CS 513 - Theory And Data Cleaning Final Project - Phase 1 Report

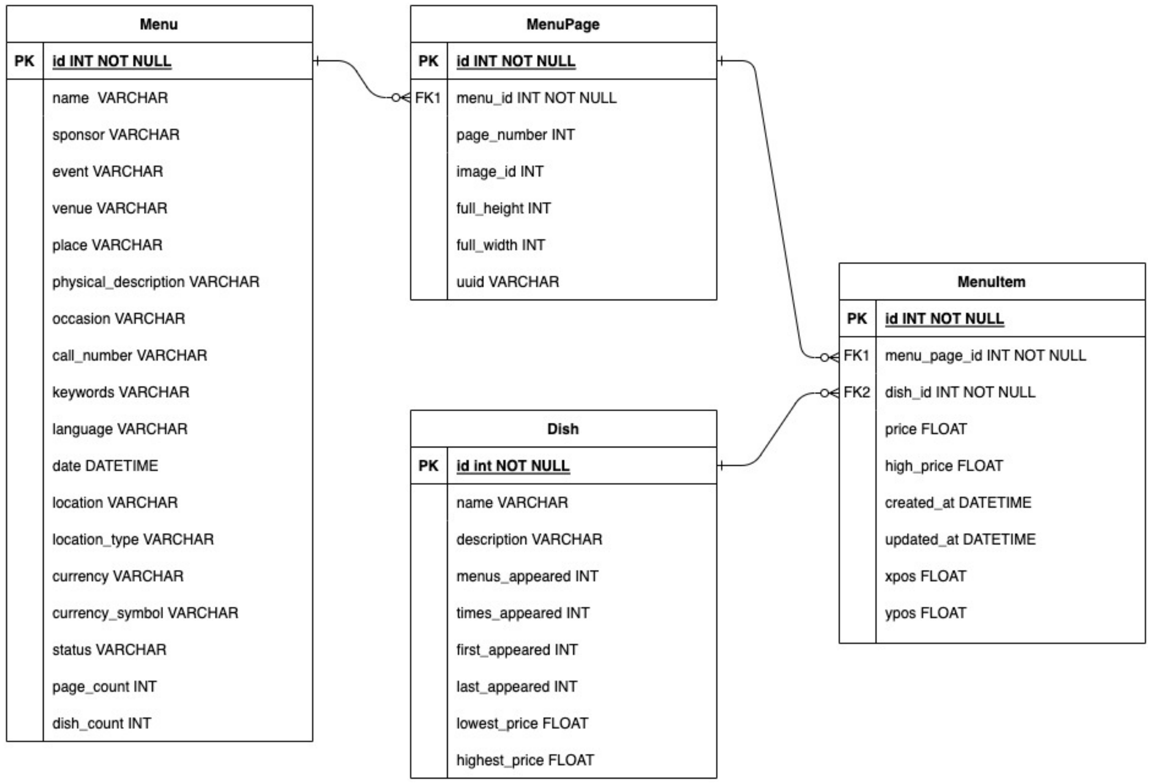
Team ID: Team 150
Sushma Ponna - ponna2@illinois.edu
Michael Inoue - mwinoue2@illinois.edu
Sanjeev Datta - sanjeev6@illinois.edu

1. Dataset Chosen

We have chosen to work with the New York Public Library (NYPL) menus dataset.

2. Description Of Dataset

The New York Public Library is digitizing and transcribing its collection of historical menus. The collection includes about 45,000 menus from the 1840s to the present, and the goal of the digitization project is to transcribe each page of each menu, creating a comprehensive database of dishes, prices, locations. As of today, the transcribed database contains 1,332,726 dishes from 17,545 menus. See the ER diagram below that depicts the schema of the dataset.



This dataset is split into four files to minimize the amount of redundant information contained in each (and thus, the size of each file). The four data files are Menu, MenuPage, MenuItem, and Dish. These four files are described briefly here, and in detail in their individual file descriptions below.

Menu

The core element of the dataset. Each Menu has a unique identifier and associated data, including data on the venue and/or event that the menu was created for; the location that the menu was used; the currency in use on the menu; and various other fields. Each menu is associated with some number of MenuPage values.

Menu.csv
Row Count: 17545

1. id : a unique identifier for the menu *(maps to the menu_id in MenuPage.csv)*
2. name : name of the Hotel/Restaurant
3. sponsor : sponsor of the menu
4. event : event the menu was created for
5. venue : type of the venue where the menu was served
6. place : address of the venue where the menu was served
7. physical_description : description of the menu's physical characteristics like paper card, dimensions, colors, design
8. occasion : name of the special occasion/holiday
9. notes : any special notes about the menu
10. call_number : the menu's call number
11. keywords : keywords for the menu
12. language : language in which the menu was printed
13. date : date the menu was created
14. location : location where the menu was served
15. location_type :type of the location
16. currency : currency used to charge for the dishes
17. currency_symbol : symbol for the currency used
18. status : the transcription status of the menu either 'complete' or 'under review'
19. page_count : number of pages in the menu
20. dish_count : number of dishes on the menu

MenuPage

Each MenuPage refers to the Menu it comes from, via the menu_id variable (corresponding to Menu:id). Each MenuPage also has a unique identifier of its own. Associated MenuPage data includes the page number of this MenuPage, an identifier for the scanned image of the page, and the dimensions of the page. Each MenuPage is associated with some number of MenuItem values.

MenuPage.csv
Row Count: 66937

1. id : identifier for the menu page (*maps to menu_page_id in MenuItem.csv*)
2. menu_id : a unique identifier for the menu (*maps to id in Menu.csv)*

3. page_number : sequential page number of the menu
4. image_id : the image id of an item in the menu
5. full_height : total height of the image
6. full_width : total width of the image
7. uuid : Universally Unique Identifier of the Menu page

MenuItem

Each MenuItem refers to both the MenuPage it is found on -- via the menu_page_id variable -- and the Dish that it represents -- via the dish_id variable. Each MenuItem also has a unique identifier of its own. Other associated data includes the price of the item and the dates when the item was created or modified in the database.

MenuItem.csv
Row Count: 1332726

1. id : a unique identifier for the menu item
2. menu_page_id : the id of the menu page (*maps to id in MenuPage.csv*)
3. price : price of the dish
4. high_price : highest price the dish was served
5. dish_id : the id of the dish (*maps to id in Dish.csv)*
6. created_at :the creation date for the dish
7. updated_at : the updation date for the dish
8. xpos : the x-axis position of the item on the scanned image of the menu page
9. ypos : the y-axis position of the item on the scanned image of the menu page

Dish

A Dish is a broad category that covers some number of MenuItems. Each dish has a unique id, to which it is referred by its affiliated MenuItems. Each dish also has a name, a description, a number of menus it appears on, and both date and price ranges.

