Web Data Integration
Open Music Data Integration

Anh-Nhat Nguyen[2034311], Ching-Yun Cheng[2112322], Shamalan Rajesvaran[2115475], Yen-An Chen[2113612], and Phelan Lee Yeuk Bun[2053019]

University of Mannheim, Germany - Team 1

1 Introduction

In this Web Data Integration project, we aim to consolidate and analyze standardization from multiple sources to gain comprehensive insights into music streaming trends, track performance, and audience preferences. The project leverages three key datasets:

1. Million Song Dataset with Spotify and Last.fm Features Dataset (csv) [4]: This dataset is an enriched version of the Million Song Dataset, a large-scale music database containing detailed metadata and audio features for over 50,000 tracks with 21 attributes. It integrates additional attributes from Spotify and Last.fm, including audio features like danceability, energy, loudness, and popularity metrics such as tags (list attribute), preview URLs, and genre classifications. The merging of these three data sources provides a comprehensive view of each song, making it suitable for analyzing music trends, listener behaviors, and track popularity across platforms.

2. Apple Music Tracks (csv) [2]: This dataset contains detailed information on 10,000 tracks with 24 attributes sourced from Apple Music. It includes attributes such as artist names, album titles, track features (e.g., tempo, key, mode), and genre classifications. In addition to audio metadata, the dataset provides insights into song popularity metrics and trends across the platform. It is ideal for exploring the characteristics of songs on Apple Music, understanding artist performance, and analyzing trends in genres and musical features.

3. Openmusic API Dataset (json) [3]: This dataset contains information on over 100,000 tracks, including track and album metadata, artist details, and playback types (clean/explicit). Approximately 1,000 track names were extracted from the two previously mentioned datasets, with fuzzy matching logic used to retrieve corresponding data from the open music database. This can be used for in-depth analysis of track consumption patterns, artist popularity, and changes in audience preferences over time.

Together, these datasets will help provide a holistic view of how various musical, commercial, and audience factors contribute to the success of music tracks on different platforms based on various algorithm approach [1].

2 Data Collection and Data Translation

2.1 Data Collection and Dataset

The dataset was obtained from Kaggle in the form of *csv* and OpenMusic API in the form of *csv* and *json*. An overview of dataset attributes is presented in Table 1 below.

Table 1: Dataset structure

Dataset	No Entities	No. Attributes	Attributes
Million Song Dataset with Spotify and Last.fm Features	50,683	21	Track ID, Name, Artist, Spotify Preview URL, Spotify ID, Tags, Genre (*MV 56%*), Year, Duration MS, Danceability, Energy, Key, Loudness, Mode, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, Time Signature
Apple Music Tracks	10,000	24	Artist ID, Artist Name, Collection Censored Name, Collection ID, Collection Name, Collection Price, Content Advisory Rating (*MV 85%*), Country, Currency, Disc Count, Disc Number, Is Streamable, Kind, Preview URL, Primary Genre Name, Release Date, Track Censored Name, Track Count, Track Explicitness, Track ID, Track Name, Track Number, Track Price, Track Time (Milliseconds)
Open Music	100,001	14	album_id, album_name, album_type, album_year, artist_id, artist_name, artist_profile_photo, artist_subscribers, track_id, track_title, is_explicit, track_length, track_views, track_features

2.2 Schema Mapping

2.2.1 Design of the Integrated Schemas The three datasets contain multiple overlapping attributes as seen in Table 2 below. There are 5 attributes within

our integrated schema that overlap across at least 2 of 3 input schemata, namely "Artist", "Track Name", "Genre", "Track Duration" and "Release Date".

Table 2: Table of Integrated Schema Attributes

Attribute Name	**Datatype**	**Datasets in which attribute found**
Artist	string	Million Song Dataset with Spotify and Last.fm Features, Apple Music Tracks, Open Music
Track	string	Million Song Dataset with Spotify and Last.fm Features, Apple Music Tracks, Open Music
Track Explicitness	string	Apple Music Tracks, Open Music
Album Year	decimal	Apple Music Tracks, Open Music
Genre	decimal	Million Song Dataset with Spotify and Last.fm Features, Apple Music Tracks
Duration	decimal	Million Song Dataset with Spotify and Last.fm Features, Apple Music Tracks, Open Music
Danceability	decimal	Spotify Streaming Statistics
Energy	decimal	Spotify Streaming Statistics
Key	decimal	Spotify Streaming Statistics
Loudness	decimal	Spotify Streaming Statistics
Speechiness	decimal	Spotify Streaming Statistics
Acousticness	decimal	Spotify Streaming Statistics
Instrumentalness	decimal	Spotify Streaming Statistics
Liveness	decimal	Spotify Streaming Statistics
Valence	decimal	Spotify Streaming Statistics
Tempo	decimal	Spotify Streaming Statistics

