ans_menu(dishcleaningtableauoperations) ans_menu(dishcleaningwith
openrefine) ans_menu(dish_105k) ans_menu(dish_210k) ans_menu(dish_315k) ans_menu(dish_420k) ans_menu(dish_525k) ans_menu(dishcleaningopenrefineoper
ations) ans_menu(splitdishfile) ans_menu(splitdishfilescript)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.113s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.107s
colans-MacBook-Pro:data_cleaning colanconnon$ 
```

Next we checked the ancestors of menupage

```
tc(X, Y) :- e(X, Y).
tc(X, Y) :- e(X, Z), tc(Z, Y).
ans_menu_page(X) :- tc(X, menupage).
```



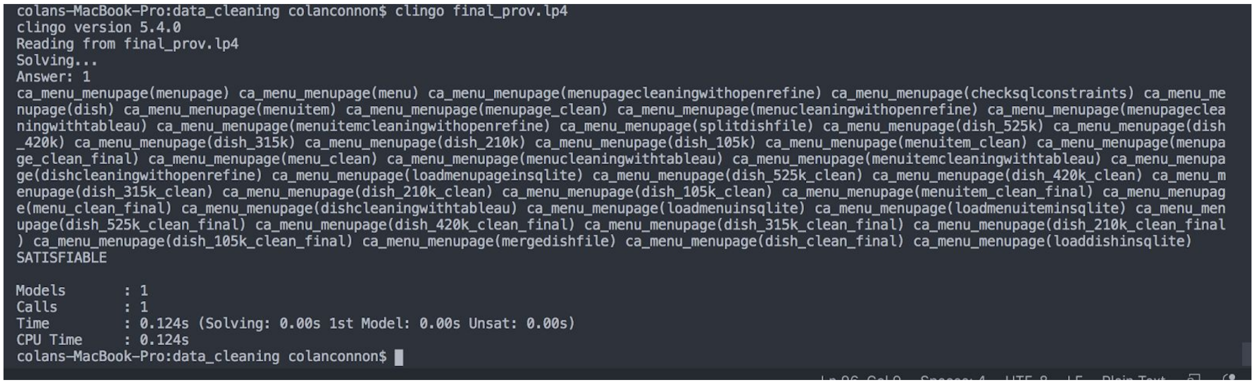
```
colans-MacBook-Pro:data_cleaning colanconnon$ clingo final_prov.lp4
clingo version 5.4.0
Reading from final_prov.lp4
Solving...
Answer: 1
ans_menu_page(loadmenupageinsqlite) ans_menu_page(checksqlconstraints) ans_menu_page(menu) ans_menu_page(menupage) ans_menu_page(menuitem) ans_menu
_page(dish) ans_menu_page(menupage_clean_final) ans_menu_page(menupagesqliteloadscript) ans_menu_page(menupagecleaningwithtableau) ans_menu_page(lo
addishinsqlite) ans_menu_page(loadmenuiteminsqlite) ans_menu_page(loadmenuinsqlite) ans_menu_page(menu_clean_final) ans_menu_page(menusqliteloadscr
ipt) ans_menu_page(menuitem_clean_final) ans_menu_page(menuitemsqliteloadscript) ans_menu_page(dish_clean_final) ans_menu_page(dishsqliteloadscript
) ans_menu_page(menupage_clean) ans_menu_page(menupagecleaningtableauoperations) ans_menu_page(menupagecleaningwithopenrefine) ans_menu_page(merged
ishfile) ans_menu_page(menuitemcleaningwithtableau) ans_menu_page(menucleaningwithtableau) ans_menu_page(menu_clean) ans_menu_page(menucleaningtabl
eauoperations) ans_menu_page(menuitem_clean) ans_menu_page(menuitemcleaningtableauoperations) ans_menu_page(dish_105k_clean_final) ans_menu_page(di
sh_210k_clean_final) ans_menu_page(dish_315k_clean_final) ans_menu_page(dish_420k_clean_final) ans_menu_page(dish_525k_clean_final) ans_menu_page(m
ergedishfilescript) ans_menu_page(menupagecleaningopenrefineoperations) ans_menu_page(dishcleaningwithtableau) ans_menu_page(menuitemcleaningwithop
enrefine) ans_menu_page(menucleaningwithopenrefine) ans_menu_page(menucleaningopenrefineoperations) ans_menu_page(menuitemcleaningopenrefineoperati
ons) ans_menu_page(dish_105k_clean) ans_menu_page(dish_210k_clean) ans_menu_page(dish_315k_clean) ans_menu_page(dish_420k_clean) ans_menu_page(dish
_525k_clean) ans_menu_page(dishcleaningtableauoperations) ans_menu_page(dishcleaningwithopenrefine) ans_menu_page(dish_105k) ans_menu_page(dish_210
k) ans_menu_page(dish_315k) ans_menu_page(dish_420k) ans_menu_page(dish_525k) ans_menu_page(dishcleaningopenrefineoperations) ans_menu_page(splitdi
shfile) ans_menu_page(splitdishfilescript)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.147s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.147s
```

As we can see they share mostly the same ancestors as the start and end in similar spots in the workflow. We can verify the steps they share by running a common ancestors query.

```
ca(X, A, A) :- tc(X, A).
ca(A, X, A) :- tc(X, A).
ca(X,Y,A) :- tc(X, A), tc(Y, A), X != Y.

ca_menu_menupage(X) :- ca(menu, menupage, X).
```

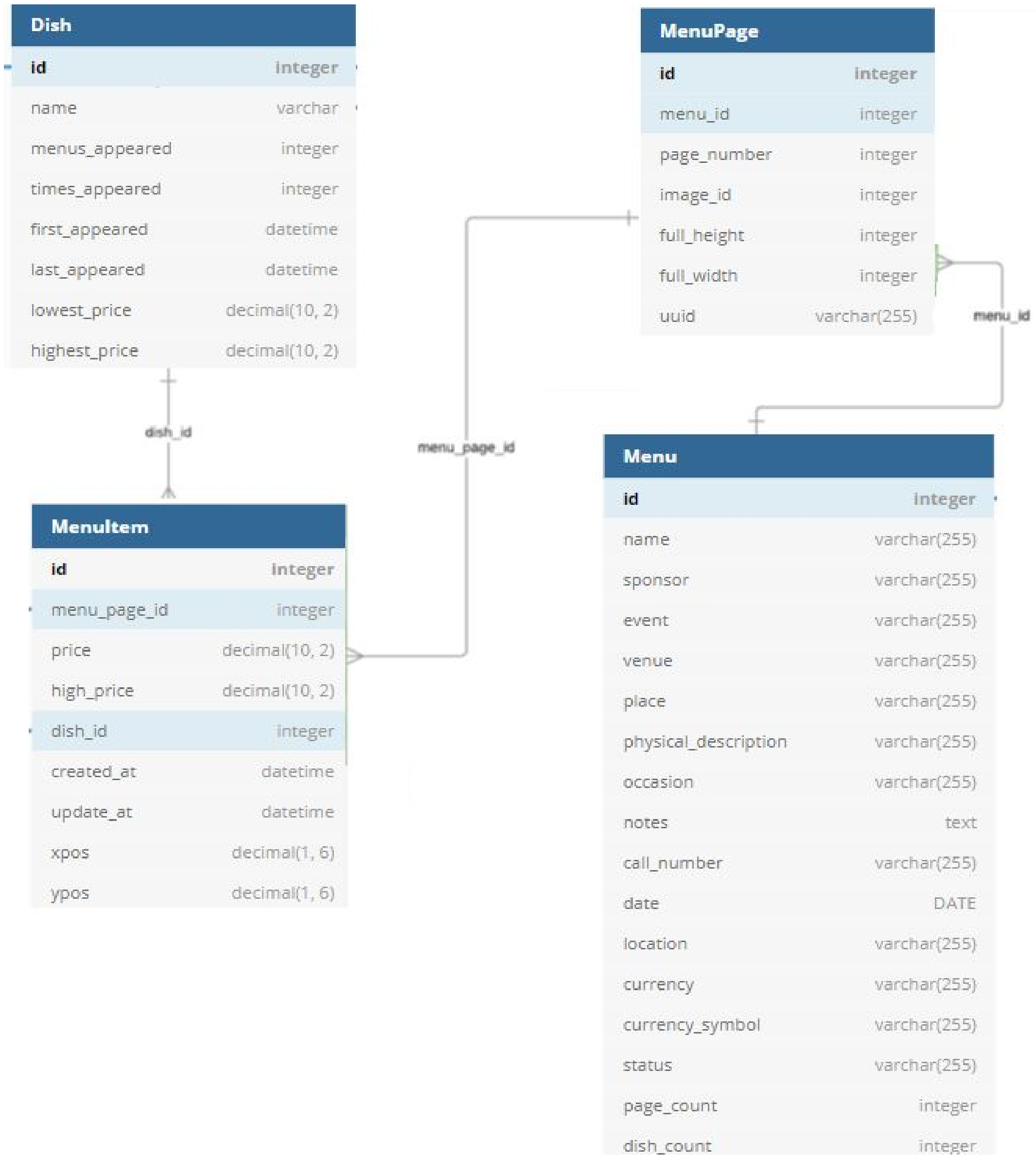


