Inner Workflow Outline:

1. **Initial OpenRefine Cleaning**: We used OpenRefine software to clean whitespace from all cells of the dataset. We also performed title case transformations to all cells. For full history see file below in github repo - "CS-513-Group-Project\cleaning_stages\1_openrefine\history.json"
2. **Tracks Column Splitting:** Using Python programming language with the NumPy and pandas libraries, we extracted the tracks column into a separate table. The tracks column values were split by the "|" separators present, given a unique row id, and a parent album id that tied each song to its corresponding row in the main dataset. Any song title extracted from the tracks column that was whitespace only was dropped at this stage.
3. **Observation Category Genre Repair**: Using Python, NumPy, Pandas, we found all rows with a null category or genre value. If they were both null, we supplied an interim value of "Data". If one value was present, we used that value to fill in the other null column since category and genre are two very related things in music.
4. **Year Repair**: Using Python, NumPy, Pandas, and Spotipy, we dynamically found the release year of rows in the dataset that had a null year value. This was a lengthy process since there were around 4500 values with missing years. We were able to repair about half of those values. Null year rows were not dropped at this point.
5. **Genre Repair**: Using the same libraries listed in the step above, we dynamically repaired genre and category column values that were given the interim value "Data" from step 3. The spotify api does not hold album genre information directly in their database, but keeps a genre value associated with the artist. We used this artist genre value to fill in the missing row values. Only 33 rows needed this fix and the api had information for 11 of them. The remaining rows that could not be fixed were not dropped at this point.
6. **OpenRefine Cleaning**: Reusing OpenRefine we cleaned our newly created tracks table as well as our main Cddb table. We again preferred title case transforms, more whitespace fixing, and a new step of unescape html cleaning. We didn't deem it appropriate as a team to merge genre and category values together due to the high level of subjectivity in the music industry. For instance, "Alt Rock" may not be considered the same as "Alternative" music to some, so we left our genre and category values unchanged. At this point in the process we also dropped all rows with null or interim values. We also replaced any unique character values such as slashes from the song titles in the tracks table where appropriate. For full history file see json files in - "CS-513-Group-Project\cleaning_stages\6_openrefine_drop_bad"
7. **Unit Code Cleaning**: Using the above Python libraries mentioned as well as regex, we cleaned out all rows that had corrupted unit code ascii characters. Between both of our datatables, there were about 400 rows of un-legible text characters that prohibited a user from being able to tell what the value was. We used this step to remove these rows and also perform a data constraint test that no rows in our main Cddb table had an album id that was not present in the

tracks table and vice versa. At this final step in the process we also renamed the Cddb “tracks” column to “album id” to be more intuitive for users to see the link between the datasets. The original “merged_values” column was also removed at this point since it had null values from the beginning.