Dish.csv
Row Count: 423397

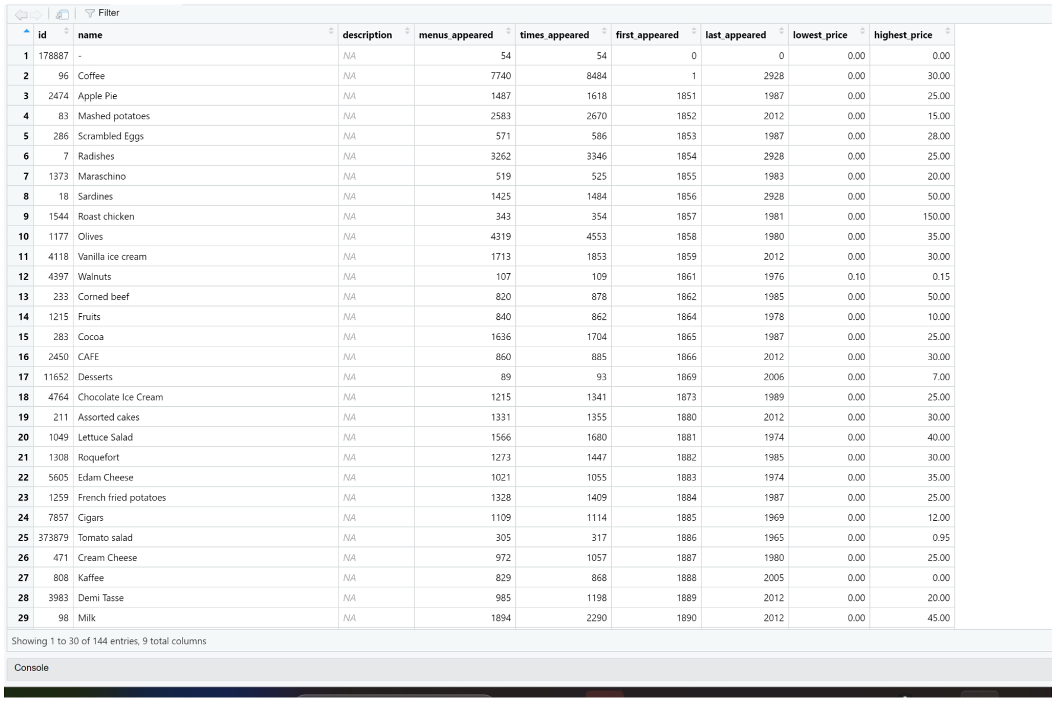
1. id : a unique identifier for the dish (*maps to dish_id in MenuItem.csv)*
2. name : name of the dish
3. description : description of the dish
4. menus_appeared : the number of menus the dish appears on
5. times_appeared : the number of times the dish appears on either multiple times or on multiple sections of a given menu
6. first_appeared : the year the dish first appeared
7. last_appeared : the year the dish last appeared
8. lowest_price : the lowest price of the dish
9. highest_price : the highest price of the dish

3. Use Cases

a. 'Zero cleaning' use case U0: data cleaning is not necessary

There is little to no data cleaning required for reporting the basic summary stats or details from the dataset. One such example of this use case is ***"To find out which dish is the most popular dish of the year"***. To answer this use case we group by the 'first_appeared' column from the Dish.csv and take the first row which has the maximum 'menus_appeared' count.

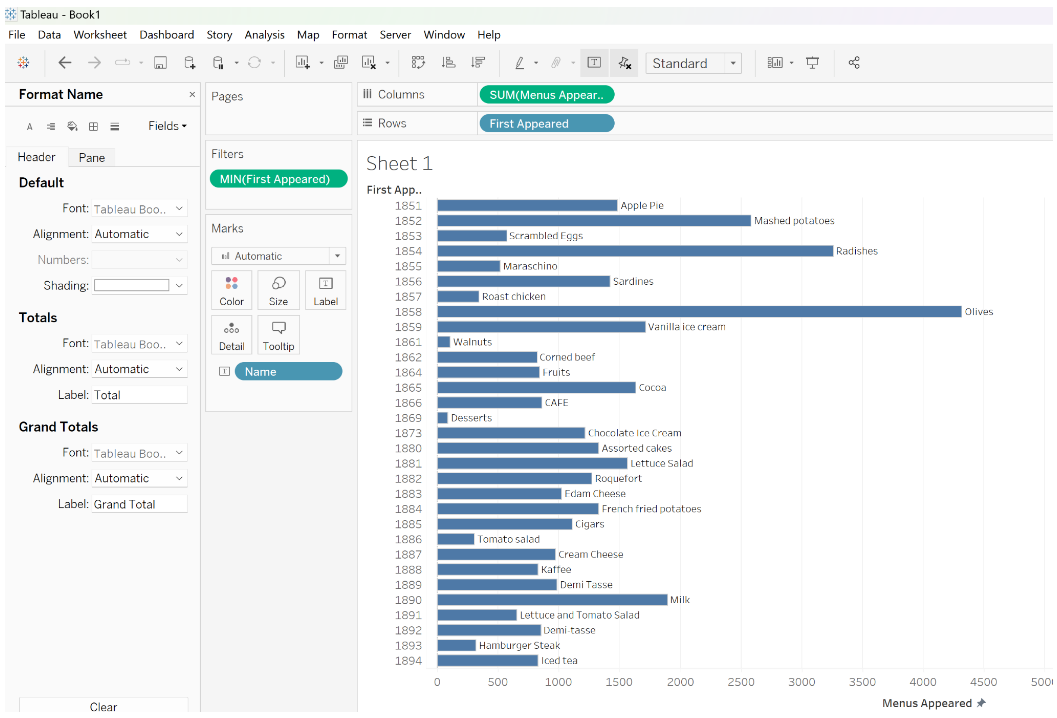
Below is a screenshot showing part of the query result.