```
colans-MacBook-Pro:data_cleaning colanconnon$ clingo final_prov.lp4
clingo version 5.4.0
Reading from final_prov.lp4
Solving...
Answer: 1
ca_menu_menupage(menupage) ca_menu_menupage(menu) ca_menu_menupage(menupagecleaningwithopenrefine) ca_menu_menupage(checksqlconstraints) ca_menu_me
nupage(dish) ca_menu_menupage(menuitem) ca_menu_menupage(menupage_clean) ca_menu_menupage(menucleaningwithopenrefine) ca_menu_menupage(menupageclea
ningwithtableau) ca_menu_menupage(menuitemcleaningwithopenrefine) ca_menu_menupage(splitdishfile) ca_menu_menupage(dish_525k) ca_menu_menupage(dish
_420k) ca_menu_menupage(dish_315k) ca_menu_menupage(dish_210k) ca_menu_menupage(dish_105k) ca_menu_menupage(menuitem_clean) ca_menu_menupage(menupa
ge_clean_final) ca_menu_menupage(menu_clean) ca_menu_menupage(menucleaningwithtableau) ca_menu_menupage(menuitemcleaningwithtableau) ca_menu_menupa
ge(dishcleaningwithopenrefine) ca_menu_menupage(loadmenupageinsqlite) ca_menu_menupage(dish_525k_clean) ca_menu_menupage(dish_420k_clean) ca_menu_m
enupage(dish_315k_clean) ca_menu_menupage(dish_210k_clean) ca_menu_menupage(dish_105k_clean) ca_menu_menupage(menuitem_clean_final) ca_menu_menupag
e(menu_clean_final) ca_menu_menupage(dishcleaningwithtableau) ca_menu_menupage(loadmenuinsqlite) ca_menu_menupage(loadmenuiteminsqlite) ca_menu_men
upage(dish_525k_clean_final) ca_menu_menupage(dish_420k_clean_final) ca_menu_menupage(dish_315k_clean_final) ca_menu_menupage(dish_210k_clean_final
) ca_menu_menupage(dish_105k_clean_final) ca_menu_menupage(mergedishfile) ca_menu_menupage(dish_clean_final) ca_menu_menupage(loaddishinsqlite)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.124s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.124s
colans-MacBook-Pro:data_cleaning colanconnon$ 
```

## 10.Relational Schema

10.1 Below is the schema in which there are four tables (one for each input file): Dish, Menuitem, Menupage, and Menu.



| Dish | |
|---|---|
| id | integer |
| name | varchar |
| menus_appeared | integer |
| times_appeared | integer |
| first_appeared | datetime |
| last_appeared | datetime |
| lowest_price | decimal(10, 2) |
| highest_price | decimal(10, 2) |

| MenuPage | |
|---|---|
| id | integer |
| menu_id | integer |
| page_number | integer |
| image_id | integer |
| full_height | integer |
| full_width | integer |
| uuid | varchar(255) |

| MenuItem | |
|---|---|
| id | integer |
| menu_page_id | integer |
| price | decimal(10, 2) |
| high_price | decimal(10, 2) |
| dish_id | integer |
| created_at | datetime |
| update_at | datetime |
| xpos | decimal(1, 6) |
| ypos | decimal(1, 6) |

| Menu | |
|---|---|
| id | integer |
| name | varchar(255) |
| sponsor | varchar(255) |
| event | varchar(255) |
| venue | varchar(255) |
| place | varchar(255) |
| physical_description | varchar(255) |
| occasion | varchar(255) |
| notes | text |
| call_number | varchar(255) |
| date | DATE |
| location | varchar(255) |
| currency | varchar(255) |
| currency_symbol | varchar(255) |
| status | varchar(255) |
| page_count | integer |
| dish_count | integer |

10.2 The following are the sql scripts that have been used to implement the above schema:

### Dish.csv

```
create table Dish(
name varchar(255) null,
id integer PRIMARY KEY asc not null,
menus_appeared integer not null,
times_appeared integer not null,
first_appeared datetime not null,
last_appeared datetime not null,
lowest_price decimal(10, 2) null,
highest_price decimal(10, 2) null);
```

### Menu.csv

```
create table Menu(
id integer primary key asc not null,
name varchar(255) null,
sponsor varchar(255) null,
event varchar(255) null,
venue varchar(255) null,
place varchar(255) null,
physical_description varchar(255) null,
occasion varchar(255) null,
notes text null,
call_number varchar(255) null,
date DATE,
location varchar(255) null,
currency varchar(255) null,
currency_symbol varchar(255) null,
status varchar(255) null,
page_count integer,
dish_count integer);
```

### MenuPage.csv

```
create table MenuPage(
id integer primary key asc not null,
menu_id integer not null,
page_number integer,
image_id integer,
full_height integer,
full_width integer,
uuid varchar(255),
FOREIGN KEY(menu_id) REFERENCES  Menu(id));
```

### MenuItem.csv

```
create table MenuItem(
id integer primary key asc not null,
menu_page_id integer not null,
price decimal(10, 2) null,
high_price decimal(10, 2) null,
dish_id integer not null,
created_at datetime not null,
update_at datetime not null,
xpos decimal(1, 6),
ypos decimal(1, 6),
FOREIGN KEY(menu_page_id) REFERENCES  MenuPage(id),
FOREIGN KEY(dish_id) REFERENCES  Dish(id));
```

## 11.Integrity Constraints

The following are the integrity constraints for the above tables that we have created:

**Dish (table)**

- Id should not be Null
- Menus_appeared or times_appeared cannot be NULL.

- First_appeared and last_appeared also cannot be null.
- Lowest price of the dish should be less than the highest price

**Menu (table)**

- Id cannot be Null
- Date cannot be null and range of date has to be between 1850 – 2020.

**MenuPage (table)**

- Id cannot be Null
- Menu_Id cannot be Null
- enforce that menu_id must be in menu table

**MenuItem (table)**

- Id cannot be Null
- Menu_page_id cannot be Null
- Dish_id cannot be null
- enforce that dish_id must occur in dish table
- enforce that menu_page_id must occur in menu_page table

## 12. Loading data into database

We loaded the data via sqlite command line interface.

