CS513: Theory and Practice of Data Cleaning
Data Cleaning Project

Team name: **Team15**
Kesavar Kabilar : kabilar2@illinois.edu
Samarth Patel: spate504@illinois.edu
Benjamin Fridkis: fridkis2@illinois.edu

Phase I Report

1. Data Description / Narrative

The data provided originates from the New York Public Library and is arranged as four separate files in tabular format using csv (comma-separated value) delineation. Each of the four files can be said to represent one of four entities, as follows: Menu, MenuPage, MenuItem, and Dish.

Menu: The 'Menu' entity consists of instances representing restaurant menus, presumably found throughout the greater New York City metropolitan area. Attributes of the Menu entity include name, sponsor, event, venue, and several others that represent the various identifying information associated with each Menu instance[1].

MenuPage: Each Menu instance is composed of one or more 'MenuPage' instances, the second of the provided entity types. MenuPage instances have page_number, image_id, height, width, and uuid attributes, as well as a menu_id attribute that establishes its relation to one (and only one) Menu instance[2].

[1] See ER diagram for a full list of Menu attributes
[2] Therefore, in the sense of a relational schema, 'menu_id' serves as a 'foreign key' to the Menu entity. In other words, the menu_id attribute of any given MenuPage instance must equal the id of a Menu instance. This is an integrity constraint that will be discussed further.

MenuItem: Next, the ‘MenuItem’ entity represents all individual items that are found on MenuPage instances. Attributes include price, created_at, updated_at, xpos[3], ypos[3], as well as the attributes ‘menu_page_id’ and ‘dish_id’, which relate MenuItem instances to both MenuPage and Dish instances respectively[4].

Dish: Finally, the ‘Dish’ entity provides specific food and/or beverage information, via a ‘description’ attribute. Dish instances also have metadata attributes including menus_appeared (a count of Menu instances on which a Dish instance appears), times_appeared (the number of times a dish appears on MenuPage instances), as well as values representing first and last appearance dates and highest and lowest price sold. As stated, MenuItem instances are related to Dish instances through the MenuItem attribute ‘dish_id’[5].

In summary, Menus are made up of MenuPages, which have MenuItems, which are Dishes that are found on specific Menus. The dataset provided represents New York City area restaurant menu data using four related entities provided in csv file format as described above.

[3] Presumably the x and y coordinates on the physical layout of the associated MenuPage.
[4] ‘menu_page_id’ and ‘dish_id’ are therefore foreign keys to establish the relationship between MenuItem instances and MenuPage and Dish instances, respectively. See footnote 2 above.
[5] Essentially, MenuItem instances are comprised of Dish instances along with additional metadata related to the particular MenuPage instance on which a Dish instance appears.

2. Use Cases

a. Main Case U1 (Requires Cleaning Sufficient to Achieve Proposed Use Case):

i. Find the set of x number of commercial venues providing lowest cost options for a particular dish. E.g. Develop an application that allows a user to specify a dish type (or select from available options) and a number of desired results. Return a list of venues [of count equal to desired results] sorted by price of dish sold, lowest to highest.

Cleaning required to achieve use case:

- Integrity constraints must be met to ensure accurate results.
  - E.g. A Dish must exist as a MenuItem to be considered eligible.
- "Reasonable" range limits must be considered:
  - E.g. Prices of '0' or NULL must be removed/filtered.

b. Zero Cleaning Case U0 (Requires no Cleaning for Use Case):

i. Find all dishes that only appeared in a single year. I.e. Find all Dish instances where first_appeared is equal to last_appeared. (Excludes consideration of dish instances with no year specified or invalid year format.)
   1. This task only requires a simple comparison between two attributes (first_appeared and last_appeared) within a single entity (Dish) to identify matching values. This operation does not need any data cleaning or transformation steps as part of its execution, as it only considers records where valid year values are already present, relying solely on direct value comparison.

ii. Count the total dishes
   1. This task only requires a simple count of rows or distinct id values to calculate the total number of dishes. This operation does not necessitate any data cleaning steps like data type conversion, format standardization, or handling of erroneous values, as it only relies on the presence of records.

iii. Determine the Referential Integrity between two tables
   1. This task only requires simple join statements to check for existing foreign key values that do not have a matching primary key in the referenced table, without needing to clean or transform the data values themselves.

c. Impossible Cleaning Case U2 (No Amount of Cleaning Possible for Use Case):

i. Determine total square area of all listed menus.