2.2.2 Tools, Approach and Challenges Altova MapForce was utilised during the schema mapping phase. Once the target schema was identified, the variables were screened to ensure that each of them had the same formatting. For example, the duration variable were transformed for 2 of the datasets so that it would be in milliseconds to conform with the data from Apple Music Track. The

challenges encountered during schema mapping included aligning attributes and handling missing data.

2.2.3 Conversion to Target Schema Datasets were converted to the target schema resulting in XML files.

3 Identity Resolution

3.1 Initiate Gold Standard and Challenges

At the start, we opted for a simple similarity metric—Edit-Distance (*Levenshtein Distance*) using the single key `Track`—to build the gold standard for the identity resolution phase. Levenshtein Distance is a widely used general-purpose string similarity metric, particularly effective for handling minor typographical errors, such as added, deleted, or substituted characters. For example, it can detect similarities between *"Someone Like You - (Adele Live)"* and *"Someone Like You - Adele"* (similarity 77%). However, it struggles with reordered components within strings, such as *"Love Song Taylor"* vs. *"Taylor - Love Song"* (similarity 17%), which can occur in music track titles.

This approach revealed limitations in coverage, as many tracks in the dataset shared the same name but differed in attributes such as `Album Name`, `Year`, and `Artist`, making them distinct. This issue arose from the dataset itself, not from the metrics, as using only the `Track` key did not capture the necessary information to distinguish between these cases. Additionally, GPT-4o-mini was used to validate 50 randomly selected cases from 1,000, and its results often diverged from manual labeling, further emphasizing the need for more comprehensive matching criteria. Details of this validation process are discussed in the next section.

The gold standard was constructed by extracting and comparing data from *Apple Music*, the *Million Song Dataset*, and *Open Music DB*. Python tools were used to parse XML files and retrieve key attributes such as `Track` and `ID`, while random sampling ensured the datasets remained manageable.

While Levenshtein Distance was chosen for its simplicity and typo tolerance, its limited coverage required incorporating additional attributes like `Artist` for better accuracy. Manual validation helped address mismatches caused by identical track names with different metadata.

Efficiency was improved using `Dask` for parallel processing and blocking techniques to reduce unnecessary comparisons. However, the resulting gold standard, though sufficient as a baseline, lacked accuracy and required significant refinements to enhance reliability.

3.2 Gold Standard Improvement

To enhance the reliability and efficiency of the gold standard, we implemented several improvements that addressed key limitations of the initial approach.

These improvements focused on refining similarity matching, reducing computational overhead, and leveraging advanced validation techniques.

Similarity Matching Enhancement We enhanced the similarity matching process by concatenating *Track Name*, *Artist Name*, and *Year* into a single string. Similarity scores were calculated using normalized Levenshtein metrics, which allowed us to categorize the results into:

- **Matched Cases**: Similarity scores in the range [0.75, 1].
- **Corner Cases**: Similarity scores in the range (0.65, 0.75).
- **Non-Matched Cases**: Similarity scores in the range [0, 0.65].

This approach improved differentiation between tracks with similar names but differing metadata, such as different release years or artists.

Integration of GPT-4o-mini for Validation We incorporated GPT-4o-mini to validate ambiguous and borderline cases. The model processed 1,173 rows from the *OpenDB and Apple Dataset* and 1,510 rows from the *OpenDB and Million Dataset*. By using contextual prompts containing attributes such as *Track Name*, *Artist Name*, and *Year*, GPT-4o-mini resolved complex cases, such as distinguishing live and studio versions of the same track. The example prompt of Apple Dataset and OpenDB Dataset below illustrates the validation process:

Prompt

You are a music data expert. Compare these music tracks to determine if they represent the same real-world song.