```
.mode csv
.import ./NYPL-Menus/Dish_Clean_Final.csv Dish
.import ./NYPL-Menus/Menu_Clean_Final.csv Menu
.import ./NYPL-Menus/MenuPage_Clean_Final.csv MenuPage
.import ./NYPL-Menus/MenuItem_Clean_Final.csv MenuItem
```

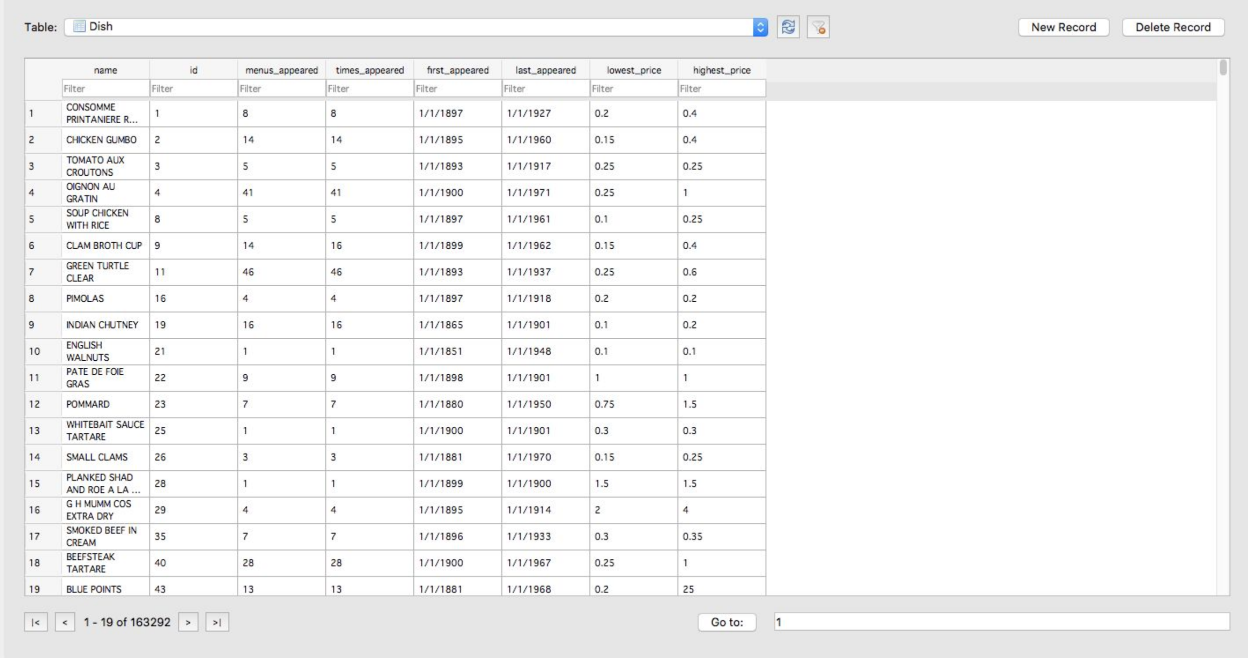


Table: Dish

| | name | id | menus_appeared | times_appeared | first_appeared | last_appeared | lowest_price | highest_price |
|---|---|---|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter |
| 1 | CONSOMME PRINTANIERE R... | 1 | 8 | 8 | 1/1/1897 | 1/1/1927 | 0.2 | 0.4 |
| 2 | CHICKEN GUMBO | 2 | 14 | 14 | 1/1/1895 | 1/1/1960 | 0.15 | 0.4 |
| 3 | TOMATO AUX CROUTONS | 3 | 5 | 5 | 1/1/1893 | 1/1/1917 | 0.25 | 0.25 |
| 4 | OIGNON AU GRATIN | 4 | 41 | 41 | 1/1/1900 | 1/1/1971 | 0.25 | 1 |
| 5 | SOUP CHICKEN WITH RICE | 8 | 5 | 5 | 1/1/1897 | 1/1/1961 | 0.1 | 0.25 |
| 6 | CLAM BROTH CUP | 9 | 14 | 16 | 1/1/1899 | 1/1/1962 | 0.15 | 0.4 |
| 7 | GREEN TURTLE CLEAR | 11 | 46 | 46 | 1/1/1893 | 1/1/1937 | 0.25 | 0.6 |
| 8 | PIMOLAS | 16 | 4 | 4 | 1/1/1897 | 1/1/1918 | 0.2 | 0.2 |
| 9 | INDIAN CHUTNEY | 19 | 16 | 16 | 1/1/1865 | 1/1/1901 | 0.1 | 0.2 |
| 10 | ENGLISH WALNUTS | 21 | 1 | 1 | 1/1/1851 | 1/1/1948 | 0.1 | 0.1 |
| 11 | PATE DE FOIE GRAS | 22 | 9 | 9 | 1/1/1898 | 1/1/1901 | 1 | 1 |
| 12 | POMMARD | 23 | 7 | 7 | 1/1/1880 | 1/1/1950 | 0.75 | 1.5 |
| 13 | WHITEBAIT SAUCE TARTARE | 25 | 1 | 1 | 1/1/1900 | 1/1/1901 | 0.3 | 0.3 |
| 14 | SMALL CLAMS | 26 | 3 | 3 | 1/1/1881 | 1/1/1970 | 0.15 | 0.25 |
| 15 | PLANKED SHAD AND ROE A LA ... | 28 | 1 | 1 | 1/1/1899 | 1/1/1900 | 1.5 | 1.5 |
| 16 | G H MUMM COS EXTRA DRY | 29 | 4 | 4 | 1/1/1895 | 1/1/1914 | 2 | 4 |
| 17 | SMOKED BEEF IN CREAM | 35 | 7 | 7 | 1/1/1896 | 1/1/1933 | 0.3 | 0.35 |
| 18 | BEEFSTEAK TARTARE | 40 | 28 | 28 | 1/1/1900 | 1/1/1967 | 0.25 | 1 |
| 19 | BLUE POINTS | 43 | 13 | 13 | 1/1/1881 | 1/1/1968 | 0.2 | 25 |

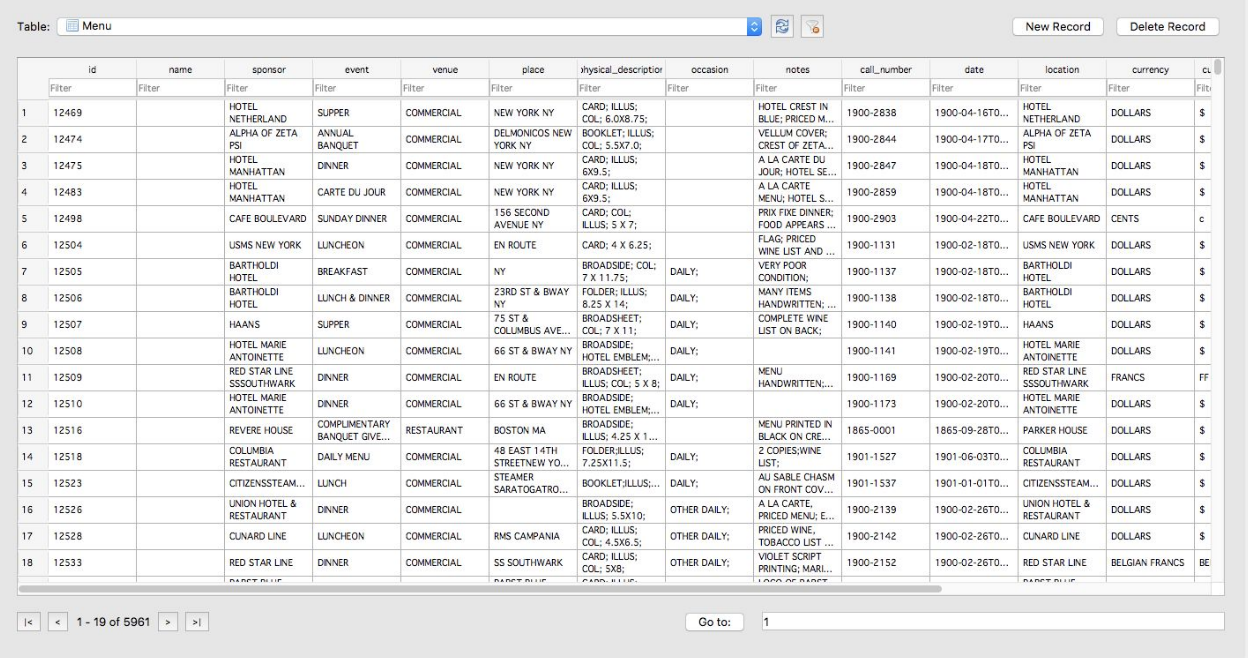
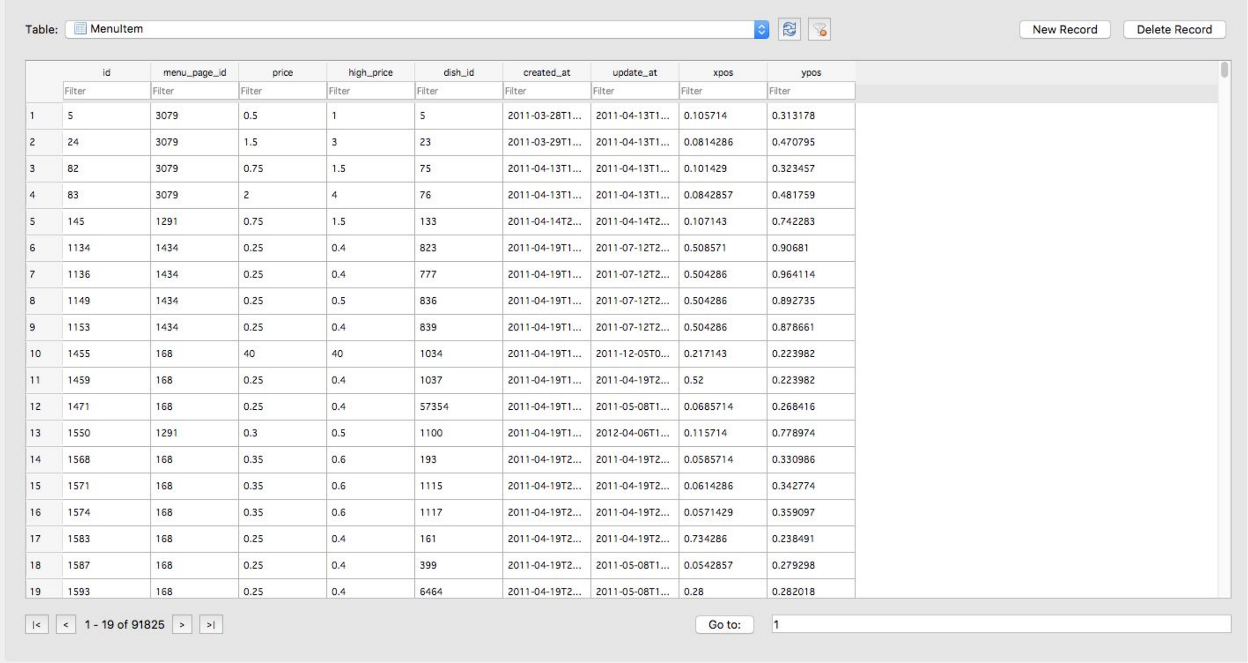
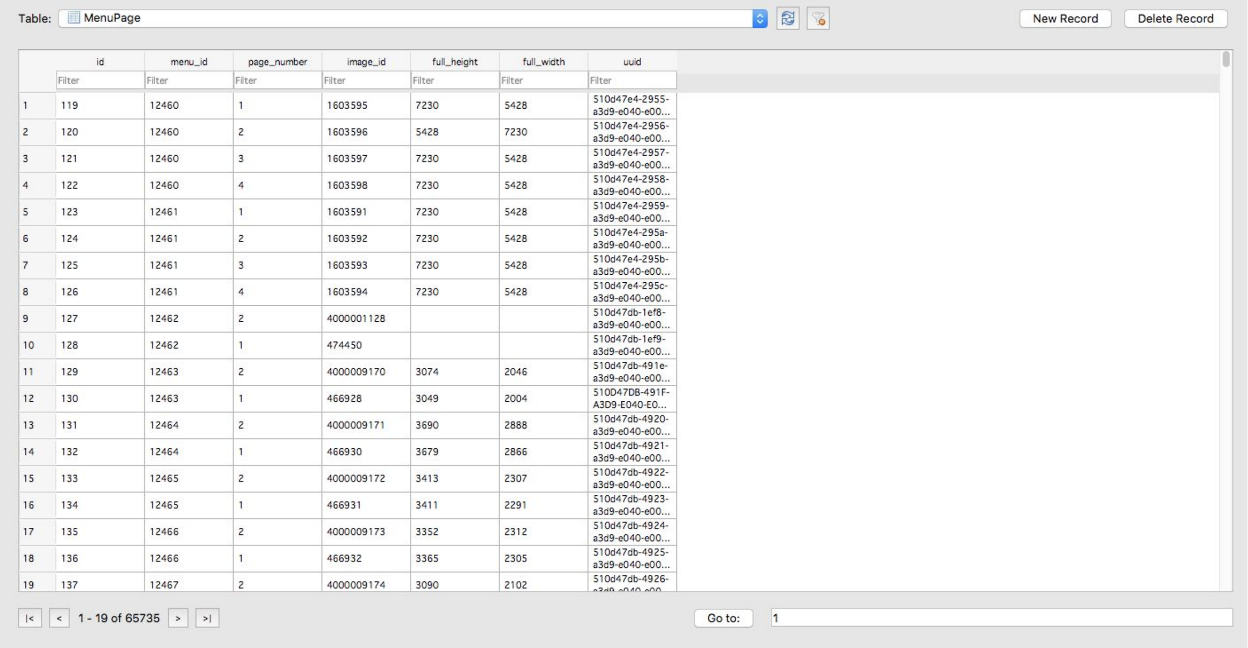


Table: Menu

| | id | name | sponsor | event | venue | place | ›hysical_descriptior | occasion | notes | call_number | date | location | currency | cu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filt |
| 1 | 12469 | | HOTEL NETHERLAND | SUPPER | COMMERCIAL | NEW YORK NY | CARD; ILLUS; COL; 6.0X8.75; | | HOTEL CREST IN BLUE; PRICED M... | 1900-2838 | 1900-04-16T0... | HOTEL NETHERLAND | DOLLARS | $ |
| 2 | 12474 | | ALPHA OF ZETA PSI | ANNUAL BANQUET | COMMERCIAL | DELMONICOS NEW YORK NY | BOOKLET; ILLUS; COL; 5.5X7.0; | | VELLUM COVER; CREST OF ZETA... | 1900-2844 | 1900-04-17T0... | ALPHA OF ZETA PSI | DOLLARS | $ |
| 3 | 12475 | | HOTEL MANHATTAN | DINNER | COMMERCIAL | NEW YORK NY | CARD; ILLUS; 6X9.5; | | A LA CARTE DU JOUR; HOTEL SE... | 1900-2847 | 1900-04-18T0... | HOTEL MANHATTAN | DOLLARS | $ |
| 4 | 12483 | | HOTEL MANHATTAN | CARTE DU JOUR | COMMERCIAL | NEW YORK NY | CARD; ILLUS; 6X9.5; | | A LA CARTE MENU; HOTEL S... | 1900-2859 | 1900-04-18T0... | HOTEL MANHATTAN | DOLLARS | $ |
| 5 | 12498 | | CAFE BOULEVARD | SUNDAY DINNER | COMMERCIAL | 156 SECOND AVENUE NY | CARD; COL; ILLUS; 5 X 7; | | PRIX FIXE DINNER; FOOD APPEARS ... | 1900-2903 | 1900-04-22T0... | CAFE BOULEVARD | CENTS | c |
| 6 | 12504 | | USMS NEW YORK | LUNCHEON | COMMERCIAL | EN ROUTE | CARD; 4 X 6.25; | | FLAG; PRICED WINE LIST AND ... | 1900-1131 | 1900-02-18T0... | USMS NEW YORK | DOLLARS | $ |
| 7 | 12505 | | BARTHOLDI HOTEL | BREAKFAST | COMMERCIAL | NY | BROADSIDE; COL; 7 X 11.75; | DAILY; | VERY POOR CONDITION; | 1900-1137 | 1900-02-18T0... | BARTHOLDI HOTEL | DOLLARS | $ |
| 8 | 12506 | | BARTHOLDI HOTEL | LUNCH & DINNER | COMMERCIAL | 23RD ST & BWAY NY | FOLDER; ILLUS; 8.25 X 14; | DAILY; | MANY ITEMS HANDWRITTEN; ... | 1900-1138 | 1900-02-18T0... | BARTHOLDI HOTEL | DOLLARS | $ |
| 9 | 12507 | | HAANS | SUPPER | COMMERCIAL | 75 ST & COLUMBUS AVE... | BROADSHEET; COL; 7 X 11; | DAILY; | COMPLETE WINE LIST ON BACK; | 1900-1140 | 1900-02-19T0... | HAANS | DOLLARS | $ |
| 10 | 12508 | | HOTEL MARIE ANTOINETTE | LUNCHEON | COMMERCIAL | 66 ST & BWAY NY | BROADSIDE; HOTEL EMBLEM;... | DAILY; | | 1900-1141 | 1900-02-19T0... | HOTEL MARIE ANTOINETTE | DOLLARS | $ |
| 11 | 12509 | | RED STAR LINE SSSOUTHWARK | DINNER | COMMERCIAL | EN ROUTE | BROADSHEET; ILLUS; COL; 5 X 8; | DAILY; | MENU HANDWRITTEN;... | 1900-1169 | 1900-02-20T0... | RED STAR LINE SSSOUTHWARK | FRANCS | FF |
| 12 | 12510 | | HOTEL MARIE ANTOINETTE | DINNER | COMMERCIAL | 66 ST & BWAY NY | BROADSIDE; HOTEL EMBLEM;... | DAILY; | | 1900-1173 | 1900-02-20T0... | HOTEL MARIE ANTOINETTE | DOLLARS | $ |
| 13 | 12516 | | REVERE HOUSE | COMPLIMENTARY BANQUET GIVE... | RESTAURANT | BOSTON MA | BROADSIDE; ILLUS; 4.25 X 1... | | MENU PRINTED IN BLACK ON CRE... | 1865-0001 | 1865-09-28T0... | PARKER HOUSE | DOLLARS | $ |