While this might be possible for some menus using the 'physical_description' attribute of the Menu entity along with a determination of the number of menu pages via the MenuPage entity, not all Menu instances have any physical_description data, or physical_description does not contain sufficient information (e.g. dimension units are missing).

3. Obvious Data Quality Problems

a. Menu

i. Completeness of Core Identifying Attributes (Completeness)

1. Description: Many Menu instances exhibit missing values (represented by null or empty strings or '?') for identifying attributes such as name, sponsor, event, venue, place, physical_description, date, keywords, language and occasion. This directly impacts the richness and usability of the menu records.
2. Evidence: For example, Menu.id = 12478 has an empty name and sponsor fields. Records in the Menu.id range 12583-12698 frequently show null fields for venue and place. All entries in Menu.csv consistently show these columns as empty/null.
3. Impact On U1: Missing identifying attributes for Menu instances directly impede the ability to accurately locate and identify commercial venues. This is crucial for U1, as the use case aims to find venues providing lowest-cost options for a particular dish. Without complete venue information, it's impossible to fulfill the request of returning a list of venues sorted by price.

ii. Potential Inaccuracy and Redundancy of Derived Attributes

1. Description: The page_count and dish_count attributes are pre-calculated values that summarize the number of pages and dishes for a given menu. While useful, their accuracy is questionable if they are not consistently updated or validated against the actual counts derivable from the MenuPage and MenuItem entities. If they are inaccurate, they introduce a data inconsistency problem. Furthermore, if they can be reliably calculated by a database engine, their storage introduces redundancy.

2. Evidence: For Menu with ID 12884, its stored dish_count is 15. But when counting the total dishes from MenuItem records that are associated with Menu.id = 12884 through their respective menu_page_id links is 19. This indicates that the stored dish_count of 15 for that menu is outdated or inaccurate.
3. Impact On U1: None, U1 is not impacted by this problem.

iii. Extreme Completeness Issues for keywords and language

1. Description: The keywords and language attributes are entirely NULL (or empty) across all inspected Menu instances. This signifies a complete lack of information in these fields, rendering them unusable.
2. Evidence: All entries in Menu.csv consistently show these columns as empty/null.
3. Impact On U1: None, U1 is not impacted by this problem.

iv. Inconsistent String Enclosures

1. Description: Some values (e.g. ‘sponsor’, ‘name’, ‘event’) are surrounded by ‘[‘ and ‘]’ characters, while most are not.
2. Evidence: For id: 12568 the event field has ‘[LUNCH]’ but for id: 12561 the event field has ‘LUNCH’.
3. Impact On U1 : Inconsistent string formats prevent accurate string matching and venue/sponsor identification, causing incorrect restaurant groupings for price comparison in U1.

b. MenuPage

i. Referential Integrity Violations on menu_id

1. Description: The menu_id values in MenuPage may not link to existing Menu.ids, indicating orphaned records and broken foreign key relationships.
2. Evidence: MenuPage.id = 119 has a menu_id of 12460, which is not found in the Menu entity.
3. Impact On U1: Broken menu_id links prevent proper connection between MenuPage records and their parent Menu entity. Since MenuItems link to MenuPages, this prevents MenuItems from being traced back to the Menu (and thus the venue) where they were offered. This directly undermines the ability to group dishes by restaurant for price comparison in U1.

ii. Non-Unique page_number within a menu_id
   1. Description: The page_number is not always unique within a given menu_id, indicating inconsistent or erroneous structural representation of menu pages.
   2. Evidence: For Menu ID 21276, page number 1 appears 2 times.
   3. Impact On U1: None, U1 is not impacted by this problem.

c. MenuItem

i. Referential Integrity Violations on dish_id
   1. Description: The dish_id values in MenuItem often lack corresponding Dish.id entries, creating orphaned menu items.
   2. Evidence: For MenuItem.id = 19171 has a 'dish_id' of nan, which is not found in the 'Dish' entity
   3. Impact On U1: If this integrity constraint is violated, ancillary application functions, such as providing a user the count of available dish alternatives at a given restaurant (in addition to the main objective of providing sought after dish and its price) could be inaccurate.