Data to compare:
- `apple_Artist` vs `opendb_Artist`
- `apple_Track` vs `opendb_Track`
- `apple_Album` vs `opendb_Album`
- `apple_Year` vs `opendb_Year`
- `apple_Duration` vs `opendb_Duration`

Rules:
1. Names should be compared case-insensitively, allowing for semantic similarity rather than exact matches.
2. Duration difference should be within ±3000ms.
3. Year difference should be within ±1 year.
4. All attributes must match within thresholds.

Output: Return `1` if the tracks are the same song, or `0` if they are different.

Stratified Sampling and Manual Review We performed stratified sampling using Python's NumPy, ensuring a balanced dataset with a ratio of **50:30:20** for Non-Matched, Corner, and Matched cases. Additionally, manual review was conducted on similarity computations and GPT evaluations to resolve ambiguities and finalize labeling. This step ensured a robust and high-quality gold standard.

Observations and Results These refinements resulted in significant improvements:

- Reduced computational overhead through blocking and Bloom Filtering.
- Improved match accuracy by incorporating additional attributes such as *Artist Name* and *Year*.
- Enhanced validation through GPT-4o-mini for resolving corner cases.

Blocking and Bloom Filtering were grouped into logical blocks based on the first two characters of the concatenated string to restrict comparisons to records within the same block, significantly reducing the number of pairwise evaluations; additionally, a Bloom filter was applied within each block to efficiently pre-filter records, ensuring that only likely matches were compared, which optimized the process by reducing pairwise comparisons by approximately 70

The refined gold standard is robust and ready for the identity resolution phase, with further confirmation expected after data fusion.

3.3 Matching Strategies

3.3.1 Blocking Methods To reduce unnecessary comparisons during identity resolution (IR), we generated blocking keys based on track names. The blocking key was derived from the bigrams of the first three tokens of each track name. We then experimented with two blocking methods: **Standard Blocking** and the **Sorted Neighborhood Method**, both using these track name-based blocking keys.

In the case of running IR for the apple + opendb datasets, the maximum number of entities sharing the same hashed blocking key was 9,696 (for the blocking key "CH"), while the minimum was 1 (for the blocking key "LAINTH"). As for the million + opendb datasets, the maximum number of entities sharing the same hashed blocking key was 67,404 (for the blocking key "RE"), while the minimum was 1,560 (for the blocking key "BL"). When using the Sorted Neighborhood Method, the window size would need to be greater than 9,696 and 67,404 respectively to ensure that no matches were missed. However, such a large window size would significantly increase resource consumption by comparing many irrelevant records.

Given this inefficiency, for both of our entity matching comparisons, we ultimately decided to adopt **Standard Blocking**, which efficiently grouped entities based on their blocking keys without requiring extensive resource expenditure or risking lost matches.

3.3.2 Similarity Metrics As mentioned above, in the entity matching for apple + opendb, we use **track name, artist name, album name,** and **album year** as the basis for determining entity matching. As for entity matching for million + opendb, since there is no album name attribute in million dataset, we only use **track name, artist name**, and **album year** as the basis. For each attribute with a data type of string, we tested various metrics, including edit-based (Levenshtein, Jaro, Jaro-Winkler), token-based (Jaccard), and phonetic (Soundex). For numeric attribute, we use absolute difference of 2 years.

3.4 Evaluation

3.4.1 Metrics and Analysis After testing over 50+ combinations of comparators for entity matching between apple + opendb datasets and million + opendb datasets, the metrics shown in Table 3 were found to be the most suitable for their respective attributes.

Table 3: Comparators Used for Entity Matching Across Datasets

Dataset	**Attribute (Comparator)**	**Reason for Effectiveness**
apple+opendb; million+opendb	Track Name (Jaccard)	Track names often include variations like additional descriptors, e.g., single, feat.
apple+opendb	Artist Name (Jaro-Winkler)	Artist names are short and structured. Most typos occur in the last name rather than the first name.
million+opendb	Artist Name (Equal)	Many entities in million + opendb have the same track name but different artist names. Within those entities, the artist names are quite similar in some cases; therefore, the equal comparator is needed to clearly distinguish ambiguous entities.
apple+opendb	Album Name (Jaccard)	Most of the singles use the track name as their album name, so the same pattern applies here.
apple+opendb; million+opendb	Album Year (Absolute Difference ± 2)	Accounts for real-world scenarios like re-releases and recording/release year discrepancies.

We organized some of our IR tests in Table 4. Those combination we finally chose are highlighted in yellow. In the results for apple + opendb, we intuitively selected the one that demonstrated the best performance across Precision, Recall, and F1-score. Upon closer inspection of the correspondences, the matches were indeed highly accurate, confirming our choice.