| 14 | 12518 | | COLUMBIA RESTAURANT | DAILY MENU | COMMERCIAL | 48 EAST 14TH STREETNEW YO... | FOLDER;ILLUS; 7.25X11.5; | DAILY; | 2 COPIES;WINE LIST; | 1901-1527 | 1901-06-03T0... | COLUMBIA RESTAURANT | DOLLARS | $ |
| 15 | 12523 | | CITIZENSSTEAM... | LUNCH | COMMERCIAL | STEAMER SARATOGATRO... | BOOKLET;ILLUS;... | DAILY; | AU SABLE CHASM ON FRONT COV... | 1901-1537 | 1901-01-01T0... | CITIZENSSTEAM... | DOLLARS | $ |
| 16 | 12526 | | UNION HOTEL & RESTAURANT | DINNER | COMMERCIAL | | BROADSIDE; ILLUS; 5.5X10; | OTHER DAILY; | A LA CARTE, PRICED MENU; E... | 1900-2139 | 1900-02-26T0... | UNION HOTEL & RESTAURANT | DOLLARS | $ |
| 17 | 12528 | | CUNARD LINE | LUNCHEON | COMMERCIAL | RMS CAMPANIA | CARD; ILLUS; COL; 4.5X6.5; | OTHER DAILY; | PRICED WINE, TOBACCO LIST ... | 1900-2142 | 1900-02-26T0... | CUNARD LINE | DOLLARS | $ |
| 18 | 12533 | | RED STAR LINE | DINNER | COMMERCIAL | SS SOUTHWARK | CARD; ILLUS; COL; 5X8; | OTHER DAILY; | VIOLET SCRIPT PRINTING; MARI... | 1900-2152 | 1900-02-26T0... | RED STAR LINE | BELGIAN FRANCS | BE |



| | id | menu_page_id | price | high_price | dish_id | created_at | update_at | xpos | ypos |
|---|---|---|---|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter |
| 1 | 5 | 3079 | 0.5 | 1 | 5 | 2011-03-28T1... | 2011-04-13T1... | 0.105714 | 0.313178 |
| 2 | 24 | 3079 | 1.5 | 3 | 23 | 2011-03-29T1... | 2011-04-13T1... | 0.0814286 | 0.470795 |
| 3 | 82 | 3079 | 0.75 | 1.5 | 75 | 2011-04-13T1... | 2011-04-13T1... | 0.101429 | 0.323457 |
| 4 | 83 | 3079 | 2 | 4 | 76 | 2011-04-13T1... | 2011-04-13T1... | 0.0842857 | 0.481759 |
| 5 | 145 | 1291 | 0.75 | 1.5 | 133 | 2011-04-14T2... | 2011-04-14T2... | 0.107143 | 0.742283 |
| 6 | 1134 | 1434 | 0.25 | 0.4 | 823 | 2011-04-19T1... | 2011-07-12T2... | 0.508571 | 0.90681 |
| 7 | 1136 | 1434 | 0.25 | 0.4 | 777 | 2011-04-19T1... | 2011-07-12T2... | 0.504286 | 0.964114 |
| 8 | 1149 | 1434 | 0.25 | 0.5 | 836 | 2011-04-19T1... | 2011-07-12T2... | 0.504286 | 0.892735 |
| 9 | 1153 | 1434 | 0.25 | 0.4 | 839 | 2011-04-19T1... | 2011-07-12T2... | 0.504286 | 0.878661 |
| 10 | 1455 | 168 | 40 | 40 | 1034 | 2011-04-19T1... | 2011-12-05T0... | 0.217143 | 0.223982 |
| 11 | 1459 | 168 | 0.25 | 0.4 | 1037 | 2011-04-19T1... | 2011-04-19T2... | 0.52 | 0.223982 |
| 12 | 1471 | 168 | 0.25 | 0.4 | 57354 | 2011-04-19T1... | 2011-05-08T1... | 0.0685714 | 0.268416 |
| 13 | 1550 | 1291 | 0.3 | 0.5 | 1100 | 2011-04-19T1... | 2012-04-06T1... | 0.115714 | 0.778974 |
| 14 | 1568 | 168 | 0.35 | 0.6 | 193 | 2011-04-19T2... | 2011-04-19T2... | 0.0585714 | 0.330986 |
| 15 | 1571 | 168 | 0.35 | 0.6 | 1115 | 2011-04-19T2... | 2011-04-19T2... | 0.0614286 | 0.342774 |
| 16 | 1574 | 168 | 0.35 | 0.6 | 1117 | 2011-04-19T2... | 2011-04-19T2... | 0.0571429 | 0.359097 |
| 17 | 1583 | 168 | 0.25 | 0.4 | 161 | 2011-04-19T2... | 2011-04-19T2... | 0.734286 | 0.238491 |
| 18 | 1587 | 168 | 0.25 | 0.4 | 399 | 2011-04-19T2... | 2011-05-08T1... | 0.0542857 | 0.279298 |
| 19 | 1593 | 168 | 0.25 | 0.4 | 6464 | 2011-04-19T2... | 2011-05-08T1... | 0.28 | 0.282018 |



| | id | menu_id | page_number | image_id | full_height | full_width | uuid |
|---|---|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | Filter | Filter | Filter |
| 1 | 119 | 12460 | 1 | 1603595 | 7230 | 5428 | 510d47e4-2955-<br>a3d9-e040-e00... |
| 2 | 120 | 12460 | 2 | 1603596 | 5428 | 7230 | 510d47e4-2956-<br>a3d9-e040-e00... |
| 3 | 121 | 12460 | 3 | 1603597 | 7230 | 5428 | 510d47e4-2957-<br>a3d9-e040-e00... |
| 4 | 122 | 12460 | 4 | 1603598 | 7230 | 5428 | 510d47e4-2958-<br>a3d9-e040-e00... |
| 5 | 123 | 12461 | 1 | 1603591 | 7230 | 5428 | 510d47e4-2959-<br>a3d9-e040-e00... |
| 6 | 124 | 12461 | 2 | 1603592 | 7230 | 5428 | 510d47e4-295a-<br>a3d9-e040-e00... |
| 7 | 125 | 12461 | 3 | 1603593 | 7230 | 5428 | 510d47e4-295b-<br>a3d9-e040-e00... |
| 8 | 126 | 12461 | 4 | 1603594 | 7230 | 5428 | 510d47e4-295c-<br>a3d9-e040-e00... |
| 9 | 127 | 12462 | 2 | 4000001128 | | | 510d47db-1ef8-<br>a3d9-e040-e00... |
| 10 | 128 | 12462 | 1 | 474450 | | | 510d47db-1ef9-<br>a3d9-e040-e00... |
| 11 | 129 | 12463 | 2 | 4000009170 | 3074 | 2046 | 510d47db-491e-<br>a3d9-e040-e00... |
| 12 | 130 | 12463 | 1 | 466928 | 3049 | 2004 | 510D47DB-491F-<br>A3D9-E040-E0... |
| 13 | 131 | 12464 | 2 | 4000009171 | 3690 | 2888 | 510d47db-4920-<br>a3d9-e040-e00... |
| 14 | 132 | 12464 | 1 | 466930 | 3679 | 2866 | 510d47db-4921-<br>a3d9-e040-e00... |
| 15 | 133 | 12465 | 2 | 4000009172 | 3413 | 2307 | 510d47db-4922-<br>a3d9-e040-e00... |
| 16 | 134 | 12465 | 1 | 466931 | 3411 | 2291 | 510d47db-4923-<br>a3d9-e040-e00... |
| 17 | 135 | 12466 | 2 | 4000009173 | 3352 | 2312 | 510d47db-4924-<br>a3d9-e040-e00... |
| 18 | 136 | 12466 | 1 | 466932 | 3365 | 2305 | 510d47db-4925-<br>a3d9-e040-e00... |
| 19 | 137 | 12467 | 2 | 4000009174 | 3090 | 2102 | 510d47db-4926- |

## 13.Check Integrity Constraints

- **IC check for Dish table**

Select * from dish where id is null
Select * from dish where menus_appeared is null or times_appeared is null