ii. Ambiguous Validity of price values (Validity)
   1. Description: The price attribute includes 0 values, which are semantically ambiguous (e.g., free, unrecorded, error) in a commercial context.
   2. Evidence: MenuItem.id = 13009 has a 'price' of 0.0. (Associated Dish ID: 256.0) A price of 0 is ambiguous. It could mean free, unrecorded, or an error.
   3. Impact On U1: These are most likely invalid and will heavily skew the results, both main and secondary, of the target use case of finding the subset of lowest price options for a given dish or set of dishes.

iii. Presence of Price Outliers
   1. Description: The price attribute exhibits significant outliers for similar dishes, suggesting data entry errors or misclassification.
   2. Evidence: MenuItem.id = 1208939, Dish Name: '"Frash Special"', Price: 0.25. This price falls outside the typical range (Q1: 0.2, Q3: 0.2) for this dish, which is calculated as between 0.2 and 0.2.
   3. Impact On U1: Price outliers could indicate a mistake/error in data reporting. If statistics such as average, median, high, or low price (among lowest price options) is desired by the user, these outliers will likely skew the results.

iv. High Missingness for high_price
1. Description: The high_price attribute is largely NULL or empty for most MenuItem instances.
2. Evidence: MenuItem.id = 1 has a 'price' of 0.4, but 'high_price' is missing.
3. Impact On U1: A user may want to note the high_price along with the current price for results returned in the lowest options search, to possibly get a sense of a “good deal” (i.e. if a dish might be on sale) or to determine if a given restaurant has lowered prices otherwise. This is not possible with missing high_price values. (However, since most entries do not include this data, this feature is probably not feasible using the given dataset.)

d. Dish

i. Invalid Year Values in first_appeared and last_appeared
1. Description: The first_appeared and last_appeared attributes contain erroneous or nonsensical year values (e.g., 1, 2928).
2. Evidence: For Dish.id = 15 ('Celery'), first_appeared: '1', last_appeared: '2928'. Both of these year values are outside the plausible range.
3. Impact On U1: None, U1 is not impacted by this problem.

ii. Inconsistency and Redundancy of Aggregate Counts
1. Description: The menus_appeared and times_appeared might be inaccurate when compared to derivable counts from other entities, and their storage is redundant if calculable.
2. Evidence: For Dish.id = 7 ('Radishes'), 'menus_appeared' stored: 3262, Calculated: 3265. The calculated cannot be larger than the stored.
3. Impact On U1: If these columns were used to provide a summary of possible dish options, they could introduce inaccuracies. Storing this information is redundant but could speed query processing time, as long as a mechanism is in place to ensure accurate/updated values are present.

iii. Invalid or Meaningless name
1. Description: Some Dish instances have invalid or meaningless name values (e.g., placeholders).
2. Evidence: For Dish.id = 637 has a name '&'.
3. Impact On U1: This dish or others like it could introduce a confusing option if an application allowed users to select from a list of available dishes (to search from for lowest price options inquiry).

iv. Dishes Without Associated Menu Items
1. Description: The Dish instances exist without any linked MenuItem records, meaning they were never offered with a price.
2. Evidence: For Dish.id = 825 ('Rice, Semolina') is listed in the Dish entity but has no corresponding entries in the MenuItem entity.
3. Impact On U1: If a dish is unavailable as a MenuItem, it is essentially not available. Including it as a possible search choice could introduce confusion for a user.

4. Initial Data Cleaning Plan

a. Description of dataset D and matching use case U1.

i. **Menu:** Contains information about individual menus
1. Id: A unique identifier assigned to each menu.
2. Sponsor: The entity or establishment sponsoring the menu.
3. Event: The specific event or mealtime for the menu
4. Venue: An identifier for the venue (Commercial, Government)
5. place: A more general geographical descriptor for the menu's origin (e.g., city, region).
6. Physical_description: Details the tangible characteristics of the menu document itself.
7. Occasion**:** Specifies the particular event
8. Notes: A free-text field for any miscellaneous information that doesn't fit into other structured fields.
9. Call number: A unique identifier used to locate the specific menu
10. Keywords: Empty field
11. Language: Empty Field
12. Date: The date or date range when the menu was in effect. This is crucial for case U1 historical analysis.
13. Location: The full physical address of the commercial venue
14. Location_type: Empty Field
15. Currency: Currency used for menu prices
16. Currency Symbol: Currency symbol
17. Status: The current state of the menu
18. Page_count: total number of pages in the menu
19. Dish_count: Total number of dishes included in the menu