However, in the results for million + opendb, after reviewing the correspondences, we decided to select the option that did not achieve the best performance metrics. This decision was based on the presence of more ambiguous data in this comparison, which, combined with an insufficient golden standard to accurately use the album year to distinguish different entities, resulted in false positives within the correspondences. Under these circumstances, the selected option demonstrated the most balanced precision and recall, minimizing false positives while still capturing the majority of true matches. This balance made it the optimal choice for this dataset.

Table 4: Entity Matching Benchmark Table

Datasets	**Matching Rule**	**B**	**P**	**R**	**F1**	**# Corr**
apple+opendb	Track (J*), Artist (JW*), Album (J), Album Year (2Y*)	SNB Track (20)	0.99	0.99	0.79	85
apple+opendb	Track (J), Artist (JW), Album (J), Album Year (2Y)	SNB Track (60)	0.99	0.82	0.89	105
apple+opendb	Track (J), Artist (JW), Album (J), Album Year (2Y)	Standard Track	0.99	0.93	0.96	120
apple+opendb	Track (L), Artist (JW), Album (L*), Album Year (2Y)	Standard Track	0.99	0.92	0.95	118
apple+opendb	Track (S*), Artist (JW), Album (S), Album Year (2Y)	Standard Track	0.97	0.95	0.96	170
apple+opendb	Track (J), Artist (J), Album (J)	Standard Track	0.87	0.94	0.91	291
million+opendb	Track (J), Artist (Equal), Album Year (2Y)	SNB Track (20)	0.94	0.72	0.82	684
million+opendb	Track (J), Artist (Equal), Album Year (2Y)	SNB Track (60)	0.92	0.85	0.88	835
million+opendb	Track (J), Artist (Equal), Album Year (2Y)	Standard Track	0.92	0.94	0.93	942
million+opendb	Track (J), Artist (JW), Album Year (2Y)	Standard Track	0.91	0.99	0.95	2.078
million+opendb	Track (S), Artist (JW), Album Year (2Y)	Standard Track	0.88	0.94	0.91	1.326
million+opendb	Track (J), Artist (Equal)	Standard Track	0.88	0.94	0.91	1.082

***Note:** J: Jaccard; JW: Jaro-Winkler; 2Y: Absolute Difference ±2; S: Soundex; L: Levenshtein

Although the data in the table suggests that the results of Entity Matching are reasonably satisfactory, a closer examination of the million + opendb correspondences reveals that the differentiation of entities based on the album year is insufficient. This limitation is also reflected in the results of our Group Size analysis.

Table 5: Group Size Analysis

Group Size	**2**	**3**	**4-7**	**8-13**	**14**
Frequency	650	104	45	4	0
Distribution	81%	13%	6%	0%	0%

3.4.2 Error That Remain Our IR results for album years continue to show inconsistencies, with mismatched or slightly inaccurate years persisting even when using a tolerance of ± 2 years. These discrepancies are particularly evident when dealing with re-releases, remasters, or cases where album metadata varies significantly across datasets.

We think that the root cause of this issue lies in the golden standard we employed, which was derived by concatenating track name, artist name, and album year into a single string and performing an initial comparison using the edit-based Levenshtein metric. Since album years were not accurately distinguished in this process, it led to the observed inconsistencies. This misalignment in album years will negatively impact attribute consistency during the Data Fusion phase. If this issue remains unresolved, the resulting fused data will likely inherit these inconsistencies, compromising its overall quality and trustworthiness.

4 Data Fusion

This section presents our approach to data fusion, beginning with our evaluation setup and gold standard creation, followed by our fusion rules and strategies, and concluding with a comprehensive quality assessment of the fused dataset.

4.1 Gold Standard Creation

A gold standard comprising 20 samples was created, based on entities from two categories: 10 entities that existed in both the *opendb* and *apple* datasets, and 10 entities that existed in both the *opendb* and *million* datasets. To ensure accuracy, the attributes used during the fusion process were manually reviewed using external sources such as the Spotify API and Wikipedia.