```
-- last_appeared can’t be before first_appeared
select * from dish where last_appeared < first_appeared;

Select * from dish where lowest_price > highest_price
```

- **IC check for Menu table**

```
Select * from menu where id is null

--- date can't be less than 1850 and greater than 2020
select id, name, event, venue, date
from menu
where date is not null and date != '' and (date < '1850-01-01' or date >
'2020-07-01');
```

- **IC check for MenuItem table**

```
 select * from menuitem where id is null
select * from menuitem where dish_id is null
 select * from menuitem where menu_page_id is null
-- price must not be higher than high price
select id, price, high_price from menuitem where price > high_price;
```

- **IC check for MenuPage table**

```
Select * from menupage where id is null
Select * from menupage where menu_id is null
```

- **Some additional constraints**

The count of the  dish id in menu items must be equal to times appeared

```
select dish_id, name, count(dish_id), times_appeared
from MenuItem
join dish on MenuItem.dish_id = dish.id
group by dish_id, name
having count(dish_id) != dish.times_appeared
```

```
-- The low price of a dish must be equal to the lowest price from menu item

select dish_id, name, dish.lowest_price, min(menuitem.price) from dish
join menuitem on menuitem.dish_id = dish.id
group by dish_id, name
having min(menuitem.price) != dish.lowest_price

-- The highest price of a dish must equal to the highest price from menu item

select dish_id, name, dish.lowest_price, max(menuitem.price) from dish
join menuitem on menuitem.dish_id = dish.id
group by dish_id, name
having max(menuitem.price) != dish.lowest_price;

 -- count of the menu ids must match menus appeared

select dish_id, name, count(menu_id), menus_appeared
from MenuItem
join dish on MenuItem.dish_id = dish.id
join MenuPage on MenuPage.id = menuitem.menu_page_id
group by dish_id, name
having count(menu_id) != dish.menus_appeared;
```

## 14. Repair integrity constraints

When doing the integrity constraints we found data that didn't pass them so we used the following queries to transform them.