	id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
1	178887	-	NA	54	54	0	0	0.00	0.00
2	96	Coffee	NA	7740	8484	1	2928	0.00	30.00
3	2474	Apple Pie	NA	1487	1618	1851	1987	0.00	25.00
4	83	Mashed potatoes	NA	2583	2670	1852	2012	0.00	15.00
5	286	Scrambled Eggs	NA	571	586	1853	1987	0.00	28.00
6	7	Radishes	NA	3262	3346	1854	2928	0.00	25.00
7	1373	Maraschino	NA	519	525	1855	1983	0.00	20.00
8	18	Sardines	NA	1425	1484	1856	2928	0.00	50.00
9	1544	Roast chicken	NA	343	354	1857	1981	0.00	150.00
10	1177	Olives	NA	4319	4553	1858	1980	0.00	35.00
11	4118	Vanilla ice cream	NA	1713	1853	1859	2012	0.00	30.00
12	4397	Walnuts	NA	107	109	1861	1976	0.10	0.15
13	233	Corned beef	NA	820	878	1862	1985	0.00	50.00
14	1215	Fruits	NA	840	862	1864	1978	0.00	10.00
15	283	Cocoa	NA	1636	1704	1865	1987	0.00	25.00
16	2450	CAFE	NA	860	885	1866	2012	0.00	30.00
17	11652	Desserts	NA	89	93	1869	2006	0.00	7.00
18	4764	Chocolate Ice Cream	NA	1215	1341	1873	1989	0.00	25.00
19	211	Assorted cakes	NA	1331	1355	1880	2012	0.00	30.00
20	1049	Lettuce Salad	NA	1566	1680	1881	1974	0.00	40.00
21	1308	Roquefort	NA	1273	1447	1882	1985	0.00	30.00
22	5605	Edam Cheese	NA	1021	1055	1883	1974	0.00	35.00
23	1259	French fried potatoes	NA	1328	1409	1884	1987	0.00	25.00
24	7857	Cigars	NA	1109	1114	1885	1969	0.00	12.00
25	373879	Tomato salad	NA	305	317	1886	1965	0.00	0.95
26	471	Cream Cheese	NA	972	1057	1887	1980	0.00	25.00
27	808	Kaffee	NA	829	868	1888	2005	0.00	0.00
28	3983	Demi Tasse	NA	985	1198	1889	2012	0.00	20.00
29	98	Milk	NA	1894	2290	1890	2012	0.00	45.00

Showing 1 to 30 of 144 entries, 9 total columns

Below is a screenshot of a barplot that displays the most popular dish in a given year. For visualizing purposes, the screenshot depicts 1800 <= 'first_appeared' <= 1894.



Below is another screenshot showing a barplot that displays the most popular dish in a given year. 1978 <= ‘first_appeared’ <= 2012.

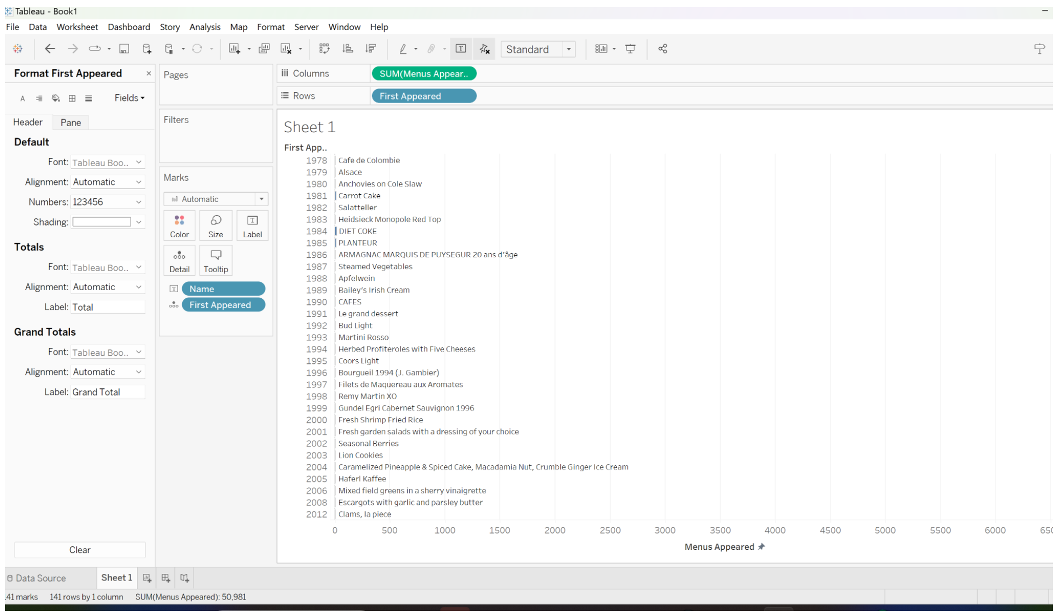
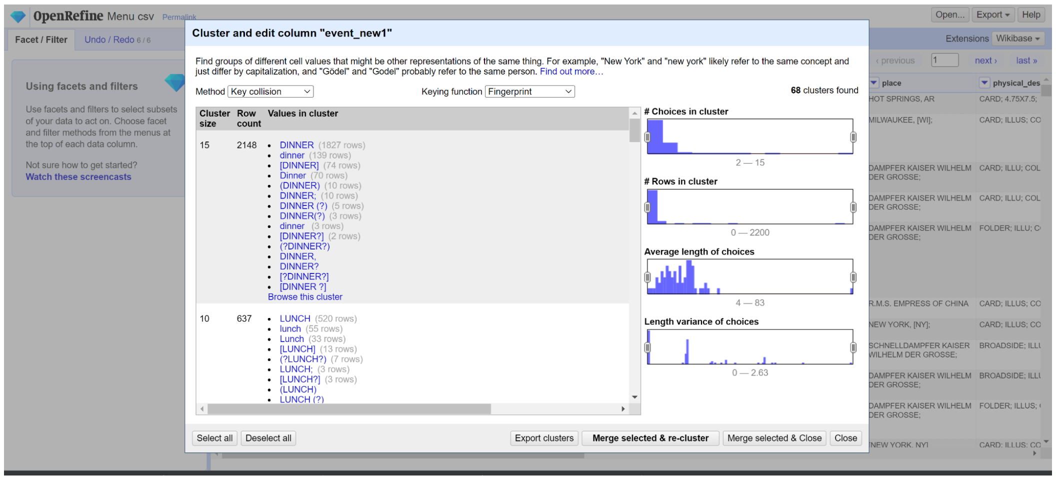


Tableau - Book1
Format First Appeared
Columns: SUM(Menus Appear..
Rows: First Appeared
Marks: Automatic
Name
First Appeared
Sheet 1
First App..
1978 Cafe de Colombie
1979 Alsace
1980 Anchovies on Cole Slaw
1981 Carrot Cake
1982 Salatteller
1983 Heidsieck Monopole Red Top
1984 DIET COKE
1985 PLANTEUR
1986 ARMAGNAC MARQUIS DE PUYSEGUR 20 ans d’âge
1987 Steamed Vegetables
1988 Apfelwein
1989 Bailey’s Irish Cream
1990 CAFES
1991 Le grand dessert
1992 Bud Light
1993 Martini Rosso
1994 Herbed Profiteroles with Five Cheeses
1995 Coors Light
1996 Bourgueil 1994 (J. Gambier)
1997 Filets de Maquereau aux Aromates
1998 Remy Martin XO
1999 Gundel Egri Cabernet Sauvignon 1996
2000 Fresh Shrimp Fried Rice
2001 Fresh garden salads with a dressing of your choice
2002 Seasonal Berries
2003 Lion Cookies
2004 Caramelized Pineapple & Spiced Cake, Macadamia Nut, Crumble Ginger Ice Cream
2005 Haferl Kaffee
2006 Mixed field greens in a sherry vinaigrette
2008 Escargots with garlic and parsley butter
2012 Clams, la piece
0 500 1000 1500 2000 2500 3000 3500 4000 4500 5000 5500 6000
Menus Appeared
Data Source
Sheet 1
41 marks 141 rows by 1 column SUM(Menus Appeared): 50,981

b. 'Main' use case U1: data cleaning is necessary and sufficient

Data cleaning is essential for most of the applications if you want to generate meaningful insights. Often times data is used to make predictions and the underlying models rely heavily on the quality of the data without which the predictions would be incorrect.