4.2 Fusion Rules

4.2.1 Conflict Resolution Strategies In the data fusion process, conflict resolution strategies are essential to derive the most accurate and reliable representation of records when combining multiple datasets. Our conflict resolution strategies include using reliable datasets with sufficient attribute intersection for fusion, as well as strong identity resolution results that maintain quality of data. We decided to use all of our datasets (apple, million, opendb) as they fulfil our prerequisites with IR results as shown above. We had assigned a provenance score to each dataset in order to reflect its reliability and accuracy. 'apple' dataset was given the highest score (3.0), followed by 'million' (2.0), and then 'opendb' (1.0). This scoring system allowed the fusion process to favor attributes from the more reliable datasets when resolving conflicts.

4.2.2 Specific Fusion Rules Our team have included specific fusion rules for resolving conflicts in key attributes to ensure that the fused dataset accurately reflects the most reliable, comprehensive, and meaningful information. These rules are tailored to the nature of each attribute and the type of conflicts typically encountered.

For the 'Artist' attribute, the Shortest String Strategy was implemented through ArtistFuserShortestString(). This approach resolves discrepancies by selecting the shortest, yet most concise, representation of an artist's name, eliminating unnecessary or redundant descriptors while preserving clarity (e.g., choosing "Adele" over "Adele Laurie Blue Adkins").

Similarly, for textual fields like 'Album' and 'Track', the Shortest String Strategy was applied through AlbumFuserShortestString() and TitleFuserShortestString() respectively. This strategy ensured that album and track names were concise and free of extraneous details, such as track version descriptors, while retaining essential information. For example, a track title such as "Imagine" was favored over "Imagine - 2011 Remastered Version" to maintain simplicity and clarity without losing its core identity.

For Album Year, the Most Recent Value Strategy, applied through AlbumYearFuserMostRecent(), prioritized the latest recorded year from the datasets. This rule is particularly effective for resolving inconsistencies stemming from re-releases or updates in metadata, ensuring that the fused dataset reflects the most up-to-date temporal information. Similarly, the Most Recent Value Strategy was utilized for the Duration and Track Explicitness attributes through DurationFuserMostRecent() and ExplicitnessMostRecent() respectively. These rules leveraged the recency of data sources to ensure accuracy for numeric values like track duration and categorical data like explicitness labels, which are more likely to evolve over time.

4.2.3 Dataset Overview The post-fusion dataset comprises a total of 803 unique records, reflecting a successful consolidation of overlapping information from the input datasets. The density of attributes, particularly for fields such as

'Artist', and 'Album', reached an impressive consistency level of 1.00, ensuring that these attributes are fully populated across all records. Other attributes, such as 'Duration', 'Track' and 'Track_Explicitness', exhibit high consistency levels of 0.88, 0.98 and 0.99 respectively, indicating minimal discrepancies. Table 6 presents a representative sample of records from our fused dataset, illustrating the structure and quality of the integrated data.

Table 6: Fused Data Records Sample

Song ID	Artist	Track	TE*	AY*	AN*	Duration(ms)
B37836+ C35617	Acid King	Silent Circle	F	1999	Busse Woods	452,000
B58733+ C15258	Wolfmother	10,000 Feet	F	2009	Cosmic Egg (Deluxe)	249,000
A1116+ B2530+ C50008	One Direction	What a Feeling	F	2015	Made In The A.M. (Deluxe Edition)	201,000
A6734+ B35951	Panic! At The Disco	One of the Drunks	T	2018	Pray for the Wicked	199,000
B71278+ C6239	Lily Allen	Friday Night	T	2006	Alright, Still	187,000
A4455+ B64467	Jelly Roll	Son Of A Sinner	F	2021	Ballads of the Broken	233,000

***Note:** TE: Track Explicitness; AY: Album Year; AN: Album Name

4.3 Quality Evaluation

We conducted extensive testing with various combinations of fusion methods. Table 7 shows four representative strategies that combine different fusion methods, including LongestString(), FavourSource(), ShortestString(), and MostRecent(). Strategy 4 demonstrated the highest accuracy at 84% and was therefore selected as our optimized fusion strategy.