```
delete from menuitem where dish_id not in (select id from dish);
37642 rows deleted
```

```
delete from menupage where menu_id not in (select id from menu);
47937 rows deleted
```

```
delete from menuitem where menu_page_id not in (select id from menupage);
1526 rows deleted
```

```
update menuitem set high_price = price where id in (select id from menuitem where price > high_price);
```

```
1268 rows updated

update dish set times_appeared = coalesce((

    select coalesce(count(dish_id), 0) as cnt
    from MenuItem
    where dish_id = dish.id
    group by dish_id
), 0) where id in (
    select dish_id
    from MenuItem
    join dish on MenuItem.dish_id = dish.id
    group by dish_id, name
    having count(dish_id) != dish.times_appeared
);
6684 rows updated

update dish set lowest_price = coalesce((
    select min(menuitem.price)
    from menuitem
    where dish_id = dish.id
    group by dish_id
    having min(menuitem.price) != dish.lowest_price
), 0) where id in (
    select dish_id
    from dish
    join menuitem on menuitem.dish_id = dish.id
    group by dish_id
    having min(menuitem.price) != dish.lowest_price
);
2294 rows updated

update dish set highest_price = coalesce((
    select max(menuitem.price)
    from menuitem
    where dish_id = dish.id
    group by dish_id
    having max(menuitem.price) != dish.highest_price
), 0) where id in (
    select dish_id
    from dish
    join menuitem on menuitem.dish_id = dish.id
    group by dish_id
    having max(menuitem.price) != dish.highest_price
);
24394 rows updated
```