An example of the 'Main' use case here would be ***"To find out what is the average number of items on a menu, lets say, for dinner."*** For this we would first want to examine the column "event" from the "Menu.csv". There are approximately 1770 choices for the column event in the file and there are 14 different variations in which the event name 'Dinner' is written. So we need to merge and re-cluster these ~2k rows with a new event name 'DINNER' given to this cluster. We used OpenRefine to cluster with the 'key collision' method and 'Fingerprint' keying function.



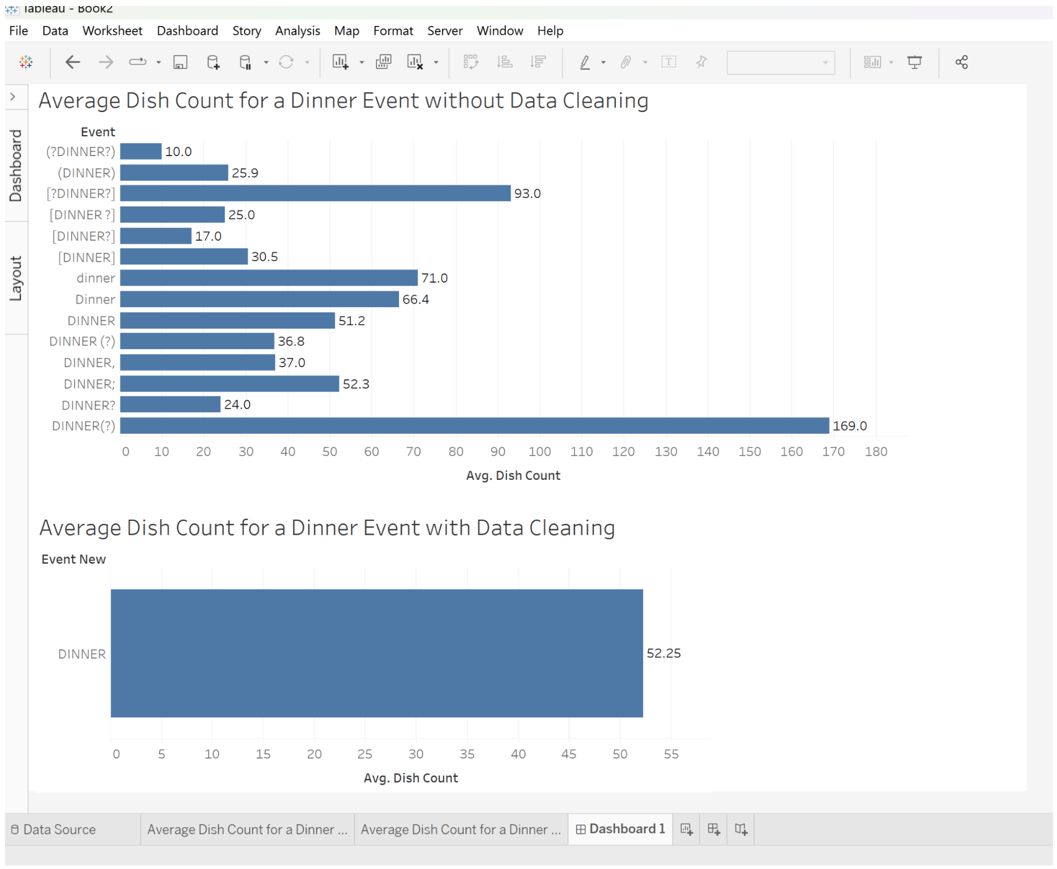
Cluster and edit column "event_new1"

Find groups of different cell values that might be other representations of the same thing. For example, "New York" and "new york" likely refer to the same concept and just differ by capitalization, and "Gödel" and "Godel" probably refer to the same person. Find out more…

Method: Key collision   Keying function: Fingerprint   **68** clusters found

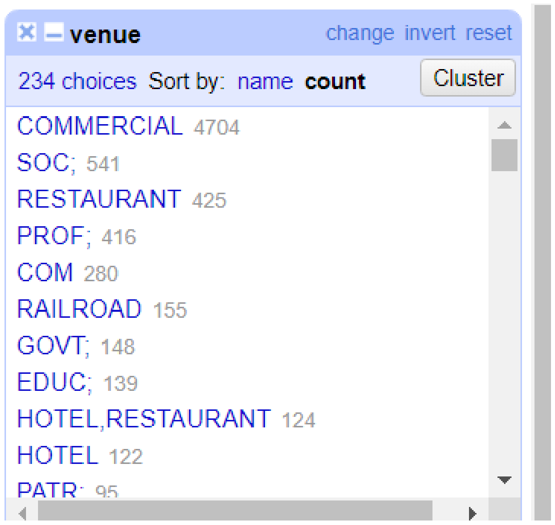
Cluster size	Row count	Values in cluster
15	2148	DINNER (1827 rows); dinner (139 rows); [DINNER] (74 rows); Dinner (70 rows); (DINNER) (10 rows); DINNER; (10 rows); DINNER (?) (5 rows); DINNER(?) (3 rows); dinner (3 rows); [DINNER?] (2 rows); (?DINNER?); DINNER,; DINNER?; [?DINNER?]; [DINNER ?]; Browse this cluster
10	637	LUNCH (520 rows); lunch (55 rows); Lunch (33 rows); [LUNCH] (13 rows); (?LUNCH?) (7 rows); LUNCH; (3 rows); [LUNCH?] (3 rows); (LUNCH); LUNCH (?)

The 'Main' use case would result in misleading values if the data cleaning is not performed and below is the evidence for the result of the query to find 'The Average Dish Count for the event - Dinner'. There are approximately 53 items on an average on a Dinner Menu and without data cleaning this metric would be largely incorrect.