II. **MenuPage:** Detailed information regarding the visual and structural aspects of the menu's presentation.
1. Id: unique identifier for each menu page
2. menu_id: Establishes a link to the Menu, indicating the menu to which this page belongs.
3. page_number: The sequential enumeration of the page within its respective menu.
4. image_id: An identifier for the digital image representation of the menu page.
5. dimensions: Specifics concerning the physical or digital dimensions of the menu page (e.g., height, width).
6. Uuid: unique identifier for each menu page

III. **MenuItem:** Specifics of their presentation on the menu.
1. Id **:**unique identifier for each menu Item
2. menu_page_id : A unique identifier for each menu item as it appears on a given page.
3. Price: The stated price of the item as listed on the menu and very vital field for Case U1.
4. High_price: high price indicator for menuItem
5. Dish_id: unique dish identifier
6. Created_at: tracking data entry for creation
7. Updated_at: tracking data entry for update
8. Xpos: numerical coordinates of the menu item on a scanned image of the menu page
9. YPos: numerical coordinates of the menu item on a scanned image of the menu page

IV. **Dish:** Specific food and beverage items themselves, irrespective of their presentation on a menu.
1. Id: A unique identifier for each distinct food or beverage item.
2. Description: The common appellation of the dish (e.g., "Spaghetti Bolognese," "Caesar Salad").
3. lowest_price: A range of prices at which this dish has been observed, reflecting variations across different menus or venues which will be most vital for U1.
4. Highest_prices: Range of observed prices for this dish across menus/venues.

5. Appearance_counts (times_appeared, first_appeared, last_appeared): A numerical value representing the frequency with which this dish appears across various menus or datasets.

b. Profiling of D to identify the quality problems P that need to be addressed to support U1.

i. **Menu:**

1. Textual Inconsistencies for Categorical Fields:
    a. Remove the leading and trailing whitespace from the Text column. Group the unique Text with similar names
    b. Remove the null values that does not support the U1 case

ii. **MenuPage:**

1. Non-Numeric Page number:
    a. Convert the Text field to integer fields.
    b. Identify instances exhibiting referential integrity constraint violations.
    c. Remove the null values that does not support the U1 case

iii. **MenuItem:**

1. Missing Price Values:
    a. Price field contains some non numeric values and missing values.
2. Invalid Price Values:
    a. Price has some Null values. U1 case can not be possible with the null values.

iv. **Dish:**

1. Inconsistencies for description:
    a. Remove the leading and trailing whitespace from the Text column. Group the unique Text with similar names
2. Non-Numeric lowest and highest prices:
    a. Convert the Text field to integer fields.
    b. If both lowest price ranges are null then remove the field since U1 is not possible to get the lowest price.

c. Performing the data cleaning process using one or more tools to address the problems P (here you should describe which tools you are planning to use, e.g., OpenRefine; Python; etc.).

i. **Menu:**

1. Inconsistent Date Formats and values:
    a. Filter/Remove any instance with a NULL name and place attribute, as these are of no use if they cannot be located.
        i. Tool: SQLite

2. Incorrect/Unnecessary Aggregate stored count values:
    a. Remove columns ‘page_count’ and ‘dish_count’.
        i. Tool: OpenRefine

3. Textual Inconsistencies for Categorical Fields:
    a. Remove the Leading and trailing space with the openrefine and group and rename the inconsistency between names.
        i. Tool: OpenRefine

ii. **MenuPage:**

1. Non-Numeric Page number:
    a. Convert the page number value to integer.
        i. Tool: OpenRefine

2. Integrity Constraint Violations:
    a. Filter out instances exhibiting referential integrity constraint violations.
        i. Tool: SQLite

iii. **MenuItem:**

1. Missing Price Values:
    a. Clean price data by removing currency symbols and non-numeric characters, converting to floats, noting potential NaN values from missing data or conversion errors.
        i. Tool: OpenRefine

2. Invalid Price Values:
    a. Filter out MenuItem rows where price is NaN or 0. Since this will

support the “Lowest price” use case. Record number of rows filtered.

i. Tool: SQLite

iv. **Dish:**

1. Inconsistencies for description:
   a. Standardize description (lowercase, strip whitespace, remove punctuation/special characters).
      i. Tool: OpenRefine