```
update dish set menus_appeared = coalesce((
    select  count(menu_id)
    from MenuItem
    join MenuPage on MenuPage.id = menuitem.menu_page_id
    where menuitem.dish_id = dish.id
    group by dish_id
    having count(menu_id) != dish.menus_appeared
), 0) where id in (
    select dish_id
    from MenuItem
    join dish on MenuItem.dish_id = dish.id
    join MenuPage on MenuPage.id = menuitem.menu_page_id
    group by dish_id, name
    having count(menu_id) != dish.menus_appeared
);
6770 rows updated
```

After running all update queries, all integrity constraints passed with no reported issues.

## 15.Create a Workflow Model

We have used YesWorkflow editor to generate the overall workflow diagram and OR2YWTool to generate workflow diagrams for the OpenRefine data cleaning steps.

The overall workflow diagram contains 4 stages - OpenRefineSequence, TableauCleaningSequence, SQLiteLoadingOperations and SQLIntegrityConstraintsCheck.

1. **OpenRefineSequence:**
   - Inputs: Menu.csv, MenuPage.csv, MenuItem.csv and Dish.csv.
   - Outputs: Menu_Clean.csv, MenuPage_Clean.csv, MenuItem_Clean.csv and Dish_Clean.csv.
   - Dependency: OpenRefine tool.

2. **TableauCleaningSequence:**
   - Inputs: Menu_Clean.csv, MenuPage_Clean.csv, MenuItem_Clean.csv and Dish_Clean.csv.
   - Outputs: Menu_Clean_Final.csv, MenuPage_Clean_Final.csv, MenuItem_Clean_Final.csv and Dish_Clean_Final.csv.
   - Dependency: Tableau Desktop Software.

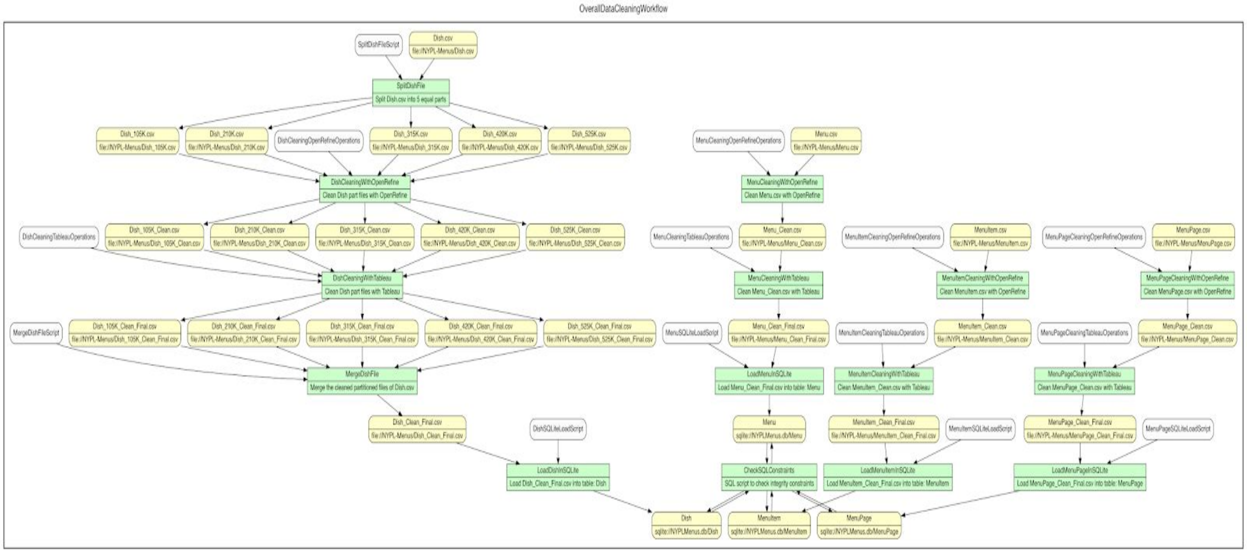
3. **SQLiteLoadingOperations:**
   - Inputs: Menu_Clean_Final.csv, MenuPage_Clean_Final.csv, MenuItem_Clean_Final.csv and Dish_Clean_Final.csv.
   - Outputs: SQLite Tables - Menu, MenuPage, MenuItem and Dish.
   - Dependency: SQLite software.

4. **SQLIntegrityConstraintsCheck:**
   - Inputs: SQLite Tables - Menu, MenuPage, MenuItem and Dish.
   - Outputs: SQLite Tables - Menu, MenuPage, MenuItem and Dish.
   - Dependency: SQLite software.

## 16. Visual representation of your overall workflow

Below diagram is the visual representation of the overall workflow of our data cleaning process.

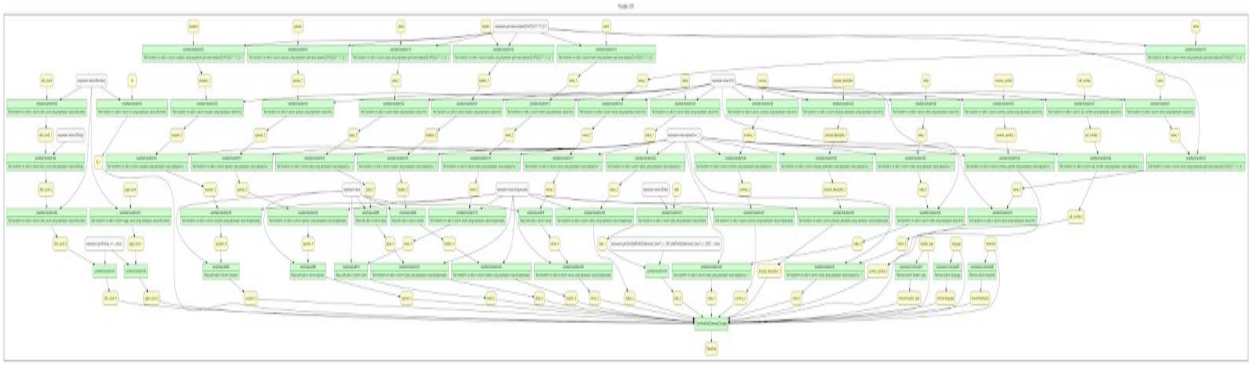
Please note that, to generate the .gv file and .png file using YesWorkflow, we renamed the “Overflow_Workflow.txt” to “Overflow_Workflow.py” as YesWorkflow doesn’t support the TXT files as input.


OverallDataCleaningWorkflow
SplitDishFileScript
Dish.csv
file://NYPL-Menus/Dish.csv
SplitDishFile
Split Dish.csv into 5 equal parts
Dish_105K.csv
file://NYPL-Menus/Dish_105K.csv
Dish_210K.csv
file://NYPL-Menus/Dish_210K.csv
DishCleaningOpenRefineOperations
Dish_315K.csv
file://NYPL-Menus/Dish_315K.csv
Dish_420K.csv
file://NYPL-Menus/Dish_420K.csv
Dish_525K.csv
file://NYPL-Menus/Dish_525K.csv
DishCleaningWithOpenRefine
Clean Dish part files with OpenRefine
DishCleaningTableauOperations
Dish_105K_Clean.csv
file://NYPL-Menus/Dish_105K_Clean.csv
Dish_210K_Clean.csv
file://NYPL-Menus/Dish_210K_Clean.csv
Dish_315K_Clean.csv
file://NYPL-Menus/Dish_315K_Clean.csv
Dish_420K_Clean.csv
file://NYPL-Menus/Dish_420K_Clean.csv
Dish_525K_Clean.csv
file://NYPL-Menus/Dish_525K_Clean.csv
DishCleaningWithTableau
Clean Dish part files with Tableau
MergeDishFileScript
Dish_105K_Clean_Final.csv
file://NYPL-Menus/Dish_105K_Clean_Final.csv
Dish_210K_Clean_Final.csv
file://NYPL-Menus/Dish_210K_Clean_Final.csv
Dish_315K_Clean_Final.csv
file://NYPL-Menus/Dish_315K_Clean_Final.csv
Dish_420K_Clean_Final.csv
file://NYPL-Menus/Dish_420K_Clean_Final.csv
Dish_525K_Clean_Final.csv
file://NYPL-Menus/Dish_525K_Clean_Final.csv
MergeDishFile
Merge the cleaned partitioned files of Dish.csv
Dish_Clean_Final.csv
file://NYPL-Menus/Dish_Clean_Final.csv
DishSQLiteLoadScript
LoadDishInSQLite
Load Dish_Clean_Final.csv into table: Dish
MenuCleaningOpenRefineOperations
Menu.csv
file://NYPL-Menus/Menu.csv
MenuCleaningWithOpenRefine
Clean Menu.csv with OpenRefine
MenuCleaningTableauOperations
Menu_Clean.csv
file://NYPL-Menus/Menu_Clean.csv
MenuItemCleaningOpenRefineOperations
MenuItem.csv
file://NYPL-Menus/MenuItem.csv
MenuPageCleaningOpenRefineOperations
MenuPage.csv
file://NYPL-Menus/MenuPage.csv
MenuCleaningWithTableau
Clean Menu_Clean.csv with Tableau
MenuItemCleaningWithOpenRefine
Clean MenuItem.csv with OpenRefine
MenuPageCleaningWithOpenRefine
Clean MenuPage.csv with OpenRefine
MenuSQLiteLoadScript
Menu_Clean_Final.csv
file://NYPL-Menus/Menu_Clean_Final.csv
MenuItemCleaningTableauOperations
MenuItem_Clean.csv
file://NYPL-Menus/MenuItem_Clean.csv
MenuPageCleaningTableauOperations
MenuPage_Clean.csv
file://NYPL-Menus/MenuPage_Clean.csv
LoadMenuInSQLite
Load Menu_Clean_Final.csv into table: Menu
MenuItemCleaningWithTableau
Clean MenuItem_Clean.csv with Tableau
MenuPageCleaningWithTableau
Clean MenuPage_Clean.csv with Tableau
Menu
sqlite://NYPLMenus.db/Menu
MenuItem_Clean_Final.csv
file://NYPL-Menus/MenuItem_Clean_Final.csv
MenuItemSQLiteLoadScript
MenuPage_Clean_Final.csv
file://NYPL-Menus/MenuPage_Clean_Final.csv
MenuPageSQLiteLoadScript
CheckSQLConstraints
SQL script to check integrity constraints
LoadMenuItemInSQLite
Load MenuItem_Clean_Final.csv into table: MenuItem
LoadMenuPageInSQLite
Load MenuPage_Clean_Final.csv into table: MenuPage
Dish
sqlite://NYPLMenus.db/Dish
MenuItem
sqlite://NYPLMenus.db/MenuItem
MenuPage
sqlite://NYPLMenus.db/MenuPage

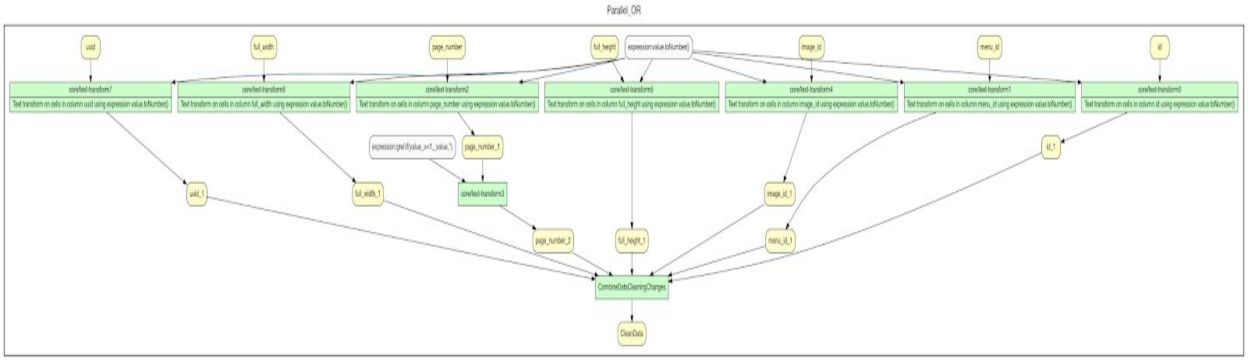
## 17. Visual representation of your OpenRefine workflow

Below diagrams provide a visual representation of the OpenRefine workflow for each of the files in the dataset. The .gv and .png files for the OpenRefine workflows are generated using OR2YWTool.

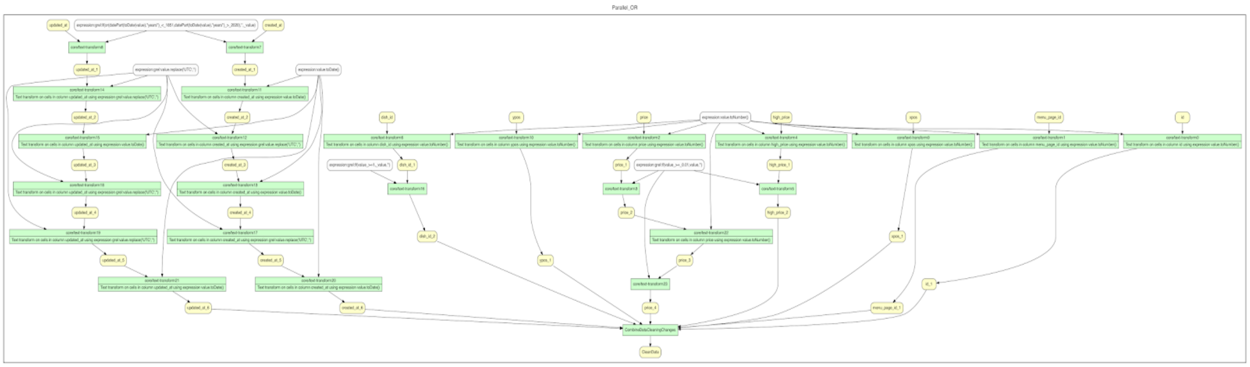
### Menu.csv



### MenuPage.csv


Parallel_OR
uuid
full_width
page_number
full_height
expression:value.toNumber()
image_id
menu_id
id
core/text-transform7
Text transform on cells in column uuid using expression value.toNumber()
core/text-transform6
Text transform on cells in column full_width using expression value.toNumber()
core/text-transform2
Text transform on cells in column page_number using expression value.toNumber()
core/text-transform5
Text transform on cells in column full_height using expression value.toNumber()
core/text-transform4
Text transform on cells in column image_id using expression value.toNumber()
core/text-transform1
Text transform on cells in column menu_id using expression value.toNumber()
core/text-transform0
Text transform on cells in column id using expression value.toNumber()
expression:grel:if(value_>=1,_value,'')
page_number_1
id_1
uuid_1
full_width_1
core/text-transform3
image_id_1
page_number_2
full_height_1
menu_id_1