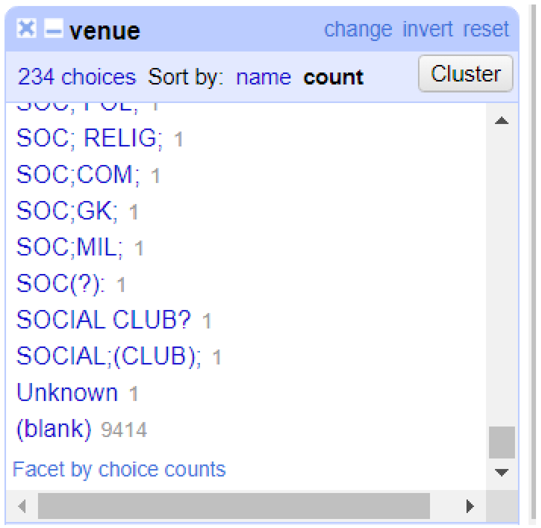
c. 'Never enough' use case U2 : data cleaning is not sufficient

"Find the most popular venue, that is, the venue with the greatest number of menu entries in the Menu dataset." At first glance, the answer seems to be "COMMERCIAL". By sorting the data in OpenRefine, we may look at the total count of the 234 distinct venues that are listed in the Menu dataset. "COMMERCIAL" is listed first.



COMMERCIAL	4704
SOC;	541
RESTAURANT	425
PROF;	416
COM	280
RAILROAD	155
GOVT;	148
EDUC;	139
HOTEL,RESTAURANT	124
HOTEL	122
PATR;	95

However, upon further analysis, that could possibly not be the case, as there are 9414 venues unaccounted for, surpassing the 4704 "COMMERCIAL" venues seen in the dataset.



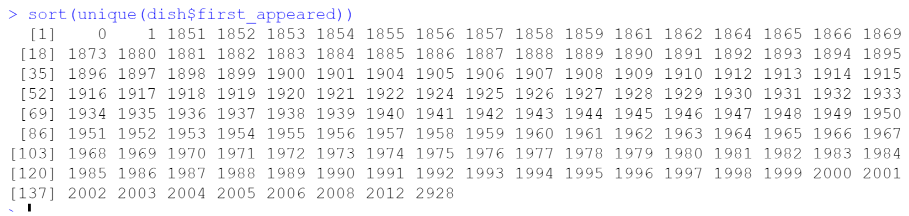
SOC; RELIG; 1
SOC;COM; 1
SOC;GK; 1
SOC;MIL; 1
SOC(?); 1
SOCIAL CLUB? 1
SOCIAL;(CLUB); 1
Unknown 1
(blank) 9414

No matter the amount of data cleaning, there is nothing which can be used to extrapolate the missing venue information from the remaining data provided. Hence, any results we get from our query will be misleading and may differ from the actual truth. We require additional information that is not present within the data, and hence the data cleaning will be "never enough."

4. Data Quality Problems

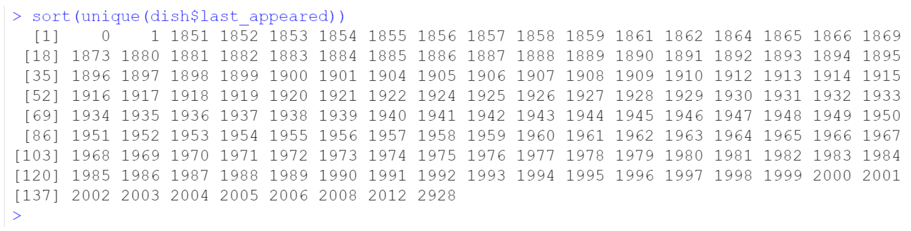
a. There are several data quality issues that we spotted during our initial exploratory analysis of the dataset.

- The 'first_appeared' column (which is the year in which the Dish was introduced on the menu for the first time) in the 'Dish.csv' has to range in between 1840 to the present year 2023. We spotted some rows having values = {0,1,2928}. These are erroneous data and have to be handled appropriately.



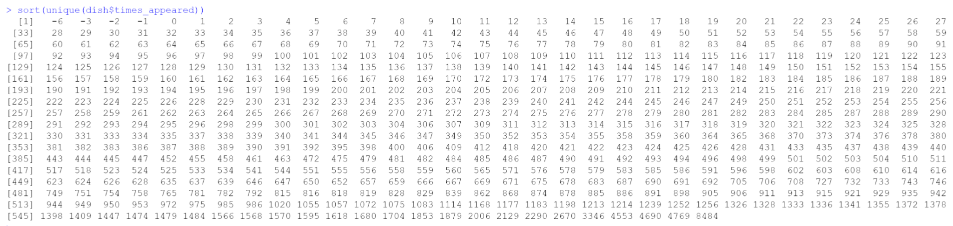
```
> sort(unique(dish$first_appeared))
  [1]    0    1 1851 1852 1853 1854 1855 1856 1857 1858 1859 1861 1862 1864 1865 1866 1869
 [18] 1873 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895
 [35] 1896 1897 1898 1899 1900 1901 1904 1905 1906 1907 1908 1909 1910 1912 1913 1914 1915
 [52] 1916 1917 1918 1919 1920 1921 1922 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933
 [69] 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950
 [86] 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967
[103] 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984
[120] 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001
[137] 2002 2003 2004 2005 2006 2008 2012 2928
```

- The 'last_appeared' column(which is the year in which the Dish was served on the menu for the last time) in the 'Dish.csv' has to range in between 1840 to the present year 2023. We spotted some rows having values = {0,1,2928}. These are erroneous data and have to be handled appropriately.



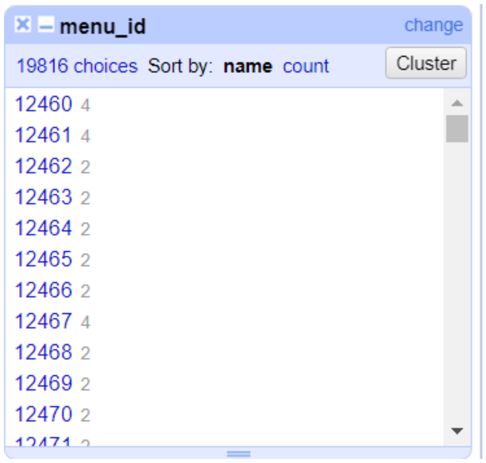
```
> sort(unique(dish$last_appeared))
  [1]    0    1 1851 1852 1853 1854 1855 1856 1857 1858 1859 1861 1862 1864 1865 1866 1869
 [18] 1873 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895
 [35] 1896 1897 1898 1899 1900 1901 1904 1905 1906 1907 1908 1909 1910 1912 1913 1914 1915
 [52] 1916 1917 1918 1919 1920 1921 1922 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933
 [69] 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950
 [86] 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967
[103] 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984
[120] 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001
[137] 2002 2003 2004 2005 2006 2008 2012 2928
>
```

- The 'times_appeared' column (which is the number of times a Dish is presented on the menu in one or many sections of the menu) in the Dish.csv cannot be negative. We found some rows having values {-6, -3, -2, -1} which are bad data.