2. Incorrect/Unnecessary aggregate stored count values:
   a. Remove columns ‘menus_appeared’ and ‘times_appeared’.
      i. Tool: OpenRefine

3. Inconsistent Date Formats:
   a. Ensure "first_appeared" and "last_appeared" are converted to datetime, specifically ISO date format.
      i. Tool: OpenRefine

d. Checking that your new dataset D’ is an improved version of D, e.g., by documenting that certain problems P are now absent and that U1 is now supported.

i. **Menu:**

1. Core identifying attributes (name, sponsor, event, venue, place, physical_description, date, keywords, language) will have fewer missing values, enabling comprehensive menu analysis. This provides U1 with accurate venue and menu data, crucial for locating venues offering the lowest-cost dishes.
   a. Run counts of NULL values on key attributes (e.g. name) before and after data cleaning to demonstrate difference between unclean and cleaned data.

ii. **MenuPage:**

1. Page number will be numeric along with the full width and full height for the menu page
   a. Double-check referential integrity constraints to ensure all menu_id values correspond to an id value in the Menu entity. If any IC violations are found, produce a count of violating instances that will be filtered to show the difference between D and D’.

iii. **MenuItem:**
   1. MenuItem instances with invalid or missing prices are filtered out, so now price comparisons are possible for all remaining instances. The data now supports the use case in which dish prices can be compared across all restaurants to find the lowest price offering subset for a given dish or set of dishes.
      a. Run count of prices = 0 or NULL/Empty to demonstrate the difference between unclean and cleaned data.

iv. **Dish:**
   1. This is vital for U1, as it allows for precise navigation and verification of dish locations within a menu when presenting the lowest-cost options and highest price options.
      a. Use OpenRefine log to demonstrate the number of fields updated when stripping out excess whitespace, etc.
      b. Produce a count of all Dishes without at least one corresponding MenuItem representing its availability. These dishes will be filtered out, and this difference will demonstrate D vs D'.

e. Documenting the types and amount of changes that have been executed on D to obtain D′.Documenting the types and amount of changes that have been executed on D to obtain D′.

i. Open Refine log:
   1. All data transformations and cleaning operations performed in OpenRefine, including steps like column splitting, type conversions, and reconciliation.

ii. SQL log and query history:
   1. All SQL commands executed, including data manipulation (INSERT, UPDATE, DELETE) and data definition (CREATE, ALTER, DROP) queries, along with execution results for data Integrity and validations.

iii. Number of rows/columns affected and empty columns are deleted.
   1. Total number of rows and columns that have been altered or removed during the cleaning process, and specifically identifies columns that were entirely empty and subsequently deleted.

iv. Number of duplicates removed.
   1. Count of duplicate records identified and eliminated from the dataset with

use of SQLite, it will improve data uniqueness and accuracy.

v. Removed the trailing and leading spaces
   1. Removed blank spaces from the beginning and end of text strings within various column values, ensuring data consistency and improving searchability and comparison operations.

vi. Number of missing values imputed
   1. Missing values (NaN, null, empty) are imputed using methods from simple (mean, median, mode) for U1 Case.

vii. Convert the numerical data to integer
   1. Data with integer/float value is converted to specific data types so it will optimize the storage and performance along with calculation required for U1 case.

viii. Data Consistency
   1. Ensuring consistency in data formats across all tables and columns to improve data quality and usability across multiple tables.

f. Tentative Task Assignment to Team Members:

Member	Task
Samarth Patel	OpenRefine tasks for Menu and MenuPage entities (Date Formats, Textual Inconsistencies, Non-Numeric Page numbers, Inconsistent String Enclosures), Also, workflow model both “outer” and “inner” data cleaning workflow models.
Kesavar Kabilar	SQLite tasks for Menu and MenuPage entities (Filtering/Removing NULLs, Integrity Constraint Violations)
Benjamin Fridkis	OpenRefine and SQLite tasks for MenuItem and Dish entities (Price Cleaning, Invalid Prices, Date Formats, Description Inconsistencies, Filtering Dishes)
All Team Members	Collaboration on documenting changes (OpenRefine log, SQL log and query history) and verifying D' improvements and peer review each other's clean data. Also, all members will look into resolution IC violations across all the tables.