The detailed performance metrics of our optimized fusion strategy across different attributes are presented in Table 8. While most attributes achieved high accuracy, Album Name information showed notably low accuracy (50%) due to the frequent occurrence of the same track appearing in different album releases (e.g., singles, original albums, compilation albums). This common practice in

Table 7: Data Fusion Strategy Comparison

Fusion Strategy	Method	Accuracy
Strategy 1	ArtistFuserLongestString(), AlbumYearFuserVoting(), AlbumFuserLongestString(), DurationFuserFavourSource(), TitleFuserLongestString(), ExplicitnessFuserFavourSource()	80%
Strategy 2	ArtistFuserFavourSource(), AlbumYearFuserFavourSource(), AlbumFuserFavourSource(), DurationFuserFavourSource(), TitleFuserFavourSource(), ExplicitnessFuserFavourSource()	80%
Strategy 3	ArtistFuserShortestString(), AlbumYearFuserVoting(), AlbumFuserShortestString(), DurationFuserFavourSource(), TitleFuserShortestString(), ExplicitnessFuserFavourSource()	82%
Strategy 4 (Optimized)	ArtistFuserShortestString(), AlbumYearFuserMostRecent(), AlbumFuserShortestString(), DurationFuserMostRecent(), TitleFuserShortestString(), ExplicitnessFuserMostRecent()	84%

the music industry creates challenges in determining the canonical album name for a track, leading to lower accuracy and consistency scores for Album Name (50%, 81%) and AlbumYear (75%, 56%) attributes.

5 Conclusion and Future Work

5.1 Limitations of the Project

The limitation that has been identified is the occurrence of incomplete data. The insufficient and inconsistent data that is spread across the attributes from the different data sources proved to be a significant stumbling block as it impact the fusion accuracy. The missing attribute such as that occurring in the Album Name and AlbumYear caused there to be gaps when the datasets were merged. This ultimately reduced the accuracy for the attributes. There was also the occurrence of inconsistent coverage of data. It was identified that across the 3 data sources, there was a varying level of detail for all the attributes which

Table 8: Optimized Fusion Strategy Performance

Attribute	Method	Accuracy	Consistency	Density
Artist	ArtistFuserShortestString()	95%	100%	100%
AlbumYear	AlbumYearFuserMostRecent()	75%	56%	100%
Album	AlbumFuserShortestString()	50%	81%	100%
Duration	DurationFuserMostRecent()	85%	88%	100%
Track	TitleFuserShortestString()	85%	98%	100%
TrackExplicitness	ExplicitnessFuserMostRecent()	90%	99%	100%

eventually led to difficulties in harmonzing the data. For example, the same song records may be listed differently across the datasets as some may list them as the original album release, while the other source may list them as compilations or re-releases at a later date. Furthermore, the computational constraints were particularly challenging to overcome. The complex fusion strategies required longer runtimes which resulting in the process being particularly challenging to test multiple strategies approaching the deadline. Moreover, there were ambiguity in the album information. The appearance of re-releases and compilations caused the tracks to appear across varying different albums such as singles and compilations. This ultimately made it particularly difficult to determine the root Album Name. The lack of a unique identifier for the Albums made it a challenge to differentiate between the different versions and variations.

5.2 Recommendations for Future Improvements

For future development of this particular project, a plausible approach would be to develop and integrate advanced machine learning methodology which shall be designed specifically for entity resolution. For example, a graph neural network (GNN) or transformer-based architecture can be employed to effectively identity corresponding entities across the datasets. An ML model can also be trained to go about the attribute weighting. For example, the ML model may be utilised to dynamically decide on the weights of the various different sources based on the reliability of the available attributes. This would ensure an accurate fusion process. Furthermore, it would be greatly advices to expand the dataset collection so as to enrich the data sources. A broader data collection would enable the inclusion of diverse attributes and data sources. This may include areas such as verified music streaming platforms and industry metadata. Moreover, in an effort to address the album name disambiguation, a specialized algorithm can be developed in order to handle re-released tracks. This approach would leverage on contextual information that accompanies the records such as the release dates, genres, and track listings to identify the root Album Name. A protocol

can also be put in place in order to go about consistency checking. For example, the AlbumYear and similar date-related attributes, can be implemented in the algorithm in order to cross check the release years with external source and highlight any inconsistencies.

References

1. AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, June 2012.
2. Kanchana1990. Song dataset: 10,000 apple music tracks. URL: https://www.kaggle.com/datasets/kanchana1990/apple-music-dataset-10000-tracks-uncovered.
3. OatsCG. OatsCG/Openmusic-Server-Specs, September 2024. original-date: 2024-01-10T01:36:38Z. URL: https://github.com/OatsCG/Openmusic-Server-Specs.
4. UndefineNull. Million song dataset + spotify + last.fm. URL: https://www.kaggle.com/datasets/undefinenull/million-song-dataset-spotify-lastfm.