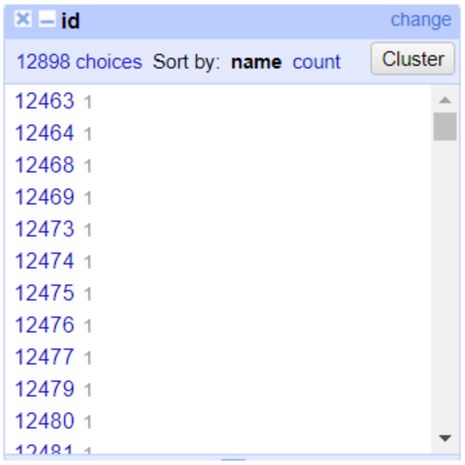
```
> sort(unique(dish$times_appeared))
  [1]   -6   -3   -2   -1    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26   27
 [33]   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56   57   58   59
 [65]   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90   91
 [97]   92   93   94   95   96   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123
[129]  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155
[161]  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  172  173  174  175  176  177  178  179  180  182  183  184  185  186  187  188  189
[193]  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221
[225]  222  223  224  225  226  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  244  245  246  247  249  250  251  252  253  254  255  256
[257]  257  258  259  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  287  288  289  290
[289]  291  292  293  294  295  296  298  299  300  301  302  303  304  306  307  309  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  328
[321]  330  331  333  334  335  337  338  339  340  341  344  345  346  347  349  350  352  353  354  355  358  359  360  364  365  368  370  373  374  376  378  380
[353]  381  382  383  386  387  388  389  390  391  392  395  398  400  406  409  412  418  420  421  422  423  424  425  426  428  431  433  435  437  438  439  440
[385]  443  444  445  447  452  455  458  461  463  472  475  479  481  482  484  485  486  487  490  491  492  493  494  496  498  499  501  502  503  504  510  511
[417]  517  518  523  524  525  533  534  541  544  551  555  556  558  559  560  565  571  576  578  579  583  585  586  591  596  598  602  603  608  610  614  616
[449]  623  624  626  628  635  637  639  646  647  650  652  657  659  666  667  669  671  675  678  683  687  690  691  692  705  706  708  727  732  733  743  746
[481]  749  751  754  758  765  781  782  792  815  816  818  819  828  829  839  862  868  874  878  885  886  891  898  905  906  911  913  915  921  929  935  942
[513]  944  949  950  953  972  975  985  986 1020 1055 1057 1072 1075 1083 1114 1168 1177 1183 1198 1213 1214 1239 1252 1256 1326 1328 1333 1336 1341 1355 1372 1378
[545] 1398 1409 1447 1474 1479 1484 1566 1568 1570 1595 1618 1680 1704 1853 1879 2006 2129 2290 2670 3346 4553 4690 4769 8484
```

- The 'menu_id' column of MenuPage.csv, which is a foreign key of 'id' in Menu.csv, has dangling foreign keys. For example, there are entries with menu_id's 12460, 12461, and 12462. Hence, we would expect these to be valid foreign keys of menu items; however, there are no entries in Menu.csv with id's 12460, 12461, or 12462



19816 choices Sort by: **name** count

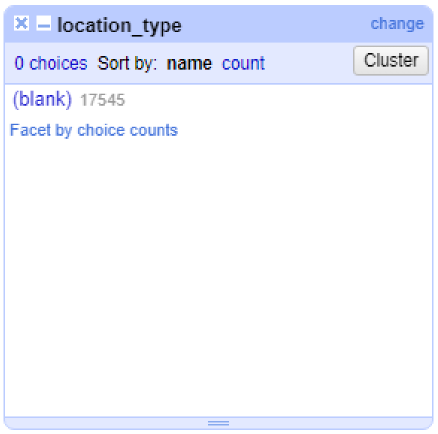
12460	4
12461	4
12462	2
12463	2
12464	2
12465	2
12466	2
12467	4
12468	2
12469	2
12470	2
12471	2



id change

12898 choices Sort by: name count Cluster

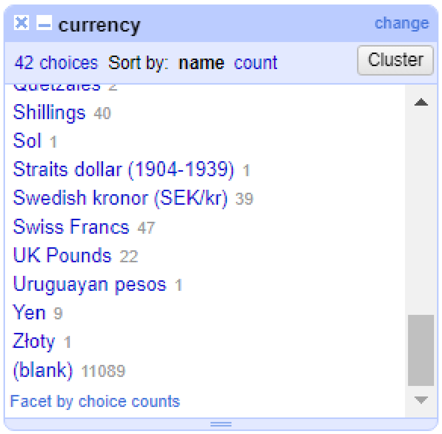
12463	1
12464	1
12468	1
12469	1
12473	1
12474	1
12475	1
12476	1
12477	1
12479	1
12480	1

- There are several issues with data quality in the dataset. One issue is that several columns are missing a substantial amount of data. For example, in the **Menu** table "location_type" is missing 17545 rows of data, "currency" is missing 11098 rows of data, and "currency_symbol" is missing 11089 rows of data.
    - 

      location_type change

      0 choices Sort by: name count Cluster

      (blank) 17545

      Facet by choice counts
    - 

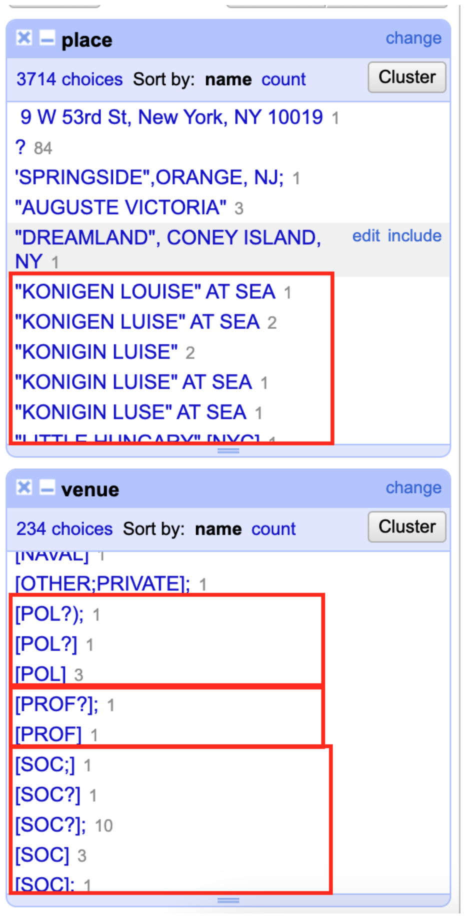
      currency change

      42 choices Sort by: name count Cluster

Shillings	40
Sol	1
Straits dollar (1904-1939)	1
Swedish kronor (SEK/kr)	39
Swiss Francs	47
UK Pounds	22
Uruguayan pesos	1
Yen	9
Złoty	1
(blank)	11089

      Facet by choice counts
- In addition to missing data, there are several instances of ambiguous data / duplication occurring. Digging into the **Menu** table, we can see several instances of duplicate instances for certain columns including "event", "venue", and

"place". In some cases, the values seem similar to existing values, but we cannot be certain due to inconsistent formatting.