CombineDataCleaningChanges
CleanData

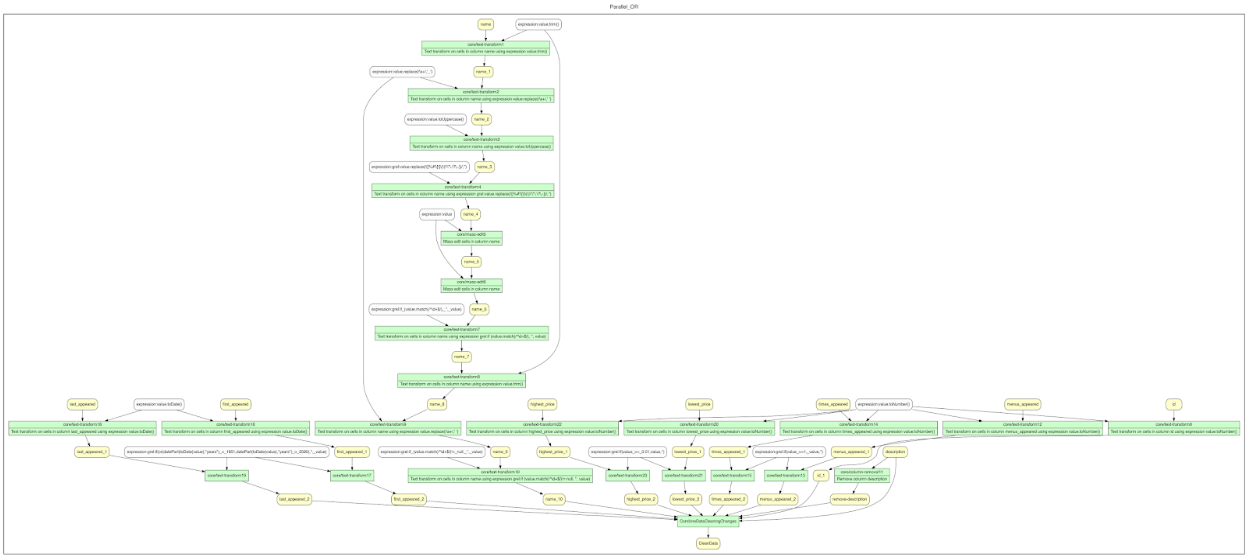
## MenuItem.csv


Parallel_OR
updated_at
expression:grel:if(or(datePart(toDate(value),"years")_<_1851,datePart(toDate(value),"years")_>_2020),"",_value)
created_at
core/text-transform8
core/text-transform7
updated_at_1
expression:grel:value.replace('UTC','')
created_at_1
expression:value.toDate()
core/text-transform14
Text transform on cells in column updated_at using expression grel:value.replace('UTC','')
core/text-transform11
Text transform on cells in column created_at using expression value.toDate()
updated_at_2
created_at_2
core/text-transform15
Text transform on cells in column updated_at using expression value.toDate()
core/text-transform12
Text transform on cells in column created_at using expression grel:value.replace('UTC','')
updated_at_3
created_at_3
core/text-transform18
Text transform on cells in column updated_at using expression grel:value.replace('UTC','')
core/text-transform13
Text transform on cells in column created_at using expression value.toDate()
updated_at_4
created_at_4
core/text-transform19
Text transform on cells in column updated_at using expression grel:value.replace('UTC','')
core/text-transform17
Text transform on cells in column created_at using expression grel:value.replace('UTC','')
updated_at_5
created_at_5
core/text-transform21
Text transform on cells in column updated_at using expression value.toDate()
core/text-transform20
Text transform on cells in column created_at using expression value.toDate()
updated_at_6
created_at_6
dish_id
ypos
price
expression:value.toNumber()
high_price
xpos
menu_page_id
id
core/text-transform6
Text transform on cells in column dish_id using expression value.toNumber()
core/text-transform10
Text transform on cells in column ypos using expression value.toNumber()
core/text-transform2
Text transform on cells in column price using expression value.toNumber()
core/text-transform4
Text transform on cells in column high_price using expression value.toNumber()
core/text-transform9
Text transform on cells in column xpos using expression value.toNumber()
core/text-transform1
Text transform on cells in column menu_page_id using expression value.toNumber()
core/text-transform0
Text transform on cells in column id using expression value.toNumber()
expression:grel:if(value_>=1,_value,'')
dish_id_1
price_1
expression:grel:if(value_>=_0.01,value,'')
high_price_1
core/text-transform16
core/text-transform3
core/text-transform5
price_2
high_price_2
dish_id_2
core/text-transform22
Text transform on cells in column price using expression value.toNumber()
xpos_1
ypos_1
price_3
id_1
core/text-transform23
price_4
menu_page_id_1
CombineDataCleaningChanges
CleanData

## Dish.csv

As we cleaned this file by partitioning into 5 parts, we got 5 OpenRefine history json files. The workflow for each of those json files is attached in the reference section.

Below image is the visual representation of the OpenRefine workflow for this file by consolidating the steps from all the 5 OpenRefine history json files.


Parallel_OR
name
expression:value.trim()
core/text-transform1
Text transform on cells in column name using expression value.trim()
expression:value.replace(/\s+/,'_')
name_1
core/text-transform2
Text transform on cells in column name using expression value.replace(/\s+/,' ')
expression:value.toUppercase()
name_2
core/text-transform3
Text transform on cells in column name using expression value.toUppercase()
expression:grel:value.replace(/[%#!{}\|\\^\[\]\/,-]/,'')
name_3
core/text-transform4
Text transform on cells in column name using expression grel:value.replace(/[%#!{}\|\\^\[\]\/,-]/,'')
expression:value
name_4
core/mass-edit5
Mass edit cells in column name
name_5
core/mass-edit6
Mass edit cells in column name
expression:grel:if_(value.match(/^\d+$/),_"",_value)
name_6
core/text-transform7
Text transform on cells in column name using expression grel:if (value.match(/^\d+$/), "", value)
name_7
core/text-transform8
Text transform on cells in column name using expression value.trim()
name_8
last_appeared
expression:value.toDate()
first_appeared
highest_price
lowest_price
times_appeared
expression:value.toNumber()
menus_appeared
id
core/text-transform18
Text transform on cells in column last_appeared using expression value.toDate()
core/text-transform16
Text transform on cells in column first_appeared using expression value.toDate()
core/text-transform9
Text transform on cells in column name using expression value.replace(/\s+/,' ')
core/text-transform22
Text transform on cells in column highest_price using expression value.toNumber()
core/text-transform20
Text transform on cells in column lowest_price using expression value.toNumber()
core/text-transform14
Text transform on cells in column times_appeared using expression value.toNumber()
core/text-transform12
Text transform on cells in column menus_appeared using expression value.toNumber()
core/text-transform0
Text transform on cells in column id using expression value.toNumber()
last_appeared_1
expression:grel:if(or(datePart(toDate(value),"years")_<_1851,datePart(toDate(value),"years")_>_2020),"",_value)
first_appeared_1
expression:grel:if_(value.match(/^\d+$/)!=_null,_"",_value)
name_9
highest_price_1
expression:grel:if(value_>=_0.01,value,'')
lowest_price_1
times_appeared_1
expression:grel:if(value_>=1,_value,'')
menus_appeared_1
description
core/text-transform19
core/text-transform17
core/text-transform10
Text transform on cells in column name using expression grel:if (value.match(/^\d+$/)!= null, "", value)
core/text-transform23
core/text-transform21
core/text-transform15
core/text-transform13
id_1
core/column-removal11
Remove column description
last_appeared_2
first_appeared_2
name_10
highest_price_2
lowest_price_2
times_appeared_2
menus_appeared_2
remove-description
CombineDataCleaningChanges
CleanData

## 18.Further analysis/takeaways/challenges

The most challenging task in this data cleaning project was how to clean up the files in a way such that we do not eliminate the important data. For eg. The dish file was extremely big and we had to partition it into 5 parts with Tableau Prep and clean individual file. We could not use OpenRefine for cleaning the dish file.
We had to rely on suggested cluster values as there were more than 3000 + clusters identified by Open Refine and Tableau Prep.
For the remaining files, since the data is so huge it was not always possible to check the integrity of each and every data item. So we had to partly rely on what the clustering suggestions were provided by OpenRefine.