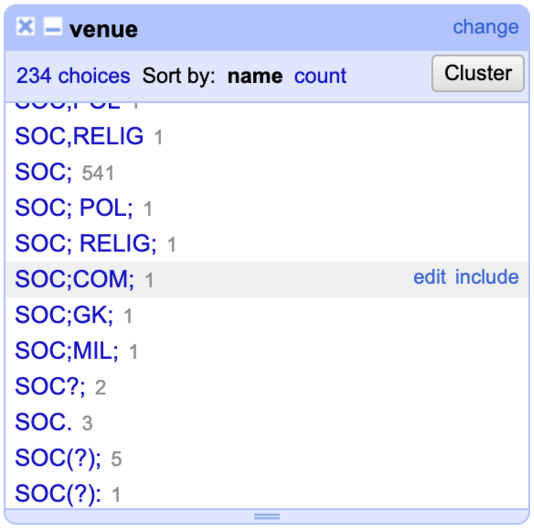

place
3714 choices Sort by: name count

9 W 53rd St, New York, NY 10019 1
? 84
'SPRINGSIDE",ORANGE, NJ; 1
"AUGUSTE VICTORIA" 3
"DREAMLAND", CONEY ISLAND, NY 1
"KONIGEN LOUISE" AT SEA 1
"KONIGEN LUISE" AT SEA 2
"KONIGIN LUISE" 2
"KONIGIN LUISE" AT SEA 1
"KONIGIN LUSE" AT SEA 1

venue
234 choices Sort by: name count

[OTHER;PRIVATE]; 1
[POL?); 1
[POL?] 1
[POL] 3
[PROF?]; 1
[PROF] 1
[SOC;] 1
[SOC?] 1
[SOC?]; 10
[SOC] 3
[SOC]; 1

- Another interesting point is that the **Menu** table has some records that include multiple "venues" separated by a semicolon. See the figure below. There is probably an opportunity to improve the schema here to have a simple association table to maintain data consistency. This structure would enable **Menu's** to be associated with as many venues as required.



venue — change

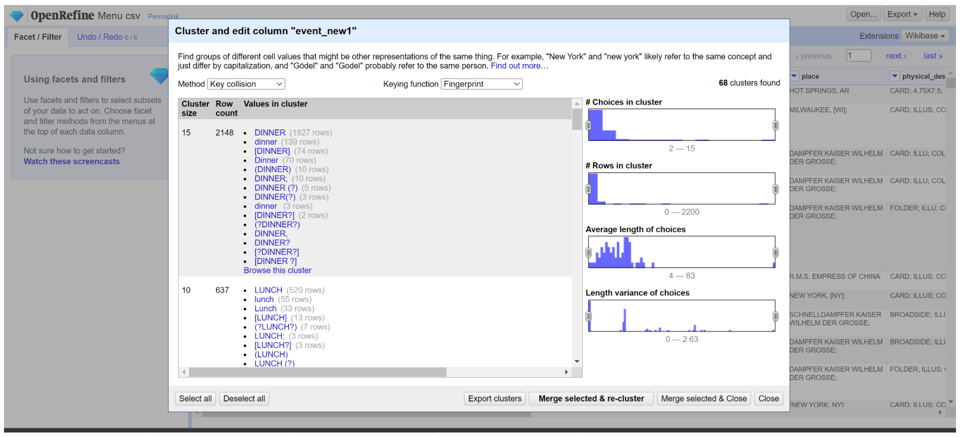
234 choices Sort by: name count — Cluster

SOC,RELIG	1
SOC;	541
SOC; POL;	1
SOC; RELIG;	1
SOC;COM;	1
SOC;GK;	1
SOC;MIL;	1
SOC?;	2
SOC.	3
SOC(?);	5
SOC(?):	1

- Most of the ancillary tables - **MenuPage, MenuItem, and Dish** generally are composed of numeric data which tends to have less blatant data quality issues. Some notes for the **MenuPage** table are that some records are missing the “full height” and “full width” values and the “uuid” has some records that appear to violate the case sensitivity pattern (all lower case). Upon a quick scan, the functional dependencies appear to align with the ER diagram above.

b. Explain why / how data cleaning is necessary to support the main use case U1

- The main use case U1 here is “To find out what is the average number of items on a Menu for Dinner”. To achieve this we analyzed the ‘event’ variable in the Menu.csv and found out that the data was very messy, containing many typos, spelling variations and ambiguities. We wanted to calculate the average dish count on a menu for a Dinner event. So we analyzed further more into the records which have the event name ‘dinner’, there were 14 different spelling variations in which the word dinner was written. Ideally, they all need to be grouped into a single cluster because they all mean the same word. So we merge and re-cluster these into a single cluster with openrefine’s ‘cluster and edit’ feature.



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

Cluster and edit column "event_new1"

Find groups of different cell values that might be other representations of the same thing. For example, "New York" and "new york" likely refer to the same concept and just differ by capitalization, and "Gödel" and "Godel" probably refer to the same person. Find out more…

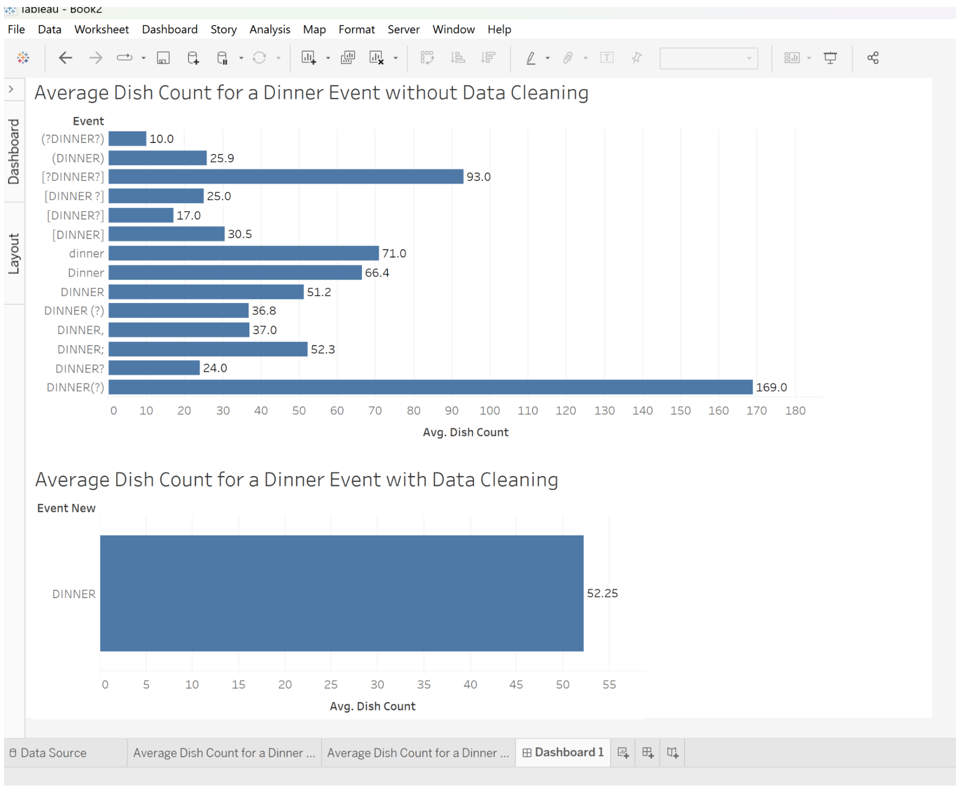
Method: Key collision — Keying function: Fingerprint — **68** clusters found

Cluster size	Row count	Values in cluster
15	2148	DINNER (1827 rows); dinner (139 rows); [DINNER] (74 rows); Dinner (70 rows); (DINNER) (10 rows); DINNER; (10 rows); DINNER (?) (5 rows); DINNER(?) (3 rows); dinner (3 rows); [DINNER?] (2 rows); (?DINNER?); DINNER,; DINNER?; [?DINNER?]; [DINNER ?]; Browse this cluster
10	637	LUNCH (520 rows); lunch (55 rows); Lunch (33 rows); [LUNCH] (13 rows); (?LUNCH?) (7 rows); LUNCH; (3 rows); [LUNCH?] (3 rows); (LUNCH); LUNCH (?)

Choices in cluster: 2 — 15
Rows in cluster: 0 — 2200
Average length of choices: 4 — 83
Length variance of choices: 0 — 2.63

place	physical_des
HOT SPRINGS, AR	CARD; 4.75X7.5;
MILWAUKEE, [WI];	CARD; ILLUS; CO
DAMPFER KAISER WILHELM DER GROSSE;	CARD; ILLU; COL
DAMPFER KAISER WILHELM DER GROSSE;	CARD; ILLU; COL
DAMPFER KAISER WILHELM DER GROSSE;	FOLDER; ILLU; C
R.M.S. EMPRESS OF CHINA	CARD; ILLUS; CO
NEW YORK, [NY];	CARD; ILLUS; CO
SCHNELLDAMPFER KAISER WILHELM DER GROSSE;	BROADSIDE; ILLU
DAMPFER KAISER WILHELM DER GROSSE;	BROADSIDE; ILLU
DAMPFER KAISER WILHELM DER GROSSE;	FOLDER; ILLUS;
NEW YORK, NY]	CARD; ILLUS; CO

Without this data cleaning step the average Dish count would be calculated separately for each of the 14 clusters containing the letters 'dinner' which indeed is not the true value of the average Dish count. The query result of the average Dish count for a Dinner event from the Cleaned up data is ~53 which is very different from the separate average values of the 14 clusters. The below visualization summarizes this point well. Therefore, Data cleaning is a very important and necessary step for U1.



Event	Avg. Dish Count
(?DINNER?)	10.0
(DINNER)	25.9
[?DINNER?]	93.0
[DINNER ?]	25.0
[DINNER?]	17.0
[DINNER]	30.5
dinner	71.0
Dinner	66.4
DINNER	51.2
DINNER (?)	36.8
DINNER,	37.0
DINNER;	52.3
DINNER?	24.0
DINNER(?)	169.0

5. Initial Plans for Phase - 2

1. S1(Michael, Sanjeev, Sushma):
   After reviewing the use case description U1, we are planning on enhancing our use case to span over multiple tables to make a more complex use case. For example, one use case could be a query such as - what venues serve the most expensive dishes (post-1900) on average for dinner? To answer this we'll need to explore and clean data across the Menu, MenuItem, MenuPage, and Dish tables.

2. S2 (Michael, Sanjeev, Sushma):
   We plan on using OpenRefine to identify data quality issues. Specifically,
   1. we plan to perform finding missing data and duplicate data using OpenRefine's facet feature
   2. We plan on making use of the clustering and regex to find duplicate data

        3. While clustering, we plan to explore the effects of using different key collision algorithms and compare the different types of data quality issues they identify

3. S3
   We plan on using OpenRefine, SQLite, Datalog, Python, YesWorkflow tools to perform data cleaning to address the issues we find in S2. Specifically,
    1. Check integrity constraints for ‘id’, cannot be NULL and cannot have duplicates, in Menu.csv, MenuPage.csv, MenuItem.csv and Dish.csv (Michael)
    2. Check the integrity constraints for the ‘menus_appeared’ ,cannot be NULL and cannot be negative, in Dish.csv (Sanjeev)
    3. Check the data quality constraint , if the ‘lowest_price’ is less than or equal to the ‘highest_price’ in Dish.csv (Sushma)
    4. Check if the data quality constraint, ‘first_appeared’ should be less than or equal to ‘last_appeared’ (Michael)
    5. Check if there are any violations for the constraint, ‘page_count’ in the Menu.csv cannot be 0 or NULL (Sanjeev)
    6. Check if there are any violations for the constraint, ‘sponsor’, in the Menu.csv. ‘sponsor’ typically is the Restaurant/Hotel/Event Place where the menu is served and it cannot be NULL or empty (Sushma)
    7. Check if there are any foreign key constraint violations for ‘menu_id’ in MenuPage.csv which should yield a unique, non-NULL/empty id, ‘id’ in Menu.csv. (Michael, Sanjeev, Sushma)

4. S4 (Michael, Sanjeev, Sushma):
   Using SQLite, we will perform queries on the old dataset D and our newly cleaned dataset D’ to verify how much cleaner D’ is than D. We will form these queries once the use case is finalized, with the edge cases in data quality issues in mind.

5. S5(Michael, Sanjeev, Sushma):
   We will use SQLite and OpenRefine to document the changes made to create our new dataset D’.

6. References:

a. https://www.kaggle.com/datasets/nypl/whats-on-the-menu
b. https://openrefine.org/docs